Iowa City Monthly Bus Ridership Prediction in 1983

Jiamin Tan

Spring 2019

Abstract

This is a technical documentation of how I build a time series model from scratch to predict monthly average bus ridership in Iowa City, IA. A Cryer (1986) dataset of the weekday bus ridership (monthly average) from September 1971 to December 1982 is used for this project. The dataset is retrieved from the Time Series Data Library (TSDL) maintained by Hyndman and Yang (2018). This project constructs a seasonal autoregressive integrated moving average (SARIMA) model and predicts the weekday bus ridership (monthly average) in the first five months in 1983. Based on the prediction, I conclude that there will be around a 1000-ridership increase in each month in 1983 from January to May comparing to the ridership in 1982.

Table of Contents

- 1. Introduction pp.2
- 2. Method pp.2
  - 2.1 The Original Data pp.2
  - 2.2 Transformation pp.3
    - 2.2.1 Box-Cox Transformation pp.3
    - 2.2.2 Log Transformation pp.3
    - 2.2.3 Square Root Transformation pp.3
  - 2.3 The ACF and PACF Plots of the Log Transformed Data pp.3
  - 2.4 Choosing Parameters for SARIMA (p, 1, q) x (P, 1, Q)$_{12}$ pp.5
  - 2.5 Fitting SARIMA Model with Fixed Q = 0 pp.6
  - 2.6 Alternative Model Fitting with AR pp.7
  - 2.7 Alternative Model Fitting with SARIMA (p, 1, q) x (P, 0, Q)$_{12}$ pp.8
  - 2.8 Diagnostic Checking pp.8
    - 2.8.1 Unit roots pp.8
    - 2.8.2 Residual Plots, ACF & PACF Plots of the Residuals pp.9
    - 2.8.3 Portmanteau Test pp.9
    - 2.8.4 Other Tests of Residuals pp.10
  - 2.9 Validation Using the Test Set pp.11
  - 2.10 Final Model Selection pp.11
  - 2.11 Predicting the “Future” pp.11
- 3. Conclusion pp.12
- 4. Acknowledgement pp.12
- 5. References pp.12
- Appendix: R Codes pp.13-69

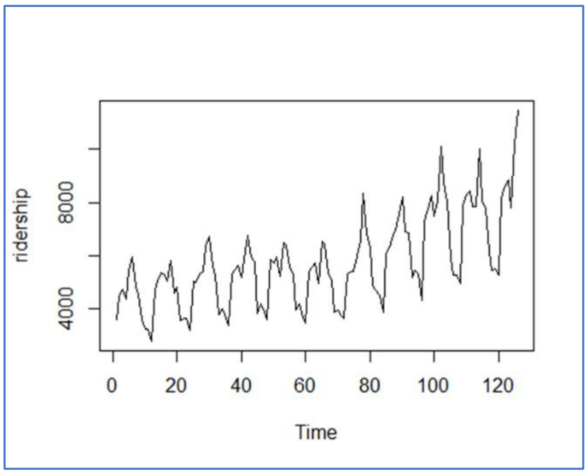
1 Introduction

Public Transportation planning is one of the most important aspects in urban planning because many people rely on buses for everyday commuting. How many buses that a bus company should put on their service is crucial to the bus operation every day, and a company should pay closely attention to when there are more passengers. In this project, I use the Cryer (1986) data of the monthly average weekday bus ridership in Iowa City from September 1971 to December 1982 to build a time series model. The model then forecasts how the ridership will change in the first five months of 1983 comparing to the ridership in the same months in 1982. The numbers predicted can then be considered in decision making by the bus company for bus operations.

2 Method

2.1 The Original Data

There are 136 observations in total, and the time range of the data is from September 1971 to December 1982. I leave 10 observations out and let them to be the test set for later use. Therefore, I use 126 observations to build our model. Figure 1 shows the time series plot of the 126 observations and the data distribution in histogram.



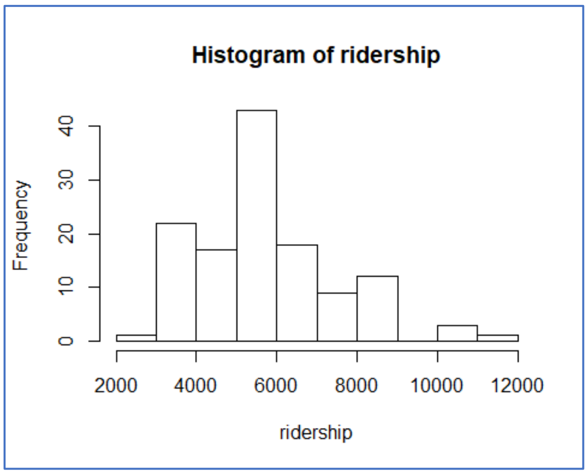


Figure 1 – Time Series Data and the Histogram of the 126 Observations used for Model Building

From the time series data, it can be observed that there is an obvious increasing trend. A yearly seasonality is also obvious from the time series plot. It can also be observed that the variance of the data is changing over the time. Moreover, the histogram suggests a log normal distribution. Thus, in the following section 2.2, I will transform the data to be ready to use.

2.2 Transformation

2.2.1 Box-Cox Transformation

I first try to use the boxcox() function from the "MASS" package in R to derive the lambda value. In figure 2 below, zero is contained in the 95% confidence interval of lambda, so I choose lambda equals to zero which suggests using the log transformation on the data. This corresponds to the log normal distribution of the original data observed in the previous section.

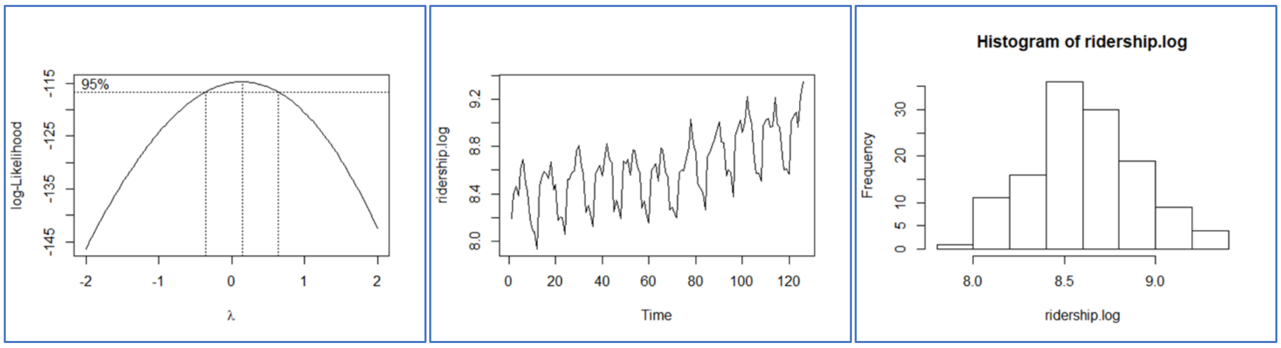


Figure 2 – 95% CI of Lambda, Time Series Plot after Log Transformation, Histogram After Log Transformation

2.2.2 Log Transformation

Therefore, I perform a log transformation to the data, and it can be observed that the time series plot of the transformed data has stable variance. In addition, the histogram in Figure 2 indicates the transformed data is close to normal distribution after the log transformation.

2.2.3 Square Root Transformation

I also try the square root transformation for the original data, but the transformed time series plot shows a change of variance over time, and the histogram shows that the square root transformed data is not normally distributed. So, I decide not to use the square root transformation and will proceed with the log transformed data. For specific plots and R codes, please refer to the Appendix 2.3.

2.3 The ACF and PACF Plots of the Log Transformed Data

Figure 3 shows the sample ACF and PACF plots of the log transformed data, the maximum lag is determined by n/4 which in this case is 126/4 ≈ 30.

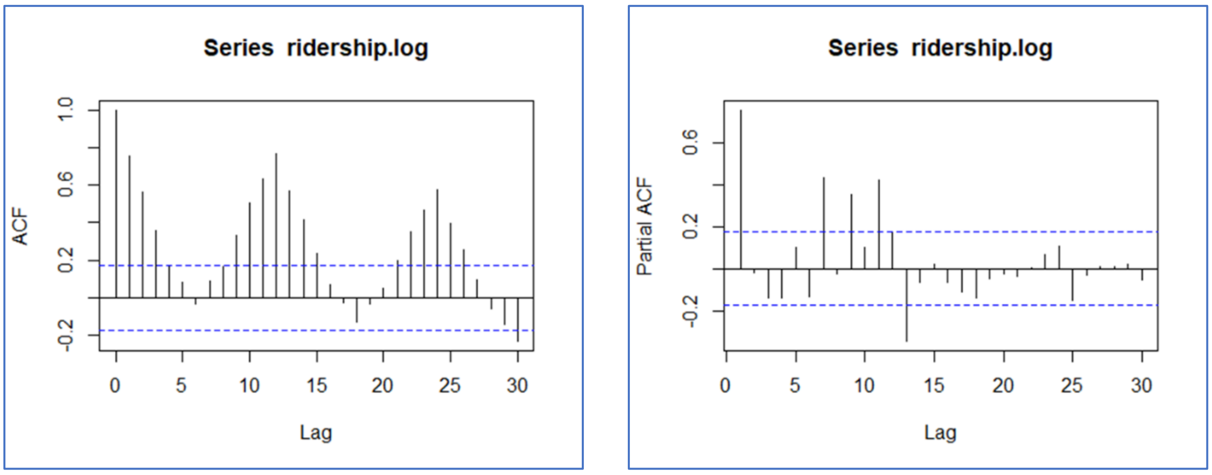


Figure 3 – ACF and PACF Plots of the Log Transformed Data

As I can tell from the ACF plot, there is an obvious seasonal pattern at lag 12 and 24, so my first step is to deseasonalize the data at lag 12. By doing so, it is observed that the variance after differencing deceases from 0.0350 to 0.0081, so this differencing should be done. After that, I get the ACF and PACF in Figure 4.

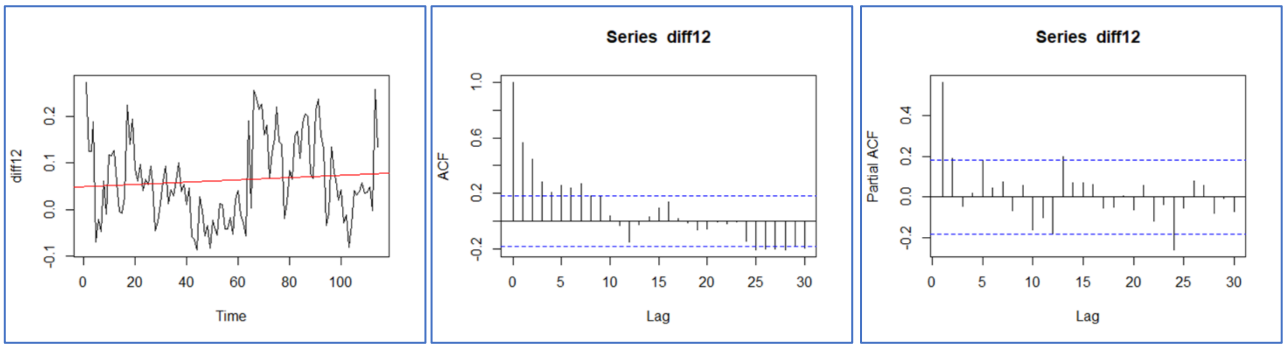


Figure 4 – Time Series Plot, ACF and PACF after Differencing at lag 12

Since the ACF does not quickly decay to 0, the stationarity is not reached. This is also indicated by the time series plot after differencing at lag 12 in figure 4. Therefore, I decide to difference at lag 1 again to eliminate the trend of the data. As I difference at lag 1 of the differenced data, the variance decreased again from 0.0081 to 0.0067, so this differencing also should be done. The time series plot of the de-seasonalized and de-trended data is in Figure 5. The data now is more stationary.

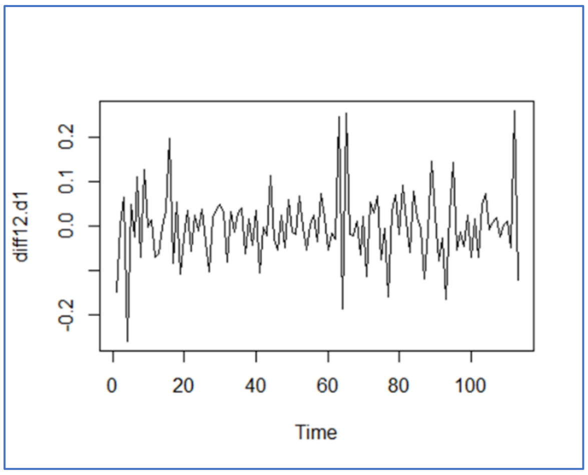


Figure 5 – Time Series Plot of After Differencing at Lag 12 and 1

I tried to difference again at lag 1, but this time the variance increases from 0.0067 to 0.1800, so I cannot proceed with that. Therefore, I will use the de-seasonlized and de-trended once data for the following steps. I can conclude that I will use a SARIMA model with d = 1 and D = 1.

2.4 Choosing Parameters for SARIMA (p, 1, q) x (P, 1, Q)$_{12}$

Since the data after two differencing processes becomes stationary, I can now identify the rest parameters from the sample ACF and PACF plots.

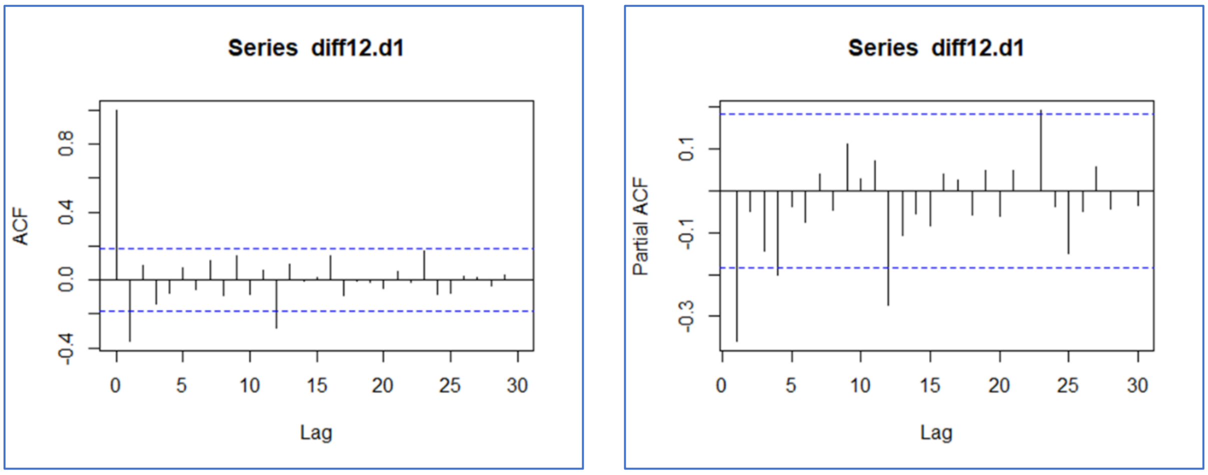


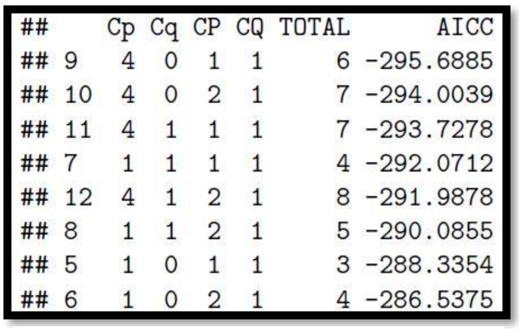
*Figure 6 – S*ample ACF and PACF Plot of the De-seasonalized and De-trended Data

To determine P and Q, I look at the seasonal components in each plot. It is observed that both ACF and PACF tails off in the plots, which indicates that I should use an ARMA for the seasonal components. The last ACF value which is outside the 95% confidence interval of zero locates at lag 12, so I conclude that Q should equal to 1. There are two seasonal lags stick outside the confidence interval in the PACF plot, and the last one is lag 23. Although lag 24 stays within the confidence interval, there is possibility that the PACF value at lag 23 can be influenced by the value at lag 24, so the situation that P equals to 2 should also be considered. However, since the PACF value at lag 23 is just touching the upper bound of confidence interval, I also want to consider the possibility that P equals to 1.

To determine the value of p and q, I look at lag 1 to lag 11 in both plots and consider the following situations:

1) both ACF and PACF values tail off, so the last lags in the range indicates that p equals 4 and q equals 1. Since at lag 4, the PACF is just touching the lower bound of the confidence interval, I also suspect p equals to 1.
2) ACF exponentially decays to 0, and PACF cuts off at lag 4 or 1, so q equals 0 and p equals 1 or 4.
3) PACF exponentially decays to 0, and ACF cuts off at 1, so p equals 0 and q equals 1.

Based on the discussion of p, q, P, Q above, I consider the combinations that p equals to 0 or 1 or 4; q equals to 0 or 1, while p and q cannot be 0 simultaneously; P equals to 1 or 2; and Q equals to 1. I run a for loop in R to print out all the possible combinations of parameters and the AICC value associated with each model fit by difference combinations. The corresponding R codes are in Appendix 6.1. The resulted data frame sorted by AICC in descending order is shown in Figure 7.

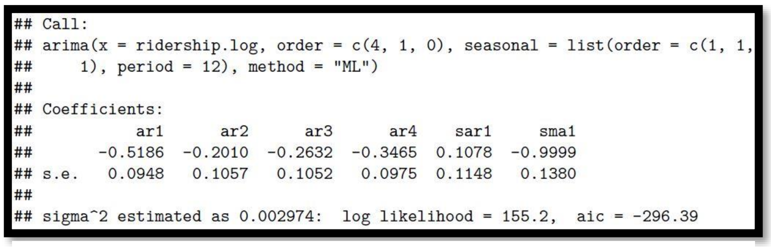


```
##     Cp Cq CP CQ TOTAL      AICC
## 9    4  0  1  1     6 -295.6885
## 10   4  0  2  1     7 -294.0039
## 11   4  1  1  1     7 -293.7278
## 7    1  1  1  1     4 -292.0712
## 12   4  1  2  1     8 -291.9878
## 8    1  1  2  1     5 -290.0855
## 5    1  0  1  1     3 -288.3354
## 6    1  0  2  1     4 -286.5375
```

Figure 7 – Result of the For Loop with d = D = 1

However, after fitting all the models from above, I find the only SMA part in all of those models has an estimated Θ_1 extremely close -1 (e.g. -0.9999). Figure 8 shows an example of the estimates of each parameter of the model ***SARIMA (4,1,0) x (1,1,1)$_{12}$***.

Such coefficient indicates a unit root $\Theta(z) = 0$ in the SMA part by calculating $\boldsymbol{1 - \Theta_1 z = 0}$, and that makes the models above to be non-invertible. Therefore, since I cannot proceed with non-invertible models, my first option is to fix the parameter Q to be 0 to eliminate the SMA part which will be discussed in the next section. The second option is to NOT eliminate the seasonality at the first hand, and this will be discussed in later sections.

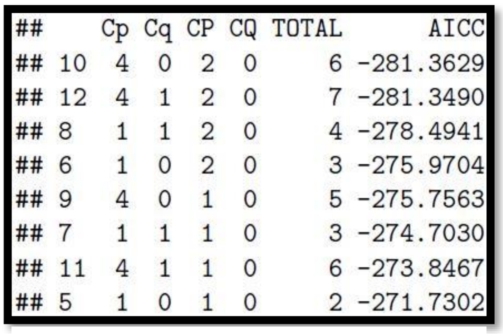


```
## Call:
## arima(x = ridership.log, order = c(4, 1, 0), seasonal = list(order = c(1, 1,
##     1), period = 12), method = "ML")
##
## Coefficients:
##           ar1      ar2      ar3      ar4    sar1     sma1
##       -0.5186  -0.2010  -0.2632  -0.3465  0.1078  -0.9999
## s.e.   0.0948   0.1057   0.1052   0.0975  0.1148   0.1380
##
## sigma^2 estimated as 0.002974:  log likelihood = 155.2,  aic = -296.39
```

Figure 8 – Coefficient Estimates of ***SARIMA (4,1,0) x (1,1,1)$_{12}$***

2.5 Fitting SARIMA Model with Fixed Q = 0

As mentioned above, I need to fix Q to be 0 to avoid an SMA part in our models. Therefore, I modify the input value of Q in for loop and keep everything else the same in the R code and get a new result show in Figure 9.



```
##     Cp Cq CP CQ TOTAL      AICC
## 10   4  0  2  0     6 -281.3629
## 12   4  1  2  0     7 -281.3490
## 8    1  1  2  0     4 -278.4941
## 6    1  0  2  0     3 -275.9704
## 9    4  0  1  0     5 -275.7563
## 7    1  1  1  0     3 -274.7030
## 11   4  1  1  0     6 -273.8467
## 5    1  0  1  0     2 -271.7302
```

Figure 9 –
Result of the For Loop after Fixing Q = 0

I fit the first four models because the first two model have the lowest AICCs among all the models here, and the third and the fourth model have fewer parameters with relatively low AICCs. I then use the arima() function in R to fit the four models and name them by "mm##" where"##" is the number in the first column in figure 9.

When fitting the model "mm10", I find that the estimated ar2 term is not significant, so I use the fixed() command to force ar2 to be 0 and refits the model. However, after forcing ar2 to be 0, the modified model has the estimate of ar3 insignificant again. Therefore, I use the fixed() command again to force both ar2 and ar3 terms to be 0. Eventually, I get the model named "mm10_b" which has all terms that are significant, and its AICC is lower than the AICC of the original model "mm10". I take similar actions to the model "mm8" and finally get "mm8_a" with all significant terms and lower AICC than the AICC of the original model "mm8". Please refer to Appendix 6.4 for R codes used for model fitting and for the output. Finally, I get four models whose parameters estimates are all significant. Their corresponding mathematical formulas are listed below, standard errors are marked in parentheses:

1) "mm10_b": *SARIMA (4,1,0) x (2,1,0)$_{12}$*
   $(1 + 0.4532_{(0.0872)}B + 0.2202_{(0.0909)}B^4)(1 + 0.5075_{(0.1008)}B^{12} + 0.3075_{(0.1044)}B^{24})\ \nabla\ \nabla_{12} \log(X_t)$
   $= Z_t$, *where estimated* $\sigma^2 = 0.004319$

2) "mm12": *SARIMA (4,1,1) x (2,1,0)$_{12}$*
   $(1 + 1.2007_{(0.1665)}B + 0.5102_{(0.1616)}B^2 + 0.3148_{(0.1490)}B^3 + 0.3125_{(0.1066)}B^4)(1 + 0.5644_{(0.1072)}B^{12} + 0.3365_{(0.1062)}B^{24})\ \nabla\ \nabla_{12} \log(X_t) = (1 - 0.7778_{(0.1707)}B)\ Z_t$, *where estimated* $\sigma^2 = 0.003958$

3) "mm8_a": *SARIMA (1,1,1) x (2,1,0)$_{12}$*
   $(1 + 0.5208_{(0.1020)}B^{12} + 0.2722_{(0.1051)}B^{24})\ \nabla\ \nabla_{12} \log(X_t) = (1 + 0.5533_{(0.0979)}B)\ Z_t$,
   where estimated $\sigma^2 = 0.004395$

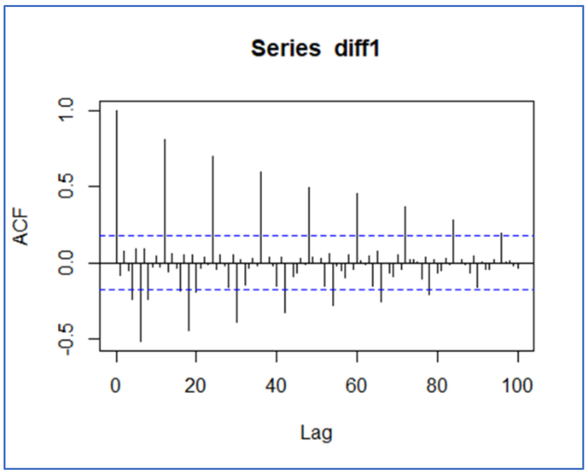
4) "mm6": *SARIMA (1,1,0) x (2,1,0)₁₂*

$(1 + 0.4191_{(0.0883)}B)\ (1 + 0.4945_{(0.1020)}B^{12} + 0.2756_{(0.1059)}B^{24})\ \nabla\ \nabla_{12} \log(X_t) = Z_t$,
where estimated $\sigma^2 = 0.004569$

A possible explanation of why Q = 0 works even though the ACF plots suggests Q = 1 is that there are only 126 observations in the data, so as I deseasonalize our data, there are even fewer observations that are taken into account for the seasonal lags. Therefore, the very small sample size will influence the outcome which assumed by large and normal data.

2.6 Alternative Model Fitting with AR

As mentioned in previous sections, since I would always get unit roots in the SMA part of with Q = 1 and D = 1, I consider the possibility that I overdifferenced the season lags in our data at the first hand. Therefore, I start with the log transformed data again, and skip the step which eliminates the seasonality of the data. That means I only difference the log transformed data at lag 1 to eliminate the trend. The variance after differencing at lag 1 (0.0830) is smaller than the variance of the log transformed data (0.0350), so I can keep the result of this differencing and derive the ACF and PACF plots (figure 10).



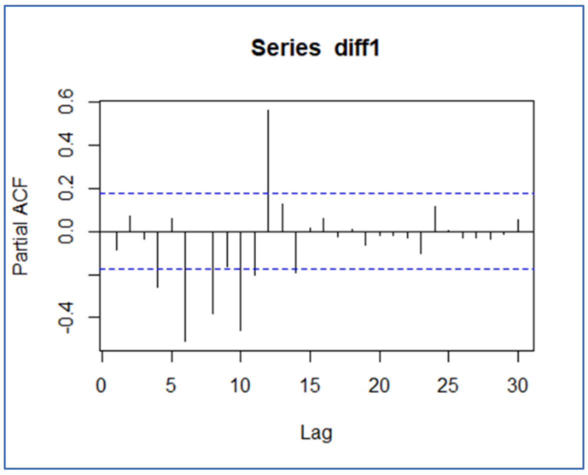
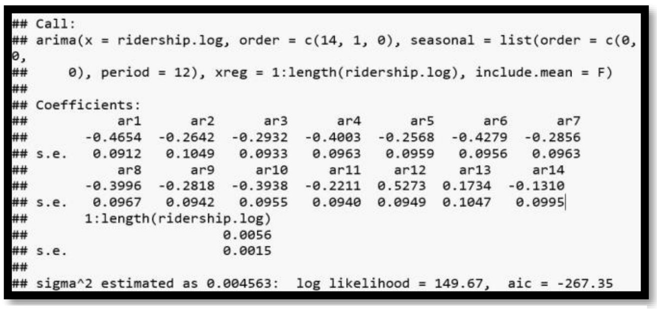


Figure 10 – Sample ACF and PACF Plots of the De-trend Only Data

By looking at the ACF plot here, I find that values exponentially decay to zero, while the PACF cuts off at lag 14. I suspect an AR model based on the behavior described above, so I perform a preliminary estimation of the AR model using Yule-Walker estimates. The R code can be found in Appendix 7.3. Without specifying the order of the AR model, the result of the AR model suggests an order of 14, which corresponds to what is observed from the PACF plot above. 12 out of the 14 estimates turned out to be significant by using the arima() function (Figure 11). However, it is undesirable to fit a model with so many parameters, and it would be infeasible to perform the Box-Pierce and Ljung-Box tests in which I would get a negative degree of freedom if I chose a model with 12 parameters while the lag h is defined as 11 as the square root of our sample sizes. Thus, I cannot fit this data using an AR model.



```
## Call:
## arima(x = ridership.log, order = c(14, 1, 0), seasonal = list(order = c(0,
0,
##     0), period = 12), xreg = 1:length(ridership.log), include.mean = F)
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5      ar6      ar7
##       -0.4654  -0.2642  -0.2932  -0.4003  -0.2568  -0.4279  -0.2856
## s.e.   0.0912   0.1049   0.0933   0.0963   0.0959   0.0956   0.0963
##           ar8      ar9     ar10     ar11    ar12    ar13     ar14
##       -0.3996  -0.2818  -0.3938  -0.2211  0.5273  0.1734  -0.1310
## s.e.   0.0967   0.0942   0.0955   0.0940  0.0949  0.1047   0.0995
##       1:length(ridership.log)
##                        0.0056
## s.e.                   0.0015
##
## sigma^2 estimated as 0.004563:  log likelihood = 149.67,  aic = -267.35
```

Figure 11 – Coefficients of the AR Model Fitted

2.7 Alternative Model Fitting with SARIMA (p, 1, q) x (P, 0, Q)$_{12}$

As I failed to use the AR model, I consider the SARIMA model again, but this time, D equals to 0 because I did not difference at the seasonal lag of 12. From the ACF and PACF plots in figure 10, I can determine that since the seasonal lags in ACF tails off to zero, and the PACF cuts off at the first seasonal lag, Q should be 0 and P should be 1. There should also be no MA terms in this model because although the ACF spikes around lag 6, the lag is influenced by the seasonal lag since I am using a data that is NOT deseasonalized. Therefore, q equals to 0. Lastly, I need to determine the value of p. By looking at the PACF, lag 4, 6, 8, 10 and 11 are outside the confidence interval. However, I will only consider the choice of p equals to 4 and 6 here, because first, I want to keep the model simple with fewer coefficients, and second, values of lags larger than 6 here could be affected by the seasonal components. Thus, I will fit two models under these circumstances, the first one is SARIMA (4, 1, 0) x (1, 0, 0)$_{12}$ (I will refer it as model "mmm4"), and the second on is SARIMA (6, 1, 0) x (1, 0, 0)$_{12}$ (I will refer it as model "mmm6").

After using the arima() function to the model "mmm4", I found that the ar2 term and the constant term in the model is not significant, so I fixed the two terms to be 0 and fit the model again and derived the model "mmm4_a" with lower AICC and all the parameters significant. For the model "mmm6", I performed similar processes to eliminate the insignificant terms ar2, ar5, ar6 and the constant term and get the final model "mmm6_a" with lower AICC and all parameters significant. I find that at this point, the model "mmm6_" became the same model as the model "mmm4_a" with the same order, the same estimated variance, and the same estimates. The R codes and results can be found in Appendix 7.4. Thus, I fit the model "mmm4_a" and get the following result:

5) "mmm4_a": SARIMA (4, 1, 0) x (1, 0, 0)$_{12}$

$(1 + 0.3669_{(0.0858)}B + 0.1867_{(0.0912)}B^3 + 0.2524_{(0.0961)}B^4)(1 + 0.9316_{(0.0238)}B^{12})\ \nabla\log(X_t)$
$= Z_t$, *where estimated* $\sigma^2 = 0.005161$

2.8 Diagnostic Checking

2.8.1 Unit roots

I first want to check if there is any unit roots in any term of all the potential models, so I use a plot.roots() function to draw all the roots and see if any of the roots (represented in red points) falls into the unit circle. I find that none of the 5 potential models generate any of the unit root in any part. Here are two examples of the plot.roots() result for the model "mm6" and "mmm4_a"(Figure 12). The root in the AR part of "mm6"is -2.386 whose absolute value is larger than 2, so the (red) point is outside the scope of the plot. The root for the SAR part is 1.073 which is larger than 1, although looks like 1 on the plot. Such result is still better than all the SARIMA model with Q = 1 mentioned in previous sections. For all plots, please refer to the Appendix 6.4.1 and 7.5. All of the five models are causal and invertible, so I can proceed with those models.

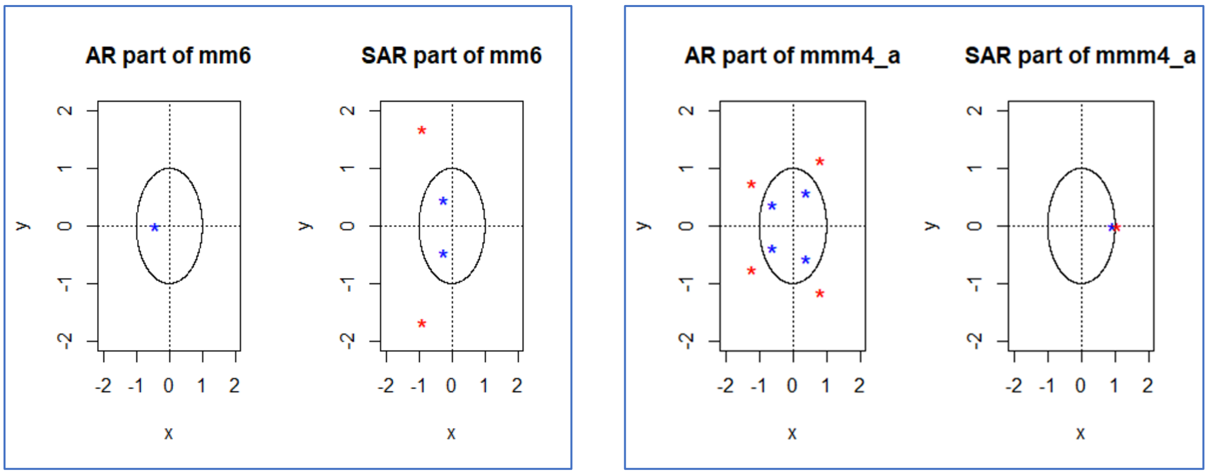


Figure 12 – Two Examples of plot.roots Result for the model “mm6” and “mmm4_a”

2.8.2 Residual Plots, ACF & PACF Plots of the Residuals

After fitting the model, I can derive the residual plots, ACF, and PACF of the residuals for each model. For the R codes and results please refer to the Appendix 6.4.2 – 6.4.5 and 7.6. It is observed from the residual plots that all of them resemble white noise. Also, the ACF plots indicate the value of ACF decays to zero fast, and for the first 4 models, there is almost no lags sticks out the confidence interval. For the ACF of the model “mmm4_a”, as shown in Figure 13, the seasonal lags stick out the confidence interval, this is because I did not de seasonalize the data used for “mmm4_a”.

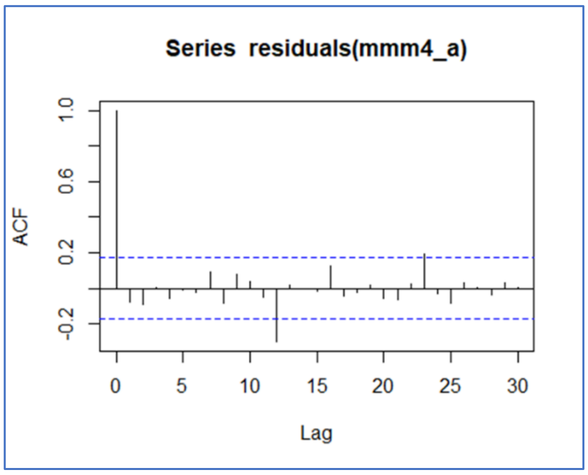


Figure 13 – Residual ACF of the Model “mmm4_a”

Besides the residual plots, ACF and PACF plots, I also use the ar() function to detect if there is any AR order that is selected to fit the residuals. However, none of the models above derives residuals that has AR orders. That show the residuals are white noise as well.

2.8.3 Portmanteau Test

I perform Box-Pierce Test and the Ljung-Box Test in this step to see if the residuals fitted by a model are correlated with each other. The hull hypothesis is that the residuals are not correlated with each other. The degree of free used in either of these tests equals to $\sqrt{n}$ – fitdf, where the fitdf = p + q + P + Q in R, and the $\sqrt{n}$ in our case equals to $\sqrt{126} \approx 11$. The p-values of all the models for these two tests are listed in table 1.

Model	df	p-value of Box-Pierce Test	p-value of the Ljung-Box Test
mm10_b	7	0.2674	0.2323
mm12	4	0.5034	0.4693
mm8_a	8	0.2394	0.2
mm6	8	0.1763	0.1438
mmm4_a	8	0.6977	0.657

Table 1 – Results of Box-Pierce Test and Ljung-Box Test

It can be observed that all the potential models get p-value larger than 0.05 in both tests, which means we fail to reject the null hypothesis and conclude that the residuals are not correlated with each other for each model.

I then perform the McLeod-Li test to see if the squares of the residuals are correlated with each other. The degree of freedom for this test will be 11. The null hypothesis is that the squares of the residuals are not correlated with each other. The results are listed in table 2.

Model	p-value of the McLeod-Li Test
mm10_b	0.852
mm12	0.9108
mm8_a	0.8561
mm6	0.9409
mmm4_a	0.9361

Table 2 – Result of McLeod Li Test

From Table 2, all the p-values are larger than 0.05, so we fail to reject the null hypothesis, and we can conclude that the square of the residuals are not correlated with each other. All the models pass the test.

2.8.4 Other Tests of Residuals

Then I test the normality of the residuals by drawing the histogram (figure 14), the QQ-plot (figure 15) and using the Shapiro-Wilk Test (table 3).

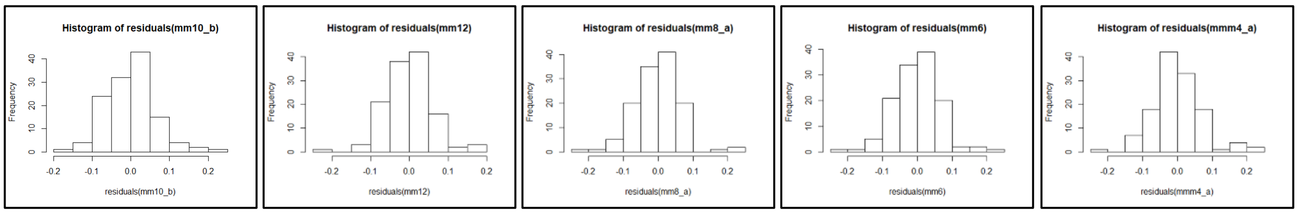


Figure 14 – Histograms of Residuals all Models

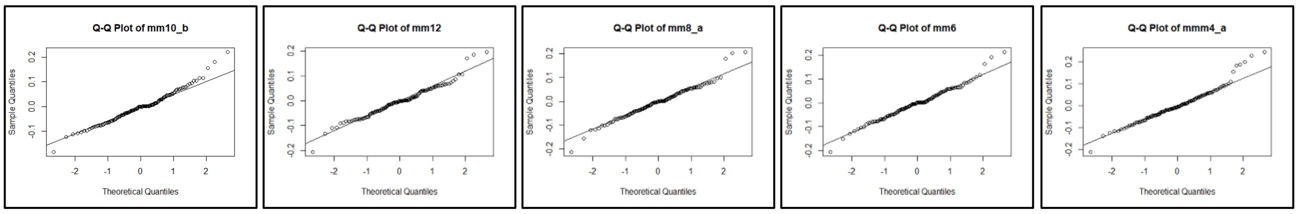


Figure 15 – Q-Q Plots of Residuals of All Models

Model	p-value of the Shapiro-Wilk Test	Conclusion
mm10_b	0.054	PASSED
mm12	0.0057	NOT PASSED
mm8_a	0.0085	NOT PASSED
mm6	0.08	PASSED
mmm4_a	0.0011	NOT PASSED

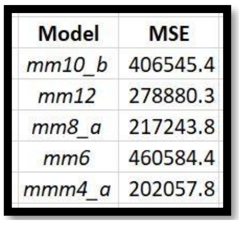
Table 3 – Results from the Shapiro-Wilk Test

From the histograms above, it is observed that the residual distribution of all the models look normal. In the normal Q-Q plots above, the model "mmm4_a" fits the normal line the best, while it probably indicates heavy tail. The Q-Q plots of the residuals of other models have fewer outliners, but the points do not fit the normal line as well as the "mmm4_a". In Table 3, only the model "mm10_b" and "mm6" passed the Sharpiro-Wilk Test for normality.

At this point, as I finished the diagnostic checking, two models pass every test, and three models pass all the test except the Shapiro-Wilk Test for residuals normality. However, as I explained above, all the histograms and Q-Q plots indicate the three models are relative normal. Therefore, I will determine the final model for prediction by comparing the test error of each model using the 10 left out observations at the beginning of this project.

2.9 Validation Using the Test Set

I use the five models to predict the monthly average of weekday bus ridership in Iowa City in March 1982 to December 1982 (10 months) and compute the mean square error (MSE) of each prediction using the test set. Table 4 below shows the MSE of the prediction made by each model. The forecast points in red with 95% confidence interval of the model mmm4_a is in figure 16. For all the results and plot, please refer to the Appendix 8.1 to 8.5.



Model	MSE
mm10_b	406545.4
mm12	278880.3
mm8_a	217243.8
mm6	460584.4
mmm4_a	202057.8

Table 4 – MSE of Prediction by Each Model

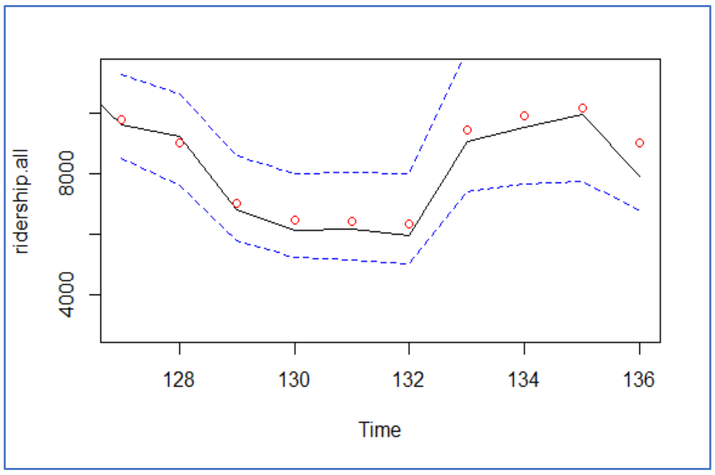


Figure 16 – Validating mmm4_a with the Test Set

2.10 Final Model Selection

Based on the performance of prediction for the test set, it is observed that the model "mmm4_a" generates the smallest error. In addition, the model "mmm4_a" has the least number of parameters among all the potential models as well. Therefore, the final model will be "mmm4_a".

SARIMA (4, 1, 0) x (1, 0, 0)$_{12}$

$$(1 + 0.3669_{(0.0858)}B + 0.1867_{(0.0912)}B^3 + 0.2524_{(0.0961)}B^4)(1 + 0.9316_{(0.0238)}B^{12})\ \nabla \log(X_t) = Z_t,$$

where estimated $\sigma^2 = 0.005161$

There is one thing to be cautious in this final selection: although the model "mmm4_a" makes the best prediction among all the potential models, the fact that it does not pass the Shapiro-Wilk Test of residual normality might cause uncertainty for long term prediction in the future. If anyone was using this model to predict the bus ridership in a long term, he/she should be cautious about this problem.

2.11 Predicting the "Future"

Finally, I use the model "mmm4_a" to predict the weekday bus ridership monthly average from January 1983 to May 1983. The predictions points and interval are shown in figure 17. The specific number of ridership predict for Jan 1983 to May 1983 is 11597, 12942, 11210, 10372, and 8228.

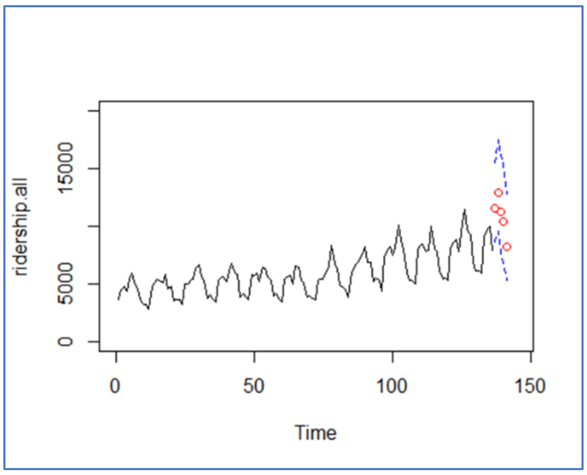


Figure 17 – Prediction from Jan 1983 to May 1983

The number for monthly average ridership in the same months in 1982 are 10179, 11460, 9641, 9243, 6824, so the differences between the same month in 1982 and 1983 are 1418, 1482, 1569, 1129, 1404.

3. Conclusion

The final SARIMA model fitted is SARIMA (4, 1, 0) x (1, 0, 0)$_{12}$. However, one should be cautious when using this model for uncertainty in long-term prediction. By using the SARIMA model fitted from all the processes above, I can conclude that in each month from January to May in 1983, the monthly average weekday ridership increases more than 1000 people. Therefore, perhaps the local bus company should adjust their original bus schedules and operation to provide more suitable service for passengers.

4. Acknowledgement

I am very thankful for Prof. Feldman, my T.A. Yuanbo Wang, and my friend Catherine Miao at UCSB for all the help they gave me when I was working for this project.

5. References

1. Rob Hyndman and Yangzhuoran Yang (2018). tsdl: Time Series Data Library. v0.1.0. https://pkg.yangzhuoranyang./tsdl/

2. Cryer, "Weekday bus ridership, Iowa city, Iowa (monthly averages) September, 1971 through December, 1982, n=148", 1986.

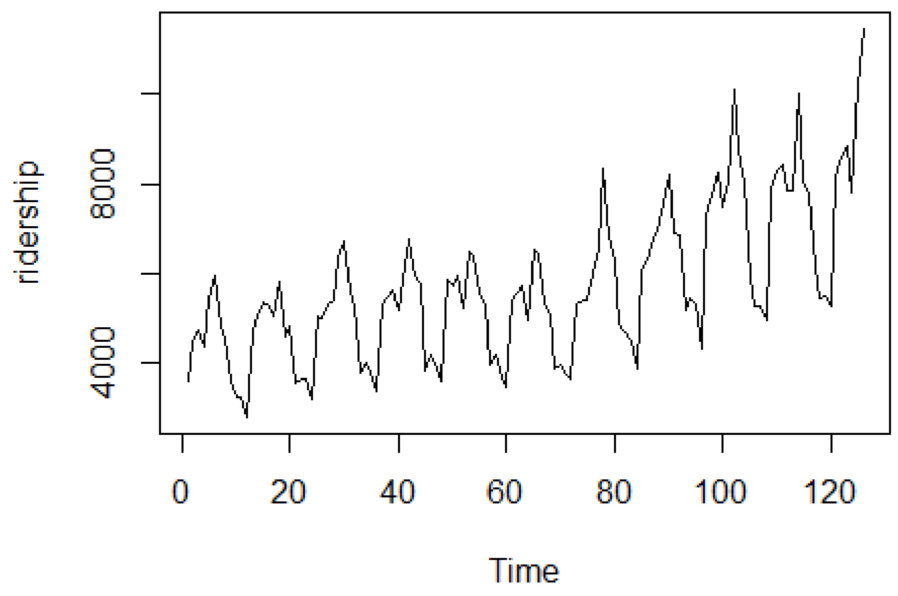
Appendix: R Codes and Results

Jiamin Tan

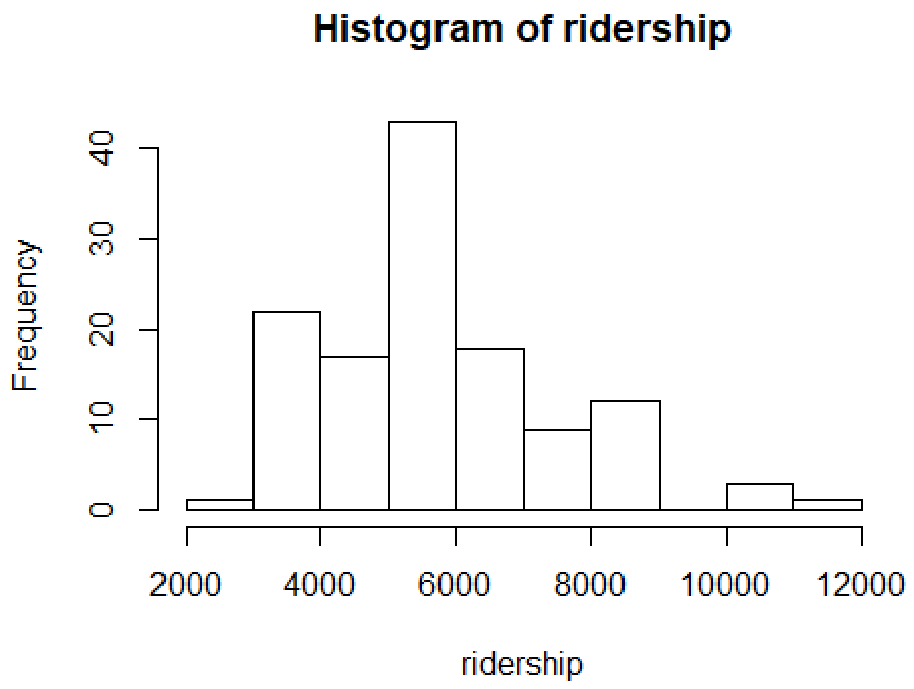
June 7, 2019

1. Importing Data

```
# install.packages("devtools")
devtools::install_github("FinYang/tsdl")

## Skipping install of 'tsdl' from a github remote, the SHA1 (d1bfa3b4) ha
s not changed since last install.
##   Use `force = TRUE` to force installation

library(tsdl)
tsdl_trans <- subset(tsdl, "Transport and tourism")

# get the description of the data used
attributes(tsdl_trans[[7]])

## $tsp
## [1] 1971.67 1982.92   12.00
##
## $class
## [1] "ts"
##
## $source
## [1] "Cryer (1986)"
##
## $description
## [1] "Weekday bus ridership, Iowa city, Iowa (monthly averages) Septembe
r, 1971 through December, 1982, n=148"
##
## $subject
## [1] "Transport and tourism"

# the sample size recored in the description is incorrect
# the true sample size is n = 136

# leave out the last 10 observation for comapring the forecasting result
ridership.all <- ts(tsdl_trans[[7]])
ridership <- ts(tsdl_trans[[7]])[1:126]
# the left out test set
ridership.test <- ts(tsdl_trans[[7]])[127:136]

# plot the time series plot of the original data
ts.plot(ridership)
```



```
# get the mean and the variance of the origianl data
mean(ridership)

## [1] 5743.937

var(ridership)

## [1] 2851465

# get the histgram of the original data
hist(ridership)
```

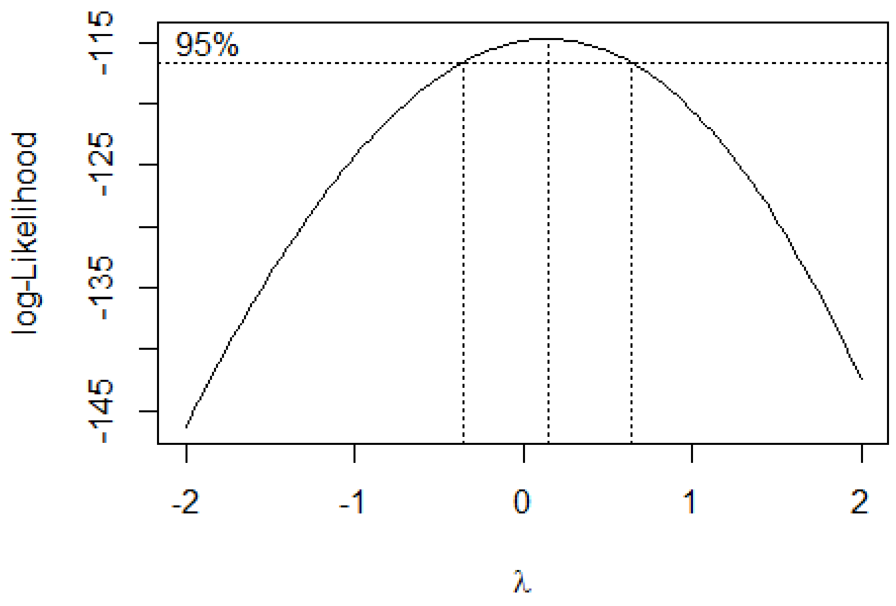


```
# the histgram looks like log normal distribution
```

2. Transformation

2.1 Box-Cox Transformation

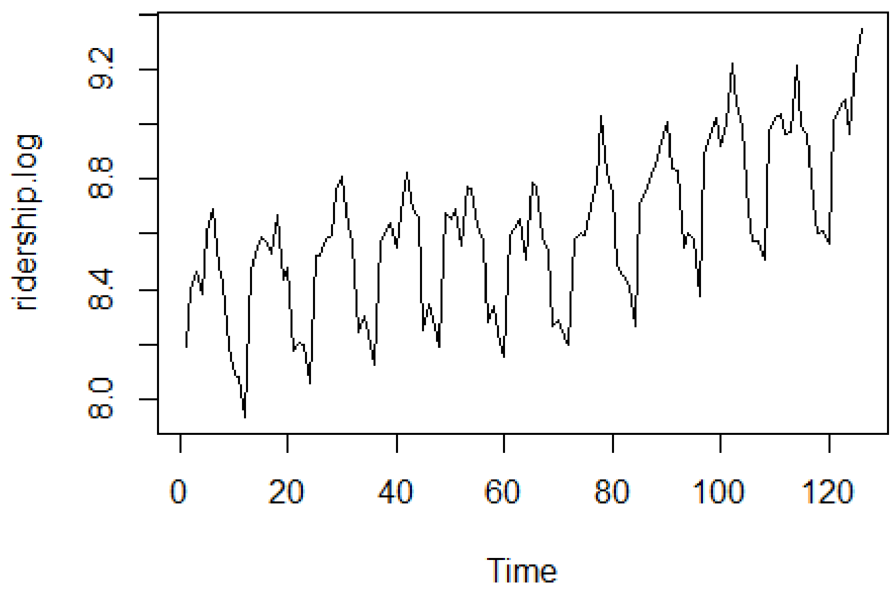
```
# box-cox transformation
library(MASS)
bcTrans <- boxcox(ridership ~ as.numeric(1:length(ridership)))
```



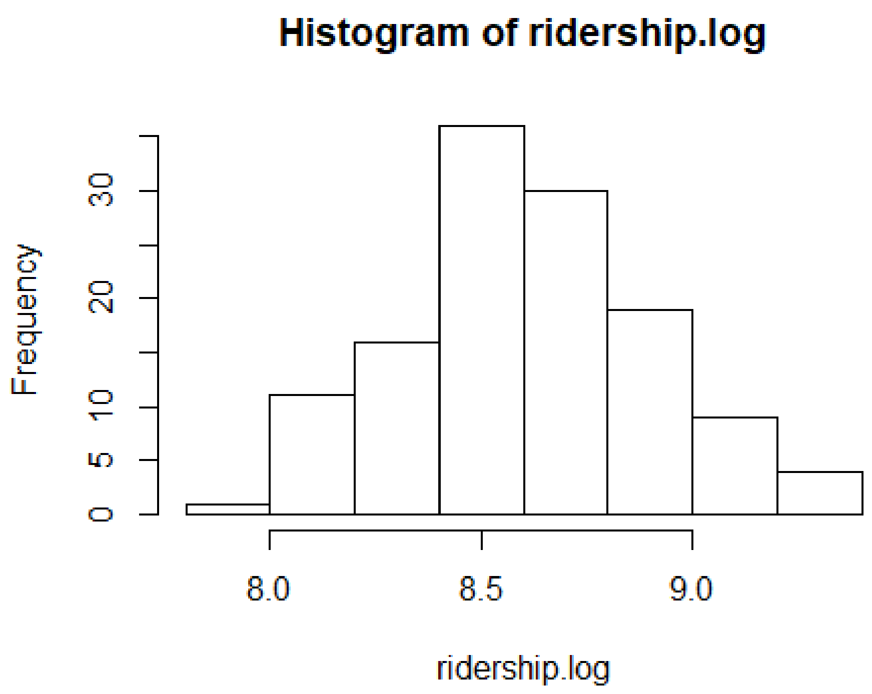
```
# since the 95% CI of lambda contains 0, we choose lambda = 0
# so we can directly use the log() transformation
# corresponds to the log normal distribution we observed at the beginning
```

2.2 Log Transformation

```
ridership.log <- log(ridership)

# plot the transformed data
ts.plot(ridership.log)
```



```
# we can see the variance is stablized compare to the original data

# histgram of the log() transformed data
hist(ridership.log)
```

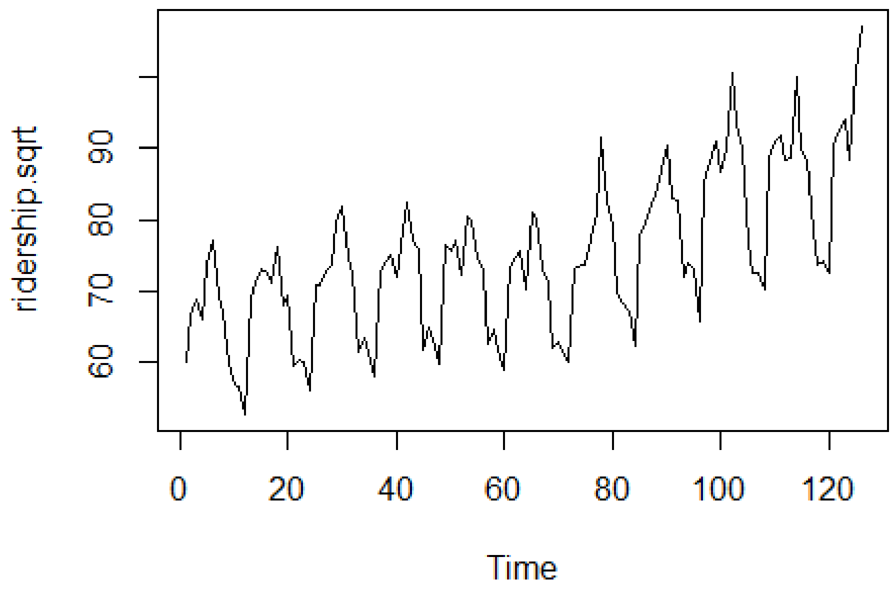


```
# get the mean and variance of the transformed data
mean(ridership.log)
```

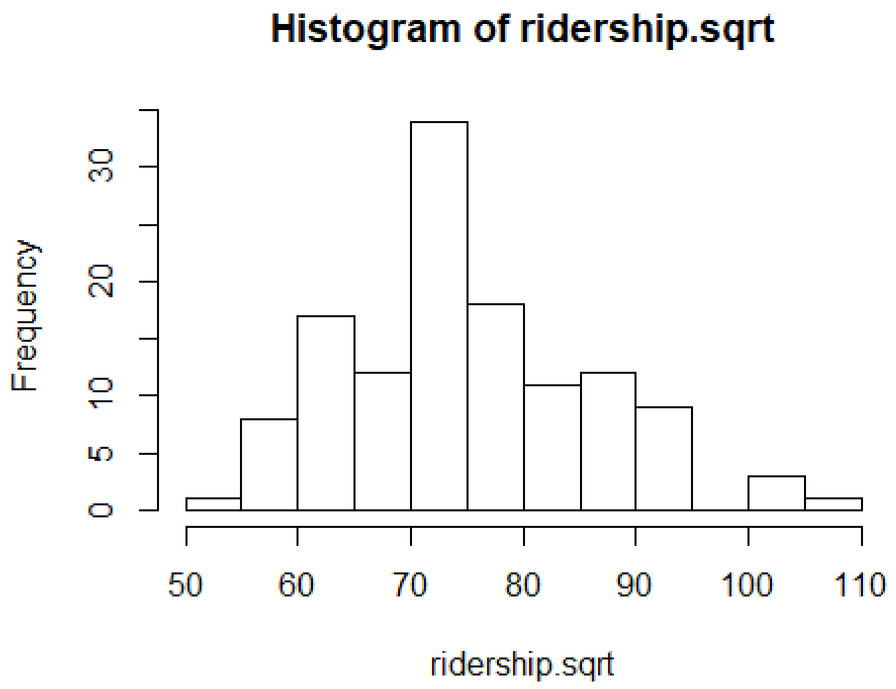
```
## [1] 8.614558

var(ridership.log)

## [1] 0.08306838
```

2.3 Square Root Transformation

```
# sqrt transformation
ridership.sqrt <- sqrt(ridership)

# plot the sqrt transformed data
ts.plot(ridership.sqrt)
```



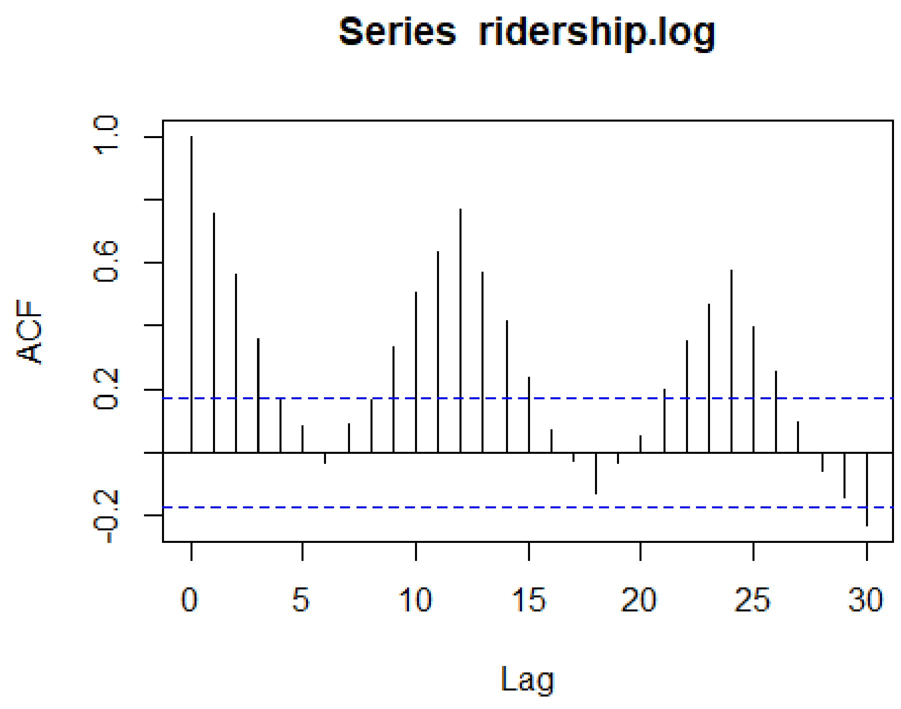
```
# histgram of the sqrt transformed data
hist(ridership.sqrt)
```



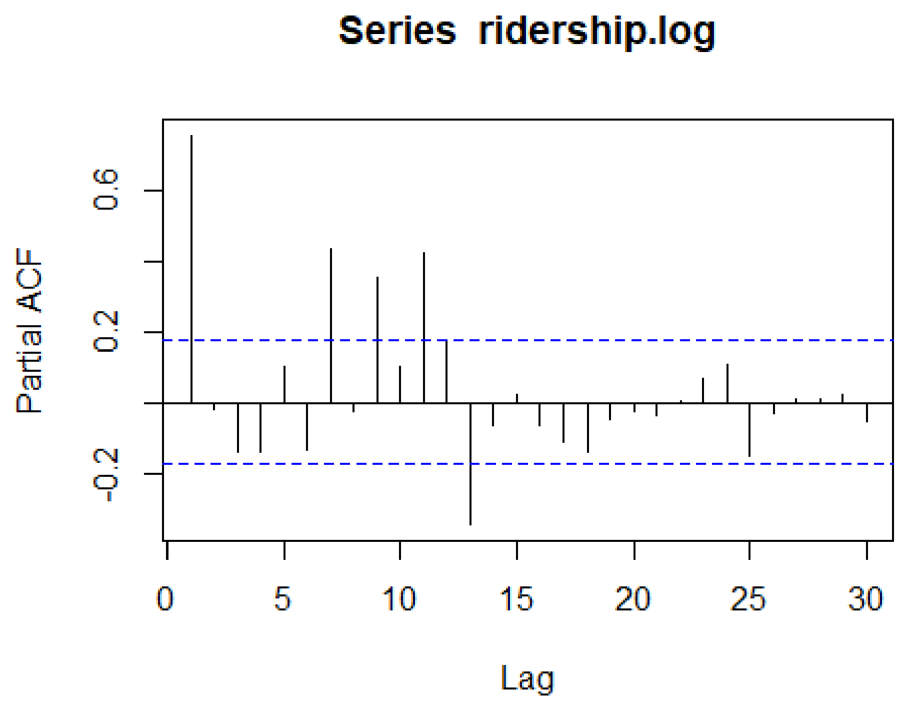
```
# get the mean and variance of the sqrt transformed data
mean(ridership.sqrt)

## [1] 75.00876

var(ridership.sqrt)

## [1] 118.5627

# from the ts plots of the log() and sqrt transformation
# the log() transformed data has more stabled variance
# from the histgram of the log() and sqrt transformation
# the log() transformed data is more normal
# we choose the log() transformation to proceed
```

3. ACF and PACF of the Log Transformed Data

```
# acf, h = 120/4 = 30
acf(ridership.log, lag.max = 30)
```



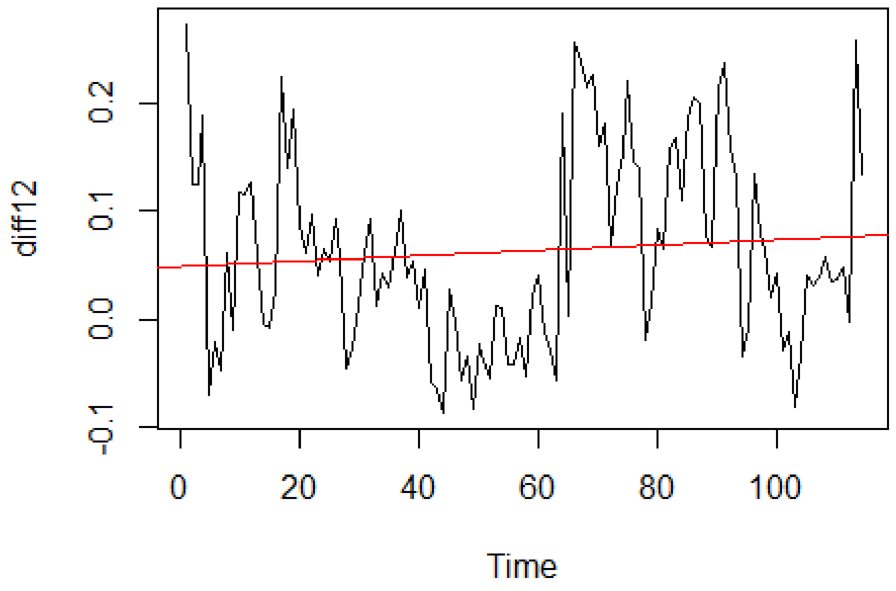
```
# if we look at neighboring peak with the same sign
# we find the seasonality is 12 months

# pacf
pacf(ridership.log, lag.max = 30)
```



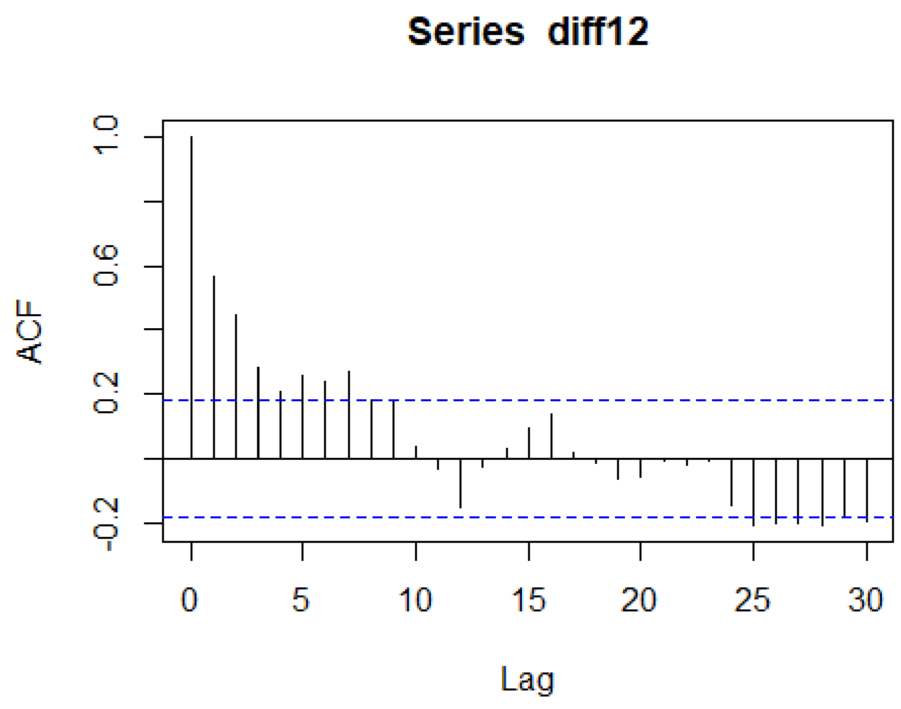
4. Differencing

4.1 Eliminating Seasonality by Differencing at Lag 12

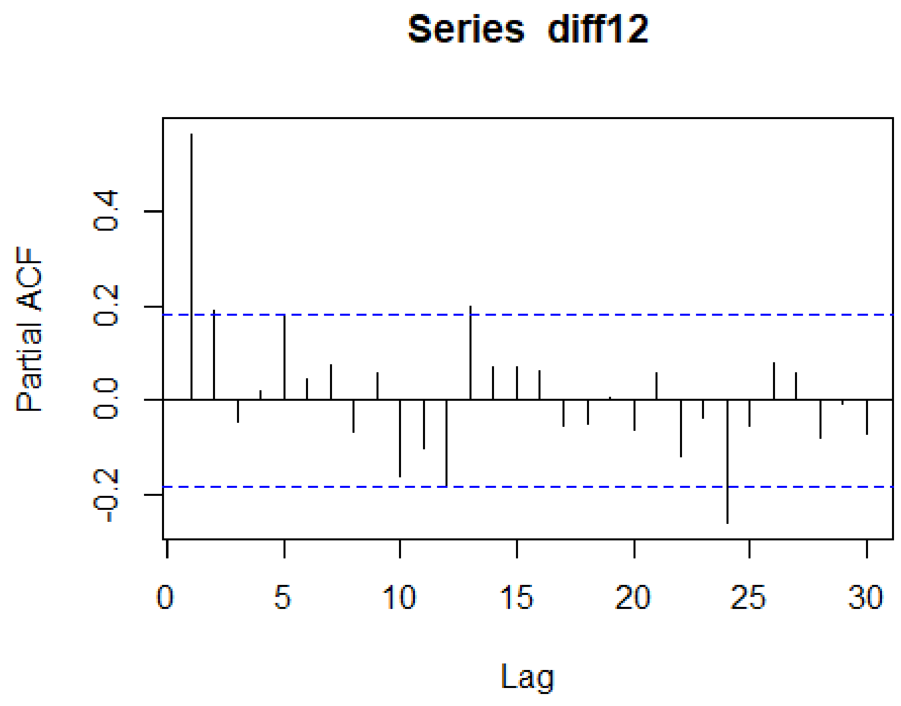
```
# difference at lag 12
diff12 <- diff(ridership.log, lag=12)

# plot the deseasonalized data
ts.plot(diff12)
# it does not look stationary

# get the variance of the deseasonalized data
var(diff12)

## [1] 0.008113734

# the variance is smaller than the transformed data, keep

# see if there is a trend in the deseasonalized data
abline(lm(diff12~as.numeric(1:length(diff12))), col = "red")
```



```
# there seems be a small trend, so try to difference at lag 1

# get the acf and pacf of the deseasonalized data
acf(diff12, lag.max = 30)
```



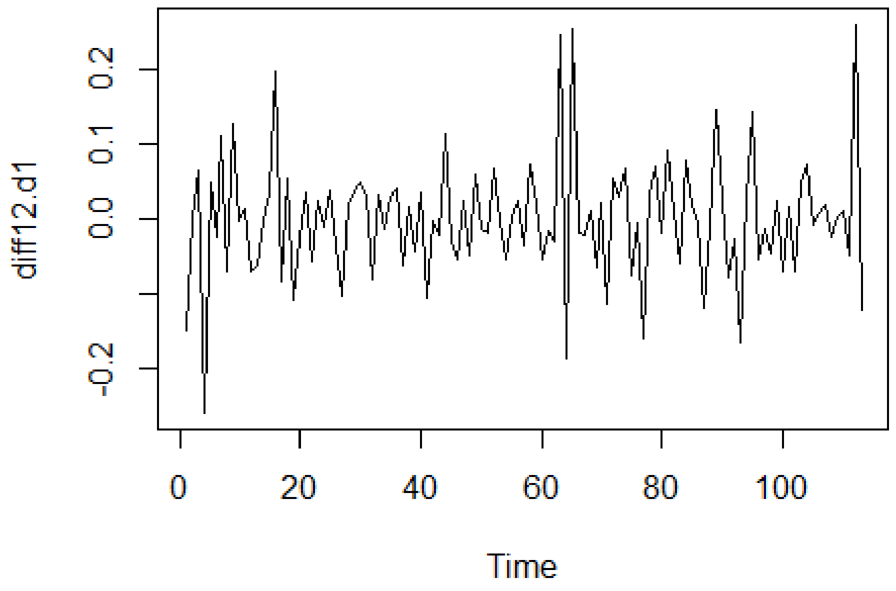
```
# the acf decays slowly, so the differenced data is not stationary
pacf(diff12, lag.max = 30)
```



4.2 Eliminating Trend by Differencing at Lag 1

```
# try to difference at lag 1 of the deseasonalized data
diff12.d1 <- diff(diff12, 1)
```

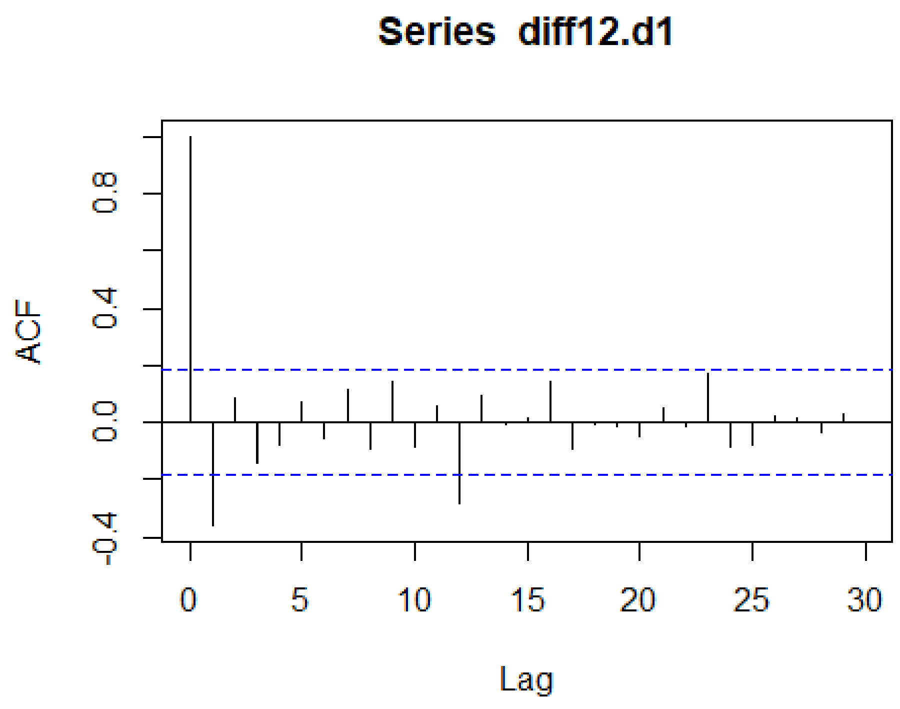
```
# get the variance
var(diff12.d1)

## [1] 0.006682884

# the variance get smaller than just deseasonalizing at lag 12, keep

# plot the deseasonalized and de-trend data
ts.plot(diff12.d1)
```



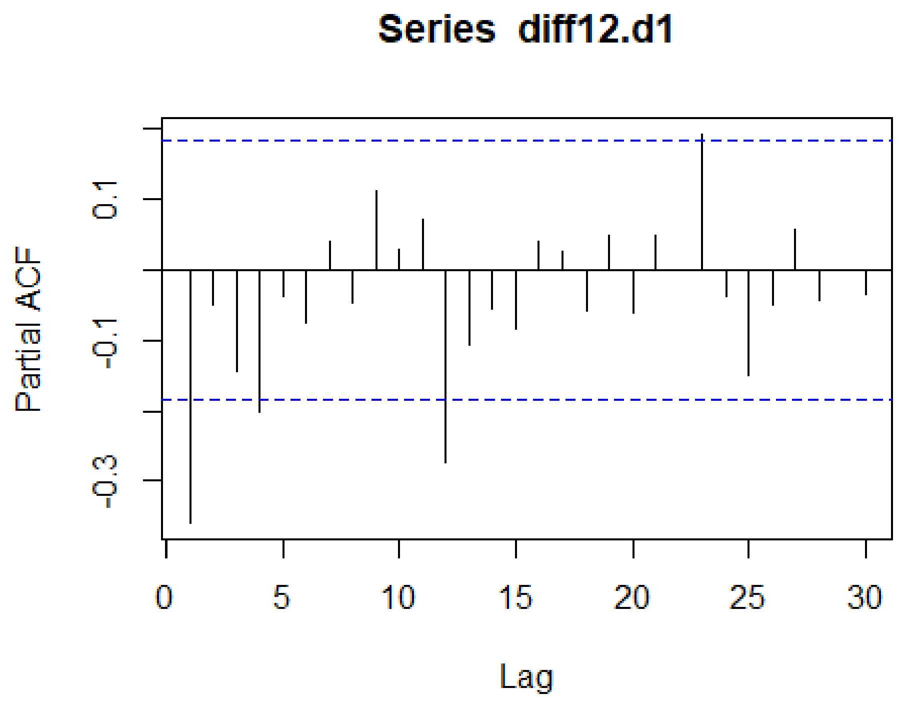
```
# the data looks more stationary now

# if we difference at lag 1 again
diff12.d1.d1 <- diff(diff12.d1, 1)
# and get the variance
var(diff12.d1.d1)

## [1] 0.01799828

# the variance increases. we cannot use this
```

5. ACF and PACF of the Log Transformed Data Differenced at Lag 12 & 1

```
# get acf of the transformed data differenced at lag 12 and 1
acf(diff12.d1, lag.max = 30)
```



```
# get pacf of the transformed data differenced at lag 12 and 1
pacf(diff12.d1, lag.max = 30)
```



6. SARIMA Model Selection Based on the ACF and PACF

6.1 Possible SARIMA Models and Their AICC Using A For Loop

```
# use the package qpcR to calculate the AICC value
library(qpcR)
```

```
## Warning: package 'qpcR' was built under R version 3.5.3

## Loading required package: minpack.lm

## Warning: package 'minpack.lm' was built under R version 3.5.3

## Loading required package: rgl

## Warning: package 'rgl' was built under R version 3.5.3

## Loading required package: robustbase

## Warning: package 'robustbase' was built under R version 3.5.3

## Loading required package: Matrix
```

```
# set up some empty columns for use later, "C" stands for column
Cp <- c()
Cq <- c()
CP <- c()
CQ <- c()
AICC <- c()

# the for loop
# although p and q start at 0, base on the ACF and PACF
# they cannot equal to 0 at the same time
for(p in c(0, 1, 4)){
  for(q in 0:1){
    for(P in 1:2){
      for (Q in 1){
        # use the try() function to skip possible errors
        try(m <- arima(ridership.log, order=c(p,1,q),
                       seasonal=list(order=c(P,1,Q), period=12),
                       method = "ML"))
        aicc <- AICc(m)

        # print out each result to see where error happens
        print(p)
        print(q)
        print(P)
        print(Q)
        print(aicc)

        Cp <- c(Cp, p)
        Cq <- c(Cq, q)
        CP <- c(CP, P)
        CQ <- c(CQ, Q)
        AICC <- c(AICC, aicc)
        }
      }
    }
}
```

```
## [1] 0
## [1] 0
## [1] 1
## [1] 1
## [1] -271.0944
## [1] 0
## [1] 0
## [1] 2
## [1] 1
## [1] -269.3914
## [1] 0
## [1] 1
## [1] 1
## [1] 1
## [1] -293.0095
## [1] 0
## [1] 1
## [1] 2
## [1] 1
## [1] -291.0321
## [1] 1
## [1] 0
## [1] 1
## [1] 1
## [1] -288.3354
## [1] 1
## [1] 0
## [1] 2
## [1] 1
## [1] -286.5375
## [1] 1
## [1] 1
## [1] 1
## [1] 1
## [1] -292.0712
## [1] 1
## [1] 1
## [1] 2
## [1] 1
## [1] -292.0712
## [1] 4
## [1] 0
## [1] 1
## [1] 1
## [1] -295.6885
## [1] 4
## [1] 0
## [1] 2
## [1] 1
## [1] -294.0039
## [1] 4
## [1] 1
## [1] 1
## [1] 1
## [1] -293.7278
## [1] 4
## [1] 1
## [1] 2
## [1] 1
## [1] -291.9878

# also get the total number of parameters
TOTAL <- Cp + Cq + CP + CQ
df1 <- data.frame(Cp, Cq, CP, CQ, TOTAL, AICC)

# fit p = 1, q = 1, P = 2, Q = 1 and calculate the AICC due to the error
M1121 <- arima(ridership.log, order=c(1,1,1),
            seasonal=list(order=c(2,1,1), period=12))

# get AICC for the model p = 1, q = 1, P = 2, Q = 1
df1[8,6] <- AICc(M1121)

# remove the situations that p = q = 0
df1 <- df1[5:12, ]

# sort the potential models by AICC
df1[order(df1$AICC, decreasing = F), ]

##    Cp Cq CP CQ TOTAL      AICC
## 9   4  0  1  1     6 -295.6885
## 10  4  0  2  1     7 -294.0039
## 11  4  1  1  1     7 -293.7278
## 7   1  1  1  1     4 -292.0712
## 12  4  1  2  1     8 -291.9878
## 8   1  1  2  1     5 -290.0855
## 5   1  0  1  1     3 -288.3354
## 6   1  0  2  1     4 -286.5375
```

6.2 Potential SARIMA Models Fitting

```
# fit the four models with the lowest AICC value above (out of eight)
# model named by "m" + the first column number of the dataframe above

########## m9 ##########
m9 <- arima(ridership.log, order=c(4,1,0),
            seasonal=list(order=c(1,1,1), period=12),
            method = "ML")
m9

##
## Call:
## arima(x = ridership.log, order = c(4, 1, 0), seasonal = list(order = c(
1, 1,
##     1), period = 12), method = "ML")
##
## Coefficients:
##           ar1      ar2      ar3      ar4     sar1     sma1
##       -0.5186  -0.2010  -0.2632  -0.3465  0.1078  -0.9999
## s.e.   0.0948   0.1057   0.1052   0.0975  0.1148   0.1380
##
## sigma^2 estimated as 0.002974:  log likelihood = 155.2,  aic = -296.39
```

```
########## m10 ##########
m10 <- arima(ridership.log, order=c(4,1,0),
            seasonal=list(order=c(2,1,1), period=12),
            method = "ML")
m10

##
## Call:
## arima(x = ridership.log, order = c(4, 1, 0), seasonal = list(order = c(
2, 1,
##     1), period = 12), method = "ML")
##
## Coefficients:
##           ar1      ar2      ar3      ar4     sar1     sar2     sma1
##       -0.5163  -0.1942  -0.2591  -0.3532  0.1000  -0.0900  -0.9998
## s.e.   0.0951   0.1063   0.1056   0.0977  0.1141   0.1187   0.1608
##
## sigma^2 estimated as 0.002902:  log likelihood = 155.48,  aic = -294.95

########## m11 ##########
m11 <- arima(ridership.log, order=c(4,1,1),
            seasonal=list(order=c(1,1,1), period=12),
            method = "ML")
m11

##
## Call:
## arima(x = ridership.log, order = c(4, 1, 1), seasonal = list(order = c(
1, 1,
##     1), period = 12), method = "ML")
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ma1    sar1     sma1
##       -0.3579  -0.1278  -0.2383  -0.3280  -0.1821  0.1133  -0.9999
## s.e.   0.2974   0.1644   0.1129   0.1087   0.3159  0.1153   0.1342
##
## sigma^2 estimated as 0.002967:  log likelihood = 155.34,  aic = -294.68

########## m7 ##########
m7 <- arima(ridership.log, order=c(1,1,1),
            seasonal=list(order=c(1,1,1), period=12),
            method = "ML")
m7

##
## Call:
## arima(x = ridership.log, order = c(1, 1, 1), seasonal = list(order = c(
1, 1,
##     1), period = 12), method = "ML")
##
## Coefficients:
##          ar1      ma1    sar1     sma1
##       0.1949  -0.7195  0.1598  -0.9999
## s.e.  0.1681   0.1236  0.1141   0.1427
```

```
##
## sigma^2 estimated as 0.003226:  log likelihood = 151.2,  aic = -292.4
```

```
# we can easily find that the sma part of all the models listed above
# has an estimate very close to -1, which indicates unit roots in sma
# none of these models will work, since they are non-invertible
```

6.3 Possible SARIMA Models with Q = 0 and Their AICC Using A For Loop

```
# set up some empty columns for use later, "C" stands for column
Cp <- c()
Cq <- c()
CP <- c()
CQ <- c()
AICC <- c()

# the for loop
# although p and q start at 0, base on the ACF and PACF
# they cannot equal to 0 at the same time
# Q = 0 at this time
for(p in c(0, 1, 4)){
  for(q in 0:1){
    for(P in 1:2){
      for (Q in 0){
        # use the try() function to skip possible errors
        try(m <- arima(ridership.log, order=c(p,1,q),
                       seasonal=list(order=c(P,1,Q), period=12),
                       method = "ML"))
        aicc <- AICc(m)

        # print out each result to see where error happens
        print(p)
        print(q)
        print(P)
        print(Q)
        print(aicc)

        Cp <- c(Cp, p)
        Cq <- c(Cq, q)
        CP <- c(CP, P)
        CQ <- c(CQ, Q)
        AICC <- c(AICC, aicc)
        }
      }
    }
}

## [1] 0
## [1] 0
## [1] 1
## [1] 0
## [1] -254.1332
## [1] 0
## [1] 0
## [1] 2
## [1] 0
## [1] -257.6404
## [1] 0
## [1] 1
## [1] 1
## [1] 0
## [1] -275.8413
## [1] 0
## [1] 1
## [1] 2
## [1] 0
## [1] -279.9949
## [1] 1
## [1] 0
## [1] 1
## [1] 0
## [1] -271.7302
## [1] 1
## [1] 0
## [1] 2
## [1] 0
## [1] -275.9704
## [1] 1
## [1] 1
## [1] 1
## [1] 0
## [1] -274.703
## [1] 1
## [1] 1
## [1] 2
## [1] 0
## [1] -278.4941
## [1] 4
## [1] 0
## [1] 1
## [1] 0
## [1] -275.7563
## [1] 4
## [1] 0
## [1] 2
## [1] 0
## [1] -281.3629
## [1] 4
## [1] 1
## [1] 1
## [1] 0
## [1] -273.8467
## [1] 4
## [1] 1
## [1] 2
## [1] 0
## [1] -281.349
```

```
# also get the total number of parameters
TOTAL <- Cp + Cq + CP + CQ
df2 <- data.frame(Cp, Cq, CP, CQ, TOTAL, AICC)

# remove the situations that p = q = 0
df2 <- df2[5:12, ]

# sort the potential models by AICC
df2[order(df2$AICC, decreasing = F), ]

##    Cp Cq CP CQ TOTAL      AICC
## 10  4  0  2  0     6 -281.3629
## 12  4  1  2  0     7 -281.3490
## 8   1  1  2  0     4 -278.4941
## 6   1  0  2  0     3 -275.9704
## 9   4  0  1  0     5 -275.7563
## 7   1  1  1  0     3 -274.7030
## 11  4  1  1  0     6 -273.8467
## 5   1  0  1  0     2 -271.7302
```

6.4 Potential SARIMA Models Fitting with Q = 0

```
# models below named by "mm" + the first column number of the dataframe ab
ove
# fit the four models with the lowest AICC values above (out of eight)

########## mm10 ##########
mm10 <- arima(ridership.log, order=c(4,1,0),
            seasonal=list(order=c(2,1,0), period=12),
            method = "ML")
mm10

##
## Call:
## arima(x = ridership.log, order = c(4, 1, 0), seasonal = list(order = c(
2, 1,
##     0), period = 12), method = "ML")
##
## Coefficients:
##           ar1      ar2      ar3      ar4     sar1     sar2
##       -0.5181  -0.1851  -0.2541  -0.3169  -0.5297  -0.3059
## s.e.   0.0944   0.1059   0.1049   0.0990   0.1040   0.1045
##
## sigma^2 estimated as 0.004075:  log likelihood = 148.03,  aic = -282.07

# the ar2 term is not significant

##### m10_a #####
# use the fixed command to remove the ar2 term and compare AICCs
mm10_a <- arima(ridership.log, order=c(4,1,0),
            seasonal=list(order=c(2,1,0), period=12),
            method = "ML", fixed = c(NA, 0, NA, NA, NA, NA))
```

```
## Warning in arima(ridership.log, order = c(4, 1, 0), seasonal = list(ord
er =
## c(2, : some AR parameters were fixed: setting transform.pars = FALSE
```

```r
mm10_a
```

```
##
## Call:
## arima(x = ridership.log, order = c(4, 1, 0), seasonal = list(order = c(
2, 1,
##     0), period = 12), fixed = c(NA, 0, NA, NA, NA, NA), method = "ML")
##
## Coefficients:
##           ar1  ar2      ar3      ar4     sar1     sar2
##       -0.4460    0  -0.1756  -0.2957  -0.5037  -0.3060
## s.e.   0.0856    0   0.0960   0.0995   0.1028   0.1053
##
## sigma^2 estimated as 0.004195:  log likelihood = 146.53,  aic = -281.06
```

```r
# we find the ar3 term is not significant again

##### m10_b #####
# remove the ar2 and ar3 term
mm10_b <- arima(ridership.log, order=c(4,1,0),
            seasonal=list(order=c(2,1,0), period=12),
            method = "ML", fixed = c(NA, 0, 0, NA, NA, NA))
```

```
## Warning in arima(ridership.log, order = c(4, 1, 0), seasonal = list(ord
er =
## c(2, : some AR parameters were fixed: setting transform.pars = FALSE
```

```r
mm10_b
```

```
##
## Call:
## arima(x = ridership.log, order = c(4, 1, 0), seasonal = list(order = c(
2, 1,
##     0), period = 12), fixed = c(NA, 0, 0, NA, NA, NA), method = "ML")
##
## Coefficients:
##           ar1  ar2  ar3      ar4     sar1     sar2
##       -0.4532    0    0  -0.2202  -0.5075  -0.3075
## s.e.   0.0872    0    0   0.0909   0.1008   0.1044
##
## sigma^2 estimated as 0.004319:  log likelihood = 144.88,  aic = -279.76
```

```r
# compare AICC
AICc(mm10_b) < AICc(mm10)
```

```
## [1] FALSE
```

```r
AICc(mm10_b)
```

```
## [1] -279.0574
```

```
# although the AICC increases, it is still smaller than AICC of mm8

########## mm12 ##########
mm12 <- arima(ridership.log, order=c(4,1,1),
            seasonal=list(order=c(2,1,0), period=12),
            method = "ML")
mm12

##
## Call:
## arima(x = ridership.log, order = c(4, 1, 1), seasonal = list(order = c(
2, 1,
##     0), period = 12), method = "ML")
##
## Coefficients:
##             ar1      ar2      ar3      ar4     ma1     sar1     sar2
##         -1.2007  -0.5102  -0.3148  -0.3125  0.7778  -0.5644  -0.3365
## s.e.     0.1665   0.1616   0.1490   0.1066  0.1707   0.1072   0.1062
##
## sigma^2 estimated as 0.003958:  log likelihood = 149.15,  aic = -282.3

# all parameters are significant

########## mm8 ##########
mm8 <- arima(ridership.log, order=c(1,1,1),
            seasonal=list(order=c(2,1,0), period=12),
            method = "ML")
mm8

##
## Call:
## arima(x = ridership.log, order = c(1, 1, 1), seasonal = list(order = c(
2, 1,
##     0), period = 12), method = "ML")
##
## Coefficients:
##            ar1      ma1     sar1     sar2
##         0.1525  -0.6703  -0.5066  -0.2653
## s.e.    0.1816   0.1425   0.1043   0.1055
##
## sigma^2 estimated as 0.004376:  log likelihood = 144.41,  aic = -278.82

# the ar1 term is not significant

##### mm8_a #####
# use the "fixed" command to remove the ar1 term and compare AICCs
mm8_a <- arima(ridership.log, order=c(1,1,1),
            seasonal=list(order=c(2,1,0), period=12),
            method = "ML", fixed = c(0, NA, NA, NA))

## Warning in arima(ridership.log, order = c(1, 1, 1), seasonal = list(ord
er =
## c(2, : some AR parameters were fixed: setting transform.pars = FALSE

mm8_a
```

```
##
## Call:
## arima(x = ridership.log, order = c(1, 1, 1), seasonal = list(order = c(
2, 1,
##     0), period = 12), fixed = c(0, NA, NA, NA), method = "ML")
##
## Coefficients:
##       ar1      ma1     sar1     sar2
##         0  -0.5533  -0.5208  -0.2722
## s.e.    0   0.0979   0.1020   0.1051
##
## sigma^2 estimated as 0.004395:  log likelihood = 144.1,  aic = -280.19
```

```
# compare AICCs
AICc(mm8_a) < AICc(mm8)
```

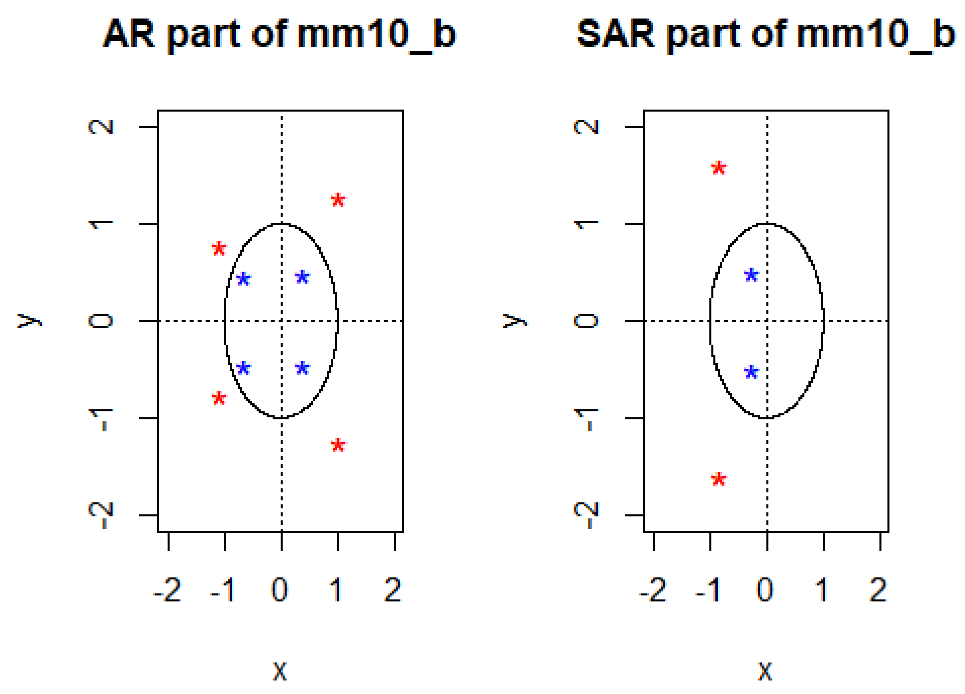
```
## [1] TRUE
```

```
# AICC of the model without ar part is better, keep the fixed one

########## mm6 ##########
mm6 <- arima(ridership.log, order=c(1,1,0),
            seasonal=list(order=c(2,1,0), period=12),
            method = "ML")
mm6
```

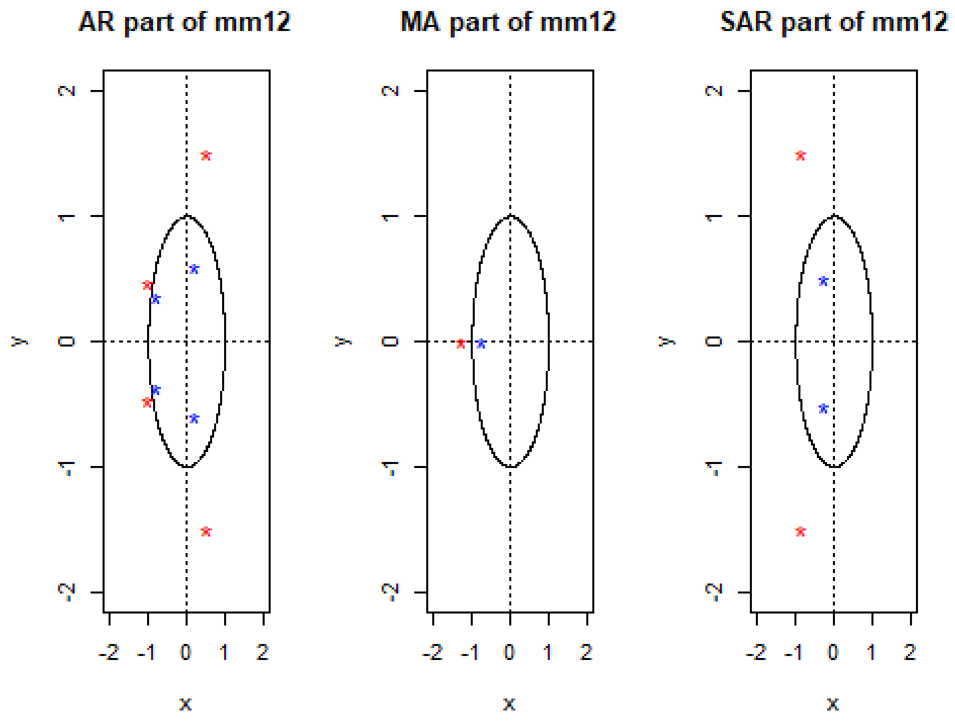
```
##
## Call:
## arima(x = ridership.log, order = c(1, 1, 0), seasonal = list(order = c(
2, 1,
##     0), period = 12), method = "ML")
##
## Coefficients:
##           ar1     sar1     sar2
##       -0.4191  -0.4945  -0.2765
## s.e.   0.0883   0.1020   0.1059
##
## sigma^2 estimated as 0.004569:  log likelihood = 142.08,  aic = -276.17
```

6.4.1 Unit Root Detection for SARIMA Models with Q = 0

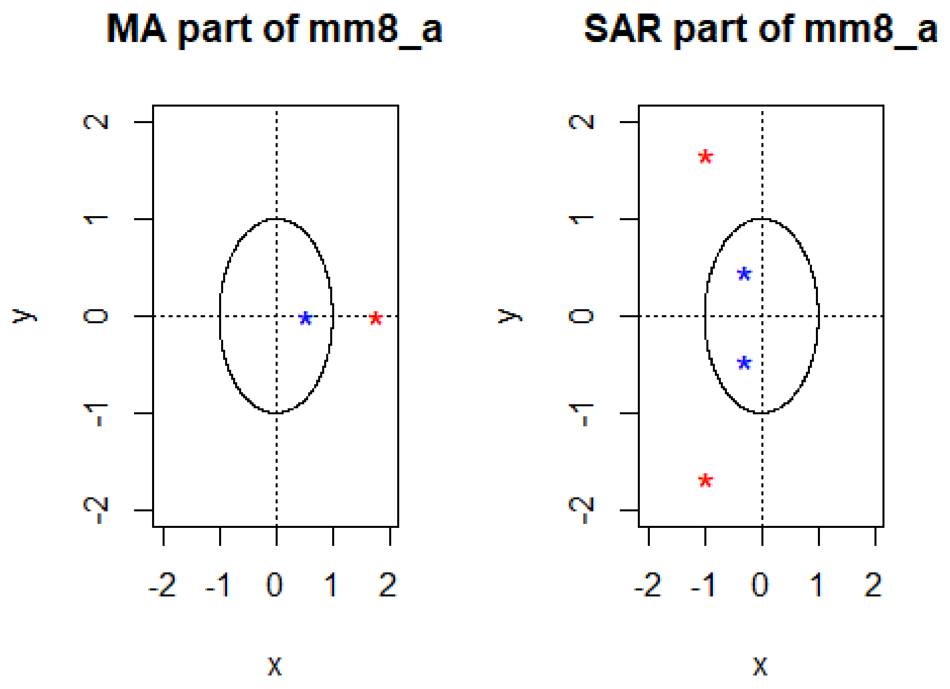
```
# import the plot.roots function given on GauchoSpace
source("plot.roots.R.txt")

# mm10_b
op <- par(mfrow=c(1,2))
plot.roots(NULL, polyroot(c(1, 0.4532, 0, 0, 0.2202)), main = "AR part of
mm10_b")
plot.roots(NULL, polyroot(c(1, 0.5075, 0.3075)), main = "SAR part of mm10_
b")
```



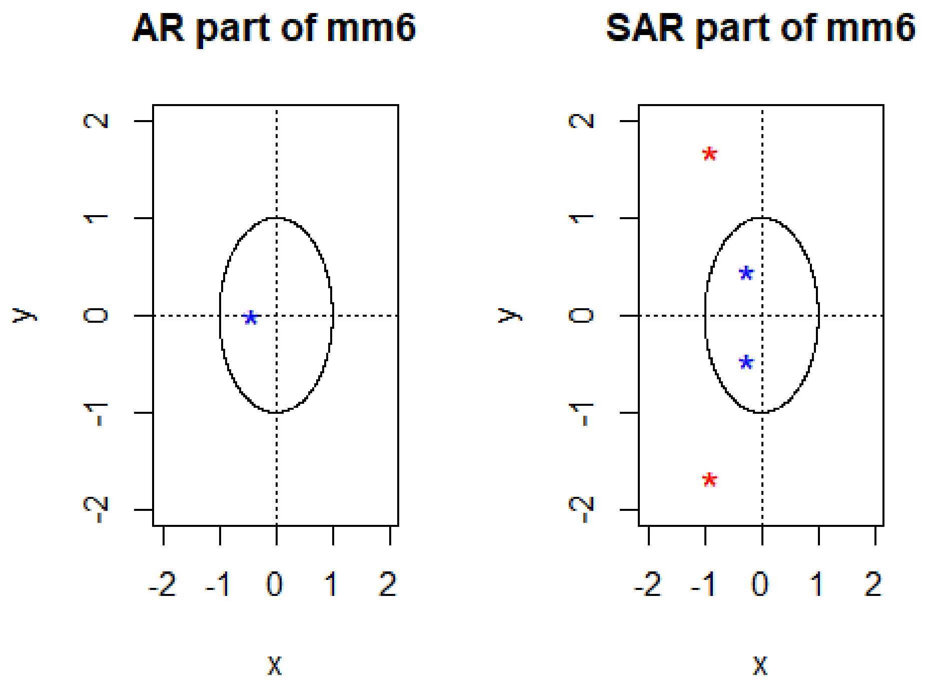
```
par(op)
# no unit roots in both parts

# mm12
op <- par(mfrow=c(1,3))
plot.roots(NULL, polyroot(c(1, 1.2007, 0.5102, 0.3148, 0.3125)), main = "A
R part of mm12")
plot.roots(NULL, polyroot(c(1, 0.7778)), main = "MA part of mm12")
plot.roots(NULL, polyroot(c(1, 0.5644, 0.3365)), main = "SAR part of mm12"
)
```



```
par(op)
# no unit roots in any part

# mm8_a
op <- par(mfrow=c(1,2))
plot.roots(NULL, polyroot(c(1, -0.5533)), main = "MA part of mm8_a")
plot.roots(NULL, polyroot(c(1, 0.5208, 0.2722)), main = "SAR part of mm8_a
")
```



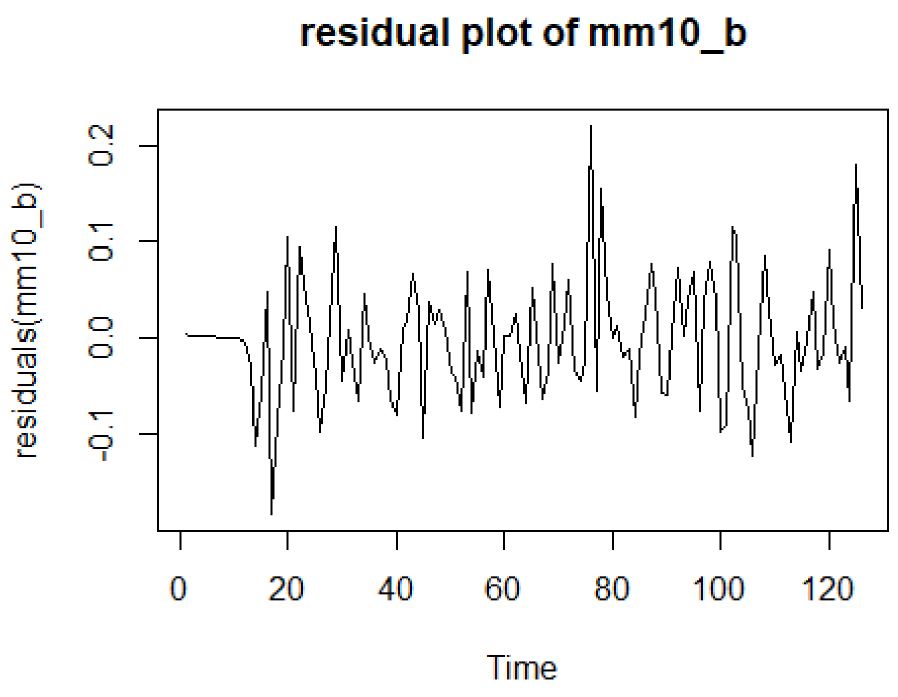
```r
par(op)
# no unit roots in any part

# mm6
op <- par(mfrow=c(1,2))
plot.roots(NULL, polyroot(c(1, 0.4191)), main = "AR part of mm6")
plot.roots(NULL, polyroot(c(1, 0.4945, 0.2765)), main = "SAR part of mm6")
```



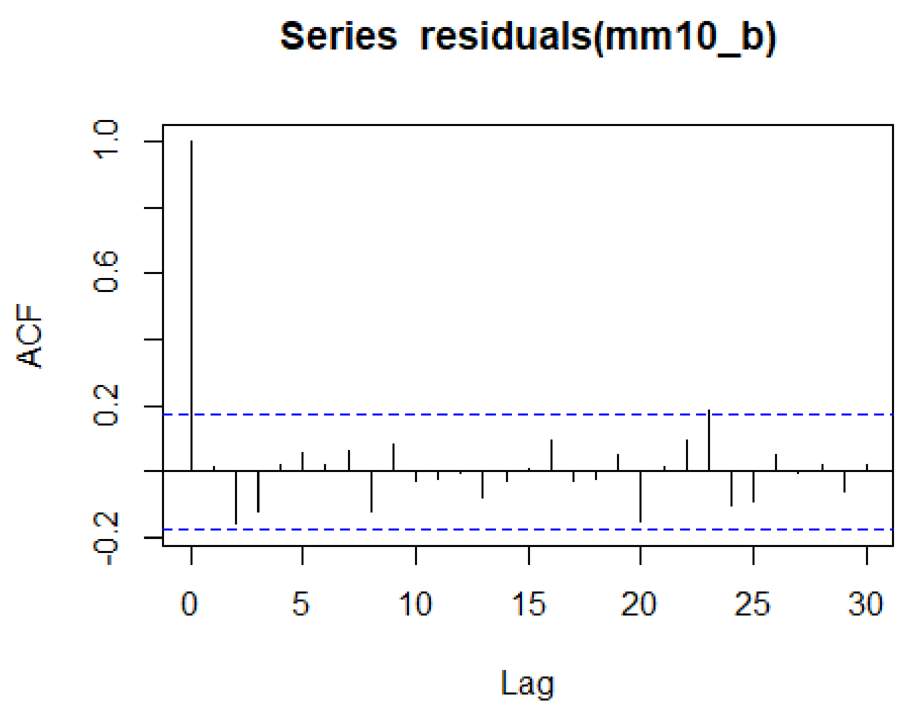
```
par(op)
# AR part root is outside the scope of the plot
# no unit roots in any part
```

6.4.2 Diagnostic Check For the Model “mm10_b”

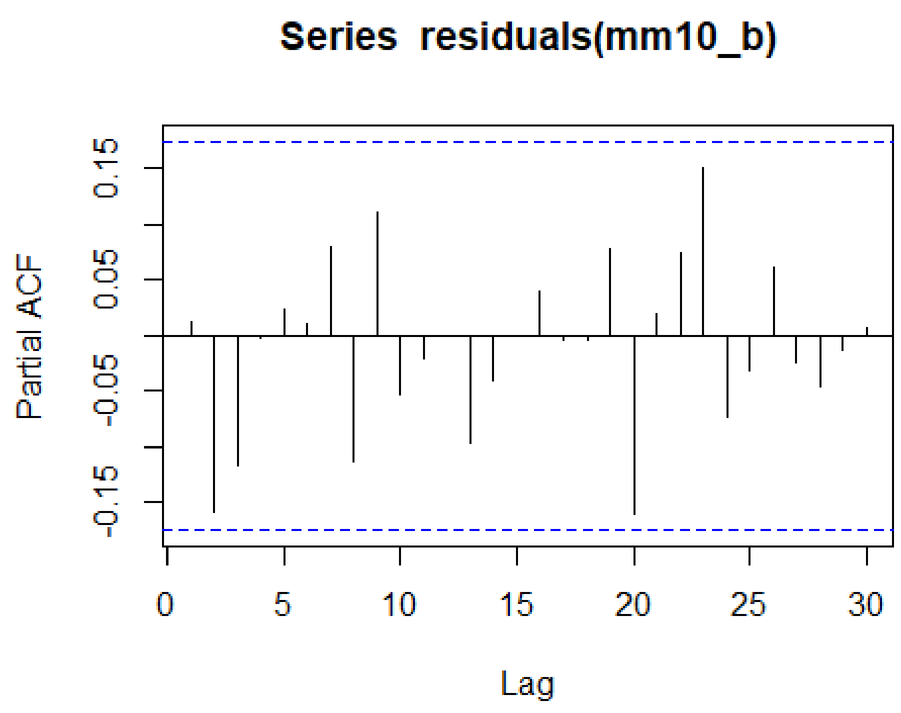
```
# plot the residuals
plot(residuals(mm10_b), main = "residual plot of mm10_b")
```



```
# acf and pacf of the residuals
acf(residuals(mm10_b), lag.max = 30)
```

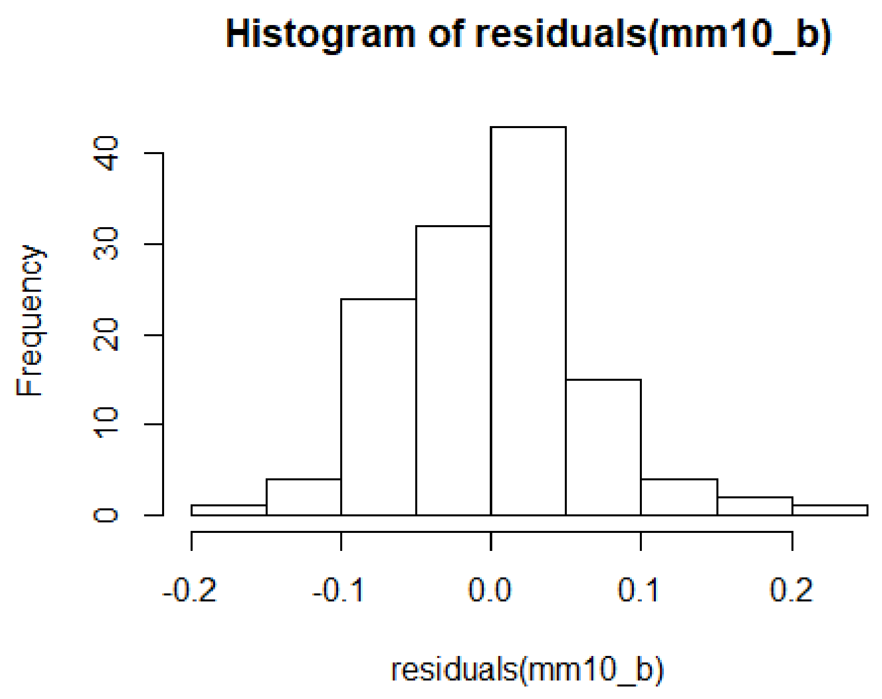


```
pacf(residuals(mm10_b), lag.max = 30)
```

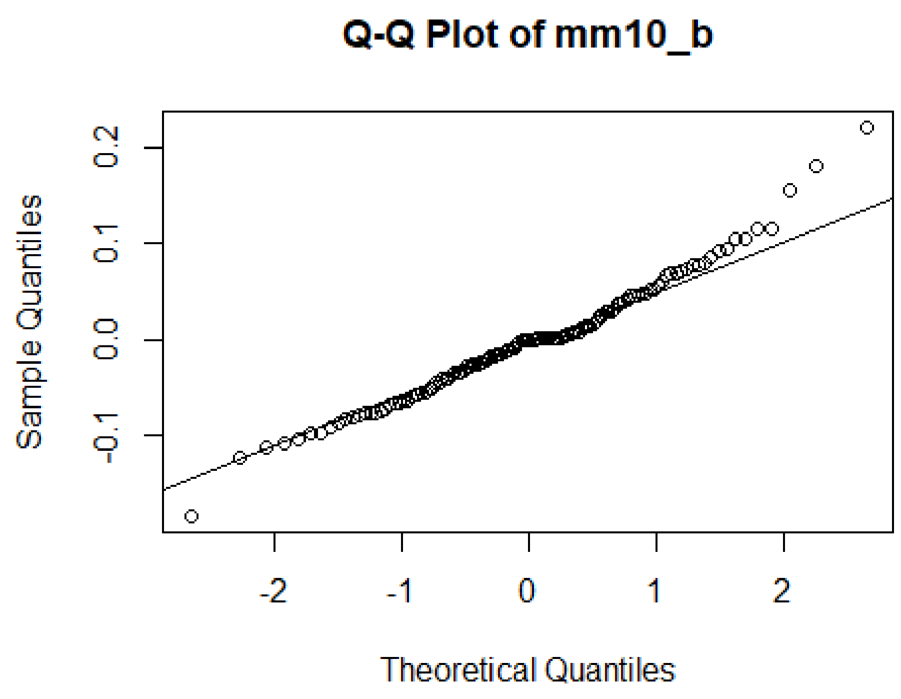


```
# Box-Pierce test for independency
Box.test(residuals(mm10_b), lag = 11, type = "Box-Pierce", fitdf = 4)

##
##  Box-Pierce test
##
```

```
## data:  residuals(mm10_b)
## X-squared = 8.7995, df = 7, p-value = 0.2674

# Ljung-Box test for independency
Box.test(residuals(mm10_b), lag = 11, type = "Ljung-Box", fitdf = 4)

##
##  Box-Ljung test
##
## data:  residuals(mm10_b)
## X-squared = 9.2937, df = 7, p-value = 0.2323

# McLeod_Li test to see if squared residuals are correlated
Box.test((residuals(mm10_b))^2, lag = 11, type = "Ljung-Box", fitdf = 0)

##
##  Box-Ljung test
##
## data:  (residuals(mm10_b))^2
## X-squared = 6.3086, df = 11, p-value = 0.852

# use the ar() function to see if residuals are white noise
ar(residuals(mm10_b))

##
## Call:
## ar(x = residuals(mm10_b))
##
##
## Order selected 0  sigma^2 estimated as  0.003906

# draw the histogram of the residuals
hist(residuals(mm10_b))
```



```r
# QQ plot of the residuals
qqnorm(residuals(mm10_b), main = "Q-Q Plot of mm10_b")
qqline(residuals(mm10_b))
```

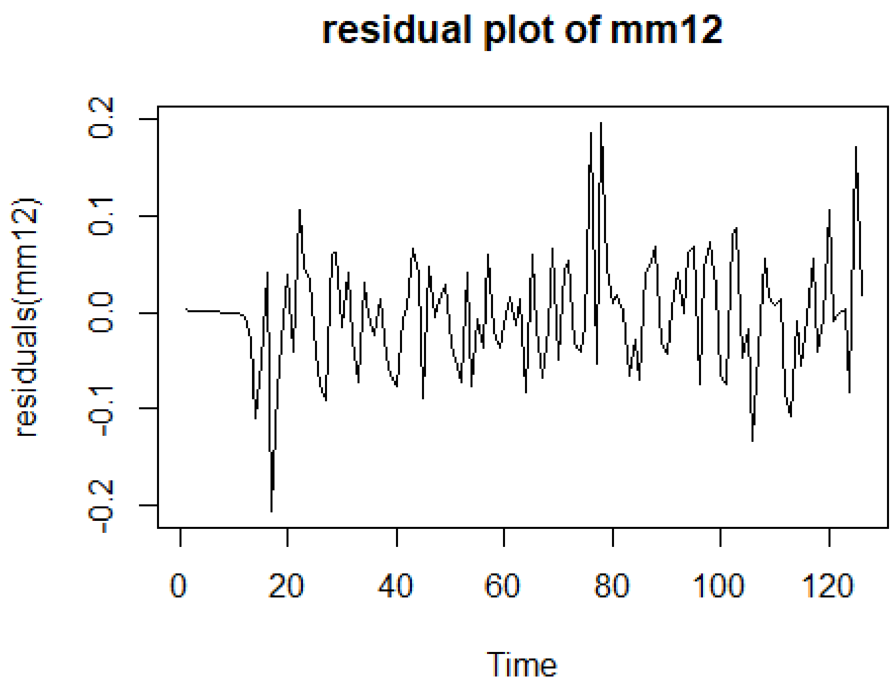


```r
# Shapiro-Wilk test for normality
shapiro.test(residuals(mm10_b))
```

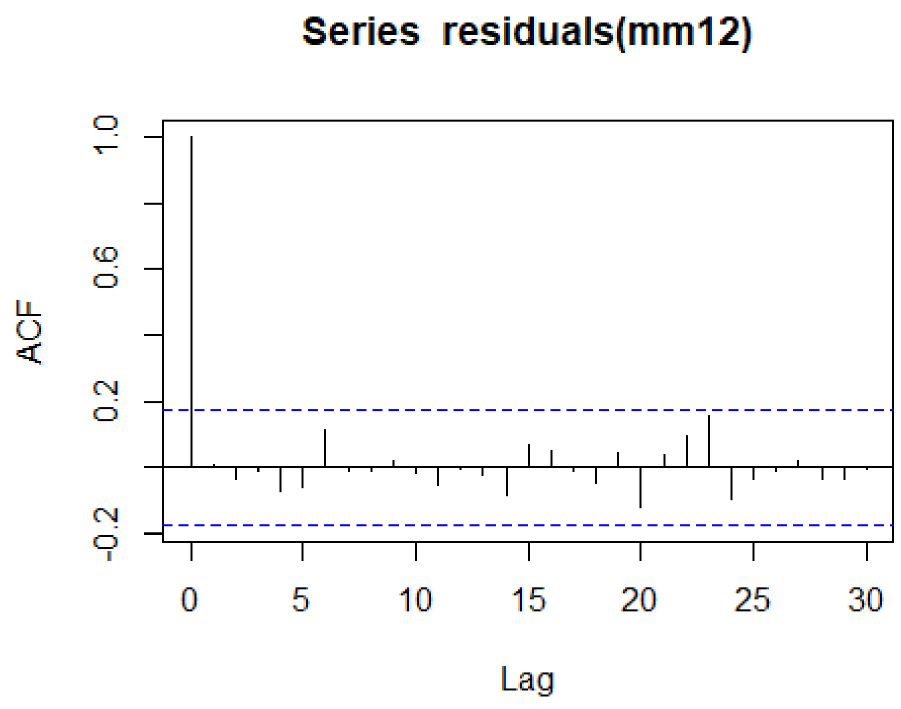
```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(mm10_b)
## W = 0.9796, p-value = 0.05396
```

6.4.3 Diagnostic Check For the Model "mm12"

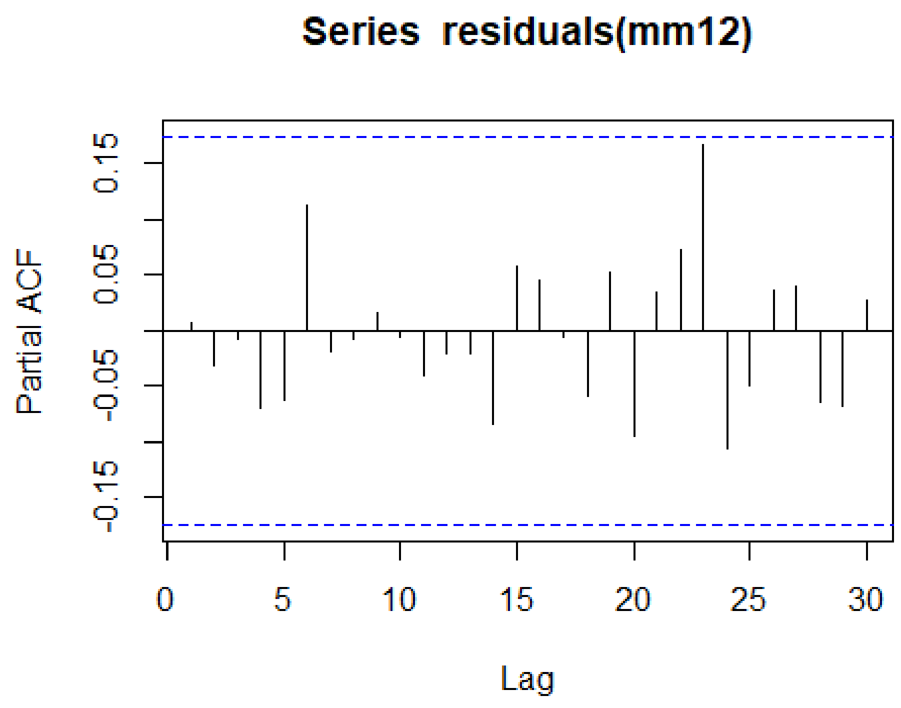
```r
# plot the residuals
plot(residuals(mm12), main = "residual plot of mm12")
```



```r
# acf and pacf of the residuals
acf(residuals(mm12), lag.max = 30)
```

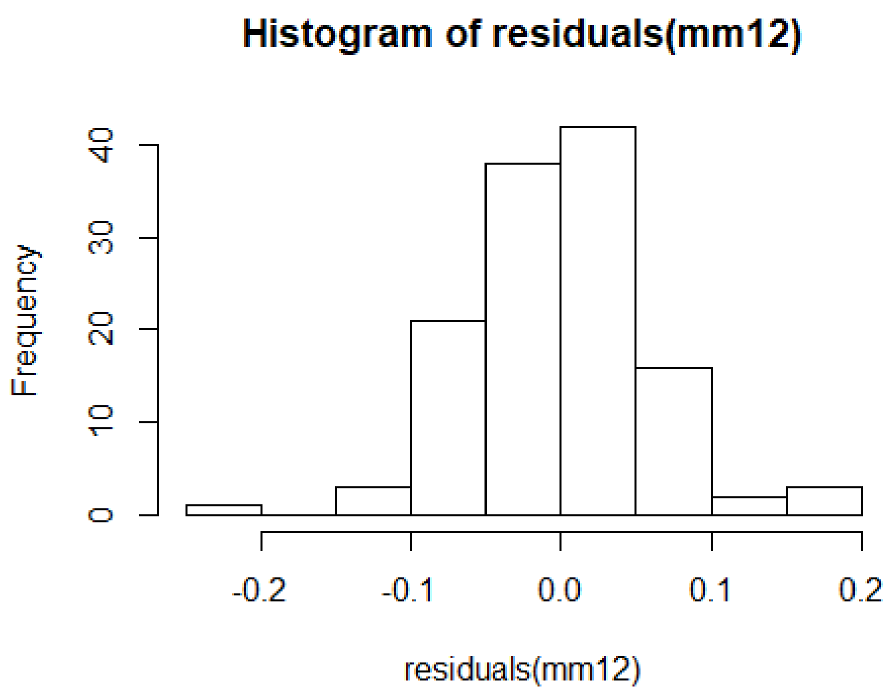


```
pacf(residuals(mm12), lag.max = 30)
```

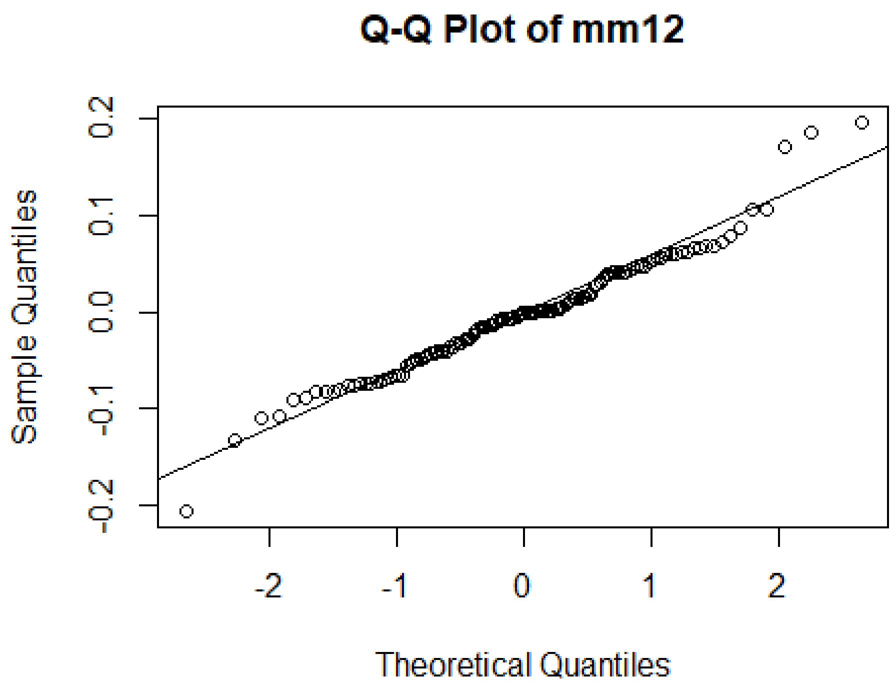


```
# Box-Pierce test for independency
Box.test(residuals(mm12), lag = 11, type = "Box-Pierce", fitdf = 7)

##
##  Box-Pierce test
##
```

```
## data:  residuals(mm12)
## X-squared = 3.3349, df = 4, p-value = 0.5034

# Ljung-Box test for independency
Box.test(residuals(mm12), lag = 11, type = "Ljung-Box", fitdf = 7)

##
##  Box-Ljung test
##
## data:  residuals(mm12)
## X-squared = 3.5567, df = 4, p-value = 0.4693

# McLeod_Li test to see if squared residuals are correlated
Box.test((residuals(mm12))^2, lag = 11, type = "Ljung-Box", fitdf = 0)

##
##  Box-Ljung test
##
## data:  (residuals(mm12))^2
## X-squared = 5.3902, df = 11, p-value = 0.9108

# use the ar() function to see if residuals are white noise
ar(residuals(mm12))

##
## Call:
## ar(x = residuals(mm12))
##
##
## Order selected 0  sigma^2 estimated as  0.003577

# draw the histogram of the residuals
hist(residuals(mm12))
```



```
# QQ plot of the residuals
qqnorm(residuals(mm12), main = "Q-Q Plot of mm12")
qqline(residuals(mm12))
```

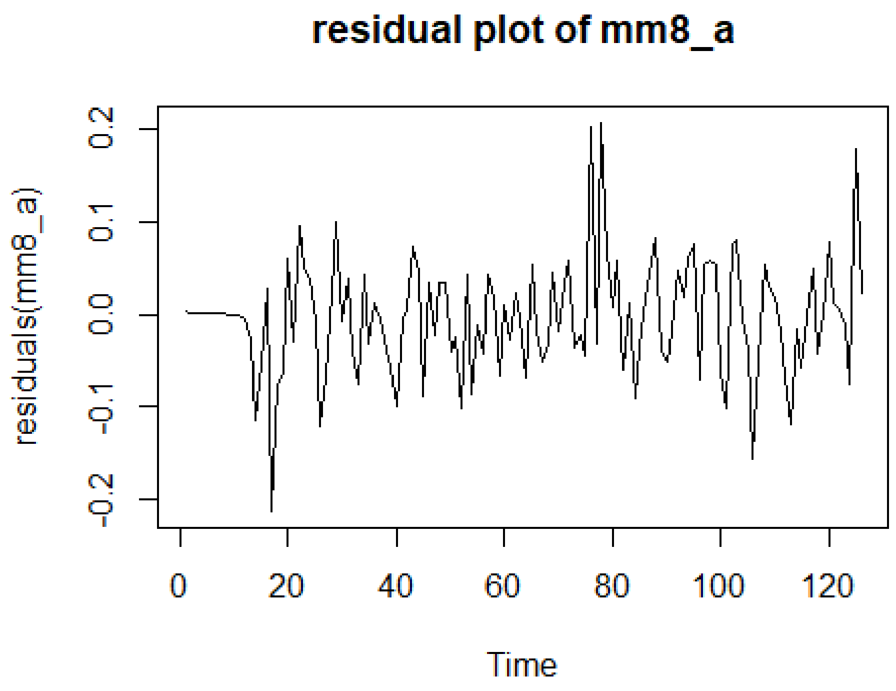


```
# Shapiro-Wilk test for normality
shapiro.test(residuals(mm12))
```

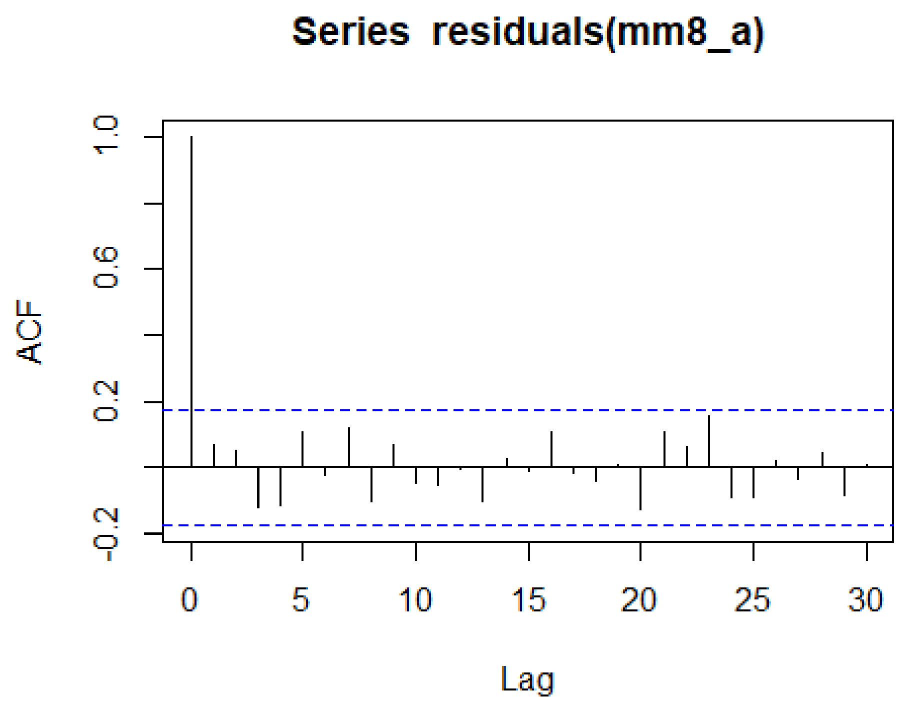
```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(mm12)
## W = 0.96922, p-value = 0.005669
```

6.4.4 Diagnostic Check For the Model “mm8_a”

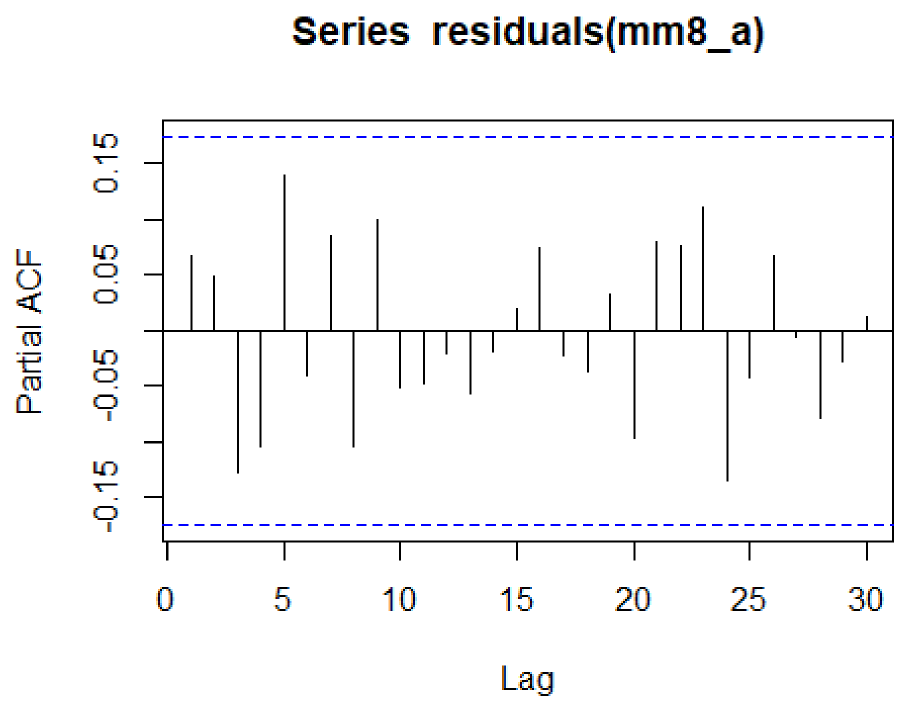
```r
# plot the residuals
plot(residuals(mm8_a), main = "residual plot of mm8_a")
```



```r
# acf and pacf of the residuals
acf(residuals(mm8_a), lag.max = 30)
```

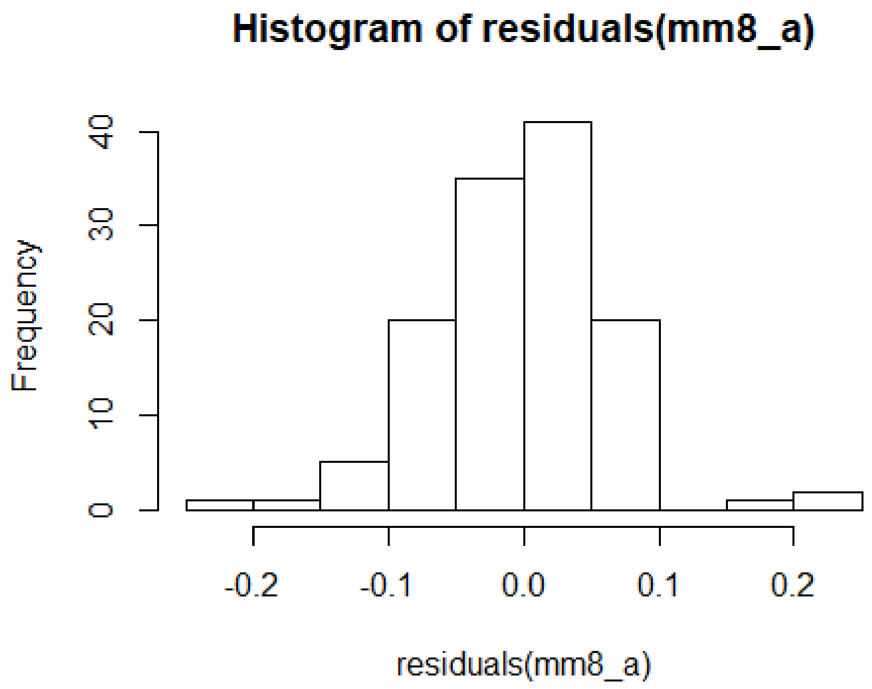


```
pacf(residuals(mm8_a), lag.max = 30)
```

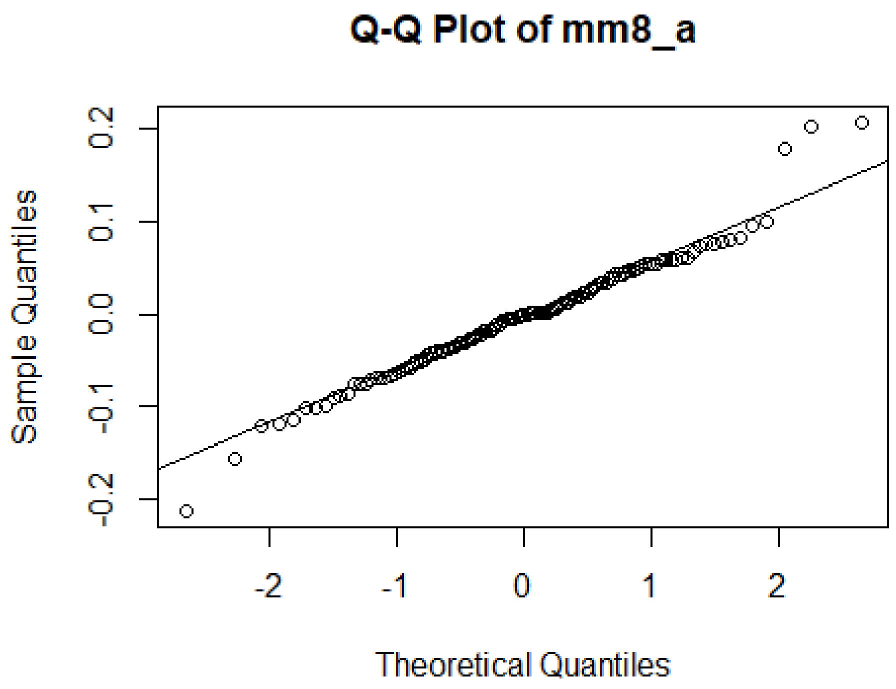


```
# Box-Pierce test for independency
Box.test(residuals(mm8_a), lag = 11, type = "Box-Pierce", fitdf = 3)

##
##  Box-Pierce test
##
```

```
## data:  residuals(mm8_a)
## X-squared = 10.38, df = 8, p-value = 0.2394

# Ljung-Box test for independency
Box.test(residuals(mm8_a), lag = 11, type = "Ljung-Box", fitdf = 3)

##
##  Box-Ljung test
##
## data:  residuals(mm8_a)
## X-squared = 11.03, df = 8, p-value = 0.2

# McLeod_Li test to see if squared residuals are correlated
Box.test((residuals(mm8_a))^2, lag = 11, type = "Ljung-Box", fitdf = 0)

##
##  Box-Ljung test
##
## data:  (residuals(mm8_a))^2
## X-squared = 6.2508, df = 11, p-value = 0.8561

# use the ar() function to see if residuals are white noise
ar(residuals(mm8_a))

##
## Call:
## ar(x = residuals(mm8_a))
##
##
## Order selected 0  sigma^2 estimated as  0.003969

# draw the histogram of the residuals
hist(residuals(mm8_a))
```



```r
# QQ plot of the residuals
qqnorm(residuals(mm8_a), main = "Q-Q Plot of mm8_a")
qqline(residuals(mm8_a))
```

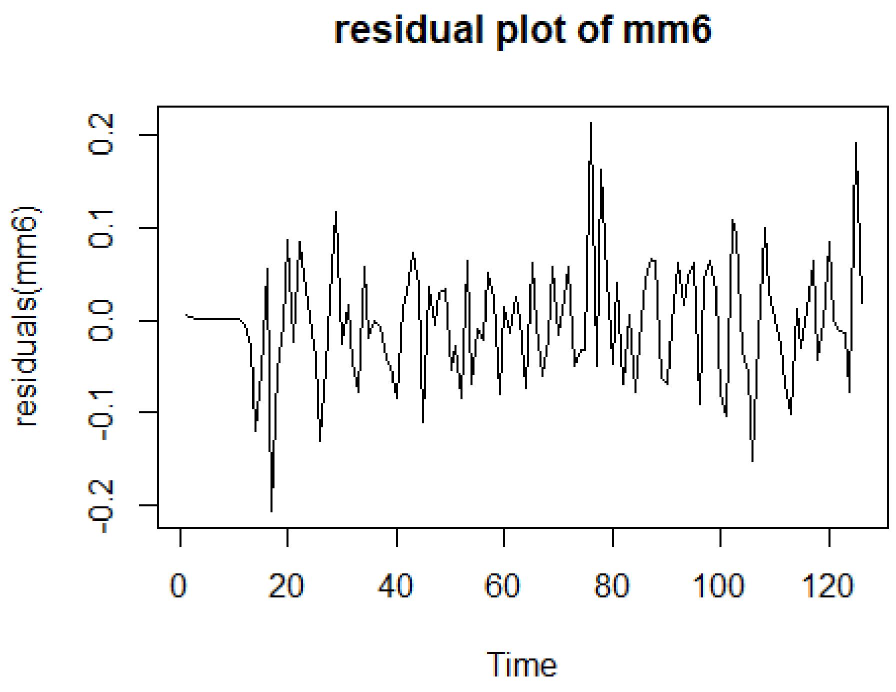


```r
# Shapiro-Wilk test for normality
shapiro.test(residuals(mm8_a))
```

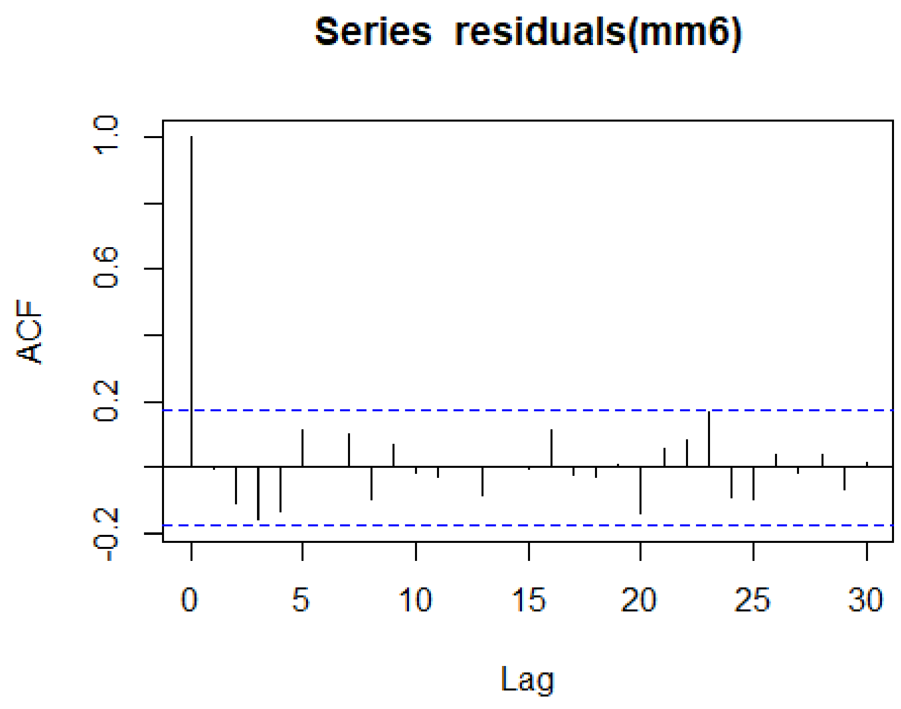
```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(mm8_a)
## W = 0.97113, p-value = 0.008467
```

6.4.5 Diagnostic Checking For the Model "mm6"

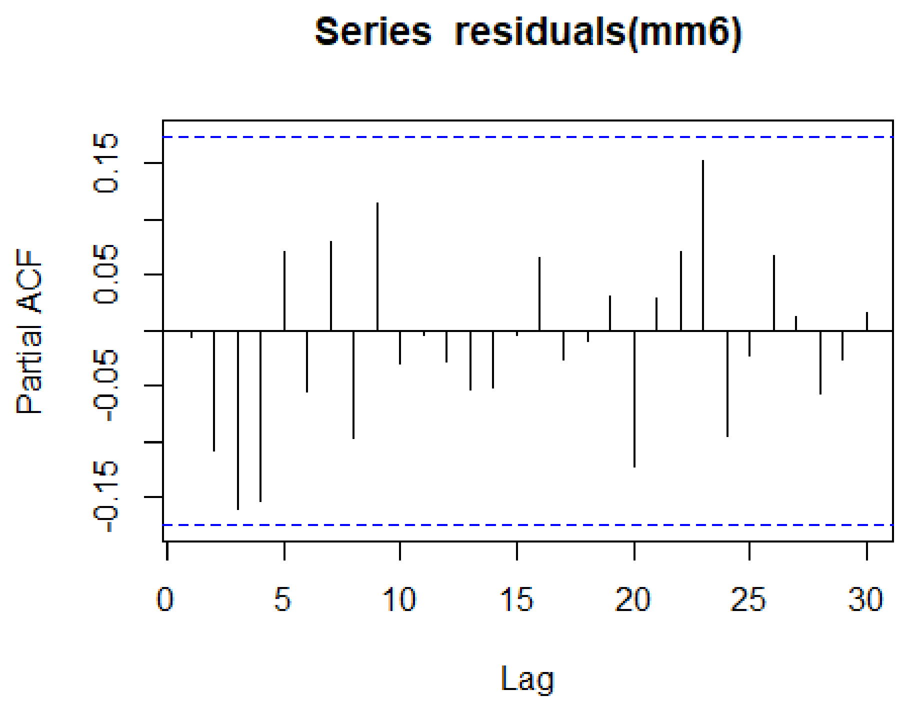
```
# plot the residuals
plot(residuals(mm6), main = "residual plot of mm6")
```



```
# acf and pacf of the residuals
acf(residuals(mm6), lag.max = 30)
```

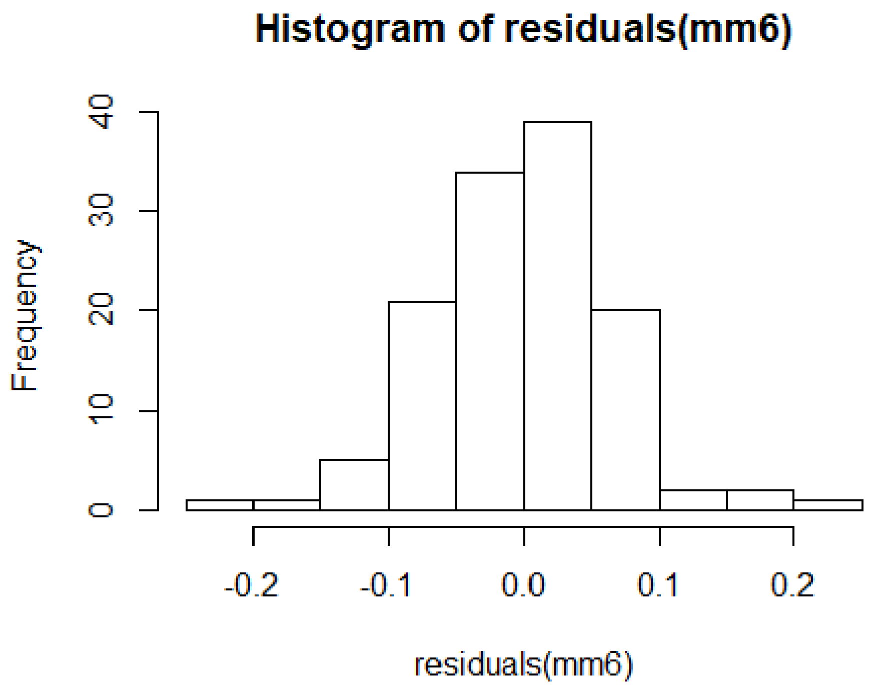


```
pacf(residuals(mm6), lag.max = 30)
```

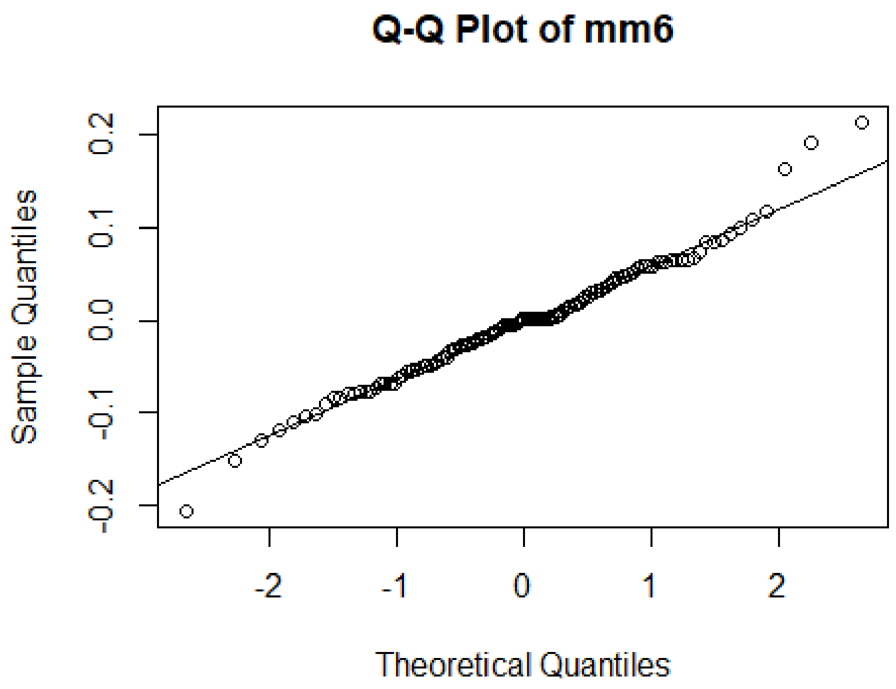


```
# Box-Pierce test for independency
Box.test(residuals(mm6), lag = 11, type = "Box-Pierce", fitdf = 3)

##
##  Box-Pierce test
##
```

```
## data:  residuals(mm6)
## X-squared = 11.528, df = 8, p-value = 0.1736

# Ljung-Box test for independency
Box.test(residuals(mm6), lag = 11, type = "Ljung-Box", fitdf = 3)

##
##  Box-Ljung test
##
## data:  residuals(mm6)
## X-squared = 12.17, df = 8, p-value = 0.1438

# McLeod_Li test to see if squared residuals are correlated
Box.test((residuals(mm6))^2, lag = 11, type = "Ljung-Box", fitdf = 0)

##
##  Box-Ljung test
##
## data:  (residuals(mm6))^2
## X-squared = 4.7899, df = 11, p-value = 0.9409

# use the ar() function to see if residuals are white noise
ar(residuals(mm6))

##
## Call:
## ar(x = residuals(mm6))
##
##
## Order selected 0  sigma^2 estimated as  0.004132

# draw the histogram of the residuals
hist(residuals(mm6))
```



```
# QQ plot of the residuals
qqnorm(residuals(mm6), main = "Q-Q Plot of mm6")
qqline(residuals(mm6))
```



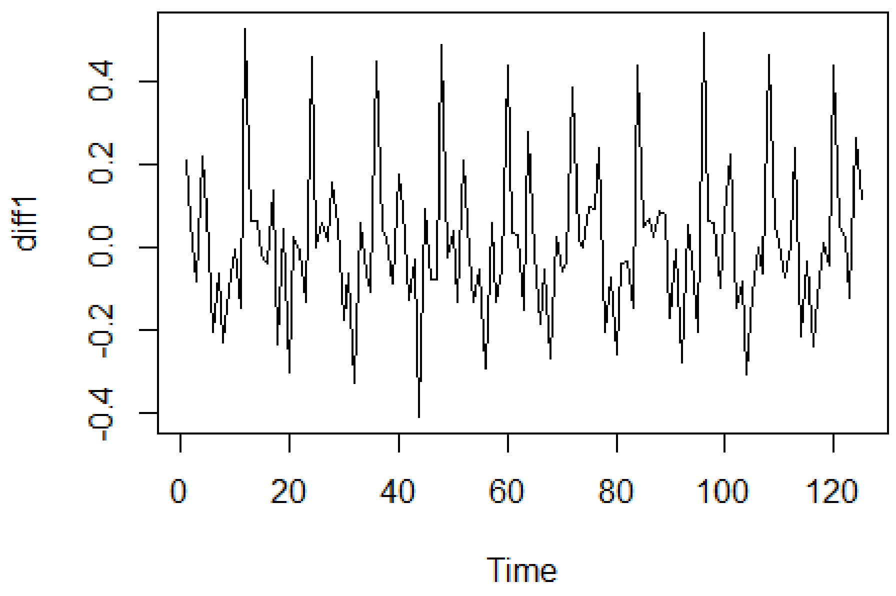
```
# Shapiro-Wilk test for normality
shapiro.test(residuals(mm6))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(mm6)
## W = 0.98136, p-value = 0.08
```

7. Alternative Model Selection and Fitting

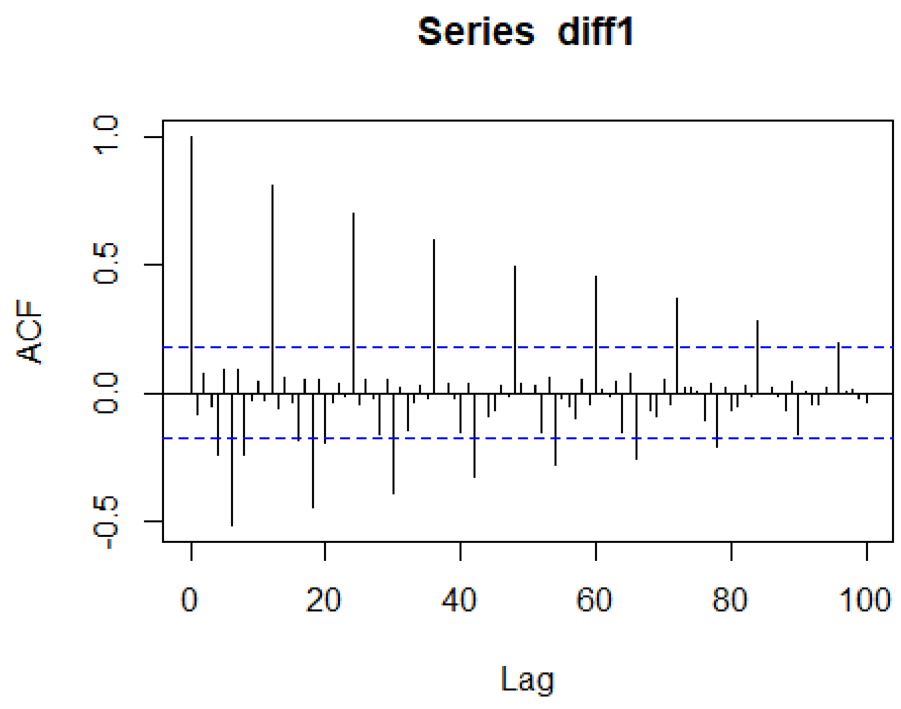
7.1 De-trend the Log Transformed Data First

```
# in codes 6.2
# the unit root in sma part indicates overdifferencing in seasonal lags
# so we de-trend to the transformed data first this time

# difference at lag 1
diff1 <- diff(ridership.log, 1)

# see if the differencing gets a lower variance
var(diff1) < var(ridership.log)

## [1] TRUE

# plot the time series plot of the de-trended data
ts.plot(diff1)
```

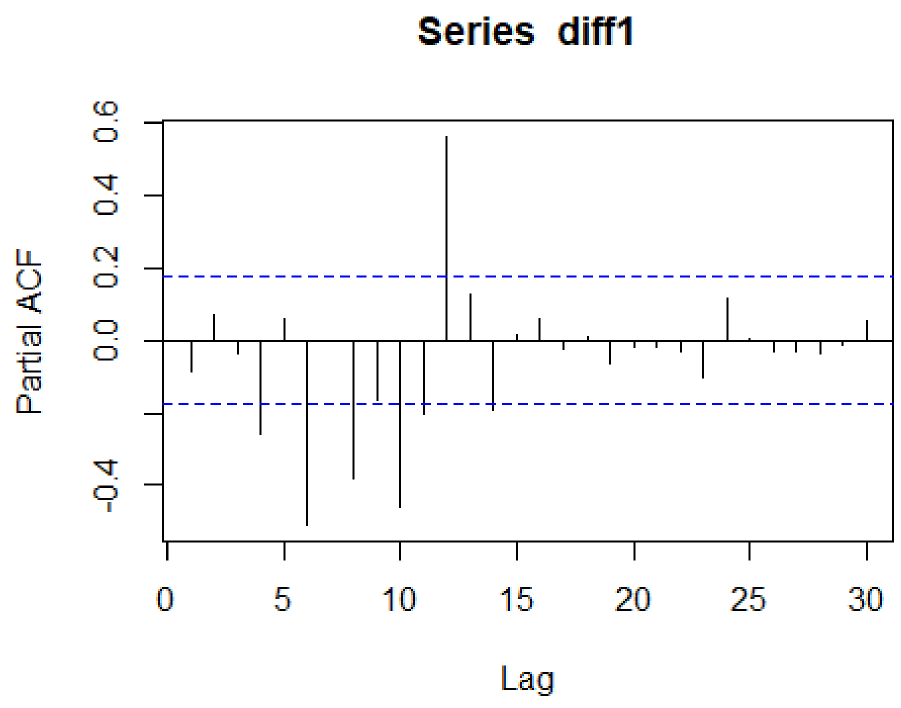


7.2 ACF and PACF of the De-trended Data

```
# plot the acf and pacf of the de-trended data
acf(diff1, lag.max = 100)
```



```r
pacf(diff1, lag.max = 30)
```



7.3 Preliminary Estimation using Yule-Walker for AR Model

```r
# ACF exponentially decays to 0, PACF cuts off at the seasonal lag 12
# consider AR model
ar(diff1, method = "yule-walker")
```

```
##
## Call:
## ar(x = diff1, method = "yule-walker")
##
## Coefficients:
##       1        2        3        4        5        6        7        8

## -0.2014  -0.0601  -0.1044  -0.2883  -0.0743  -0.3256  -0.0944  -0.2613

##       9       10       11       12       13       14
## -0.1053  -0.2680  -0.0462   0.5484   0.0809  -0.1913
##
## Order selected 14  sigma^2 estimated as  0.01106

# fit the ar model with arima function to see the significance of each par
ameter
arm <- arima(ridership.log, order=c(14,1,0),
             seasonal=list(order=c(0,0,0), period=12),
             xreg=1:length(ridership.log), include.mean=F)
arm

##
## Call:
## arima(x = ridership.log, order = c(14, 1, 0), seasonal = list(order = c
(0, 0,
##     0), period = 12), xreg = 1:length(ridership.log), include.mean = F)
##
## Coefficients:
##             ar1      ar2      ar3      ar4      ar5      ar6      ar7
##         -0.4654  -0.2642  -0.2932  -0.4003  -0.2568  -0.4279  -0.2856
## s.e.     0.0912   0.1049   0.0933   0.0963   0.0959   0.0956   0.0963
##             ar8      ar9     ar10     ar11    ar12    ar13     ar14
##         -0.3996  -0.2818  -0.3938  -0.2211  0.5273  0.1734  -0.1310
## s.e.     0.0967   0.0942   0.0955   0.0940  0.0949  0.1047   0.0995
##         1:length(ridership.log)
##                          0.0056
## s.e.                     0.0015
##
## sigma^2 estimated as 0.004563:  log likelihood = 149.67,  aic = -267.35

# only the ar13 and ar14 terms are not significant
# that means we have at least 12 parameters
# the sample size is not large enough to have such many parameters
```

7.4 SARIMA Model with D = 0

```
len <- length(ridership.log)
# fit sarima model with D = 0

########## mmm4 ##########
# pacf is outside the CI at lag 4, and the seasonal lag is 12
mmm4 <- arima(ridership.log, order=c(4,1,0),
              seasonal=list(order=c(1,0,0), period=12),
              xreg=1:len, include.mean=F)
mmm4
```

```
##
## Call:
## arima(x = ridership.log, order = c(4, 1, 0), seasonal = list(order = c(
1, 0,
##     0), period = 12), xreg = 1:len, include.mean = F)
##
## Coefficients:
##           ar1      ar2      ar3      ar4    sar1    1:len
##       -0.3913  -0.0748  -0.2092  -0.2565  0.9341  0.0117
## s.e.   0.0911   0.0981   0.0954   0.0959  0.0234  0.0261
##
## sigma^2 estimated as 0.00511:  log likelihood = 139.78,  aic = -265.57
```

```
# ar2 term and the constant are not significant

##### mmm4_a #####
# use the fixed command to set those to be 0
mmm4_a <- arima(ridership.log, order=c(4,1,0),
                seasonal=list(order=c(1,0,0), period=12),
                xreg=1:len, include.mean=F,
                fixed = c(NA, 0, NA, NA, NA, 0))
```

```
## Warning in arima(ridership.log, order = c(4, 1, 0), seasonal = list(ord
er =
## c(1, : some AR parameters were fixed: setting transform.pars = FALSE
```

```
mmm4_a
```

```
##
## Call:
## arima(x = ridership.log, order = c(4, 1, 0), seasonal = list(order = c(
1, 0,
##     0), period = 12), xreg = 1:len, include.mean = F, fixed = c(NA, 0,
NA, NA,
##     NA, 0))
##
## Coefficients:
##           ar1  ar2      ar3      ar4    sar1  1:len
##       -0.3669    0  -0.1867  -0.2524  0.9316      0
## s.e.   0.0858    0   0.0912   0.0961  0.0238      0
##
## sigma^2 estimated as 0.005161:  log likelihood = 139.4,  aic = -268.8
```

```
# all the coefs are significant now

# see if the AICC decreases
AICc(mmm4_a) < AICc(mmm4)
```

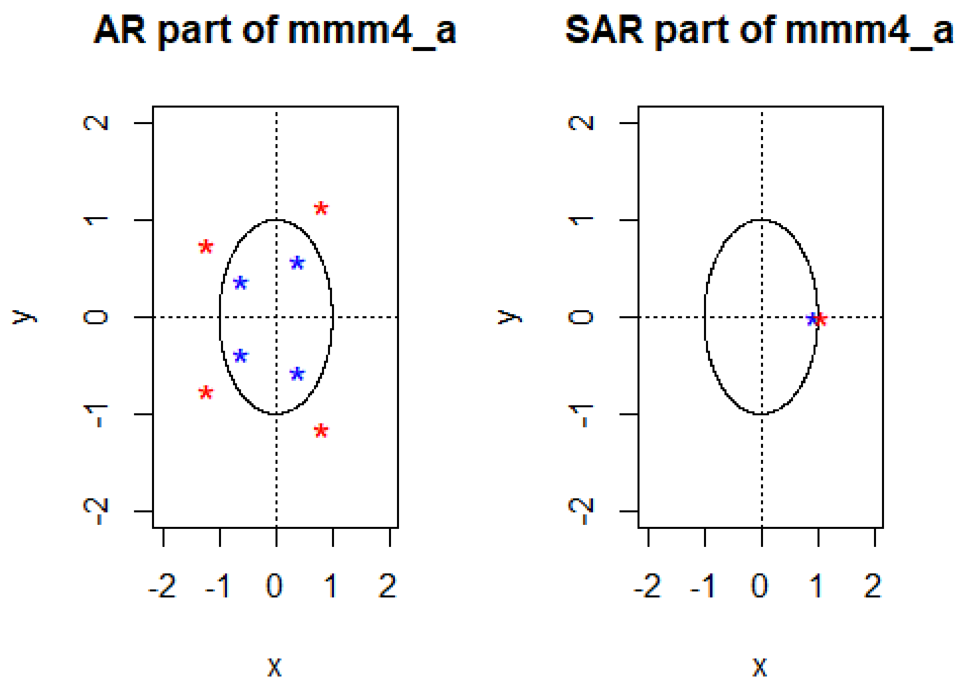
```
## [1] TRUE
```

```
########## mmm6 ##########
# pacf is also outside the CI at lag 6, and the seasonal lag is 12
mmm6 <- arima(ridership.log, order=c(6,1,0),
              seasonal=list(order=c(1,0,0), period=12),
              xreg=1:len, include.mean=F)
mmm6

##
## Call:
## arima(x = ridership.log, order = c(6, 1, 0), seasonal = list(order = c(
1, 0,
##     0), period = 12), xreg = 1:len, include.mean = F)
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5      ar6    sar1   1:
len
##       -0.3999  -0.1282  -0.2618  -0.2864  -0.1127  -0.1664  0.9274  0.0
099
## s.e.   0.0946   0.1035   0.0997   0.1030   0.1075   0.0991  0.0263  0.0
201
##
## sigma^2 estimated as 0.005031:  log likelihood = 141.27,  aic = -264.54

# ar2, ar5, ar6 and the constant are not significant

##### mmm6_a #####
# use the fixed command to set those to be 0
mmm6_a <- arima(ridership.log, order=c(6,1,0),
                seasonal=list(order=c(1,0,0), period=12),
                xreg=1:len, include.mean=F,
                fixed = c(NA, 0, NA, NA, 0, 0, NA, 0))

## Warning in arima(ridership.log, order = c(6, 1, 0), seasonal = list(ord
er =
## c(1, : some AR parameters were fixed: setting transform.pars = FALSE

# see if the AICC decreases
AICc(mmm6_a) < AICc(mmm6)

## [1] TRUE

mmm6_a

##
## Call:
## arima(x = ridership.log, order = c(6, 1, 0), seasonal = list(order = c(
1, 0,
##     0), period = 12), xreg = 1:len, include.mean = F, fixed = c(NA, 0,
NA, NA,
##     0, 0, NA, 0))
##
## Coefficients:
##           ar1  ar2      ar3      ar4  ar5  ar6    sar1  1:len
##       -0.3668    0  -0.1866  -0.2523    0    0  0.9316      0
## s.e.   0.0858    0   0.0912   0.0961    0    0  0.0238      0
##
## sigma^2 estimated as 0.005161:  log likelihood = 139.4,  aic = -268.8
```

```
# after removing the non-significant coefs, mmm6_a becomes the same as mmm
4_a
```

7.5 Unit Root Detection for SARIMA Model with D = 0

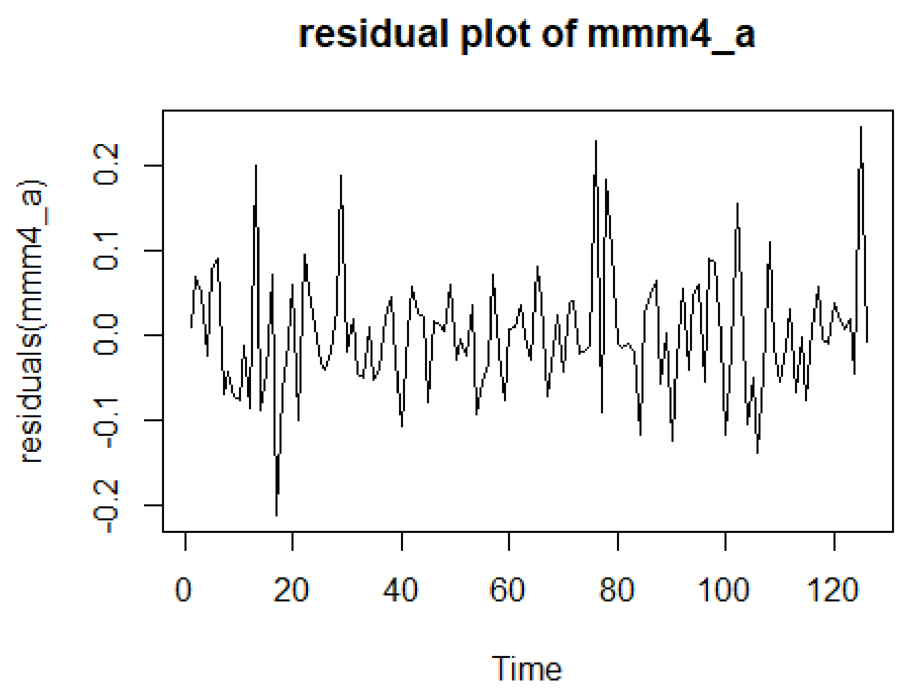
```
# mmm4_a (or mmm6_a)
op <- par(mfrow=c(1,2))
plot.roots(NULL, polyroot(c(1, 0.3669, 0, 0.1867, 0.2524)), main = "AR par
t of mmm4_a")
plot.roots(NULL, polyroot(c(1, -0.9316)), main = "SAR part of mmm4_a")
```



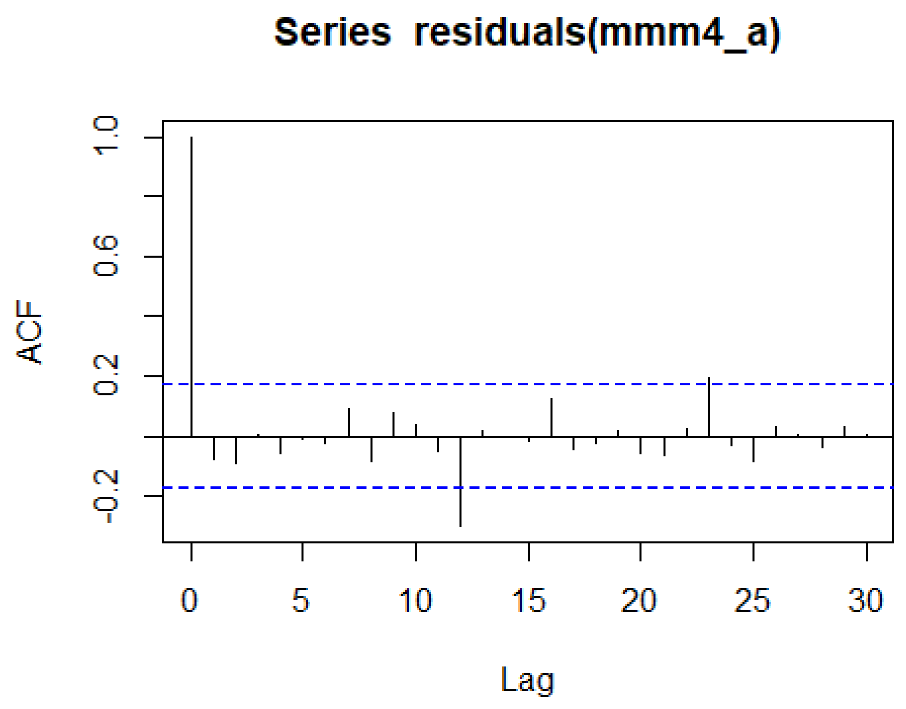
```
par(op)
# no unit root in both parts
```

7.6 Diagnostic Check for the Model mmm4_a

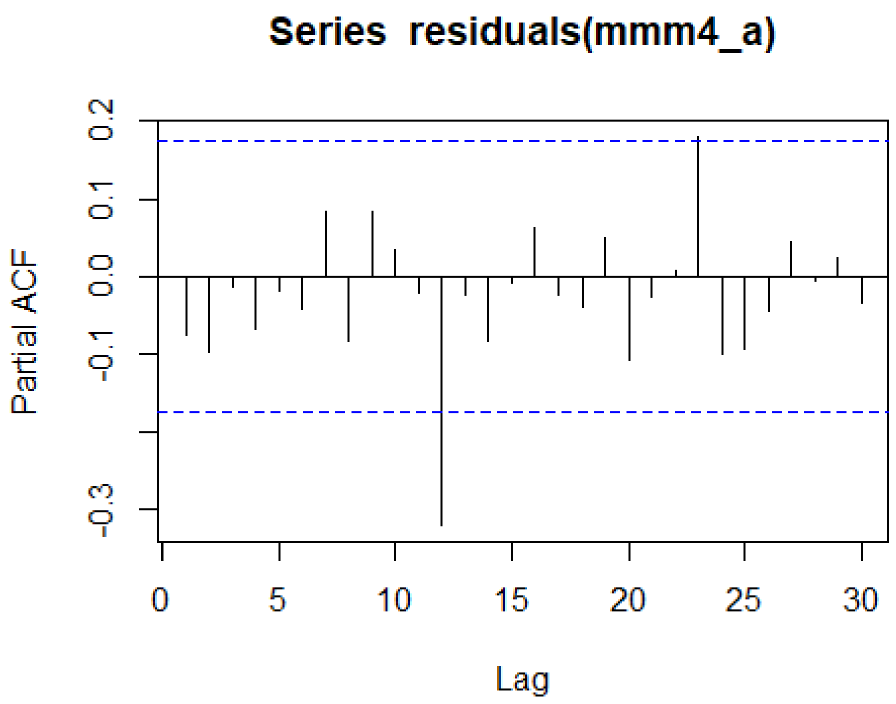
```
# plot the residuals
plot(residuals(mmm4_a), main = "residual plot of mmm4_a")
```



```
# acf and pacf of the residuals
acf(residuals(mmm4_a), lag.max = 30)
```



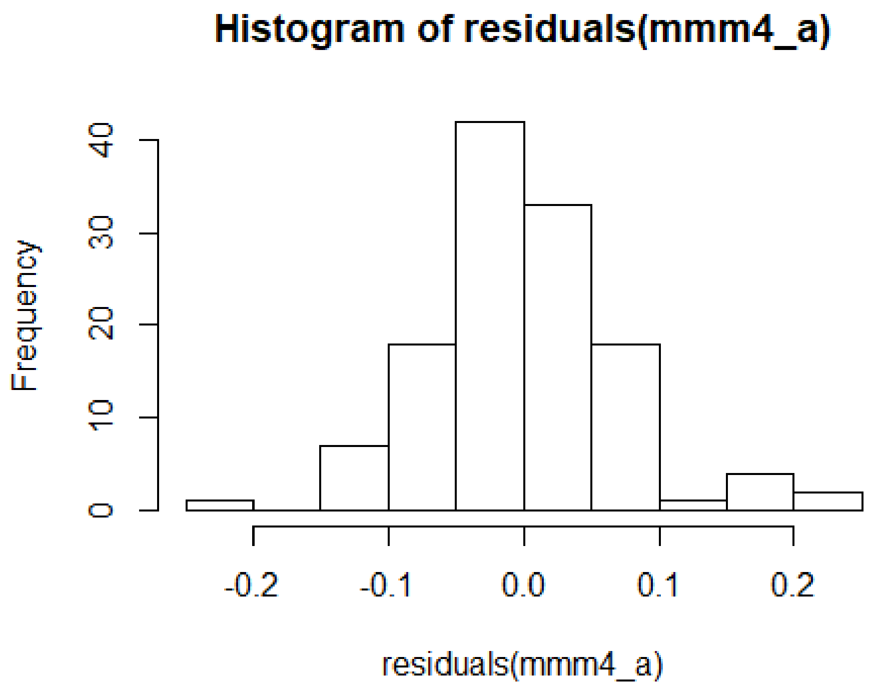
```
pacf(residuals(mmm4_a), lag.max = 30)
```



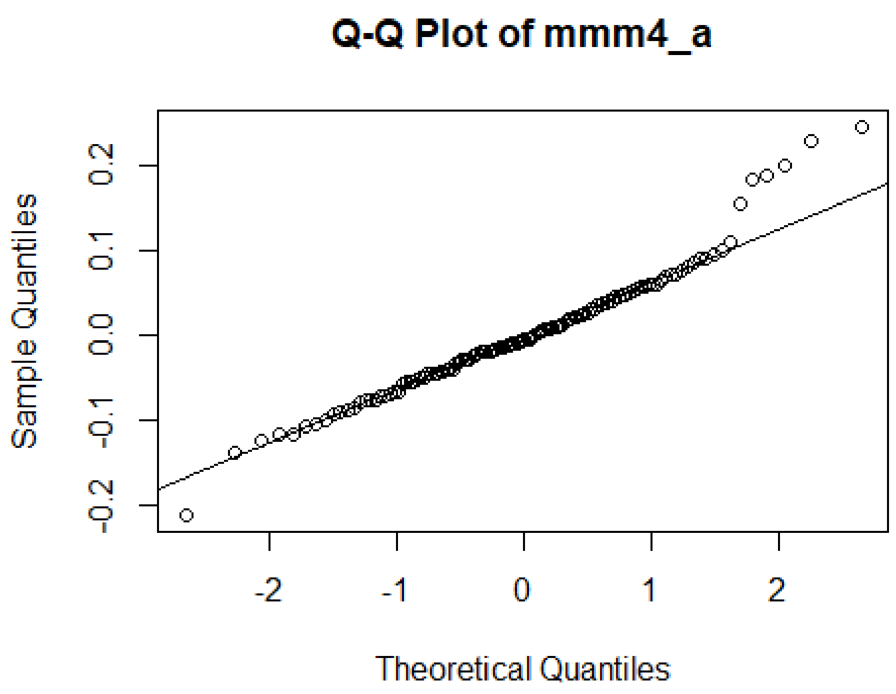
```
# Box-Pierce test for independency
Box.test(residuals(mmm4_a), lag = 11, type = "Box-Pierce", fitdf = 3)

##
##  Box-Pierce test
##
## data:  residuals(mmm4_a)
## X-squared = 5.5478, df = 8, p-value = 0.6977

# Ljung-Box test for independency
Box.test(residuals(mmm4_a), lag = 11, type = "Ljung-Box", fitdf = 3)

##
##  Box-Ljung test
##
## data:  residuals(mmm4_a)
## X-squared = 5.9128, df = 8, p-value = 0.657

# McLeod_Li test to see if squared residuals are correlated
Box.test((residuals(mmm4_a))^2, lag = 11, type = "Ljung-Box", fitdf = 0)

##
##  Box-Ljung test
##
## data:  (residuals(mmm4_a))^2
## X-squared = 4.8949, df = 11, p-value = 0.9361

# use the ar() function to see if residuals are white noise
ar(residuals(mmm4_a))

##
## Call:
```

```
## ar(x = residuals(mmm4_a))
##
##
## Order selected 0  sigma^2 estimated as  0.005161
```

```
# draw the histogram of the residuals
hist(residuals(mmm4_a))
```



```
# QQ plot of the residuals
qqnorm(residuals(mmm4_a), main = "Q-Q Plot of mmm4_a")
qqline(residuals(mmm4_a))
```

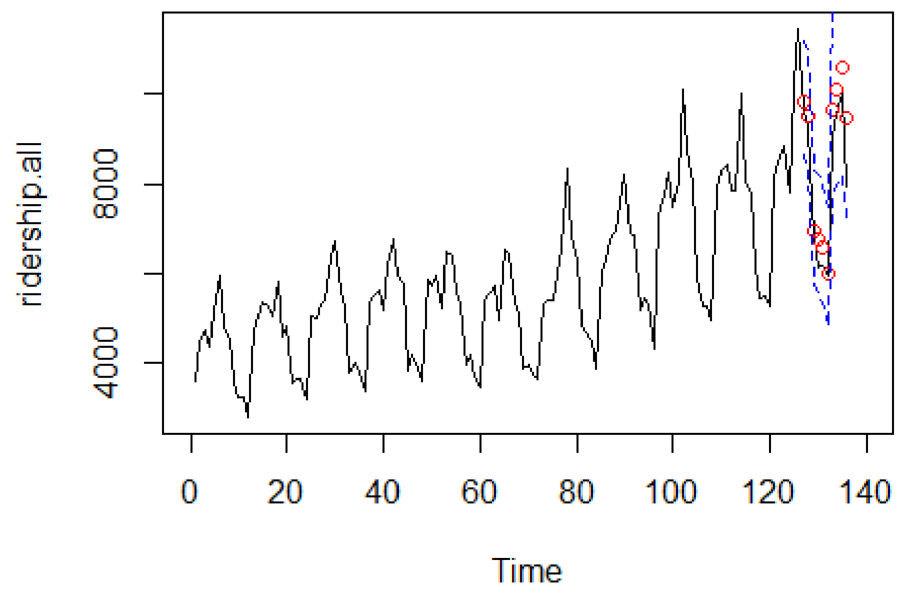


```
# Shapiro-Wilk test for normality
shapiro.test(residuals(mmm4_a))

##
##  Shapiro-Wilk normality test
##
## data:  residuals(mmm4_a)
## W = 0.96123, p-value = 0.001144
```

8. Forecasting

8.1 Forecasting Using the Model mm10_b

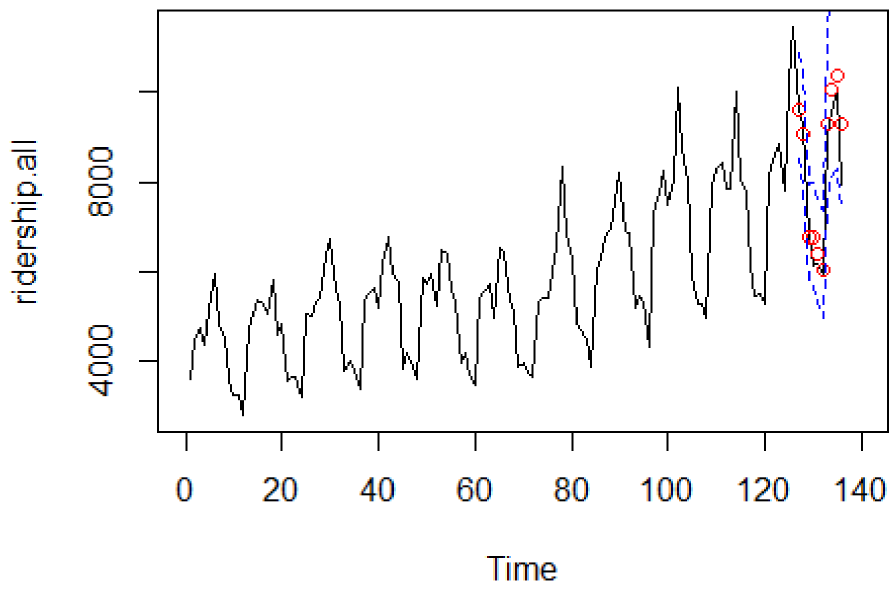
```
# define e to be the base of natural log
e <- exp(1)

# predicited values in the next 10 lags
predmm10_b <- predict(mm10_b, n.ahead = 10)

# plot the time series with the test set
ts.plot(ridership.all, xlim =c(0, 140))

# predicted values
points(127:136, e^(predmm10_b$pred), col = "red")

# 95% CI
lines(127:136, e^(predmm10_b$pred+1.96*predmm10_b$se), lty=2, col = "blue"
)
lines(127:136, e^(predmm10_b$pred-1.96*predmm10_b$se), lty=2, col = "blue"
)
```



```
# get the mse of mm8_a
mse <- sum((ridership.test - e^(predmm10_b$pred))^2)*(1/10)
mse

## [1] 406545.4
```

8.2 Forecasting Using the Model mm12

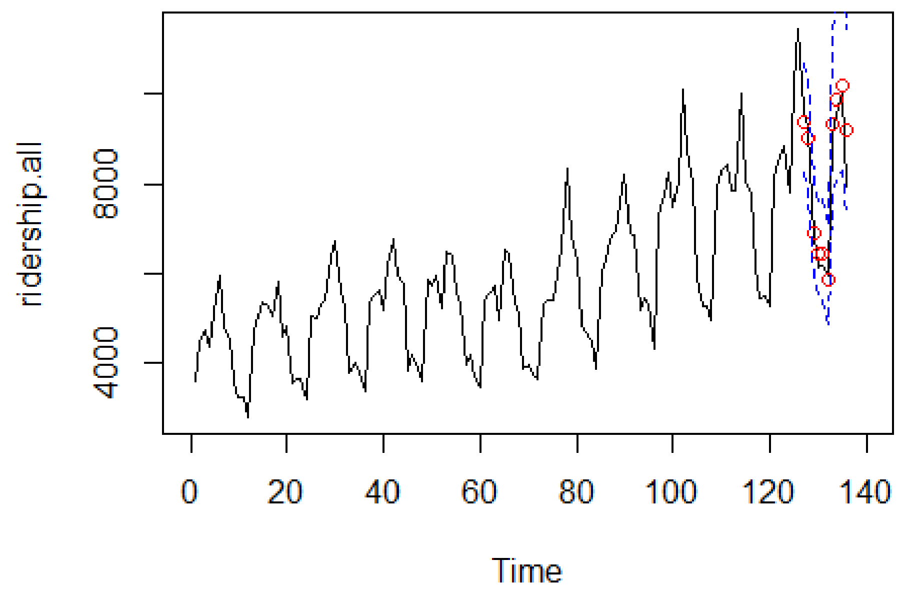
```
# predicited values in the next 10 lags
predmm12<- predict(mm12, n.ahead = 10)

# plot the time series with the test set
ts.plot(ridership.all, xlim =c(0, 140))

# predicted values
points(127:136, e^(predmm12$pred), col = "red")

# 95% CI
lines(127:136, e^(predmm12$pred+1.96*predmm12$se), lty=2, col = "blue")
lines(127:136, e^(predmm12$pred-1.96*predmm12$se), lty=2, col = "blue")
```



```
# get the mse of mm8_a
mse <- sum((ridership.test - e^(predmm12$pred))^2)*(1/10)
mse

## [1] 278880.3
```

8.3 Forecasting Using the Model mm8_a

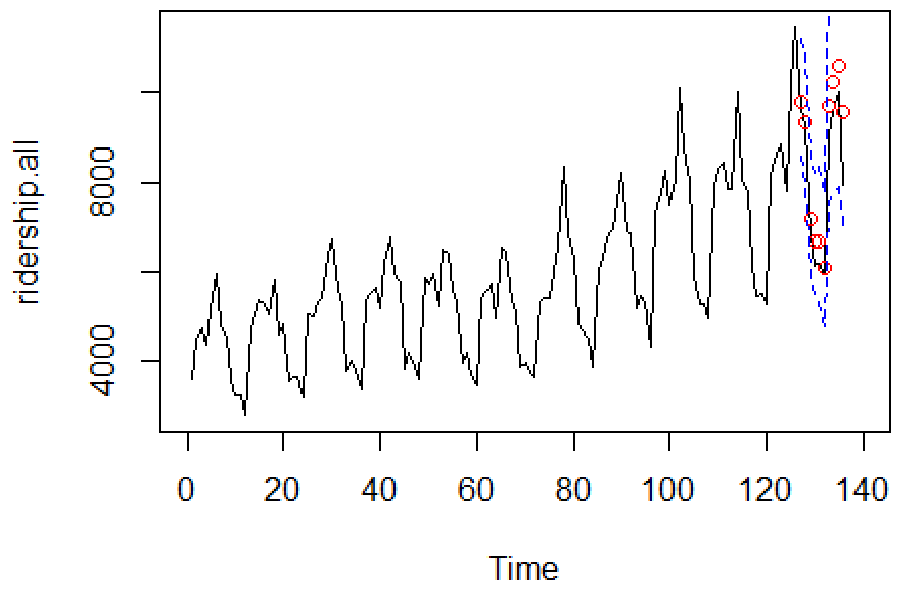
```
# predicted values in the next 10 lags
predmm8_a <- predict(mm8_a, n.ahead = 10)

# plot the time series with the test set
ts.plot(ridership.all, xlim =c(0, 140))

# predicted values
points(127:136, e^(predmm8_a$pred), col = "red")

# 95% CI
lines(127:136, e^(predmm8_a$pred+1.96*predmm8_a$se), lty=2, col = "blue")
lines(127:136, e^(predmm8_a$pred-1.96*predmm8_a$se), lty=2, col = "blue")
```



```
# get the mse of mm8_a
mse <- sum((ridership.test - e^(predmm8_a$pred))^2)*(1/10)
mse

## [1] 217243.8
```

8.4 Forecasting Using the Model mm6

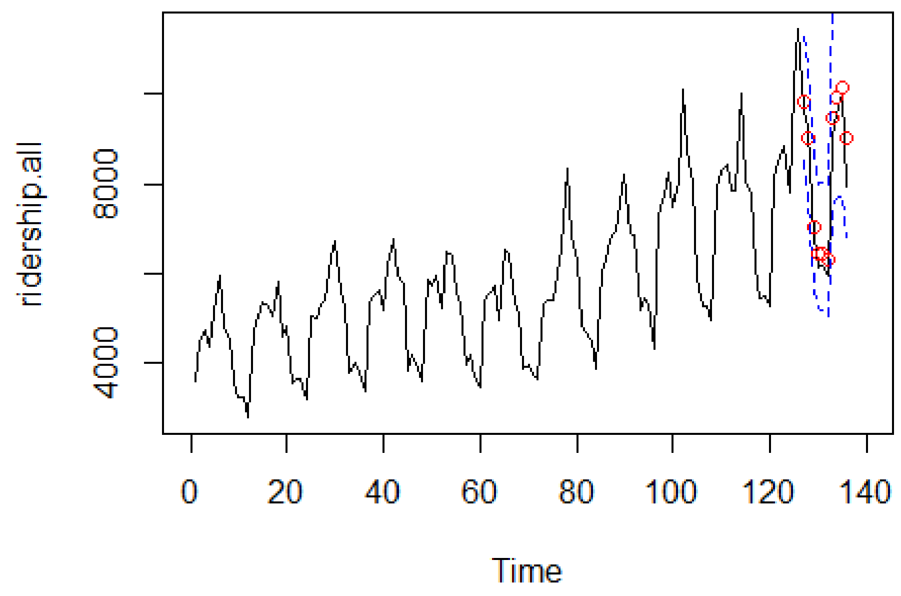
```
# predicted values in the next 10 lags
predmm6 <- predict(mm6, n.ahead = 10)

# plot the time series with the test set
ts.plot(ridership.all, xlim =c(0, 140))

# predicted values
points(127:136, e^(predmm6$pred), col = "red")

# 95% CI
lines(127:136, e^(predmm6$pred+1.96*predmm6$se), lty=2, col = "blue")
lines(127:136, e^(predmm6$pred-1.96*predmm6$se), lty=2, col = "blue")
```



```
# get the mse of mm8_a
mse <- sum((ridership.test - e^(predmm6$pred))^2)*(1/10)
mse

## [1] 460584.4
```

8.5 Forecasting Using the Model mmm4_a

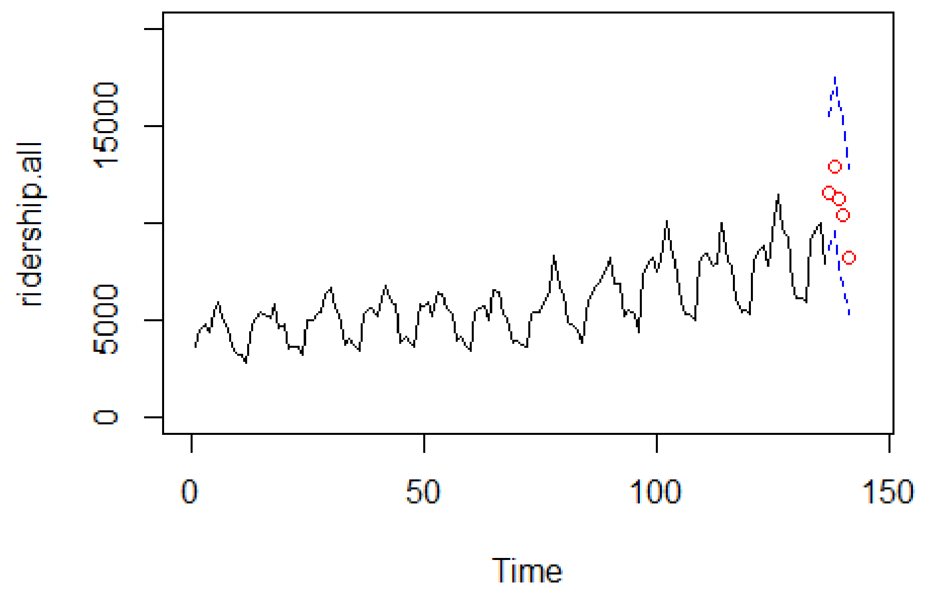
```
# predicted values in the next 10 lags
predmmm4_a <- predict(mmm4_a, n.ahead = 10,
                      newxreg = ((len+1):(len+10)))

# plot the time series with the test set
ts.plot(ridership.all, xlim =c(0, 140))

# predicted values
points(127:136, e^(predmmm4_a$pred), col = "red")

# 95% CI
lines(127:136, e^(predmmm4_a$pred+1.96*predmmm4_a$se), lty=2, col = "blue"
)
lines(127:136, e^(predmmm4_a$pred-1.96*predmmm4_a$se), lty=2, col = "blue"
)
```



```
# get the mse of mm8_a
mse <- sum((ridership.test - e^(predmmm4_a$pred))^2)*(1/10)
mse

## [1] 202057.8
```

8.6 Forecasting Jan 1983 - May 1983 Value Using mmm4_a

```
# predicited values in the next 15 lags
predmmm4_a <- predict(mmm4_a, n.ahead = 15,
                      newxreg = ((len+1):(len+15)))

# plot the time series with the test set
ts.plot(ridership.all, xlim =c(0, 145), ylim = c(0, 20000))

# predicted values
points(137:141, e^(predmmm4_a$pred[11:15]), col = "red")

# 95% CI
lines(137:141, e^(predmmm4_a$pred[11:15]+1.96*predmmm4_a$se[11:15]), lty=2
, col = "blue")
lines(137:141, e^(predmmm4_a$pred[11:15]-1.96*predmmm4_a$se[11:15]), lty=2
, col = "blue")
```



```
# specific value of the ridership from Jan 1983 to May 1983
pred.ridership <- e^(predmmm4_a$pred[11:15])
pred.ridership

## [1] 11597.089 12941.545 11209.670 10372.051  8228.208

# the monthly ridership from Jan to May in 1982
old.ridership <- ridership.all[125:129]

# the difference
pred.ridership - old.ridership

## [1] 1418.089 1481.545 1568.670 1129.051 1404.208

# increasing over 1000 people a month.
```