Journal of Progress in Engineering and Physical Science
ISSN 2709-4006
www.pioneerpublisher.com/jpeps
Volume 2 Number 3 September 2023

Analyzing and Predicting Temperature Data in Baoji: A SARIMA Model Approach

Xinyue Niu[1]

[1] Ocean University of China, China
Correspondence: Xinyue Niu, Ocean University of China, China.

doi:10.56397/JPEPS.2023.09.08

Abstract

In this study, I analyzed and predicted the temperature data of Baoji. First, I processed the data with outliers and explored the seasonal variation pattern of the temperature data through seasonal decomposition. Through the ADF unit root test, I confirm that the temperature data are non-stationary time series.

To make the prediction, I built the SARIMA model and adjusted the model parameters to get the best performance. I tried different autoregressive orders, difference orders, moving average orders, and seasonal periods, and selected the optimal model by comparing the model fitting effect and prediction ability.

Finally, I made a temperature prediction for some time in the future based on the SARIMA model I built. The forecast results show that the temperature shows a gradually increasing trend in the given time range. This conclusion is based on my evaluation of the model and analysis of the predicted results.

Although I have achieved some useful results through the analysis and prediction of temperature data, there are still some limitations to this study. The method of handling outliers in the data processing stage may need further improvement to improve the accuracy of the data. In addition, I only used the SARIMA model for the prediction, and the application of other time series models may yield more accurate results.

In summary, this study predicted the temperature data through time series analysis and SARIMA model and came to the conclusion that the temperature gradually increased. Future research can be carried out from the aspects of improving data processing methods and applying more time series models to improve the accuracy and reliability of temperature prediction.

Keywords: seasonal model, difference, temperature variation, ACF

1. Introduction

Time series analysis has important application value in many fields. By modeling and forecasting the trend, seasonality, and periodicity of data, it can provide a useful reference for decision-making. The purpose of this study is to analyze the time series of the temperature data of a city and to provide support for climate research and related decision-making of the city by predicting future temperature change.

1.1 Data Source and Description

The temperature data used in this study were obtained from historical meteorological records provided by the Bureau of Meteorology. The city studied is Baoji City, and the data cover the daily temperature observation data for nearly 10 years. The data records the maximum and minimum temperatures in the city on a daily basis.

1.2 Data Preprocessing

It is necessary to preprocess the data before time series analysis. First, I calculated the descriptive statistics of the data, including the minimum, maximum, lower quartile, upper quartile, and median, to get an idea of the overall distribution of the data. Second, I checked the data for missing values and outliers. For missing values, I use interpolation to fill in or adopt an appropriate missing value handling strategy. For outliers, I do this by identifying extreme values that are outside the expected range and correcting or excluding them.

In order to better understand the characteristics of the data, I drew various graphs of the data. Includes scatter plots to show the overall distribution and outliers of the temperature data; Histogram to reveal the distribution pattern of temperature data; Box plot to show outliers and central trends of temperature data; And time series charts to observe trends and seasonal changes in temperature data.

Through data preprocessing and visual analysis, I can better understand the characteristics of temperature data and provide a reliable data basis for subsequent time series modeling and prediction.

2. Relevant Theories and Models Are Introduced

Time series analysis is a statistical method of studying a collection of data that changes over time, in which the data is arranged in the order of time. The time series model is a tool for modeling and forecasting such data sets. Common time series models include AR model, MA model, ARMA model, and so on.

AR models (autoregressive models) are those that predict the value of the present moment based on the observed value of the past moment.

MA model (moving average model) refers to the prediction of the value of the present moment based on the error term of the past moment.

The ARMA model (autoregressive moving average model) is a combination of AR model and MA model, which takes into account the observed values and error terms in the past time. The order of the ARMA model (P, Q) represents the number of past observations and error terms considered in the model.

In practical application, determining the type and order of the time series model is a key step. Generally speaking, the type and order of the model can be preliminarily judged by observing the autocorrelation function (ACF) and partial autocorrelation function (PACF). ACF measures the relationship between time series observations and their lag values, and PACF measures the relationship between time series observations and their lag values, eliminating the influence of other lag values.

- The specific steps are as follows:
- Check the stationarity of time series data. If the time series is non-stationary, differential operations are performed until a stationary series is obtained.
- Draw ACF and PACF images. According to the peak and trailing characteristics of ACF and PACF images, the type and order of the model are initially determined.
- If ACF is trailing and PACF is truncated, it may be an MA model.
- If ACF is truncated and PACF is trailed, it may be an AR model.
- If ACF and PACF are trailing, it may be an ARMA model.

The order of the model is determined by information criteria (such as AIC, BIC). AR, MA, or ARMA models of different order are used to calculate the corresponding information criteria, and the model with smaller information criteria values is selected as the best model.

Model diagnosis is performed on the selected model. The selected model can be judged by checking the stationarity of model residuals, autocorrelation graphs, normal distribution, and so on.

3. Empirical Analysis

- Import the necessary packages and read the data set:

First, import the required packages, including

TIDYVERSE, ASTSA, FORECAST, and so on. Then use the READ_CSV function to read the data set and save the data in a data box called BAOJI.

3.1 Statistical Descriptive Statistics

Calculate the MEAN, standard deviation, and descriptive statistics of the data using the SUMMARY, mean, and SD functions, and output the results.

Data interpolation or filling in missing values:

Depending on the code comments, you can choose to use interpolation or the mean to fill in missing values in the data. Corresponding code examples are provided in the comments.

Plot time series charts and seasonal breakdown charts:

PLOT the time series data using the plot function to see the overall trend of the data. The DECOMPOSE function was then used for seasonal decomposition and the PLOT function was used to plot seasonal subsequences.

3.2 Outlier Handling

Use the BOXPLOT function to plot a boxplot of the data to detect the presence of outliers. Based on the code in the comment, you can compute a threshold range for outliers, use the SUBSET function to remove outliers from the dataset, and then use the BOXPLOT function to draw a box plot again to verify the processing.

To draw a histogram:

Plot a histogram of the data using the HIST function to observe the distribution of the data.

Stationarity analysis:

The unit root test is performed on the data using the ADF. TEST function to determine the stationarity of the data.

3.3 Differential Operation

Differential manipulation of the data using the DIFF function to eliminate trends or seasonality in the data. The number of differences can be judged by observing the data graph and autocorrelation graph after the difference.

Autocorrelation and partial autocorrelation analysis:

The ACF2 function is used to draw the autocorrelation and partial autocorrelation of the data after the difference.

3.4 Model Fitting and Prediction

Use the SARIMA function to fit the ARIMA model and select the appropriate parameters (P, D, Q, S), then use the SARMI function to make the model prediction and determine the time range of the prediction.

4. Conclusion and Prospect

4.1 Conclusion

1) Data processing: In the data processing stage, we processed the abnormal value of the temperature data, determined the threshold range of the abnormal value through the box plot, and removed the abnormal value from the data.
2) Data characteristics: Seasonal decomposition of temperature data was carried out, and a seasonal subsequence diagram was drawn. In addition, through the ADF unit root test, we confirm that the temperature data are non-stationary time series.
3) Model building: We tried the SARIMA model to predict the temperature data. By adjusting the model parameters, including P (autoregressive order), D (difference order), Q (moving average order), and S (seasonal period), we performed multiple model training and predictions and compared the performance of different models.

4.2 Outlook

1) Data processing improvement: In the data processing stage, we used a simple boxplot method to deal with outliers, but this method may not capture all outliers. The next step can be to try using more advanced outlier detection methods to improve the accuracy of data processing.
2) Data characteristics research: Seasonal decomposition of temperature data is an important step, but we can further delve into the degree of influence and change pattern of seasonality to better understand and interpret the characteristics of data.
3) Model improvement and optimization: When building the SARIMA model, we made many attempts and selected the best model. However, we can try to use other time series models such as VAR (vector autoregressive model) or ES (exponential smoothing model) to get more accurate predictions.
4) Result evaluation and feedback: For the established model, we need to evaluate and verify the predicted results. A number of

evaluation measures, such as root mean square error (RMSE) and mean absolute percentage error (MAPE), can be used to assess the accuracy of the model and provide feedback and improvements to the results.

Overall, although the above code provides some useful analysis and prediction results, there is still room for improvement. By improving data processing methods, delving into data characteristics, optimizing model selection, and evaluating forecast results, we can improve the accuracy and reliability of time series analysis and provide better support for future predictions and decisions.

References

Hu J F, Wu B, Ye P, et al. (2019). *Time series analysis and prediction*. Beijing: China Statistics Press.

Ji J X, Shen C. (2018). *Time series analysis method and R language implementation*. Beijing: China Renmin University Press.

Shumway, Robert H. and Stauffer, David S. (2022). *Time Series Analysis — an R-based approach to data analysis*. China Machine Press.

Appendix

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidy
verse 2.0.0 ──
## ✔ dplyr     1.1.2     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.2     ✔ tibble    3.2.1
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.1
## ── Conflicts ──────────────────────────────────────────────────
───── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the □]8;;http://conflicted.r-lib.org/□conflicted package□]8;;□ to force all confli
cts to become errors
```

```
#library(tidymodels)
library(glue)
library(astsa)
#library(imputeTS)
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
##
## 载入程辑包：'forecast'
##
## The following object is masked from 'package:astsa':
##
##     gas
```

```
#library(TSA)
#library(fUnitRoots)
library(tseries)
library(readr)
library(dplyr)
library(forecast)
library(stats)
library(vars)
```

```
## 载入需要的程辑包：MASS
##
## 载入程辑包：'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
##
## 载入需要的程辑包：strucchange
## 载入需要的程辑包：zoo
##
## 载入程辑包：'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## 载入需要的程辑包：sandwich
##
## 载入程辑包：'strucchange'
##
## The following object is masked from 'package:stringr':
##
##     boundary
##
## 载入需要的程辑包：urca
## 载入需要的程辑包：lmtest
```

```
library(MASS)
library(zoo)
```

```
getwd()
```

```
## [1] "C:/Users/牛哈哈/Desktop"
```

```
baoji<-read_csv("C:/Users/牛哈哈/Desktop/宝鸡月平均.csv")
```

```
## Rows: 121 Columns: 2
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (1): date
## dbl (1): air_temp_month
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
baoji
```

```
## # A tibble: 121 × 2
##    date   air_temp_month
##    <chr>           <dbl>
##  1 Jan-12           -3.6
##  2 Feb-12           -1.2
##  3 Mar-12            3.8
##  4 Apr-12            7.8
##  5 May-12           10.5
##  6 Jun-12           16.7
##  7 Jul-12           24.5
##  8 Aug-12           23.7
##  9 Sep-12           19.5
## 10 Oct-12           10.6
## # ℹ 111 more rows
```

```
#计算数据的mean、sd
baoji$air_temp_month
```

```
##   [1] -3.6 -1.2  3.8  7.8 10.5 16.7 24.5 23.7 19.5 10.6  6.7 -2.5 -4.3 -1.2  4.7
##  [16]  7.8 13.6 20.4 24.7 25.4 14.5 10.3 -1.4 -3.1 -5.8 -1.9  4.7 11.3 17.9 20.5
##  [31] 26.1 24.7 15.4  8.3  3.5 -1.5 -2.5 -3.6  4.6  8.5 16.5 21.8 23.4 25.3 20.1
##  [46] 14.2  4.5 -1.2 -3.5 -2.4  5.8 10.4 18.4 22.0 24.3 25.3 19.0 10.7  3.9 -1.9
##  [61] -5.3 -2.3  4.8  9.6 13.0 20.2 22.5 21.6 20.1 13.6  6.7  1.2 -3.6 -1.2  5.7
##  [76]  7.8 14.6 18.7 23.1 22.6 18.6 12.3  4.8  2.3  1.4  2.7  7.9 10.8 19.0 20.1
##  [91] 23.7 23.8 19.4 10.2  6.8  2.1 -1.9  2.7  8.9 13.7 19.3 20.4 25.3 23.7 21.2
## [106] 16.4  9.2  3.6 -2.3  2.5  3.7  9.5 14.7 22.4 23.7 26.4 18.2  8.5 10.5  3.1
## [121]  1.7
```

```
summary(baoji$air_temp_month)#描述性统计分析
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    -5.8     3.1    10.4    10.9    20.1    26.4
```

```
mean(baoji$air_temp_month)
```

```
## [1] 10.90165
```

```
sd(baoji$air_temp_month)
```

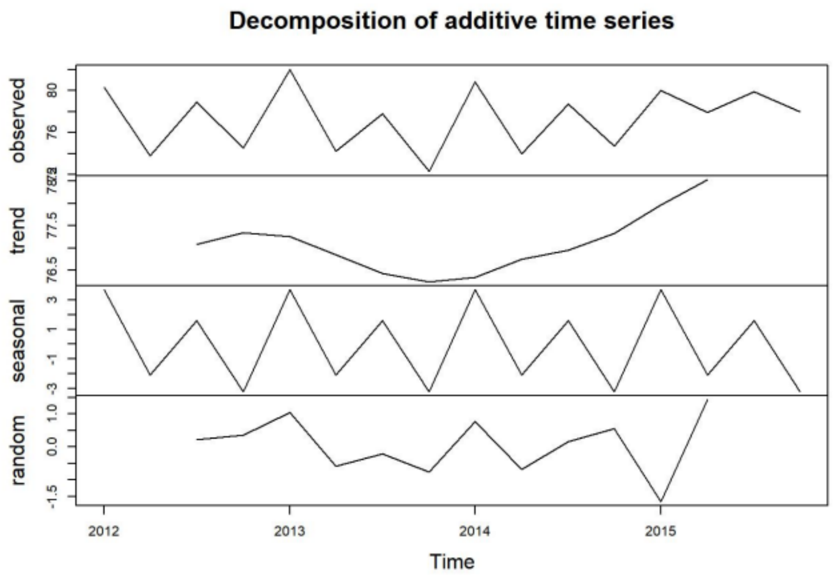
```
## [1] 9.531693
```

```
max(baoji$air_temp_month)
```

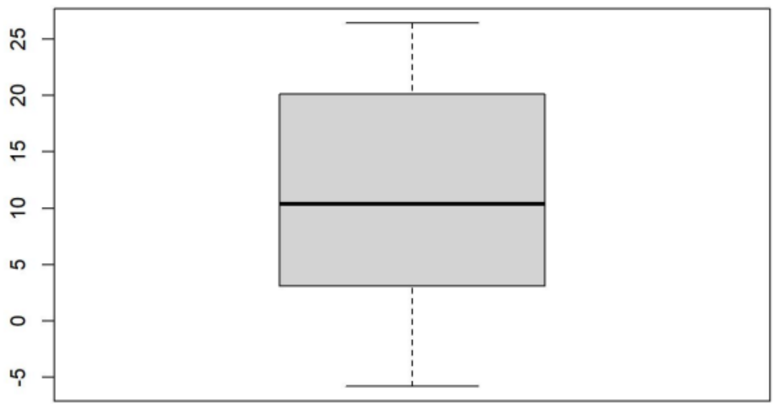
```
## [1] 26.4
```

```
#使用插值法或者均值填补缺失值
#xz$air_temp_month<-na.interpolation(xz$air_temp_month)#插值法填补缺失值
#xz$air_temp_month[is.na(xz$air_temp_month)]<-mean(xz$air_temp_month)均值填补缺失值
```

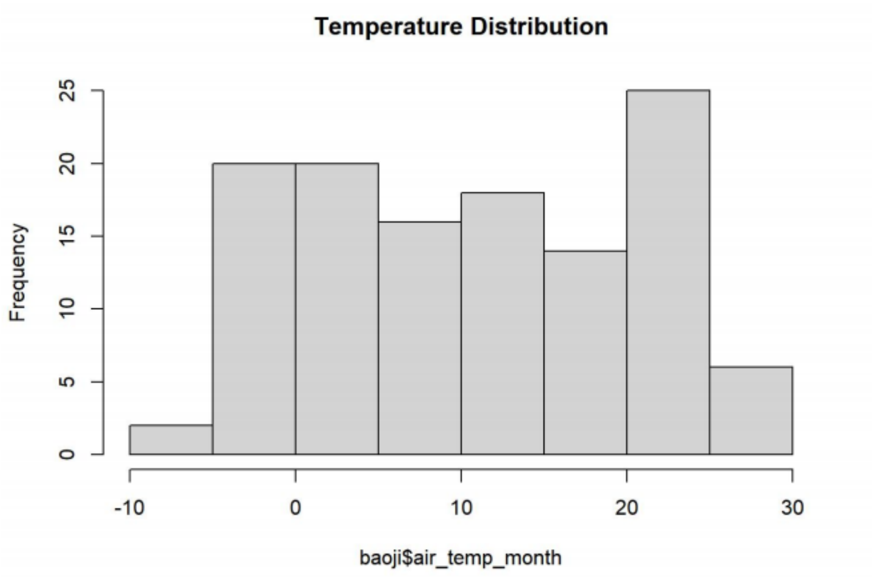
```
x=window(hor,start=2012)
plot(decompose(x))
```



```
#异常值处理
boxplot(baoji$air_temp_month,main="")
#确定异常值的阈值范围，处理
#距离设置为 1.5 倍的四分位差（IQR）。根据 IQR，下限的阈值为 Q1 - 1.5 x IQR，上限的阈值为 Q3 +
 1.5 x IQR。其中，Q1 和 Q3 分别为数据的下四分位数和上四分位数，IQR 为四分位距
Q1<-quantile(baoji$air_temp_month,0.25,na.rm = TRUE)
Q2<-quantile(baoji$air_temp_month,0.75,na.rm = TRUE)
#计算分位数时存在缺失值或NaN（不是数字）
IQR=Q2-Q1
upper<-Q2+1.25*IQR
lower<-Q1-1.25*IQR
xz<-subset(baoji,baoji$air_temp_month>=lower&baoji$air_temp_month<=upper)
boxplot(baoji$air_temp_month,main="")
```



```
hist(baoji$air_temp_month, breaks = "Sturges", main = "Temperature Distribution")
```

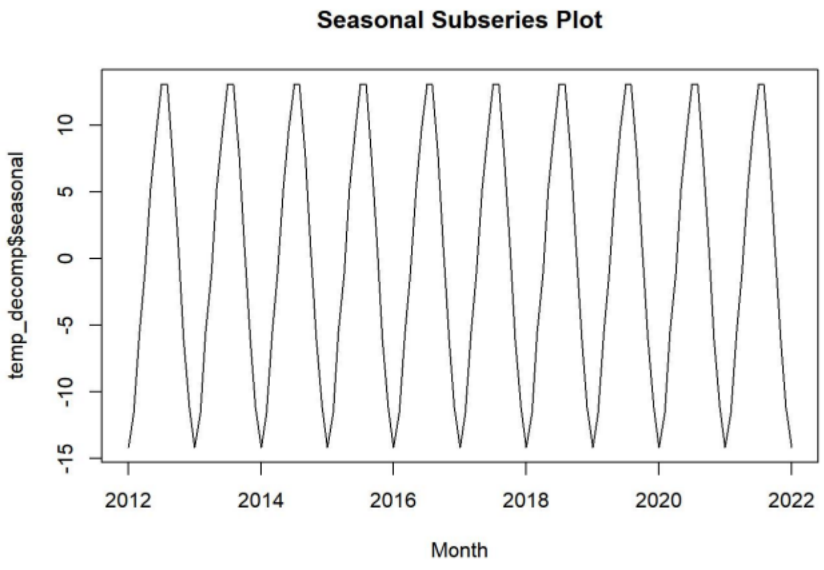


```
#, breaks = "Sturges"确定直方图数量
```

```
# 将温度数据转换为时间序列对象
baoji$air_temp_month_ts <- ts(baoji$air_temp_month, start = c(2012, 1), frequency = 12)

# 进行季节性分解
temp_decomp <- decompose(baoji$air_temp_month_ts)

# 绘制季节性子序列图
plot(temp_decomp$seasonal, main = "Seasonal Subseries Plot", xlab = "Month")
```

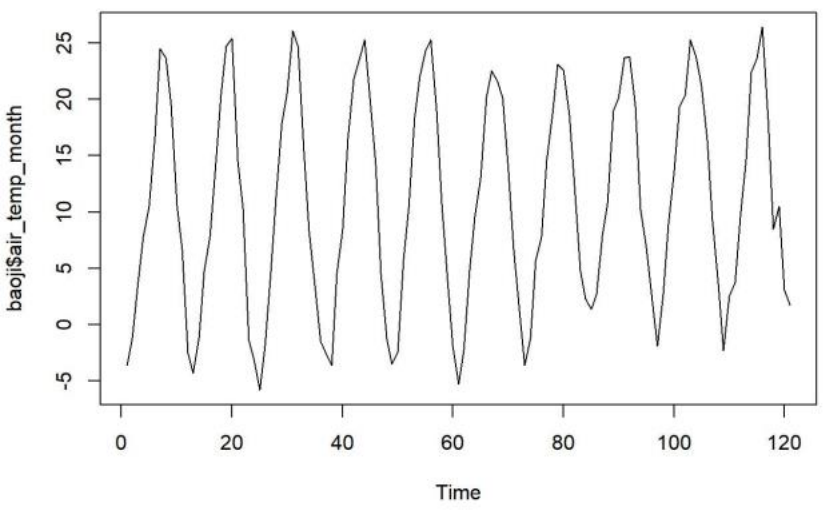


```
library(tseries)
adf.test(baoji$air_temp_month)
```

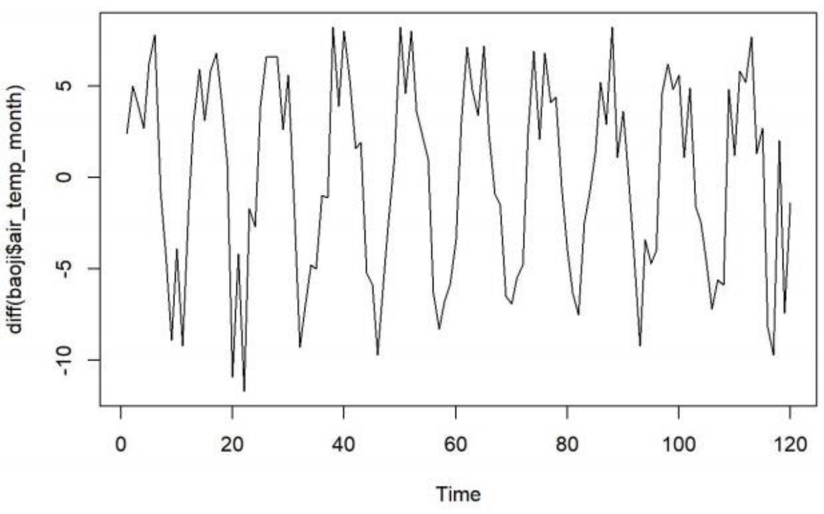
```
## Warning in adf.test(baoji$air_temp_month): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  baoji$air_temp_month
## Dickey-Fuller = -11.774, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

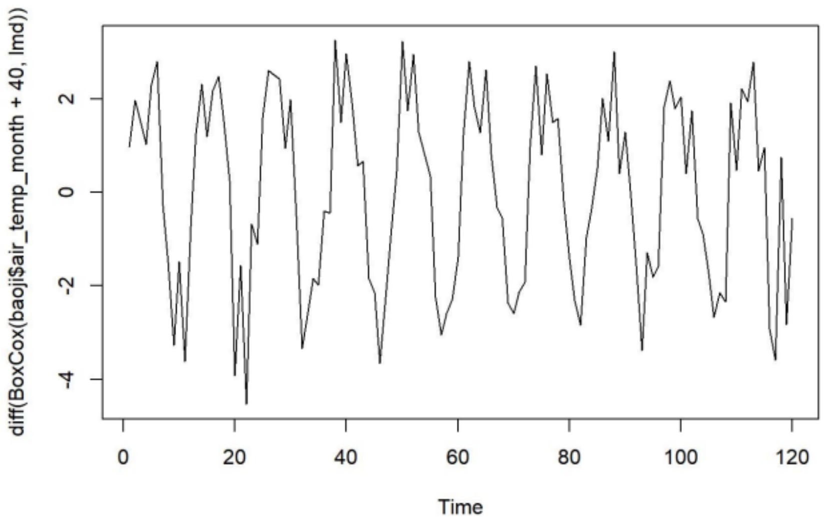
```
plot.ts(baoji$air_temp_month)
```



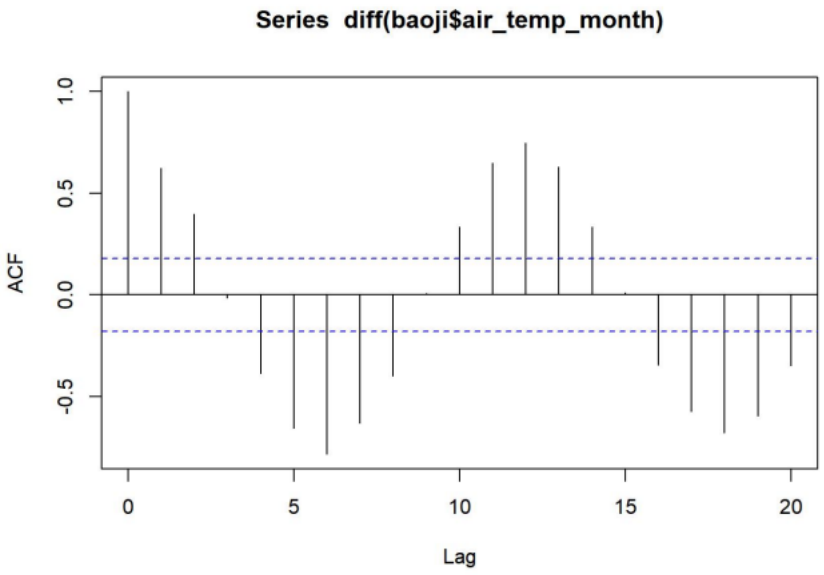
```
plot.ts(diff(baoji$air_temp_month))
```



```
lmd <- BoxCox.lambda(baoji$air_temp_month+40, method = "loglik")
plot.ts(diff(BoxCox(baoji$air_temp_month+40, lmd)))
```

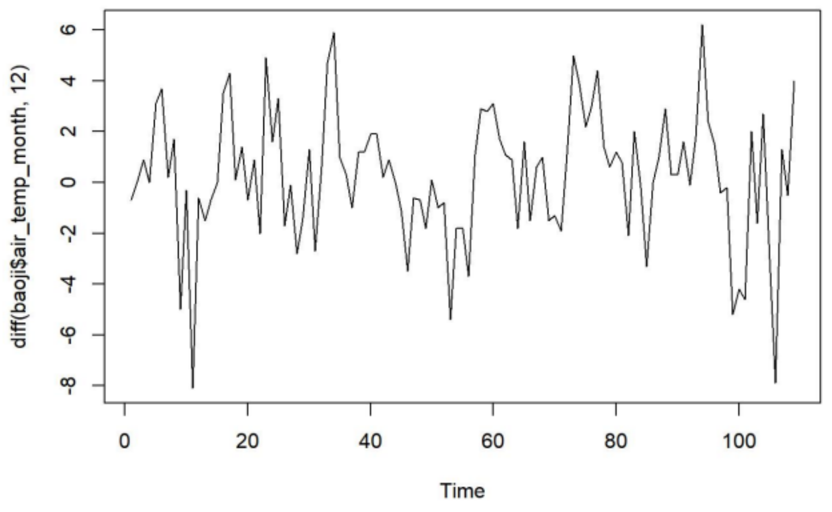


```
acof <- acf(diff(baoji$air_temp_month))
```

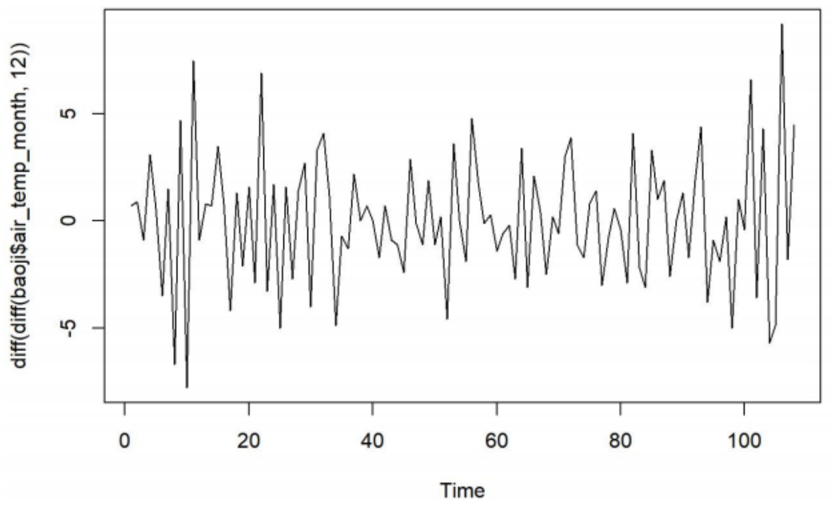


```
#eacf(diff(baoji$air_temp_month))
```

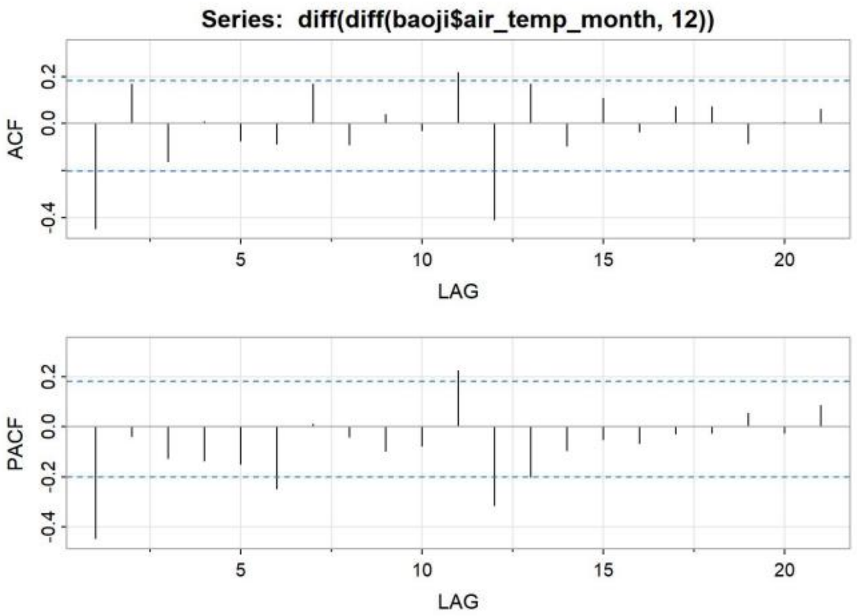
```
plot.ts(diff(baoji$air_temp_month,12))
```



```
plot.ts(diff(diff(baoji$air_temp_month,12)))
```



```
lmd <- BoxCox.lambda(baoji$air_temp_month+40, method = "loglik")
acf2(diff(diff(baoji$air_temp_month,12)))
```



```
##       [,1]  [,2]  [,3]  [,4]  [,5]  [,6] [,7]  [,8]  [,9] [,10] [,11] [,12]
## ACF  -0.45  0.17 -0.16  0.01 -0.07 -0.09 0.17 -0.09  0.04 -0.03  0.22 -0.41
## PACF -0.45 -0.04 -0.13 -0.13 -0.15 -0.25 0.01 -0.04 -0.10 -0.08  0.23 -0.31
##      [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21]
## ACF   0.17 -0.09  0.11 -0.03  0.07  0.07 -0.08  0.01  0.06
## PACF -0.20 -0.10 -0.05 -0.07 -0.03 -0.03  0.05 -0.02  0.09
```

```
#eacf(diff(diff(baoji$air_temp_month,12)))
```

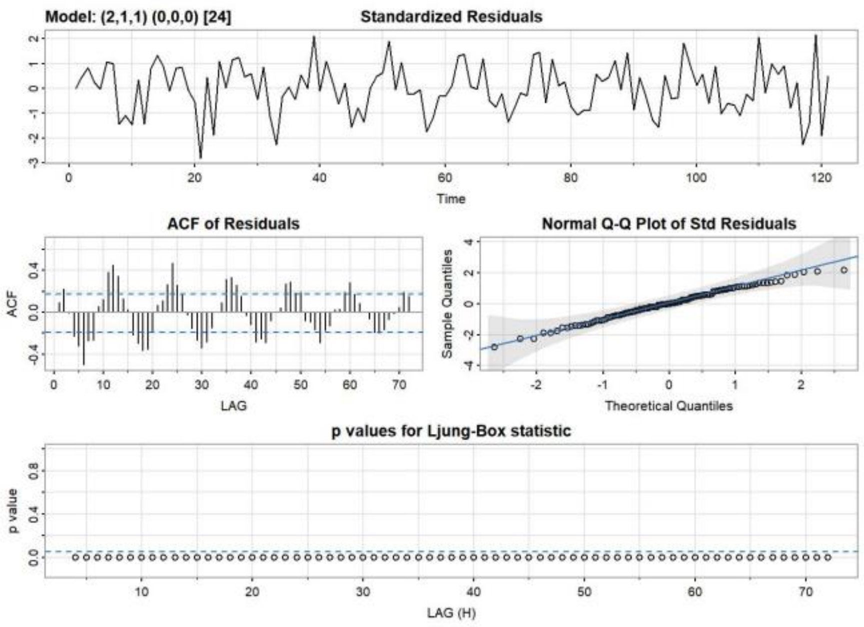
```
library(forecast)
adf.test(diff(diff(baoji$air_temp_month,12)))
```

```
## Warning in adf.test(diff(diff(baoji$air_temp_month, 12))): p-value smaller than
## printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(diff(baoji$air_temp_month, 12))
## Dickey-Fuller = -6.5772, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

```
library(forecast)
sarima(baoji$air_temp_month, p=2, d=1, q=1, S=24)
```

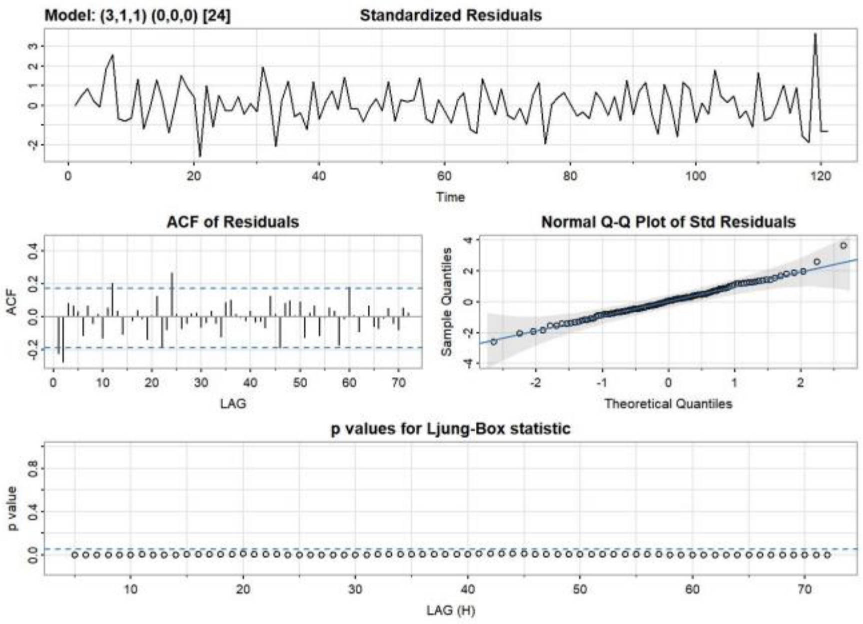
```
## initial  value 1.676580
## iter   2 value 1.662969
## iter   3 value 1.467417
## iter   4 value 1.443555
## iter   5 value 1.424883
## iter   6 value 1.424381
## iter   7 value 1.424199
## iter   8 value 1.422976
## iter   9 value 1.422629
## iter  10 value 1.422527
## iter  11 value 1.422515
## iter  12 value 1.422489
## iter  13 value 1.422466
## iter  14 value 1.422455
## iter  15 value 1.422453
## iter  16 value 1.422452
## iter  17 value 1.422450
## iter  18 value 1.422449
## iter  19 value 1.422449
## iter  19 value 1.422449
## final  value 1.422449
## converged
## initial  value 1.418308
## iter   2 value 1.418274
## iter   3 value 1.418220
## iter   4 value 1.418193
## iter   5 value 1.418171
## iter   6 value 1.418164
## iter   7 value 1.418151
## iter   8 value 1.418140
## iter   9 value 1.418134
## iter  10 value 1.418133
## iter  10 value 1.418133
## final  value 1.418133
## converged
```



```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = constant, transform.pars = trans, fixed = fixed, optim.control = list(trace = tr
c,
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1     ar2     ma1  constant
##       0.0487  0.4303  0.5035    0.0427
## s.e.  0.2144  0.1402  0.2128    1.0671
##
## sigma^2 estimated as 16.98:  log likelihood = -340.45,  aic = 690.9
##
## $degrees_of_freedom
## [1] 116
##
## $ttable
##          Estimate     SE t.value p.value
## ar1        0.0487 0.2144  0.2269  0.8209
## ar2        0.4303 0.1402  3.0702  0.0027
## ma1        0.5035 0.2128  2.3660  0.0196
## constant   0.0427 1.0671  0.0400  0.9681
##
## $AIC
## [1] 5.757476
##
## $AICc
## [1] 5.760375
##
## $BIC
## [1] 5.873622
```

```
sarima(baoji$air_temp_month,p=3,d=1,q=1,S=24)
```

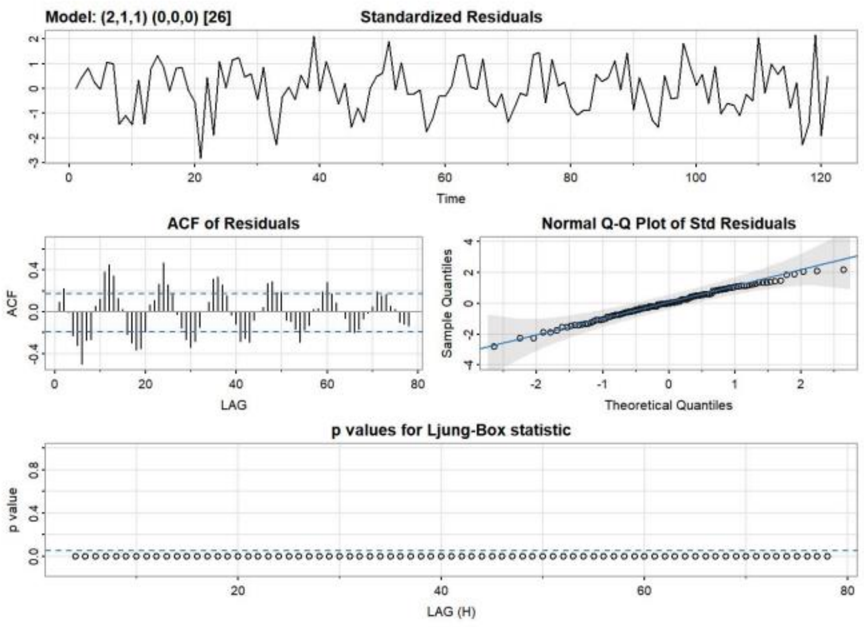
```
## initial  value 1.678511
## iter   2 value 1.660708
## iter   3 value 1.405015
## iter   4 value 1.374399
## iter   5 value 1.355445
## iter   6 value 1.338564
## iter   7 value 1.300116
## iter   8 value 1.265222
## iter   9 value 1.216215
## iter  10 value 1.093061
## iter  11 value 1.066229
## iter  12 value 1.062028
## iter  13 value 1.048628
## iter  14 value 1.046426
## iter  15 value 1.040889
## iter  16 value 1.037253
## iter  17 value 1.030098
## iter  18 value 1.024836
## iter  19 value 1.023750
## iter  20 value 1.023662
## iter  21 value 1.023661
## iter  21 value 1.023661
## iter  21 value 1.023661
## final  value 1.023661
## converged
## initial  value 1.017386
## iter   2 value 1.015206
## iter   3 value 1.011544
## iter   4 value 1.005339
## iter   5 value 1.004146
## iter   6 value 1.003300
## iter   7 value 1.002278
## iter   8 value 1.001875
## iter   9 value 1.001633
## iter  10 value 1.001468
## iter  11 value 1.001391
## iter  12 value 1.001386
## iter  13 value 1.001386
## iter  14 value 1.001386
## iter  14 value 1.001386
## iter  14 value 1.001386
## final  value 1.001386
## converged
```



```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = constant, transform.pars = trans, fixed = fixed, optim.control = list(trace = tr
c,
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1     ar2      ar3      ma1  constant
##       1.0234  0.1295  -0.6003  -1.0000    0.0295
## s.e.  0.0716  0.1174   0.0722   0.0214    0.0158
##
## sigma^2 estimated as 6.964:  log likelihood = -290.44,  aic = 592.88
##
## $degrees_of_freedom
## [1] 115
##
## $ttable
##          Estimate     SE  t.value p.value
## ar1        1.0234 0.0716  14.2913  0.0000
## ar2        0.1295 0.1174   1.1036  0.2721
## ar3       -0.6003 0.0722  -8.3111  0.0000
## ma1       -1.0000 0.0214 -46.8351  0.0000
## constant   0.0295 0.0158   1.8650  0.0647
##
## $AIC
## [1] 4.94065
##
## $AICc
## [1] 4.945036
##
## $BIC
## [1] 5.080024
```

```
sarima(baoji$air_temp_month,p=2,d=1,q=1,S=26)
```

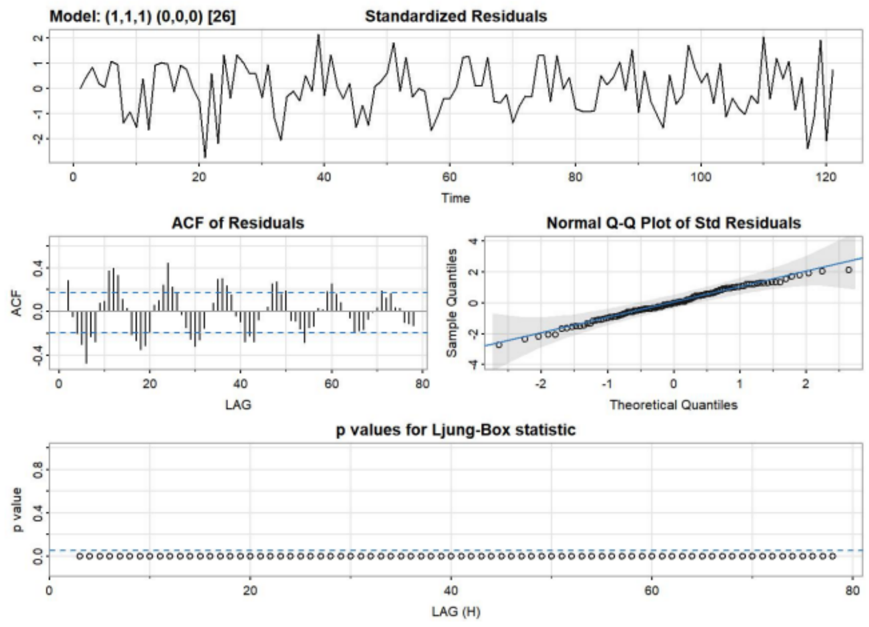
```
## initial  value 1.676580
## iter   2 value 1.662969
## iter   3 value 1.467417
## iter   4 value 1.443555
## iter   5 value 1.424883
## iter   6 value 1.424381
## iter   7 value 1.424199
## iter   8 value 1.422976
## iter   9 value 1.422629
## iter  10 value 1.422527
## iter  11 value 1.422515
## iter  12 value 1.422489
## iter  13 value 1.422466
## iter  14 value 1.422455
## iter  15 value 1.422453
## iter  16 value 1.422452
## iter  17 value 1.422450
## iter  18 value 1.422449
## iter  19 value 1.422449
## iter  19 value 1.422449
## final  value 1.422449
## converged
## initial  value 1.418308
## iter   2 value 1.418274
## iter   3 value 1.418220
## iter   4 value 1.418193
## iter   5 value 1.418171
## iter   6 value 1.418164
## iter   7 value 1.418151
## iter   8 value 1.418140
## iter   9 value 1.418134
## iter  10 value 1.418133
## iter  10 value 1.418133
## final  value 1.418133
## converged
```

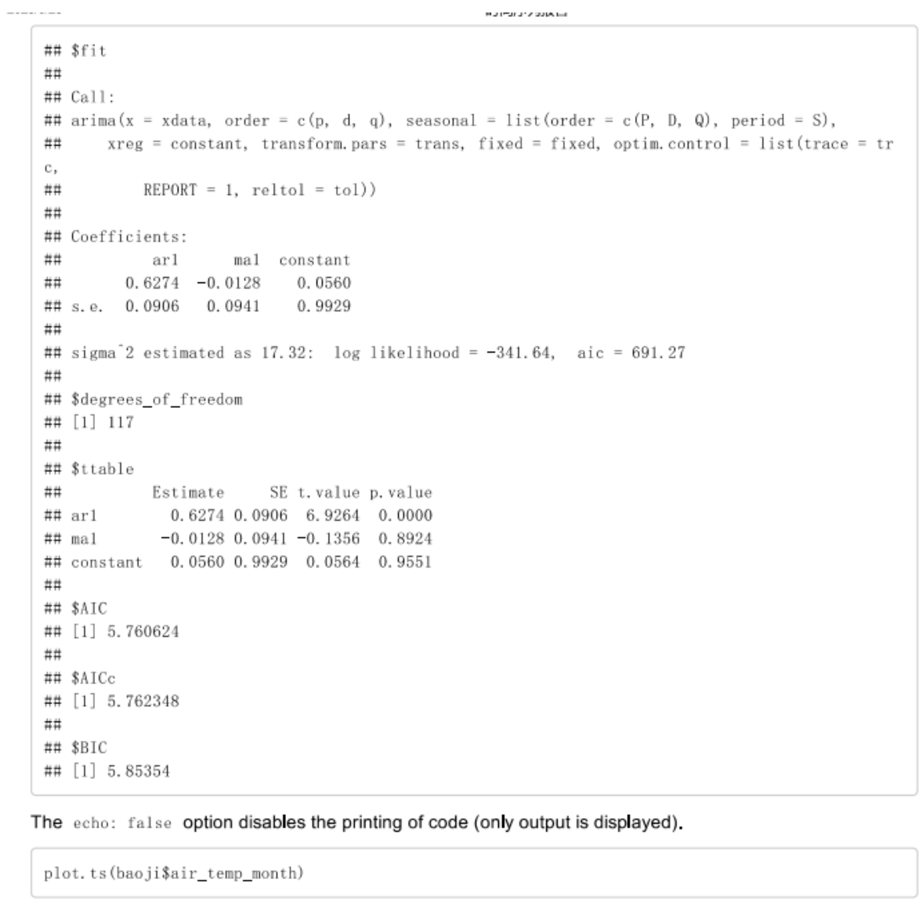


```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = constant, transform.pars = trans, fixed = fixed, optim.control = list(trace = tr
c,
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1     ar2     ma1  constant
##       0.0487  0.4303  0.5035    0.0427
## s.e.  0.2144  0.1402  0.2128    1.0671
##
## sigma^2 estimated as 16.98:  log likelihood = -340.45,  aic = 690.9
##
## $degrees_of_freedom
## [1] 116
##
## $ttable
##          Estimate     SE t.value p.value
## ar1        0.0487 0.2144  0.2269  0.8209
## ar2        0.4303 0.1402  3.0702  0.0027
## ma1        0.5035 0.2128  2.3660  0.0196
## constant   0.0427 1.0671  0.0400  0.9681
##
## $AIC
## [1] 5.757476
##
## $AICc
## [1] 5.760375
##
## $BIC
## [1] 5.873622
```

```
sarima(baoji$air_temp_month,p=1,d=1,q=1,S=26)
```

```
## initial  value 1.675987
## iter   2 value 1.544208
## iter   3 value 1.475972
## iter   4 value 1.440938
## iter   5 value 1.431421
## iter   6 value 1.429623
## iter   7 value 1.429323
## iter   8 value 1.429316
## iter   9 value 1.429314
## iter  10 value 1.429312
## iter  11 value 1.429312
## iter  12 value 1.429312
## iter  12 value 1.429312
## iter  12 value 1.429312
## final  value 1.429312
## converged
## initial  value 1.428090
## iter   2 value 1.428070
## iter   3 value 1.428046
## iter   4 value 1.428041
## iter   5 value 1.428040
## iter   6 value 1.428040
## iter   6 value 1.428040
## iter   6 value 1.428040
## final  value 1.428040
## converged
```

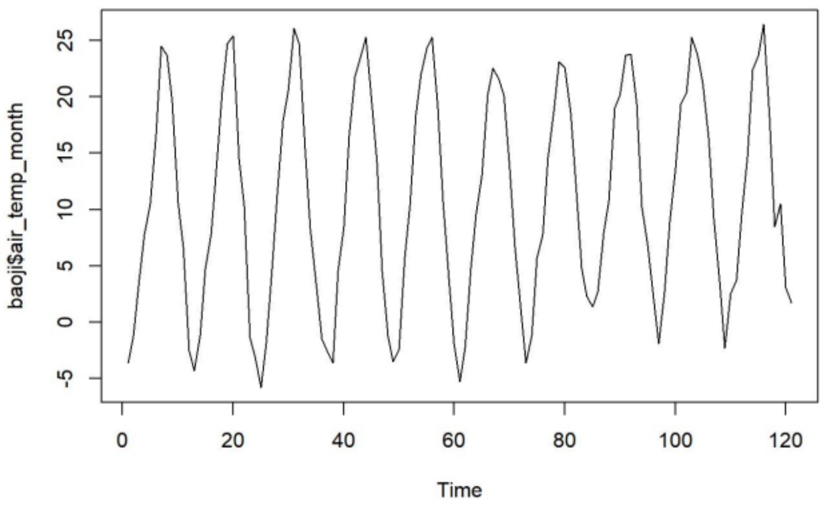




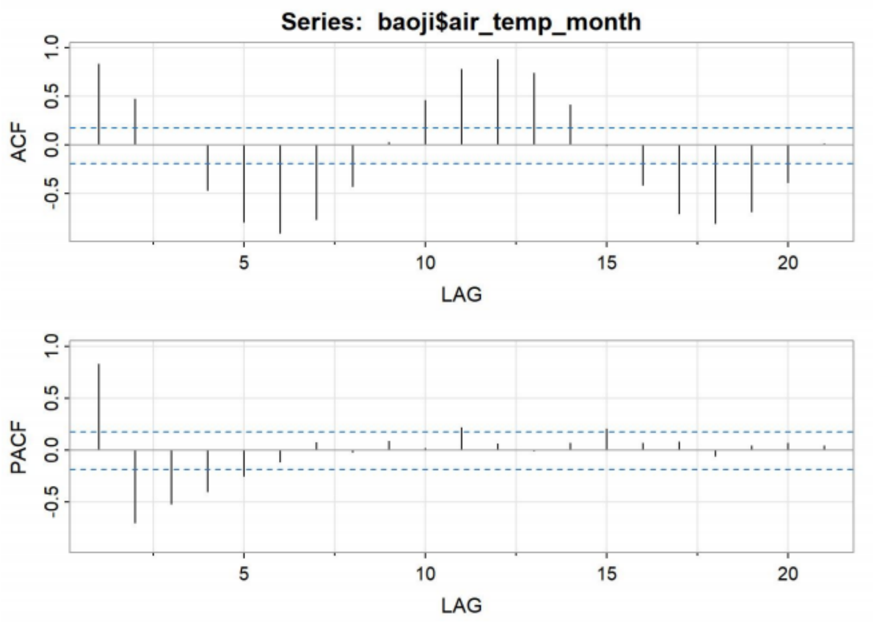
```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = constant, transform.pars = trans, fixed = fixed, optim.control = list(trace = tr
c,
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1      ma1  constant
##       0.6274  -0.0128    0.0560
## s.e.  0.0906   0.0941    0.9929
##
## sigma^2 estimated as 17.32:  log likelihood = -341.64,  aic = 691.27
##
## $degrees_of_freedom
## [1] 117
##
## $ttable
##          Estimate     SE t.value p.value
## ar1        0.6274 0.0906  6.9264  0.0000
## ma1       -0.0128 0.0941 -0.1356  0.8924
## constant   0.0560 0.9929  0.0564  0.9551
##
## $AIC
## [1] 5.760624
##
## $AICc
## [1] 5.762348
##
## $BIC
## [1] 5.85354
```

The `echo: false` option disables the printing of code (only output is displayed).

```
plot.ts(baoji$air_temp_month)
```

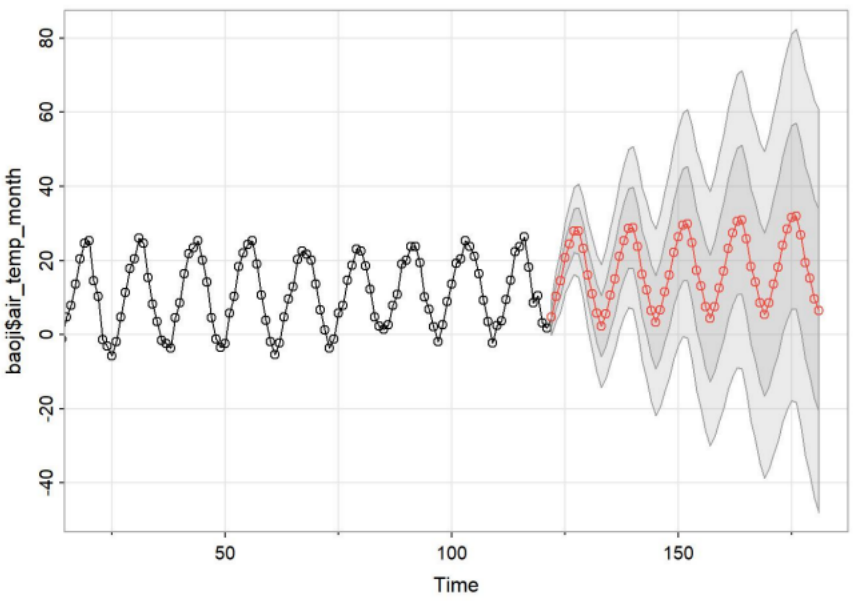


```
acf2(baoji$air_temp_month)
```



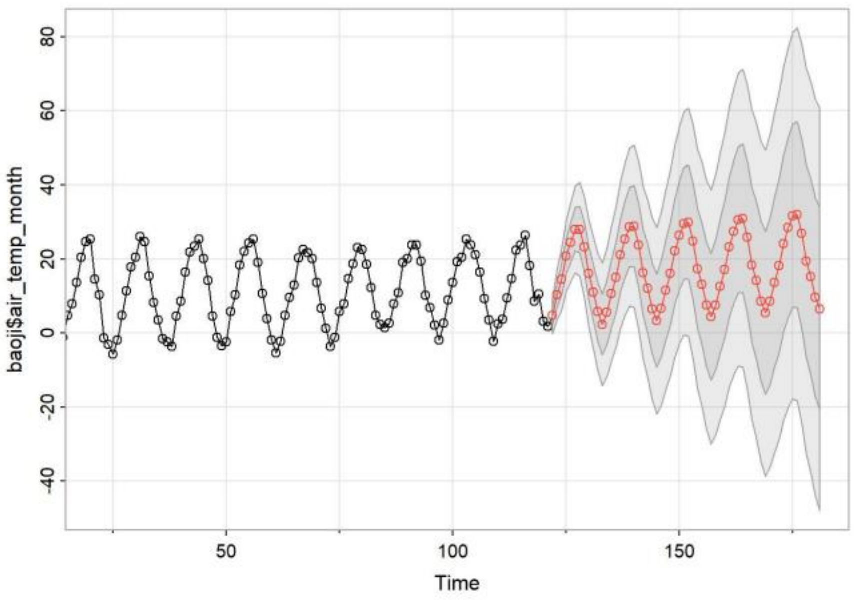
```
##      [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8] [,9] [,10] [,11] [,12]
## ACF  0.83  0.47  0.00 -0.46 -0.79 -0.90 -0.76 -0.42 0.03  0.46  0.77  0.88
## PACF 0.83 -0.71 -0.52 -0.40 -0.25 -0.11  0.08 -0.02 0.09  0.01  0.22  0.06
##      [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21]
## ACF   0.74  0.41   0.0 -0.41 -0.70 -0.80 -0.69 -0.38  0.01
## PACF -0.01  0.07   0.2  0.06  0.08 -0.06  0.04  0.07  0.04
```

```
sarima.for(baoji$air_temp_month,0,1,0,0,1,2,12,n.ahead = 60)
```



```
## $pred
## Time Series:
## Start = 122
## End = 181
## Frequency = 1
##  [1]  4.653746 10.311327 14.482534 20.719185 24.440817 27.997700 27.986926
##  [8] 23.301597 16.157936 10.958084  5.722005  2.294226  5.536148 10.585636
## [15] 14.991771 21.077176 25.281685 28.556430 28.885793 23.764267 16.312131
## [22] 12.072403  6.530742  3.326848  6.568770 11.618258 16.024393 22.109798
## [29] 26.314307 29.589052 29.918415 24.796889 17.344753 13.105025  7.563364
## [36]  4.359470  7.601392 12.650880 17.057015 23.142420 27.346929 30.621674
## [43] 30.951037 25.829511 18.377375 14.137648  8.595986  5.392093  8.634015
## [50] 13.683502 18.089637 24.175043 28.379551 31.654296 31.983660 26.862133
## [57] 19.409997 15.170270  9.628608  6.424715
##
## $se
## Time Series:
## Start = 122
## End = 181
## Frequency = 1
##  [1]  2.386437  3.374931  4.133430  4.772874  5.336235  5.845553  6.313919
##  [8]  6.749863  7.159311  7.546576  7.914916  8.266860  8.708296  9.128410
## [15]  9.530022  9.915380 10.286312 10.644325 10.990683 11.326454 11.652554
## [22] 11.969772 12.278799 12.580236 13.021598 13.448482 13.862227 14.263976
## [29] 14.654715 15.035303 15.406492 15.768946 16.123254 16.469942 16.809481
## [36] 17.142296 17.599501 18.045126 18.480008 18.904890 19.320429 19.727218
## [43] 20.125786 20.516613 20.900132 21.276740 21.646796 22.010632 22.488103
## [50] 22.955645 23.413853 23.863264 24.304367 24.737605 25.163386 25.582081
## [57] 25.994032 26.399556 26.798945 27.192468
```

```
sarima.for(baoji$air_temp_month,0,1,0,0,1,2,12,n.ahead = 60)
```



```
## $pred
## Time Series:
## Start = 122
## End = 181
## Frequency = 1
##  [1]  4.653746 10.311327 14.482534 20.719185 24.440817 27.997700 27.986926
##  [8] 23.301597 16.157936 10.958084  5.722005  2.294226  5.536148 10.585636
## [15] 14.991771 21.077176 25.281685 28.556430 28.885793 23.764267 16.312131
## [22] 12.072403  6.530742  3.326848  6.568770 11.618258 16.024393 22.109798
## [29] 26.314307 29.589052 29.918415 24.796889 17.344753 13.105025  7.563364
## [36]  4.359470  7.601392 12.650880 17.057015 23.142420 27.346929 30.621674
## [43] 30.951037 25.829511 18.377375 14.137648  8.595986  5.392093  8.634015
## [50] 13.683502 18.089637 24.175043 28.379551 31.654296 31.983660 26.862133
## [57] 19.409997 15.170270  9.628608  6.424715
##
## $se
## Time Series:
## Start = 122
## End = 181
## Frequency = 1
##  [1]  2.386437  3.374931  4.133430  4.772874  5.336235  5.845553  6.313919
##  [8]  6.749863  7.159311  7.546576  7.914916  8.266860  8.708296  9.128410
## [15]  9.530022  9.915380 10.286312 10.644325 10.990683 11.326454 11.652554
## [22] 11.969772 12.278799 12.580236 13.021598 13.448482 13.862227 14.263976
## [29] 14.654715 15.035303 15.406492 15.768946 16.123254 16.469942 16.809481
## [36] 17.142296 17.599501 18.045126 18.480008 18.904890 19.320429 19.727218
## [43] 20.125786 20.516613 20.900132 21.276740 21.646796 22.010632 22.488103
## [50] 22.955645 23.413853 23.863264 24.304367 24.737605 25.163386 25.582081
## [57] 25.994032 26.399556 26.798945 27.192468
```