MH4500 Lab 3 Report

Chen Zeyi (U2040386J)

Contents

```r
library(rootSolve)
library(forecast)
```

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

```r
library(FitAR)
```

```
## Loading required package: lattice
```

```
## Loading required package: leaps
```

```
## Loading required package: ltsa
```

```
## Loading required package: bestglm
```

```
##
## Attaching package: 'FitAR'
```

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

```r
library(knitr)
knitr::opts_chunk$set(tidy.opts = list(width.cutoff = 60), tidy = TRUE)
```

1 Data "cow.dat"

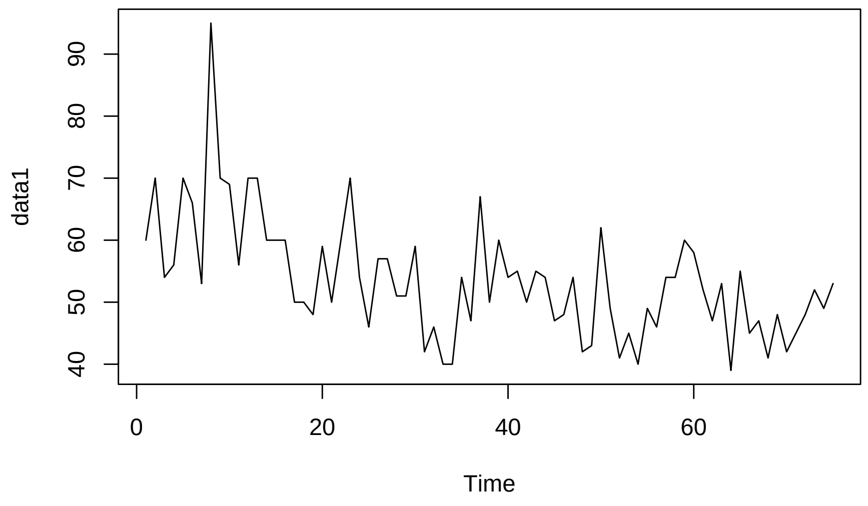
1.1 Model Identification

We import the data in type of numeric and give a basic visualisation, as follows.

```
getwd()
```

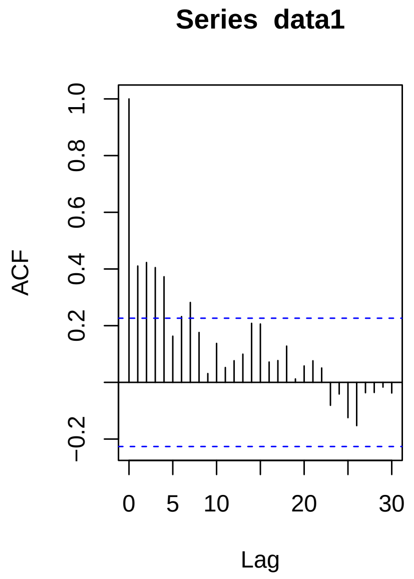
```
## [1] "/Users/zeyichen/Desktop/Courses/Y2S2/Time Series Analysis/Lab/Lab 3"
```

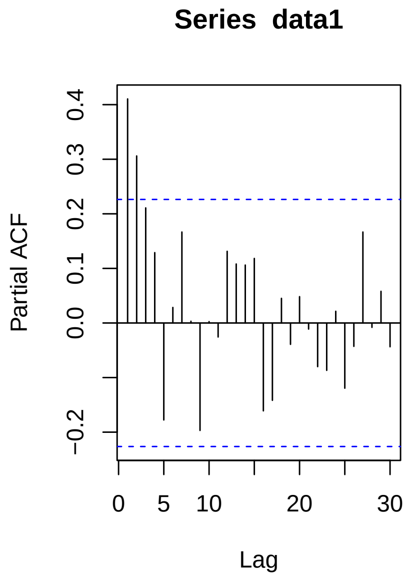
```
setwd("/Users/zeyichen/Desktop/Courses/Y2S2/Time Series Analysis/Lab/Lab 3")
data1 = scan("cow.dat")
plot.ts(data1)
```



To check whether the time series is stationary, we observe cut-offs from ACF and PACF plots.

```
par(mfrow = c(1, 2))
acf(data1, lag.max = 30)
pacf(data1, lag.max = 30)
```





As the ACF and PACF plots cut off rapidly, we may conclude that the process is stationary. It is also plausible that MA(7) and AR(2) are two potential models behind the sample data. Thus, in the following parts, we conduct model fitting and diagnostic checking of the two models successively.

1.2 Parameter Estimation

1.2.1 ARIMA Model Fitting

We use the *arima()* function in R to fit the models.

```
set.seed(22)
fit1_MA = arima(data1, order = c(0, 0, 7))
fit1_AR = arima(data1, order = c(2, 0, 0))
fit1_MA
```

```
##
## Call:
## arima(x = data1, order = c(0, 0, 7))
##
## Coefficients:
##          ma1     ma2     ma3     ma4     ma5     ma6     ma7  intercept
##       0.2041  0.3494  0.1530  0.4361  0.0614  0.0405  0.3797    53.6552
## s.e.  0.1163  0.1114  0.1219  0.1179  0.1267  0.1664  0.1139     2.2413
##
## sigma^2 estimated as 58.08:  log likelihood = -260,  aic = 537.99
```

```
fit1_AR
```

```
##
## Call:
## arima(x = data1, order = c(2, 0, 0))
##
## Coefficients:
##          ar1     ar2  intercept
##       0.2794  0.3183    53.8638
## s.e.  0.1079  0.1100     2.2969
##
## sigma^2 estimated as 67.78:  log likelihood = -264.73,  aic = 537.46
```

Based on the output, the MA(7) fitted model is

$$X_t = 53.6552 + Z_t + 0.2041Z_{t-1} + 0.3494Z_{t-2} + 0.1530Z_{t-3} + 0.4361Z_{t-4} + 0.0614Z_{t-5} + 0.0405Z_{t-6} + 0.3797Z_{t-7}$$

and the AR(2) model is

$$X_t = 21.6694 + 0.2794X_{t-1} + 0.3183X_{t-2} + Z_t, \text{ where } 21.6694 = 53.8638 \times (1 - 0.2794 - 0.3183)$$

1.2.2 Invertibility Checking

One may test the invertibility of the MA(7) model

```
f = function(x) 1 + 0.2041 * x + 0.3494 * x^2 + 0.153 * x^3 +
    0.4361 * x^4 + 0.0614 * x^5 + 0.0405 * x^6 + 0.3797 * x^7
uniroot.all(f, c(-5, 5))
```

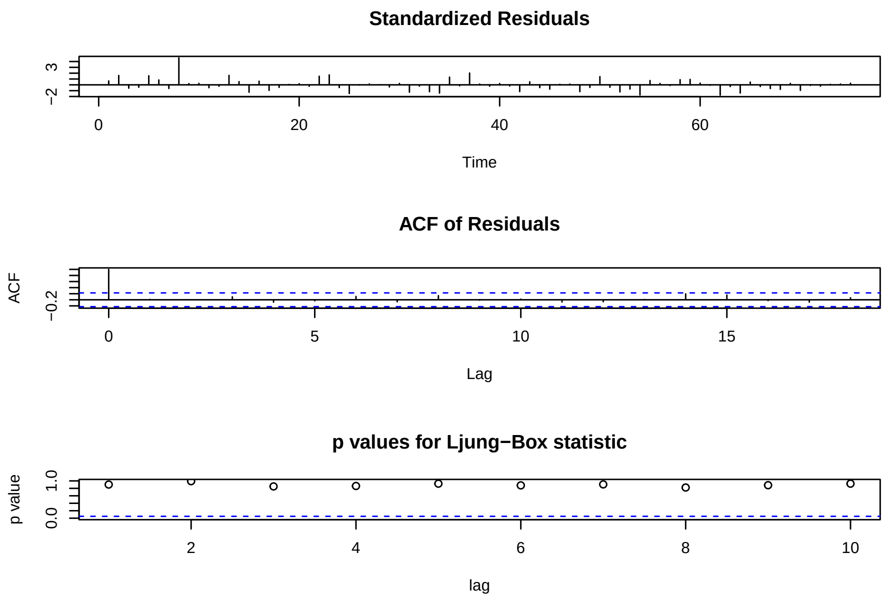
```
## [1] -1.278903
```

As the root lies outside the unit circle, the process is not invertible, whereas we still regard the model as a condidate for later AIC comparison.

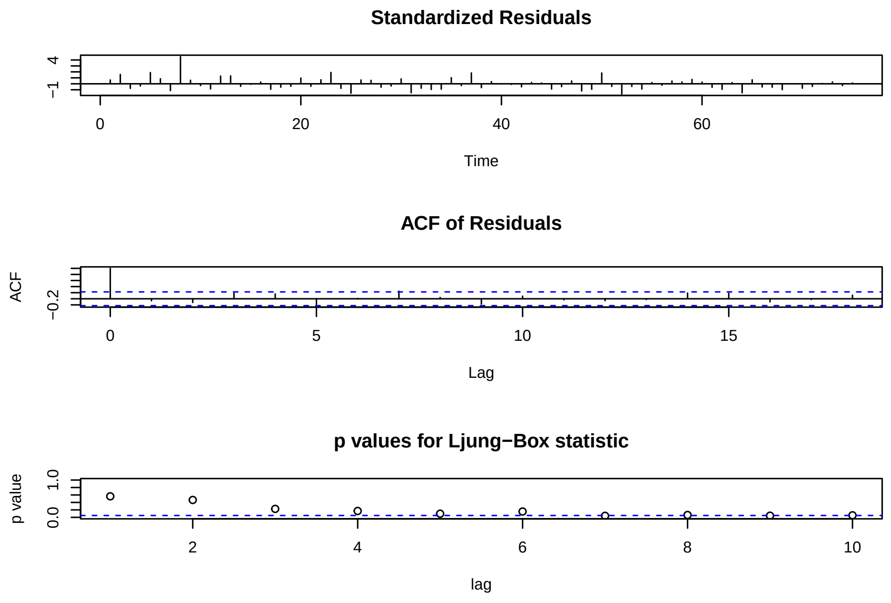
1.3 Diagnostic Checking

We perform diagnostic checking by using *tsdiag()* function in R.

```
par(mfrow = c(1, 2))
tsdiag(fit1_MA)
```



```
tsdiag(fit1_AR)
```



From the plots given, we observe that the scatter points of MA(7) model is high above the dotted blue line, which means the residuals are satisfactory to meet the properties of white noises. However, AR(2) model has several residual points close to or even beneath the dotted blue line, which indicates that there is still space for modification.

To further verify our conclusion above, we may also give the result of Ljung-Box statistics using *Box.test()* function, using lag equal to the 1/5 length of the data.

```
Box.test(fit1_MA$residuals, lag = length(data1)/5)
```

```
##
##  Box-Pierce test
##
## data:  fit1_MA$residuals
## X-squared = 9.3624, df = 15, p-value = 0.8578
```

```
Box.test(fit1_AR$residuals, lag = length(data1)/5)
```

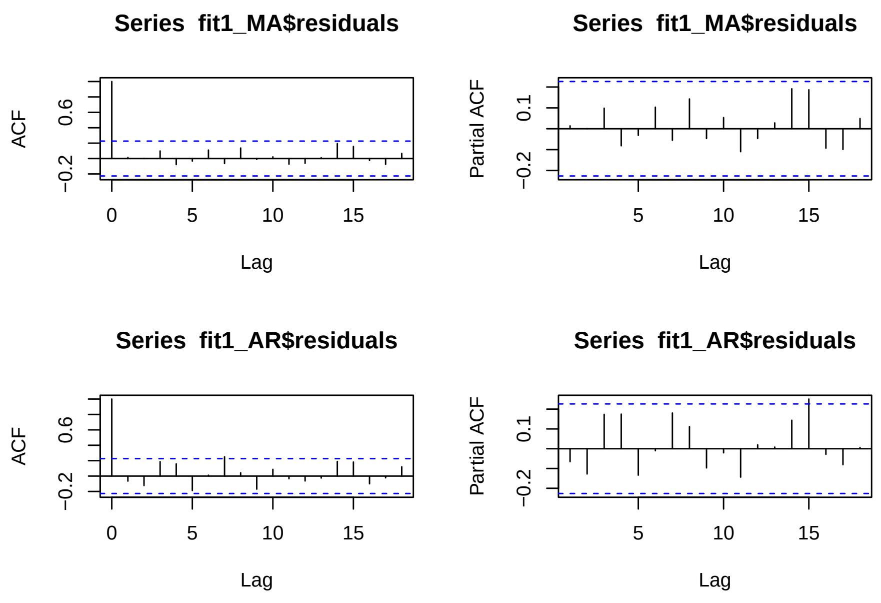
```
##
##  Box-Pierce test
##
## data:  fit1_AR$residuals
## X-squared = 21.836, df = 15, p-value = 0.1122
```

Indeed, the MA(7) model has high p-value of 0.8578 supporting the null hypothesis of satisfactory residual behaviors, while the AR(2) model has relatively small p-value of 0.1122 against the null hypothesis.

1.4 Model Improvement & Selection

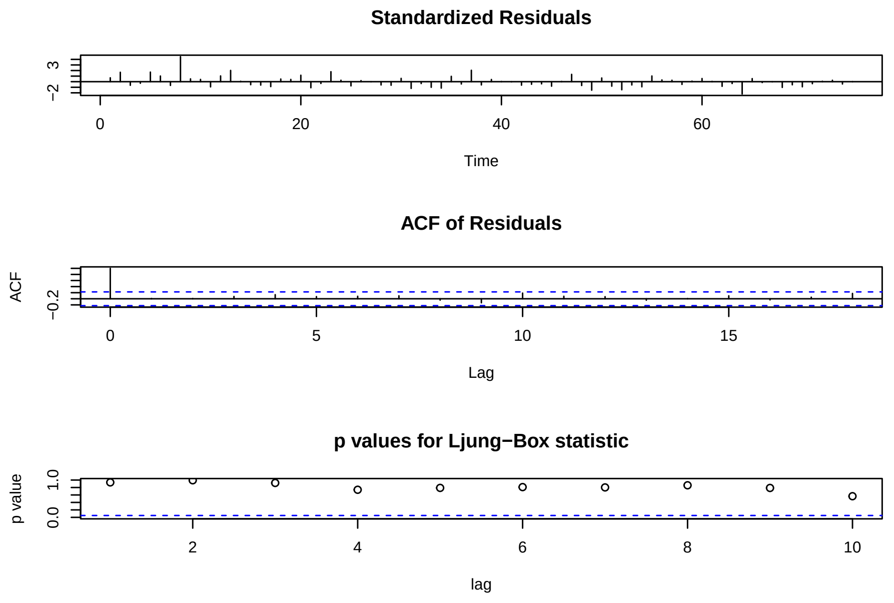
We now observe the behaviors of residuals to further improve our model.

```r
par(mfrow = c(2, 2))
acf(fit1_MA$residuals)
pacf(fit1_MA$residuals)
acf(fit1_AR$residuals)
pacf(fit1_AR$residuals)
```

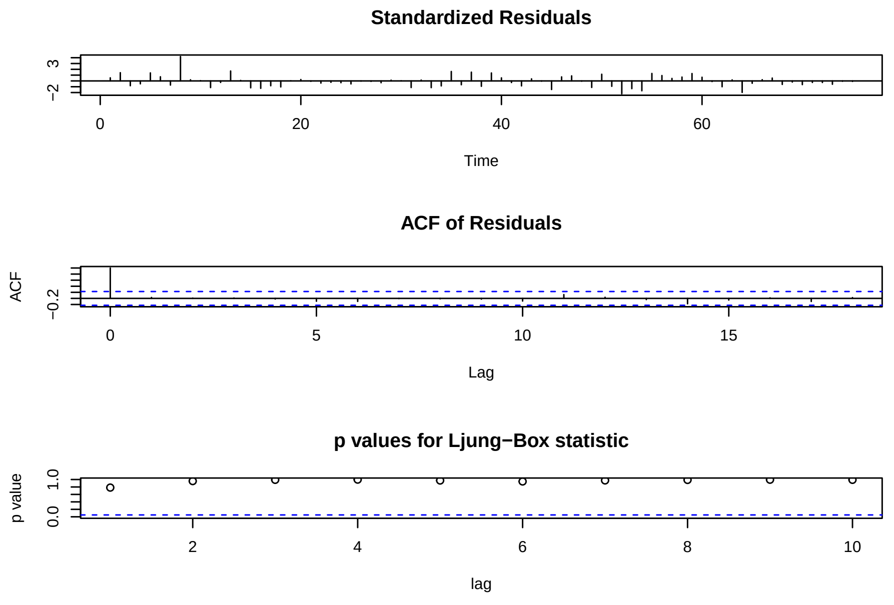


Notice that the residuals for model **fit__MA** do not have cut-offs and strictly stay between the boundary lines. For residuals for **fit__AR**, however, the residuals follow either an MA(7) process or an AR(15) process. Taking such residual behaviors into consideration, we may try two new models: ARMA(2, 7) and AR(17) for original data.

```r
fit1_ARMA = arima(data1, order = c(2, 0, 7))
fit1_AR17 = arima(data1, order = c(17, 0, 0))
tsdiag(fit1_ARMA)
```



```
tsdiag(fit1_AR17)
```



```
fit1_ARMA
```

```
##
## Call:
## arima(x = data1, order = c(2, 0, 7))
##
## Coefficients:
##          ar1      ar2      ma1     ma2     ma3     ma4     ma5     ma6     ma7
##       1.2724  -0.9699  -1.2469  1.3399  0.0138  0.1506  0.0161  0.4003  0.0613
## s.e.  0.0350   0.0372   0.1360  0.2356  0.2450  0.2515  0.2335  0.1759  0.1147
##       intercept
##         53.7605
## s.e.     1.8791
##
## sigma^2 estimated as 44.85:  log likelihood = -254.36,  aic = 530.71
```

```
fit1_AR17
```

```
##
## Call:
## arima(x = data1, order = c(17, 0, 0))
##
## Coefficients:
##          ar1     ar2     ar3     ar4      ar5      ar6     ar7     ar8      ar9
##       0.1089  0.3661  0.2894  0.1687  -0.2420  -0.0830  0.1515  0.0765  -0.0589
## s.e.  0.1108  0.1073  0.1112  0.1070   0.1063   0.1131  0.1060  0.1348   0.1252
##         ar10     ar11     ar12    ar13    ar14    ar15     ar16     ar17
##       0.1412  -0.2695  -0.0584  0.1427  0.3224  0.3363  -0.2287  -0.2993
## s.e.  0.1245   0.1265   0.1290  0.1285  0.1285  0.1278   0.1295   0.1315
##       intercept
##         54.4494
## s.e.     4.7498
##
## sigma^2 estimated as 41.87:  log likelihood = -249.86,  aic = 537.71
```

The diagnostic checking on both models seems good enough. Therefore, we complete our model selection by comparing AIC of each alternative model.

```
fit1_MA$aic
```

```
## [1] 537.9924
```

```
fit1_AR$aic
```

```
## [1] 537.4617
```

```
fit1_ARMA$aic
```

```
## [1] 530.7144
```

```
fit1_AR17$aic
```

```
## [1] 537.7101
```

Among all, the ARMA(2, 7) model (i.e. **fit1__ARMA**) has the least AIC value, thus we use it as our final model.

1.5 Prediction

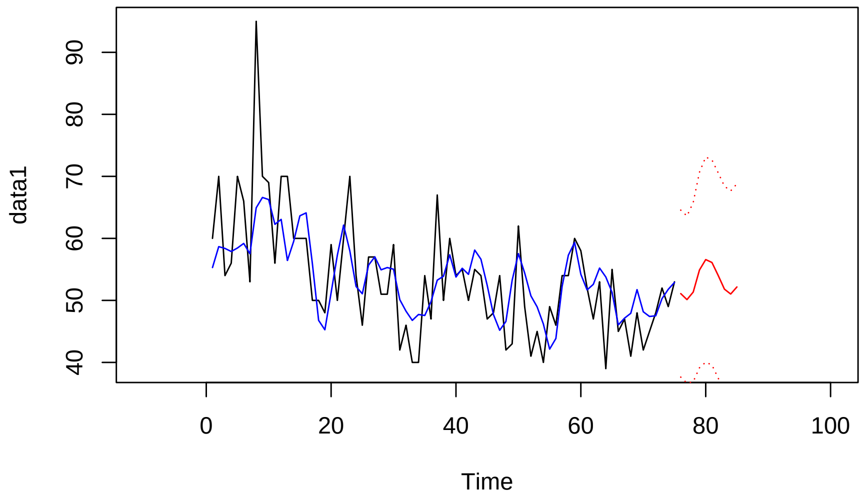
We predict the time series data 10 steps forward.

```
data1_predict = predict(fit1_ARMA, n.ahead = 10)
data1_predict
```

```
## $pred
## Time Series:
## Start = 76
## End = 85
## Frequency = 1
##  [1] 51.09682 50.12694 51.34175 54.90263 56.58067 56.11331 54.02484 51.81483
##  [9] 51.02848 52.17146
##
## $se
## Time Series:
## Start = 76
## End = 85
## Frequency = 1
##  [1] 6.868428 6.876345 7.373115 8.044330 8.437367 8.437514 8.443398 8.469837
##  [9] 8.485670 8.484690
```

We finally put the original data, our fitted data and predictions into one plot.

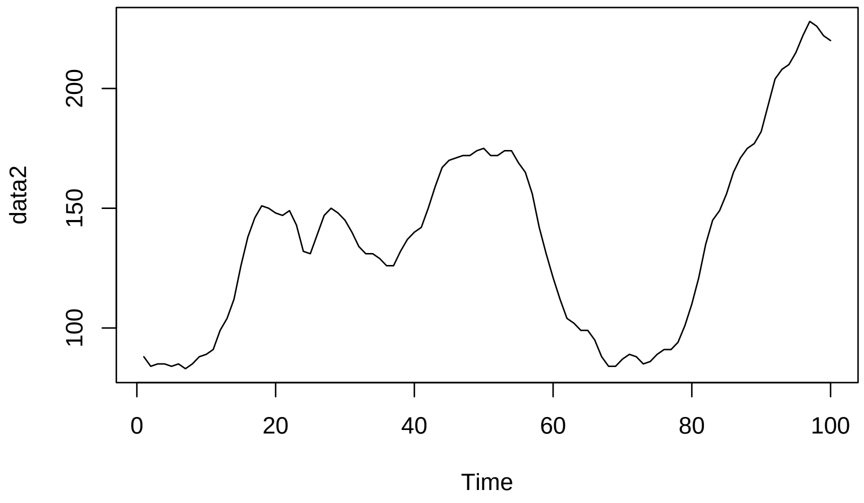
```
plot.ts(data1, xlim = c(-10, 100))
lines(1:length(data1), data1 - fit1_ARMA$residuals, type = "l",
    col = "blue")
lines(data1_predict$pred, col = "red")
lines(data1_predict$pred + 1.96 * data1_predict$se, col = "red",
    lty = 3)
lines(data1_predict$pred - 1.96 * data1_predict$se, col = "red",
    lty = 3)
```


data1
90
80
70
60
50
40
0
20
40
60
80
100
Time

2 Data “wwwusage.txt”

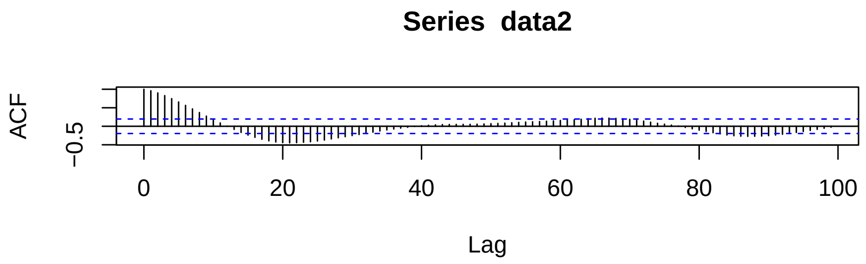
2.1 Stationarity Checking

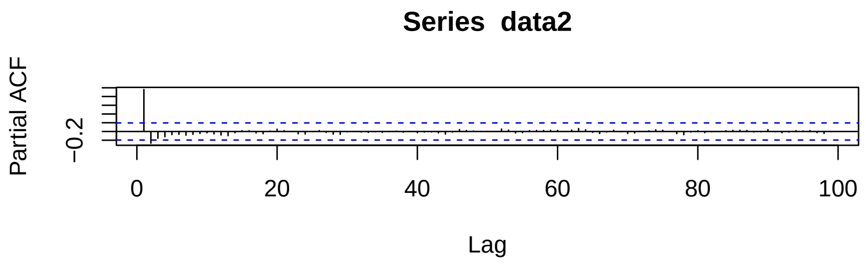
```
data2 = read.table("/Users/zeyichen/Desktop/Courses/Y2S2/Time Series Analysis/Lab/Lab 3/wwwu
    header = TRUE)
data2 = as.numeric(as.matrix(data2))
ts.plot(data2)
```



Check the stationarity by ACF and PACF plots:

```
par(mfrow = c(2, 1))
acf(data2, lag.max = 100)
pacf(data2, lag.max = 100)
```





Observe that the ACF of the time series data cuts off extremely slowly, thus the data can be interpreted as non-stationary. At the inital stage, we need to transform and difference on the data to make it stationary.

2.2 Model Identification with Data Transformed & Differenced

Before applying differencing operator, it is always necessary to use Box-Cox Transformation on the dataset to stablise the variance. Here the transformation is achieved by using *BoxCox.lambda()* and *bxcx()* functions in R.

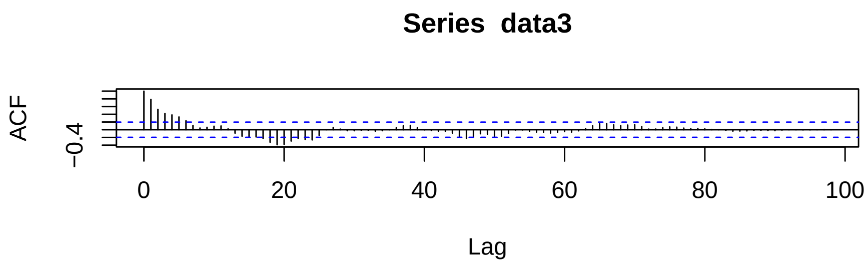
```
lambda = BoxCox.lambda(data2)  # This gives the optimal lambda
lambda
```

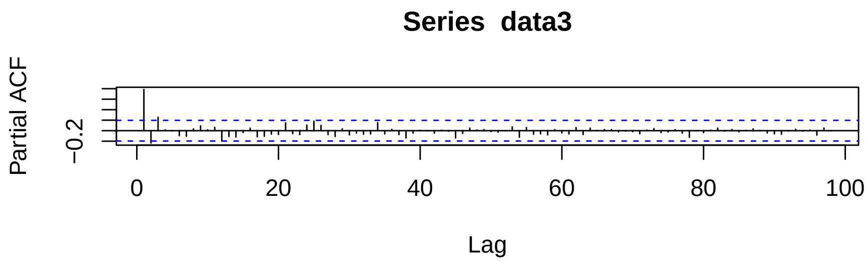
```
## [1] 0.3596253
```

```
data2_transformed = bxcx(data2, lambda)
# data2_transformed
```

Now we try to operate 1-lag diffrence operator on the transformed data to see whether its ACF and PACF cut off.

```
data3 = diff(bxcx(data2, lambda))
# data3
par(mfrow = c(2, 1))
acf(data3, lag.max = 100)
pacf(data3, lag.max = 100)
```





From the plots generated, the ACF of **data3** cuts off at 24 and its PACF cuts off at 3. Therefore, MA(24) and AR(3) are two model options.

2.3 Parameter Estimation

We fit the data with the MA(24) model and AR(3) model in respective. Then we list the coefficients of the induced MA(24) and AR(3) model.

```
fit_trans_MA = arima(data2_transformed, order = c(0, 1, 24))
fit_trans_AR = arima(data2_transformed, order = c(3, 1, 0))
fit_trans_MA
```

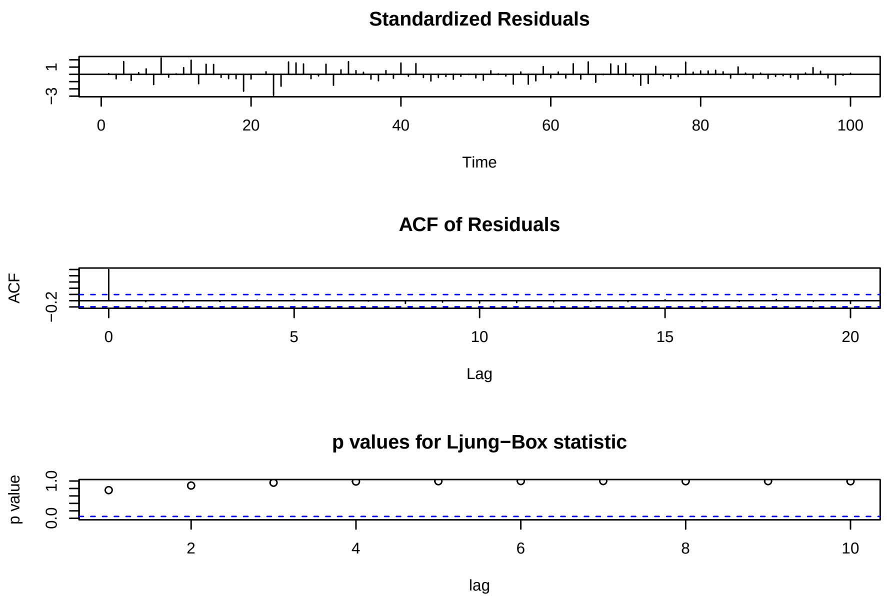
```
##
## Call:
## arima(x = data2_transformed, order = c(0, 1, 24))
##
## Coefficients:
##          ma1     ma2     ma3     ma4     ma5     ma6     ma7     ma8     ma9
##       1.0680  0.7358  0.5633  0.7435  0.5915  0.4606  0.4052  0.3489  0.5051
## s.e.  0.1296  0.1875  0.2145  0.2408  0.2467  0.2271  0.2178  0.1969  0.1847
##         ma10    ma11    ma12    ma13    ma14    ma15    ma16    ma17    ma18
##       0.6457  1.0268  1.2757  1.1261  0.9878  0.6567  0.5891  0.7344  0.3848
## s.e.  0.2133  0.2165  0.2025  0.2146  0.2080  0.2167  0.2242  0.2285  0.2273
##         ma19    ma20    ma21    ma22    ma23    ma24
##       0.0729  0.1676  0.2817  0.4922  0.6606  0.2972
## s.e.  0.2417  0.2272  0.2142  0.2051  0.1982  0.1394
##
## sigma^2 estimated as 0.01119:  log likelihood = 68.97,  aic = -87.95
```

```
fit_trans_AR
```

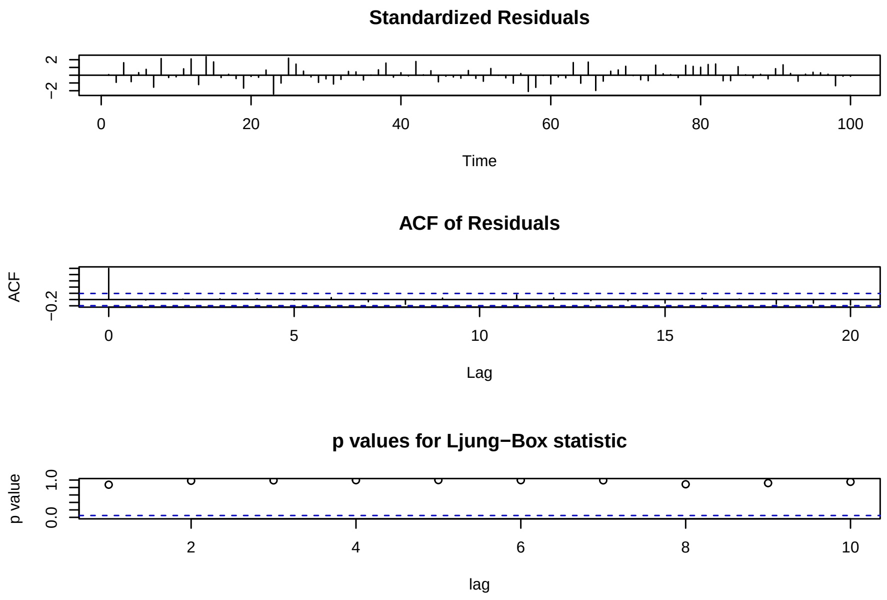
```
##
## Call:
## arima(x = data2_transformed, order = c(3, 1, 0))
##
## Coefficients:
##          ar1      ar2     ar3
##       1.0957  -0.5776  0.3055
## s.e.  0.0970   0.1366  0.0961
##
## sigma^2 estimated as 0.01913:  log likelihood = 54.65,  aic = -101.29
```

2.4 Diagnostic Checking

```
par(mfrow = c(1, 2))
tsdiag(fit_trans_MA)
```



```
tsdiag(fit_trans_AR)
```



Both models give satisfactory residual behaviors as the residual points lie much above the dotted blue line in the third plots.

Again we use *Box.test()* to further convince:

```
Box.test(fit_trans_MA$residuals, lag = length(data3)/5)
```

```
##
##  Box-Pierce test
##
## data:  fit_trans_MA$residuals
## X-squared = 2.9875, df = 19.8, p-value = 1
```

```
Box.test(fit_trans_AR$residuals, lag = length(data3)/5)
```

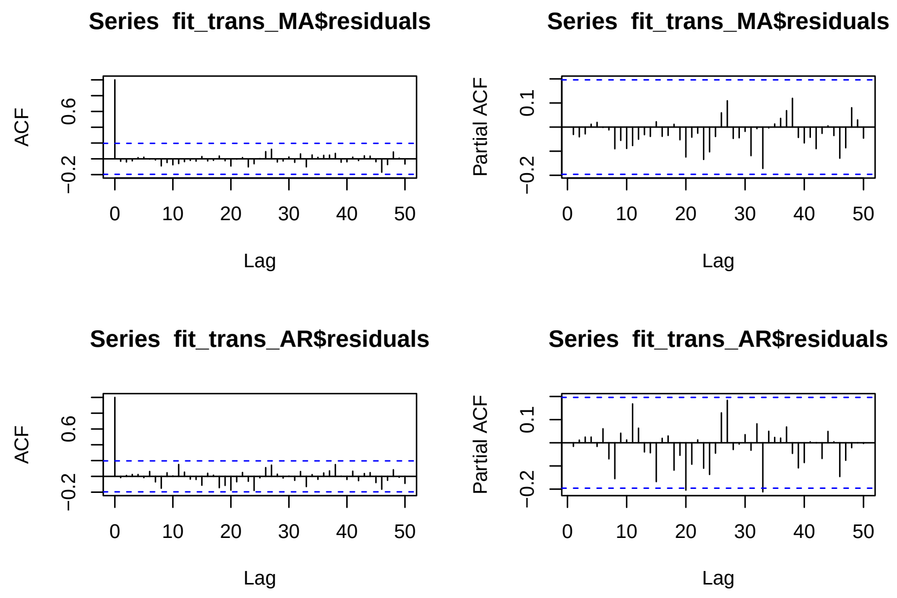
```
##
##  Box-Pierce test
##
## data:  fit_trans_AR$residuals
## X-squared = 11.168, df = 19.8, p-value = 0.9375
```

Both p-values approach to 1, which strongly support the null hypothesis of satisfactory residual behaviors.

2.5 Model Improvement & Selection

We now observe the behaviors of residuals to further improve our model.

```r
par(mfrow = c(2, 2))
acf(fit_trans_MA$residuals, lag.max = 50)
pacf(fit_trans_MA$residuals, lag.max = 50)
acf(fit_trans_AR$residuals, lag.max = 50)
pacf(fit_trans_AR$residuals, lag.max = 50)
```



The residual points in all plots are generally satisfactory, except for a tiny violation in PACF of **fit_trans_AR** where the $ r_{33, 33} $ oversteps the boundary a little bit. However, this could be explained by an exception outside the confidence interval thus ignored.

Now we compare the AIC of **fit_trans_MA** and **fit_trans_AR**.

```r
fit_trans_MA$aic
```

```
## [1] -87.9456
```

```r
fit_trans_AR$aic
```

```
## [1] -101.2934
```

The ARIMA(3, 1, 0) (i.e. **fit_trans_AR**) model gives lower AIC value, thus is our final choice.

2.6 Prediction & Backward Transformation

We predict the data in the transformed form in the next 10 steps.

```
data2_trans_prediction = predict(fit_trans_AR, n.ahead = 10)
data2_trans_prediction
```

```
## $pred
## Time Series:
## Start = 101
## End = 110
## Frequency = 1
##  [1] 16.54742 16.52833 16.49727 16.46944 16.45105 16.43749 16.42475 16.41301
##  [9] 16.40336 16.39567
##
## $se
## Time Series:
## Start = 101
## End = 110
## Frequency = 1
##  [1] 0.1383021 0.3211510 0.4944938 0.6521302 0.8070814 0.9647958 1.1220015
##  [8] 1.2751088 1.4234844 1.5676628
```

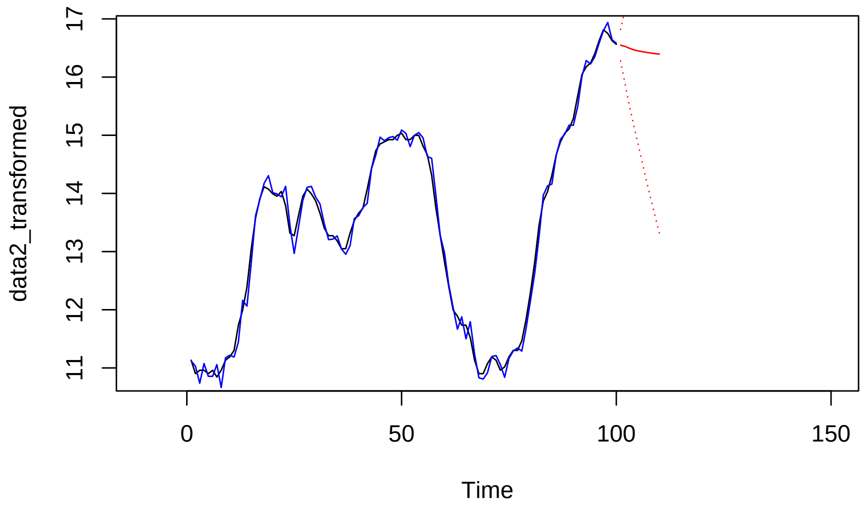
The fitted plot of the transformed data is as follows.

```
data2_transformed
```

```
##   [1] 11.13278 10.90195 10.96030 10.96030 10.90195 10.96030 10.84314 10.96030
##   [9] 11.13278 11.18943 11.30153 11.73478 11.99428 12.39334 13.04988 13.57635
##  [17] 13.91123 14.11459 14.07427 13.99310 13.95225 14.03377 13.78706 13.31695
##  [25] 13.27299 13.61888 13.95225 14.07427 13.99310 13.87002 13.66121 13.40424
##  [33] 13.27299 13.27299 13.18441 13.04988 13.04988 13.31695 13.53363 13.66121
##  [41] 13.74530 14.07427 14.43119 14.73774 14.85027 14.88750 14.92459 14.92459
##  [49] 14.99835 15.03503 14.92459 14.92459 14.99835 14.99835 14.81290 14.66200
##  [57] 14.31369 13.74530 13.27299 12.82103 12.39334 11.99428 11.89146 11.73478
##  [65] 11.73478 11.52108 11.13278 10.90195 10.90195 11.07571 11.18943 11.13278
##  [73] 10.96030 11.01822 11.18943 11.30153 11.30153 11.46675 11.83956 12.29533
##  [81] 12.82103 13.44758 13.87002 14.03377 14.31369 14.66200 14.88750 15.03503
##  [89] 15.10799 15.28810 15.67347 16.04503 16.17695 16.24231 16.40396 16.62629
##  [97] 16.81331 16.75133 16.62629 16.56323
```

```
plot.ts(data2_transformed, xlim = c(-10, 150))
lines(1:length(data2_transformed), data2_transformed - fit_trans_AR$residuals,
    type = "l", col = "blue")
```

```r
lines(data2_trans_prediction$pred, col = "red")
lines(data2_trans_prediction$pred + 1.96 * data2_trans_prediction$se,
    col = "red", lty = 3)
lines(data2_trans_prediction$pred - 1.96 * data2_trans_prediction$se,
    col = "red", lty = 3)
```



Before producing the final fitted plot, we need to apply the backward transformation on the fitted and predicted values, using *InverseQ = TRUE* in function *bxcx()*.

```r
fitted_values = bxcx(data2_transformed - fit_trans_AR$residuals,
    lambda, InverseQ = TRUE)
fitted_values
```

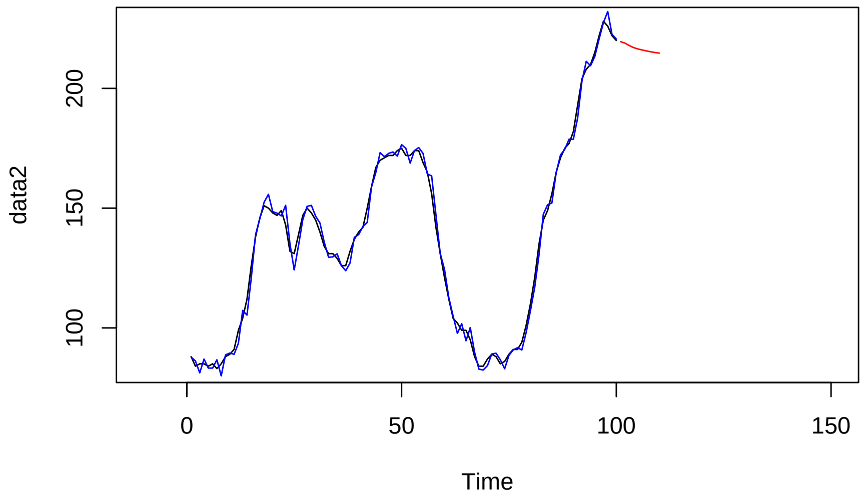
```
## Time Series:
## Start = 1
## End = 100
## Frequency = 1
##   [1]  87.80434  86.18881  81.22129  86.99678  83.21935  83.21509  86.69035
##   [8]  79.98789  88.73008  89.51240  88.96890  93.58906 107.31762 105.34293
##  [15] 120.82843 138.98042 145.62785 152.49674 155.77529 148.53585 147.91359
##  [22] 146.75113 151.18152 135.22197 124.22323 134.38590 145.23396 150.74482
##  [29] 151.16107 146.59170 143.77548 135.72148 129.47842 129.66159 130.93907
##  [36] 125.94193 123.88137 127.12412 137.81566 138.95072 142.29031 144.03141
##  [43] 158.84452 164.89360 173.17466 171.48597 172.87534 173.44943 171.72865
##  [50] 176.51952 174.93400 168.78678 174.03357 175.28447 172.84297 164.21825
##  [57] 163.47274 147.25134 131.10448 124.40134 112.65080 104.92630  97.70070
##  [64] 101.75254  94.62202 100.12469  89.88174  82.79011  82.40629  84.25141
##  [71]  89.10925  89.43009  86.69281  82.92608  88.55025  90.80900  91.74999
##  [78]  90.77653  98.00299 107.12919 116.92218 130.42697 147.42620 151.42327
##  [85] 152.14585 164.81245 172.20042 174.55996 178.79009 178.75573 187.66516
##  [92] 203.06642 211.30022 209.45752 213.36033 220.64636 227.45625 232.09671
##  [99] 222.54135 220.60103
```

```
predicted_values = bxcx(data2_trans_prediction$pred, lambda,
    InverseQ = TRUE)
predicted_values
```

```
## Time Series:
## Start = 101
## End = 110
## Frequency = 1
##  [1] 219.5004 218.8979 217.9203 217.0467 216.4706 216.0464 215.6484 215.2819
##  [9] 214.9810 214.7416
```

Finally, we put the original data, the fitted value and predicted values together in one plot.

```
plot.ts(data2, xlim = c(-10, 150))
lines(1:length(data2), fitted_values, type = "l", col = "blue")
lines(predicted_values, col = "red")
```



From the plot, we see the fitted effect is satisfactory.

2.7 On the Necessity of Transformation

Notice that the data can be stationarised merely by applying a single differencing operator and the model chosen without Box-Cox Transformation is also an Arima(3, 1, 0) model. (Repeat the same procedure as described previously to select the ideal model)

This section aims to compare the models generated with and without data transformed, thus emphasising the necessity of conducting transformation on the raw data. We use MAD (Mean Absolute Deviation) to measure the accuracy oc prediction and prefer the model with smaller MAD. To train the model suitable for prediction, we divide our sample data into training and testing set, where the size of the training data is 90% of the whole sample.

```
set.seed(2022)
length(data2)
```

```
## [1] 100
```

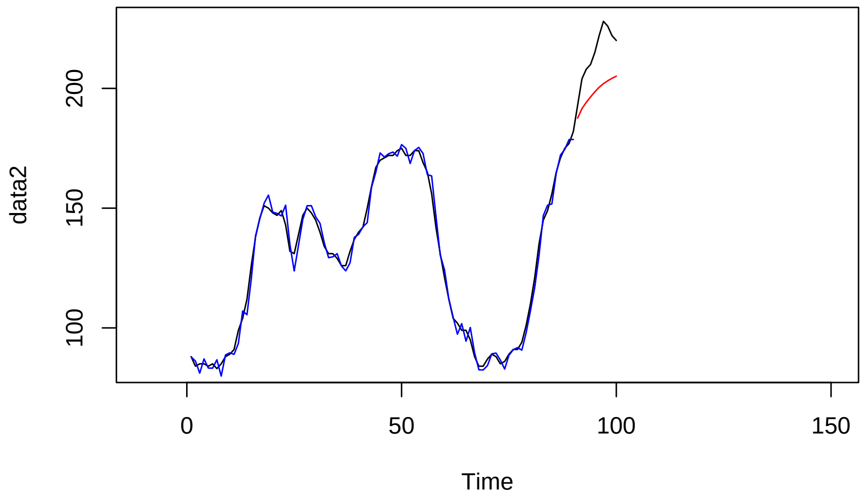
```
train.data <- data2[1:90]
train.data
```

```
##  [1]  88  84  85  85  84  85  83  85  88  89  91  99 104 112 126 138 146 151 150
## [20] 148 147 149 143 132 131 139 147 150 148 145 140 134 131 131 129 126 126 132
## [39] 137 140 142 150 159 167 170 171 172 172 174 175 172 172 174 174 169 165 156
## [58] 142 131 121 112 104 102  99  99  95  88  84  84  87  89  88  85  86  89  91
## [77]  91  94 101 110 121 135 145 149 156 165 171 175 177 182
```

```
test.data <- data2[91:100]
test.data
```

```
##  [1] 193 204 208 210 215 222 228 226 222 220
```

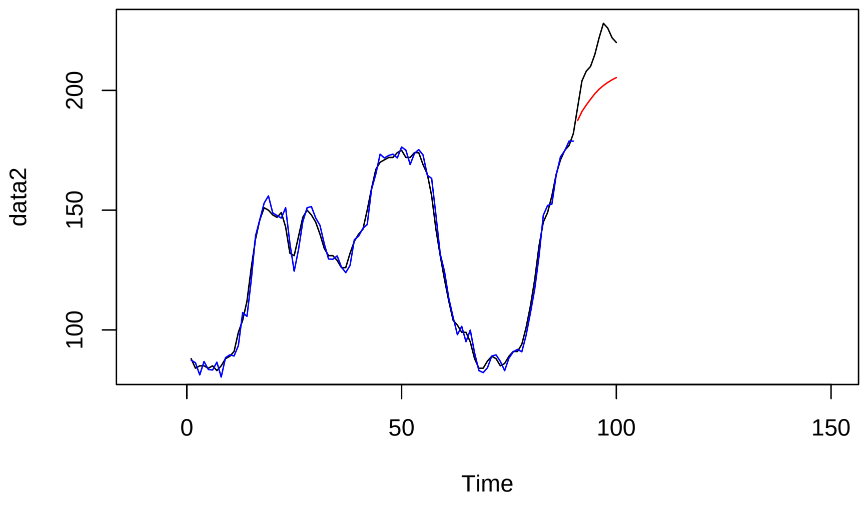
```
# Without Transformation
fit_orig = arima(train.data, order = c(3, 1, 0))
predict_orig = predict(fit_orig, n.ahead = length(test.data))
plot.ts(data2, xlim = c(-10, 150))
lines(1:length(train.data), train.data - fit_orig$residuals,
    type = "l", col = "blue")
lines(predict_orig$pred, col = "red")
```



```
mad(test.data - predict_orig$pred)
```

```
## [1] 3.952968
```

```
# With Transformation
lambda1 = BoxCox.lambda(train.data)
training_trans <- bxcx(train.data, lambda1)
fit_training_trans <- arima(training_trans, order = c(3, 1, 0))
predict_trans = predict(fit_training_trans, n.ahead = length(test.data))
predict_backtrans = bxcx(predict_trans$pred, lambda1, InverseQ = TRUE)
plot.ts(data2, xlim = c(-10, 150))
lines(1:length(train.data), bxcx(training_trans - fit_training_trans$residuals,
    lambda1, InverseQ = TRUE), type = "l", col = "blue")
lines(predict_backtrans, col = "red")
```



```
mad(test.data - predict_backtrans)
```

```
## [1] 3.556958
```

The MADs for the 2 models turned out to be 3.952 and 3.557 in respective. Therefore, we may conclude that the model derived from transformed data is more robust.

3 Potential Improvement

1. The data sizes of both datasets are limited, thus give limited information and interactions. When more data are obtained, there might be seasonal factors lying in the time series, and more complicated model like SARIMA could be involved.

2. Although training and testing sets were determined, cross validation techniques could be applied when constructing models to optimize the bias-variance trade-off and avoid the overfitting problem.