Analyzing Monthly Average Temperatures in Shanghai

PSTAT 174 Final Project

Sunrise Gao

12/07/2022

Contents

Abtract

With the global warming problem being more and more serious, meteorological research has become particularly important. Temperature Analysis and prediction as big parts of meteorological research can help us to prevent potential natural disasters at an early stage as well as being a strong statistical support to alarm us to protect our environment, our homeland, our earth.

Therefore, the main purpose of this project is whether I can successfully construct and choose a model that forecasts the future values of monthly average temperatures accurately for my home-country city, Shanghai. In addition, parameter estimation, diagnostic check and model comparison are also big parts of my project after I choose and/or construct a model which can impact my final model's accuracy.

Introduction

A very important part of statistical analysis that is the data. Every data set can be analyzed, but a good data set can save times from doing a lot of unnecessary work. After searching a lot of data set. I found one interesting data set from Kaggle.com called "Climate Change: Earth Surface Temperature Data". It contains many major cities around the world with decades of monthly average temperatures. After viewing most of this data set, I decided to choose the city that from my home country, Shanghai.

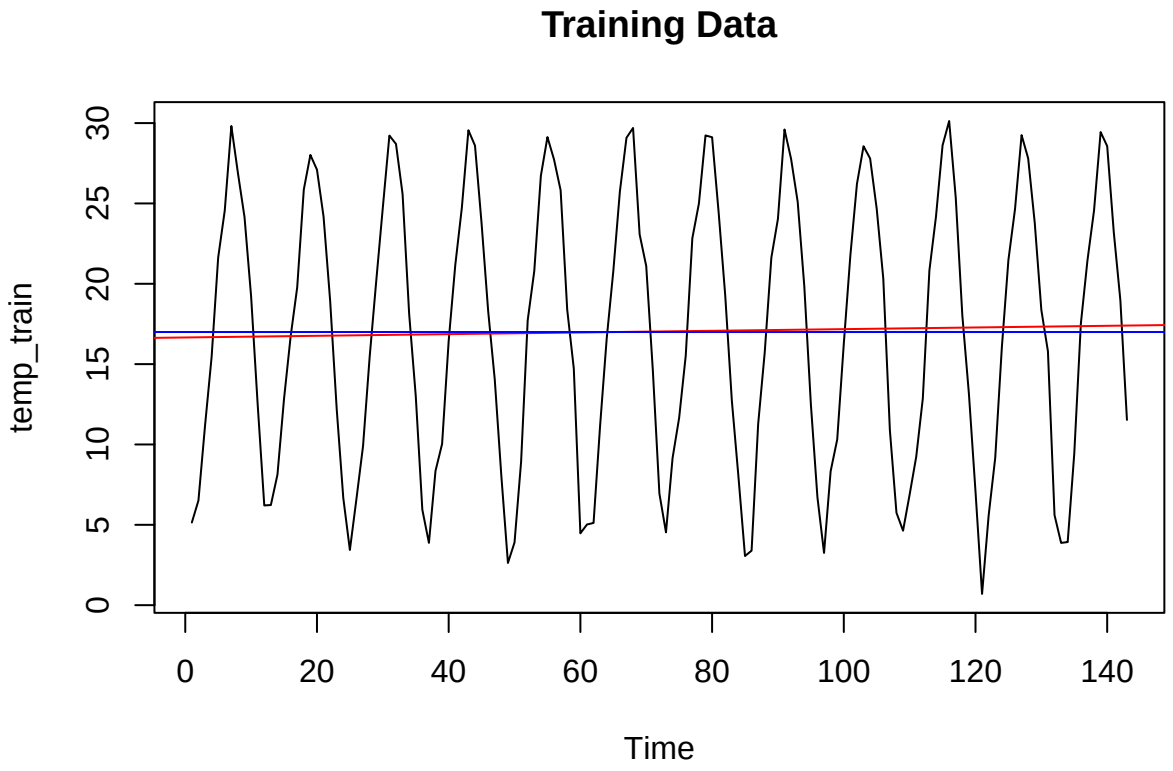
Shanghai, as known as China's New York, has the largest Gross Domestic Product (GDP) in the country. As the richest, busiest and most globalized city in the country, we can expect that human activities and land developments are excessive, and these are often the key factors of temperature rising. Analyzing the temperatures in Shanghai is essentially important. Now, I will be using techniques that I have learned from PSTAT 174/274 class. Following the steps that start with data splitting, ploting time serise, early comparison, data transformation, then model identification, model checking, diagnostic check, and eventually forecasting. Let's get it!

1 Ploting and Analyzing the Time Series

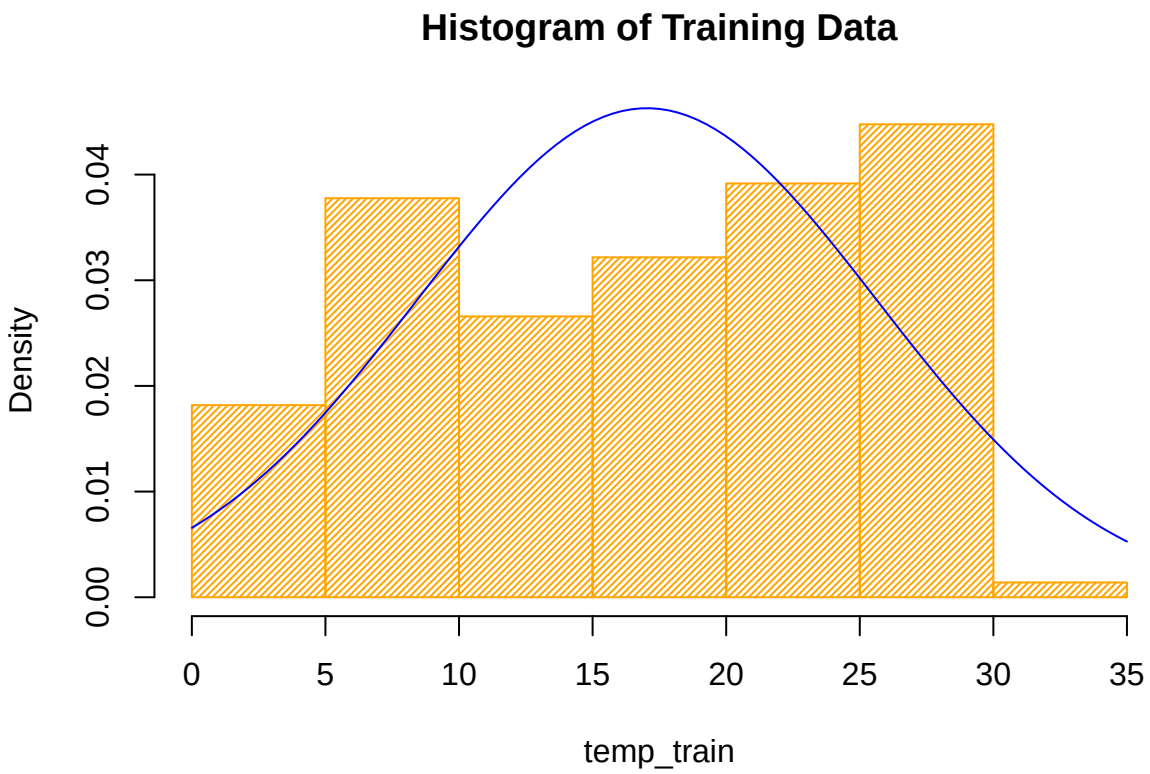
In this step, we will load the data, splitting the data, plot the data, and perform box-cox transformation.

1a. Plot the Training Data

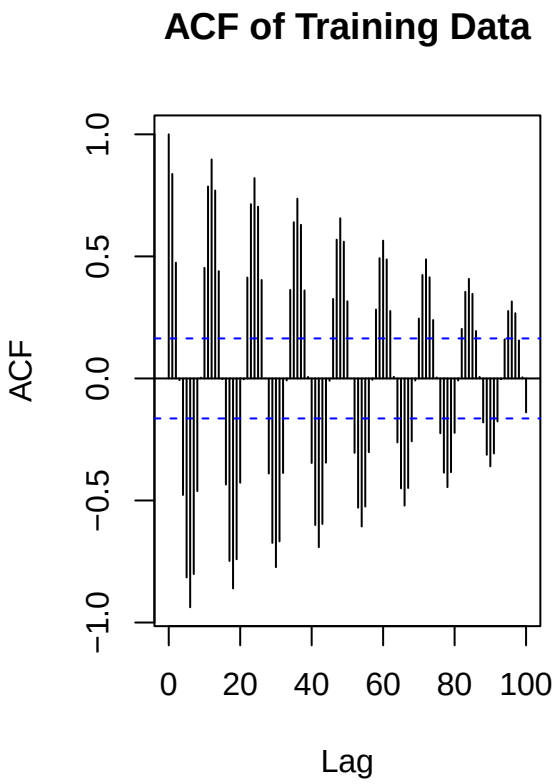
```
# plot training data set
plot.ts(temp_train, main = "Training Data")

fit <- lm(temp_train ~ as.numeric(1:length(temp_train)))

# add trend line and mean line to training data
abline(fit, col = "red")
abline(h = mean(sh_temp_raw$AverageTemperature), col = "blue")
```

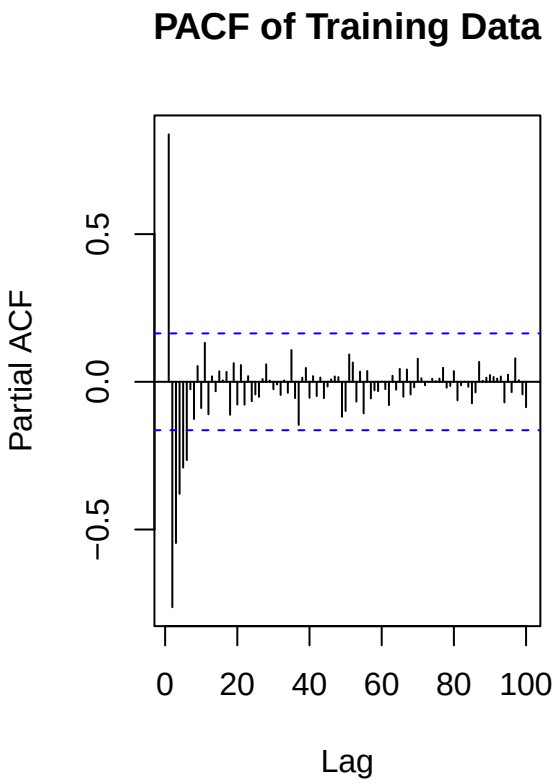


Interpretation:
a. As we can see from the result above, the plot doesn't have an obvious trend for our training data.
b. Since the data contains monthly average temperature year by year, it makes sense that the data set already has seasonality. However, there is no stationary at the moment, so we will discuss and explore later on.
c. The variations of temperature remain the same mostly over time, and there is no many sharp changes in behavior besides seasonal effect.



Interpretation:
The histogram doesn't really show any normality for our training data, so we will perform box-cox transformation.





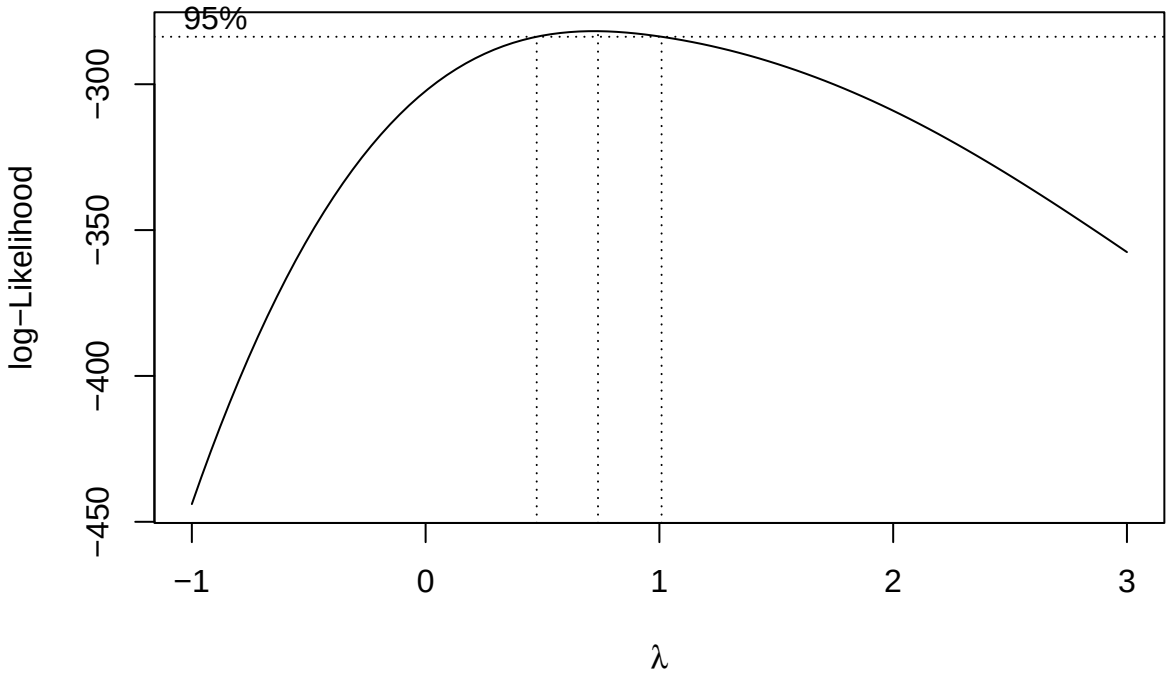
Interpretation:
Base on the ACF (slowly decaying oscillation) and PACF, the data set is an AR model. This might help us to skip invertibility check in the later step.

1b. Performing Box-cox Transformation

```
#variance of original data
var(temp_train)
```

```
## [1] 74.29737
```

```
# using Box-Cox transformation
bc_Transform <- boxcox(temp_train ~ as.numeric(1:length(temp_train)),
                       lambda = seq(-1, 3), plotit=TRUE)
```




```
lambda <- bc_Transform$x[which.max(bc_Transform$y)]
lambda
```

```
## [1] 0.7373737
```

```
temp_train_BC <- (1/lambda)*(temp_train^lambda - 1)
```

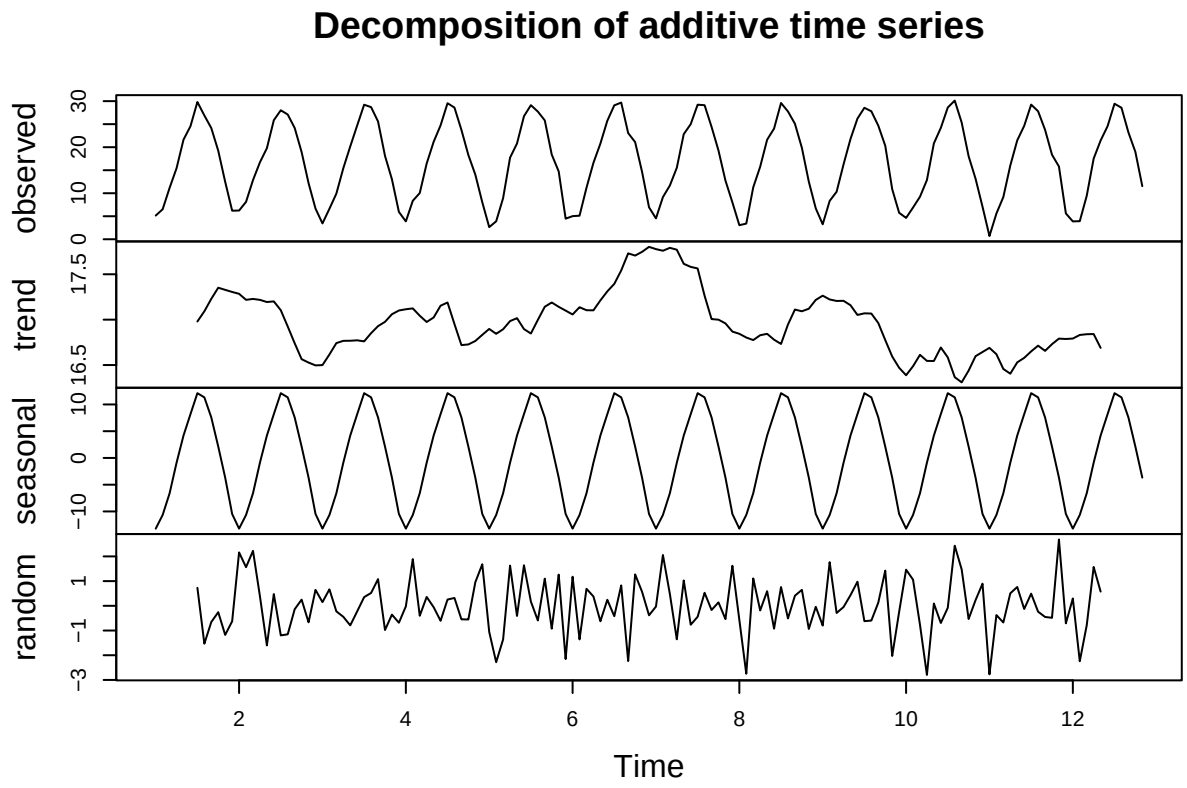
Interpretation:
The lamba is 0.7373737 and 1 is within the confidence interval which indicates the box-cox transformation is not needed. We will just use the original data in the next step. In addition, the data set has a big variance and doesn't show a standard normality, we might come back later if the all the candidate models' residuals are off and not passing diagnostic check.

2. Differencing data

In this step, we will proceed differencing process.

2a. Decomposition of Original Data

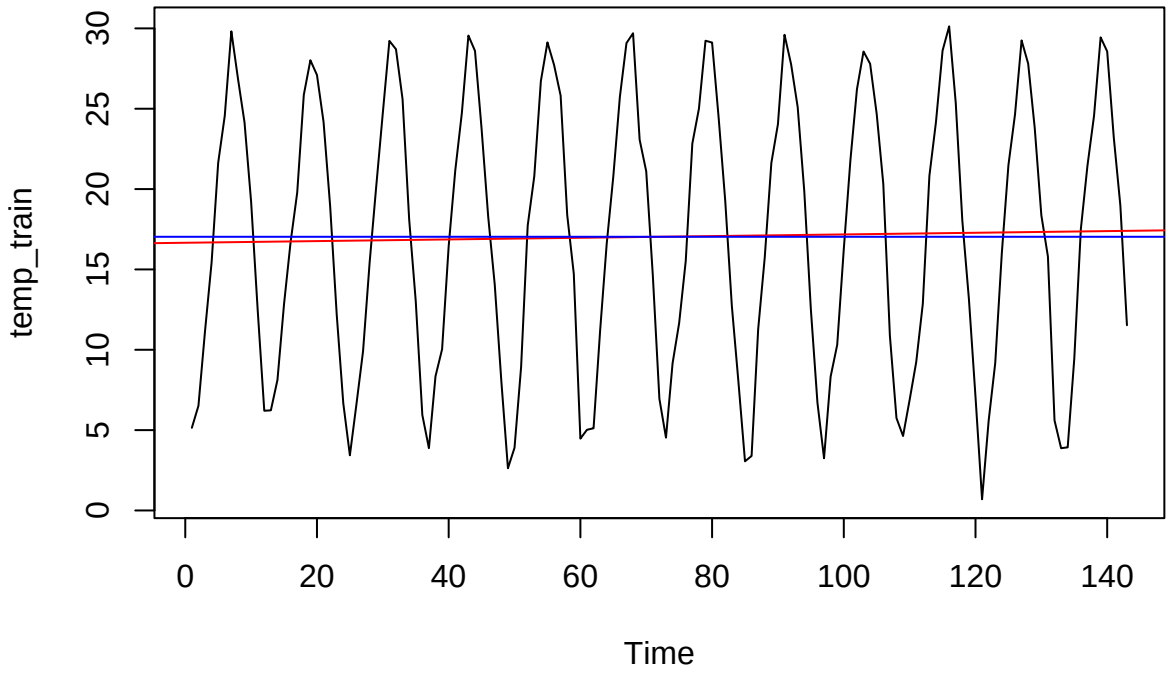
```
# decomposition of original data
temp_new <- ts(temp_train, frequency = 12)
decomp <- decompose(temp_new)
plot(decomp)
```



Interpretation:
The decomposition plot shows that there is a trend and seasonal, so we will proceed differencing process to remove trend and seasonal.

2b. Data Differencing Process

```
# No differencing
fit <- lm(temp_train ~ as.numeric(1:length(temp_train)));
plot.ts(temp_train)
abline(reg = fit, col = "red")
abline(h = mean(temp_train), col = "blue")
```

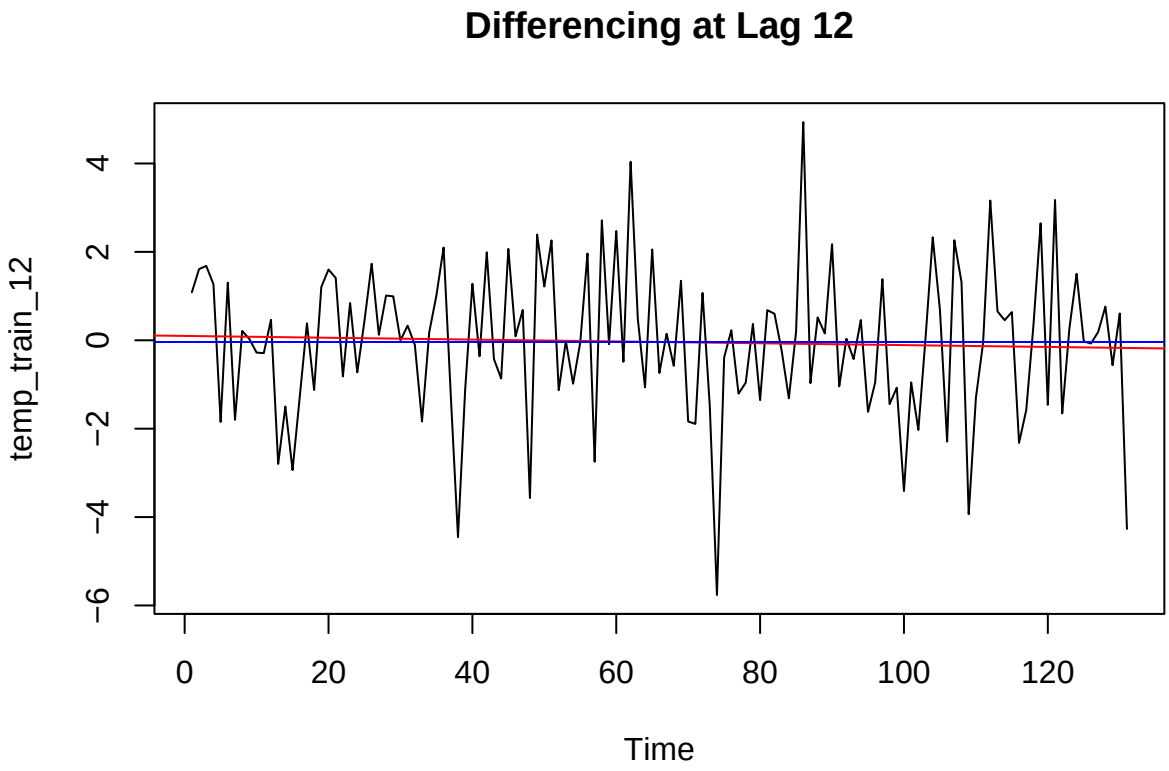



```
##   Differencing Variance
## A None         74.2973684705013
## B
## C
```

Recall that the original data's variance is 74.297 and there is a seasonal component and may contains a trend as we mention in previous step. Now, we will start differencing at lag 12 first.

i. Differencing at Lag 12

```
# difference at 12 to remove the seasonality
temp_train_12 <- diff(temp_train, lag = 12)
plot.ts(temp_train_12, main = "Differencing at Lag 12")
fit_diff_12 <- lm(temp_train_12 ~ as.numeric(1:length(temp_train_12)))

abline(reg = fit_diff_12, col = "red")
abline(h=mean(temp_train_12), col = "blue")
```

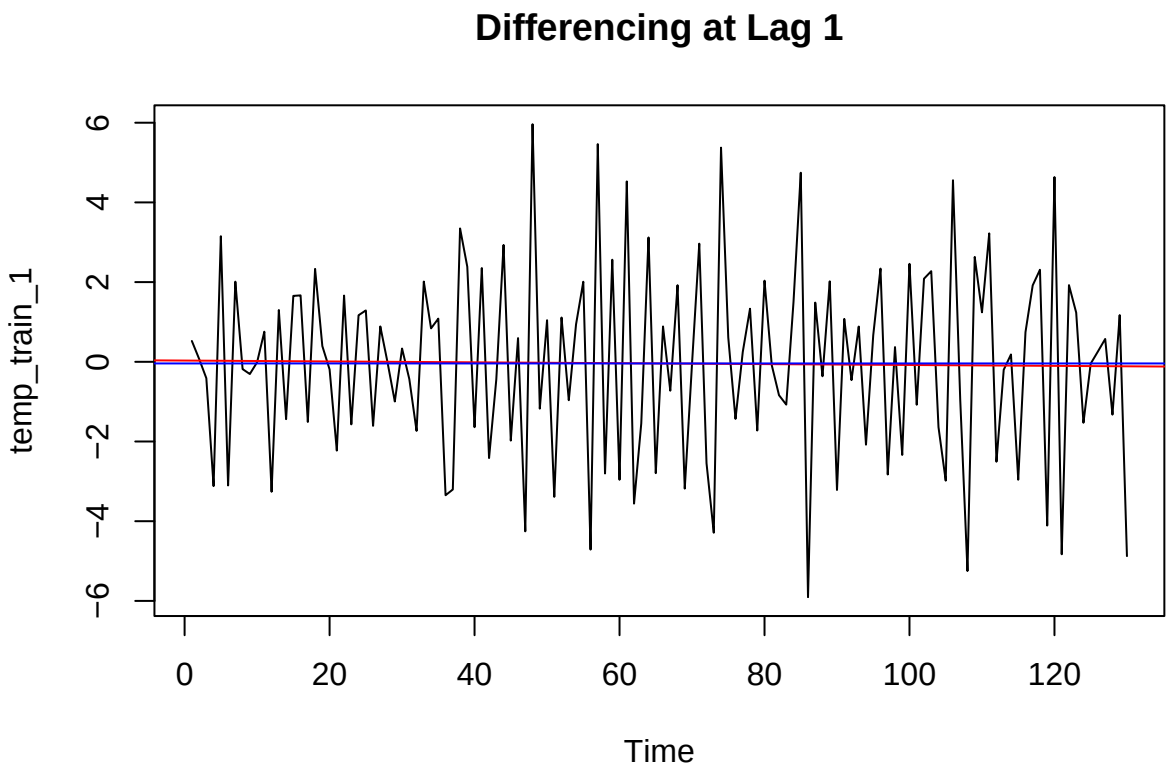



```
##   Differencing Variance
## A None         74.2973684705013
## B lag_12       2.90508725061656
## C
```

The seasonal component has been removed and the variance decreases as well. Furthermore, the plot shows that the data is nearly stationary but still has a slightly trend. Next, we are going do a differencing at lag 1.

ii. Differencing at Lag 1

```
# difference at 1
temp_train_1 <- diff(temp_train_12, lag=1)
plot.ts(temp_train_1, main = "Differencing at Lag 1")
fit_diff_1 <- lm(temp_train_1 ~ as.numeric(1:length(temp_train_1)))
abline(reg = fit_diff_1, col = "red")
abline(h = mean(temp_train_1), col = "blue")
```

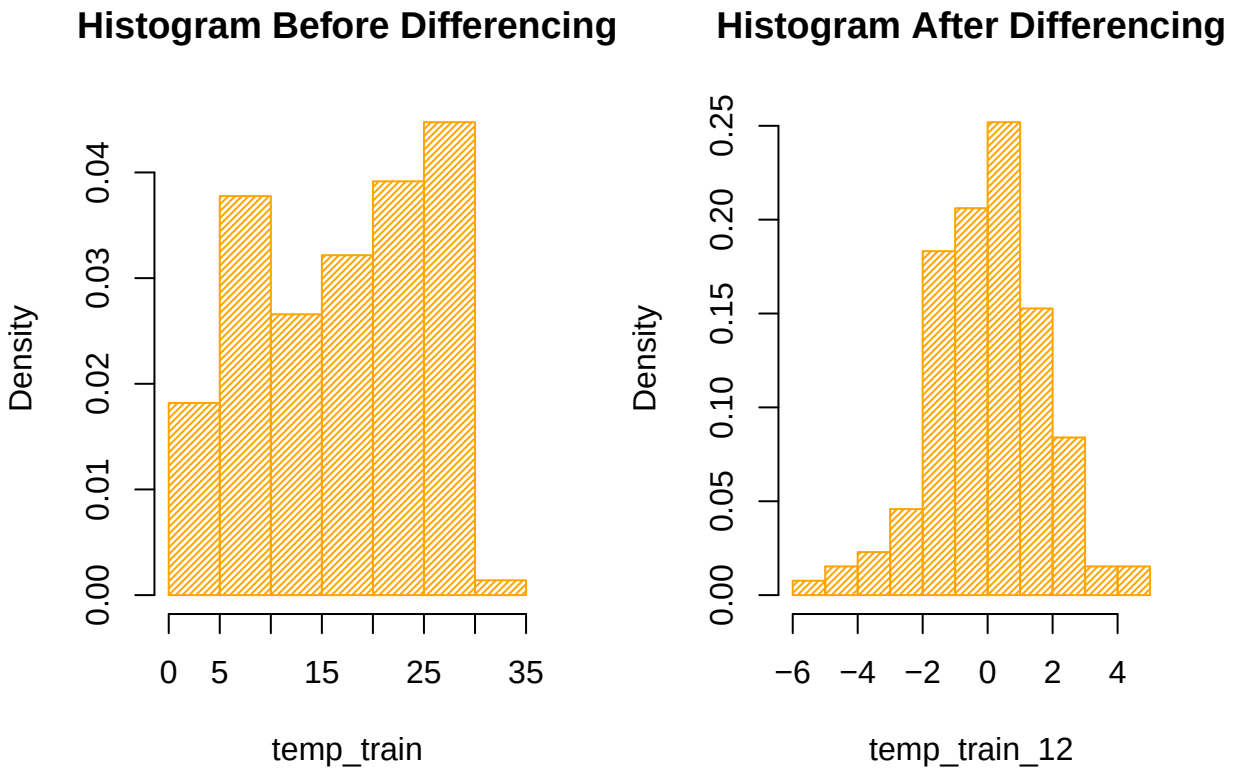



```
##   Differencing Variance
## A None         74.2973684705013
## B lag_12       2.90508725061656
## C lag_12&1     5.9575762602266
```

As the result from the table above, with the variance is increasing, differencing at lag 1 is not needed. Therefore, we will not keep the differencing at lag 1.

iii. Comparing Histograms

```
# compare histogram before and after differecing
par(mfrow=c(1, 2))
hist(temp_train, freq = F, main = "Histogram Before Differencing",
     col = "orange", density = 40)
hist(temp_train_12, freq = F, main = "Histogram After Differencing",
     col = "orange", density = 40)
```

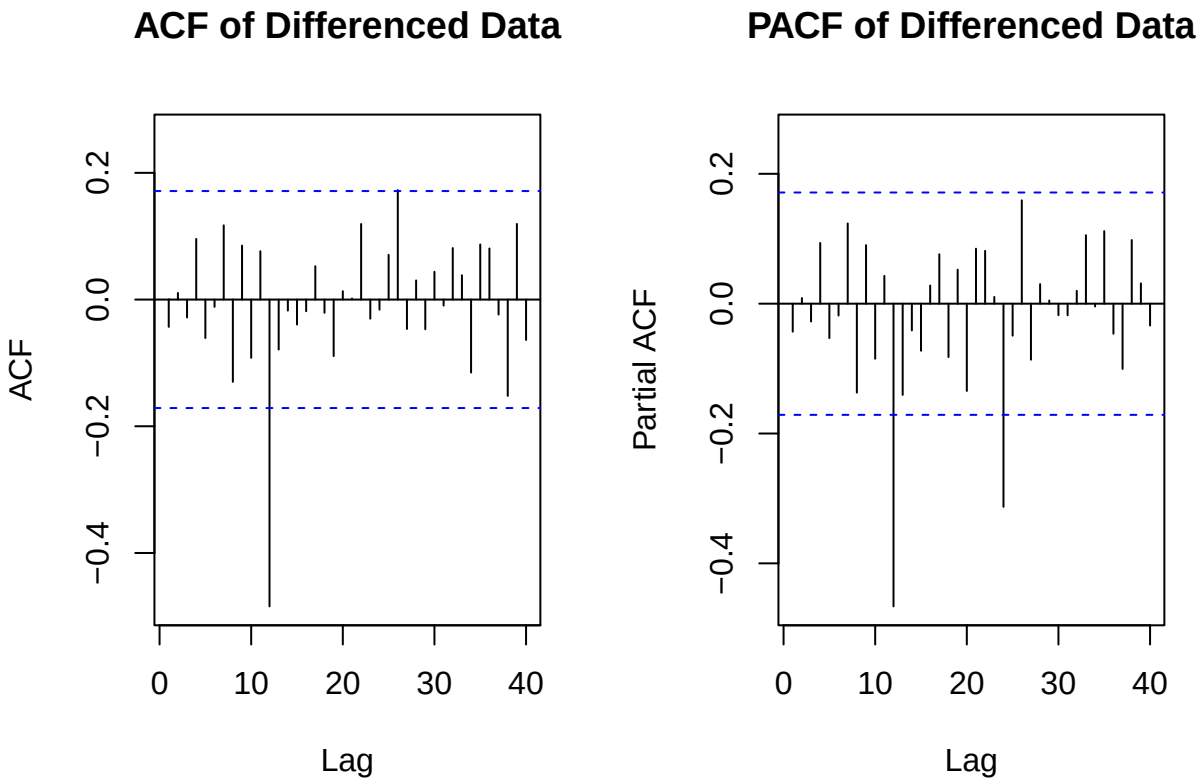


Interpretation:
By comparing the histograms before and after differencing at lag 12, the histogram seems to be more approximately normally distributed after differencing. We will be using the differencing data to do model identification.

3. Model Identification by ACF/PACF

In this step, we will be doing identification the parameters of our SARIMA model.

```
# plot ACF and PACF
par(mfrow=c(1, 2))
Acf(temp_train_12, lag.max = 40, main = "ACF of Differenced Data")
Pacf(temp_train_12, lag.max = 40, main = "PACF of Differenced Data")
```



Now we have estimated the parameters below base on ACF and PACF and we are ready to construct the models.
s = 12 – SARIMA model.
p = 0 – All values are within the first period from PACF.
q = 0 – All values are within the first period from ACF.
d = 0 – No differencing at 1.
D = 1 – Differencing at 12.
P = 2 – There are spikes at 12, 24 from PACF.
Q = 1 – There are spikes at 12 from ACF.

4. Comparing AICc and Choosing Candidate Models

In this step, we are going to create some candidate models base on pervious identification step and compare their AICcs by using AICc() and arima() functions as well as fixing before we choose candidate models for next step.

```
df <- expand.grid(P = 1:2, Q = 0:2 )
df <- cbind(df, AICc = NA)

for (i in 1:nrow(df)) {
  sarima_ob <- NULL
  try(arima_ob <- arima(temp_train, order=c(0, 0, 0),
                        seasonal = list(order = c(df$P[i], 1, df$Q[i]),
                        period = 12), method = "ML"))
  if (!is.null(arima_ob)) { df$AICc[i] <- AICc(arima_ob) }
  }

df[order(df$AICc), ]
```

```
##   P Q     AICc
## 5 1 2 447.5659
## 3 1 1 447.9023
## 4 2 1 449.3769
## 6 2 2 451.3062
## 2 2 0 461.3647
## 1 1 0 476.9722
```

Base on identification step, I will be only picking the models which P = 2 as my candidate models.
Model A: SARIMA(0,0,0)(2,1,1) AICc: 449.3769
Model B: SARIMA(0,0,0)(2,1,2) AICc: 451.3062
Model C: SARIMA(0,0,0)(2,1,0) AICc: 476.9722

5. Estimating Coefficients & Checking Invertibility/Stationarity

In this step, we will be estimating coefficients of every candidate model, fixing their coefficient if needed, and check their invertibility and stationarity.

5a. Model A

$$SARIMA(0,0,0)(2,1,1)_{12}$$

i. Estimating Coefficients

```
#estimating coefficients for model A
model_A <- arima(temp_train, order = c(0, 0, 0),
                 seasonal = list(order = c(2, 1, 1), period = 12), method = "ML")
model_A
```

```
##
## Call:
## arima(x = temp_train, order = c(0, 0, 0), seasonal = list(order = c(2, 1, 1),
##     period = 12), method = "ML")
##
## Coefficients:
##          sar1     sar2     sma1
##       -0.0787  -0.0886  -0.8966
## s.e.   0.1232   0.1115   0.1974
##
## sigma^2 estimated as 1.437:  log likelihood = -220.6,  aic = 449.2
```

As we can see, there is a coefficient that its absolute value is close to 1. So, we will proceed fixed() to model A.

ii. Fixing Coefficients

```
#fixing model A
model_A_fixed <- arima(temp_train, order = c(0, 0, 0),
                       seasonal = list(order = c(2, 1, 1), period = 12),
                       fixed = c(NA, 0, 0), method = "ML")
```

```
## Warning in arima(temp_train, order = c(0, 0, 0), seasonal = list(order = c(2, :
## some AR parameters were fixed: setting transform.pars = FALSE
```

```
model_A_fixed
```

```
##
## Call:
## arima(x = temp_train, order = c(0, 0, 0), seasonal = list(order = c(2, 1, 1),
##     period = 12), fixed = c(NA, 0, 0), method = "ML")
##
## Coefficients:
##          sar1  sar2  sma1
##       -0.5098     0     0
## s.e.   0.0744     0     0
##
## sigma^2 estimated as 2.106:  log likelihood = -236.47,  aic = 476.94
```

```
#AICc for model A
AICc(arima(temp_train, order = c(0, 0, 0),
           seasonal = list(order = c(2, 1, 1), period = 12),
           fixed = c(NA, 0, 0), method = "ML"))
```

```
## Warning in arima(temp_train, order = c(0, 0, 0), seasonal = list(order = c(2, :
## some AR parameters were fixed: setting transform.pars = FALSE
```

```
## [1] 477.1165
```

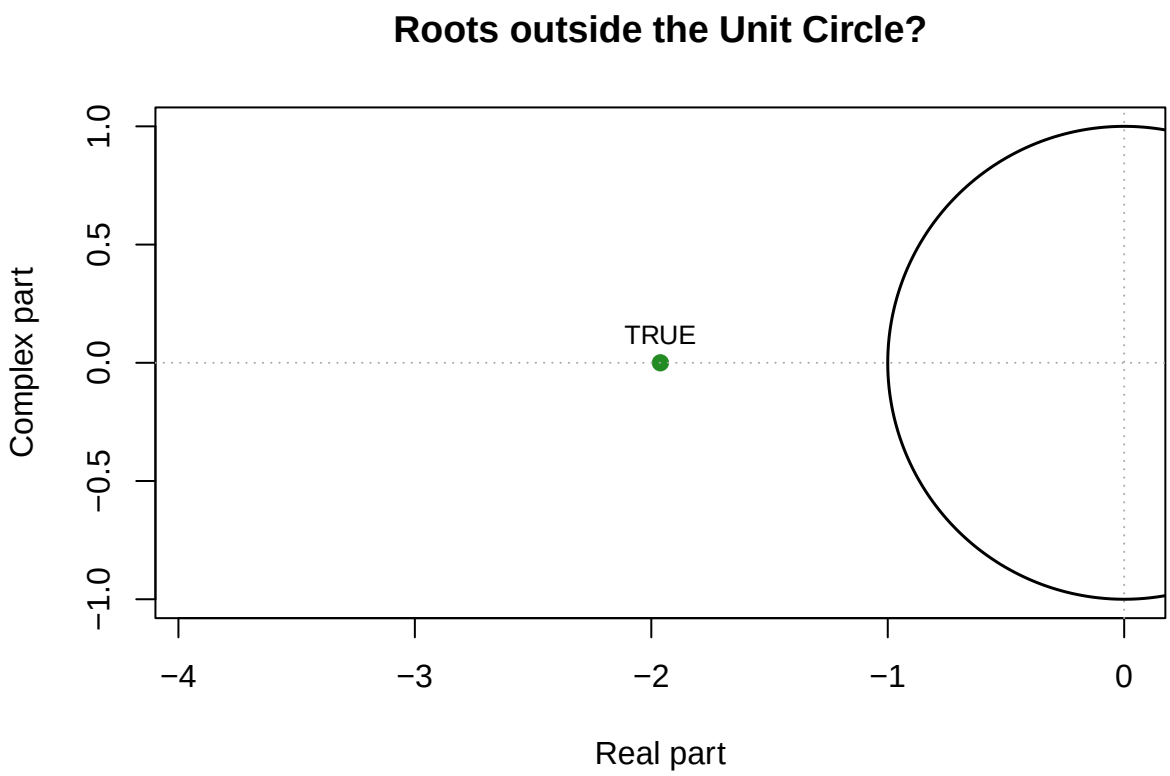
Now, the coefficient looks good, and we will apply it to expand model A.

iii. Model Expansion with Coefficients

$$(1 + 0.5098B^{12})X_t = Z_t$$

iv. Checking Invertibility/Stationarity

```
##        real complex outside
## 1 -1.961554       0    TRUE
## *Results are rounded to 6 digits.
```



Here, we will only check if the root is outside the unit circle for stationary since the data is pure AR model as we mentioned in step 1. AR model is always invertible. As we can see the result above, the root is outside the unit circle. Therefore, model A after fixed is stationary and invertible with AICc: 477.1165.

5b. Model B

$$SARIMA(0,0,0)(2,1,2)_{12}$$

i. Estimating Coefficients

```
#estimating coefficients for model B
model_B <- arima(temp_train, order = c(0, 0, 0),
                 seasonal = list(order = c(2, 1, 2), period = 12), method = "ML")
model_B
```

```
##
## Call:
## arima(x = temp_train, order = c(0, 0, 0), seasonal = list(order = c(2, 1, 2),
##     period = 12), method = "ML")
##
## Coefficients:
```

```
##          sar1     sar2     sma1     sma2
##       -0.5916  -0.1061  -0.3887  -0.4899
## s.e.   0.8316   0.1134   0.8409   0.8322
##
## sigma^2 estimated as 1.421:  log likelihood = -220.51,  aic = 451.02
```

ii. Fixing Coefficients

```
#fixing model B
model_B_fixed <- arima(temp_train, order = c(0, 0, 0),
                       seasonal = list(order = c(2, 1, 2), period = 12),
                       fixed = c(NA, 0, NA, NA), method = "ML")
```

```
## Warning in arima(temp_train, order = c(0, 0, 0), seasonal = list(order = c(2, :
## some AR parameters were fixed: setting transform.pars = FALSE
```

```
## Warning in arima(temp_train, order = c(0, 0, 0), seasonal = list(order = c(2, :
## possible convergence problem: optim gave code = 1
```

```
model_B_fixed
```

```
##
## Call:
## arima(x = temp_train, order = c(0, 0, 0), seasonal = list(order = c(2, 1, 2),
##     period = 12), fixed = c(NA, 0, NA, NA), method = "ML")
##
## Coefficients:
##         sar1  sar2     sma1    sma2
##       0.5362     0  -2.0679  0.9938
## s.e.  0.2574     0   0.7626  0.7291
##
## sigma^2 estimated as 0.8486:  log likelihood = -220.51,  aic = 449.02
```

```
#AICc for model B
AICc(arima(temp_train, order = c(0, 0, 0),
           seasonal = list(order = c(2, 1, 2), period = 12), method = "ML"))
```

```
## [1] 451.3062
```

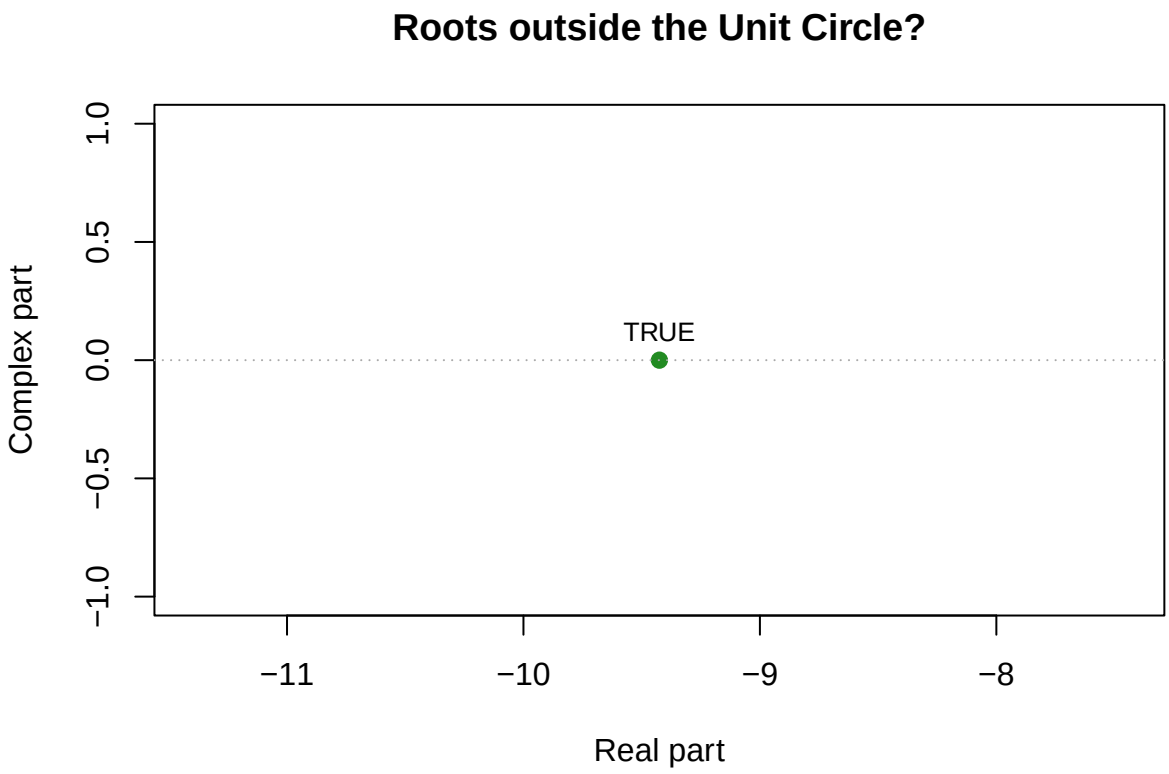
After fixed model B, we get a coefficient which its absolute value is greater than 1. So, we might not use the fixed coefficients but the original coefficients to expand the model.

iii. Model Expansion with Coefficients

$$(1 + 0.1061B^{24})(1 - B^{12})X_t = Z_t$$

iv. Checking Invertibility/Stationarity

```
##        real complex outside
## 1 -9.425071       0    TRUE
## *Results are rounded to 6 digits.
```



Again, we will only check for stationary because the model is invertible as the data is AR model. As we can see that it is same the result as model A, the root is outside the unit circle. Therefore, model B stationary and invertible. However, all the absolute values of coefficients must be smaller than 1 for linear model. We might drop model B from this step since there is a large absolute value of coefficient after fixed.

5c. Model C

$$SARIMA(0,0,0)(2,1,0)_{12}$$

i. Estimating Coefficients

```
#estimating coefficients for model C
model_C <- arima(temp_train, order = c(0, 0, 0),
                 seasonal = list(order=c(2, 1, 0), period = 12), method = "ML")
model_C
```

```
##
## Call:
## arima(x = temp_train, order = c(0, 0, 0), seasonal = list(order = c(2, 1, 0),
##     period = 12), method = "ML")
##
## Coefficients:
##          sar1     sar2
##       -0.7041  -0.3829
## s.e.   0.0826   0.0849
##
## sigma^2 estimated as 1.788:  log likelihood = -227.64,  aic = 461.28
```

ii. Fixing Coefficients

```
#fixing model C
model_C_fixed <- arima(temp_train, order = c(0, 0, 0),
                       seasonal = list(order = c(2, 1, 0), period = 12),
                       fixed = c(NA,0), method = "ML")
```

```
## Warning in arima(temp_train, order = c(0, 0, 0), seasonal = list(order = c(2, :
## some AR parameters were fixed: setting transform.pars = FALSE
```

```
model_C_fixed
```

```
##
## Call:
## arima(x = temp_train, order = c(0, 0, 0), seasonal = list(order = c(2, 1, 0),
##     period = 12), fixed = c(NA, 0), method = "ML")
##
## Coefficients:
##          sar1  sar2
##       -0.5098     0
## s.e.   0.0744     0
##
## sigma^2 estimated as 2.106:  log likelihood = -236.47,  aic = 476.94
```

```
#AICc for model C
AICc(arima(temp_train,order = c(0, 0, 0),
           seasonal = list(order = c(2, 1, 0), period = 12),
           fixed = c(NA, 0), method = "ML"))
```

```
## Warning in arima(temp_train, order = c(0, 0, 0), seasonal = list(order = c(2, :
## some AR parameters were fixed: setting transform.pars = FALSE
```

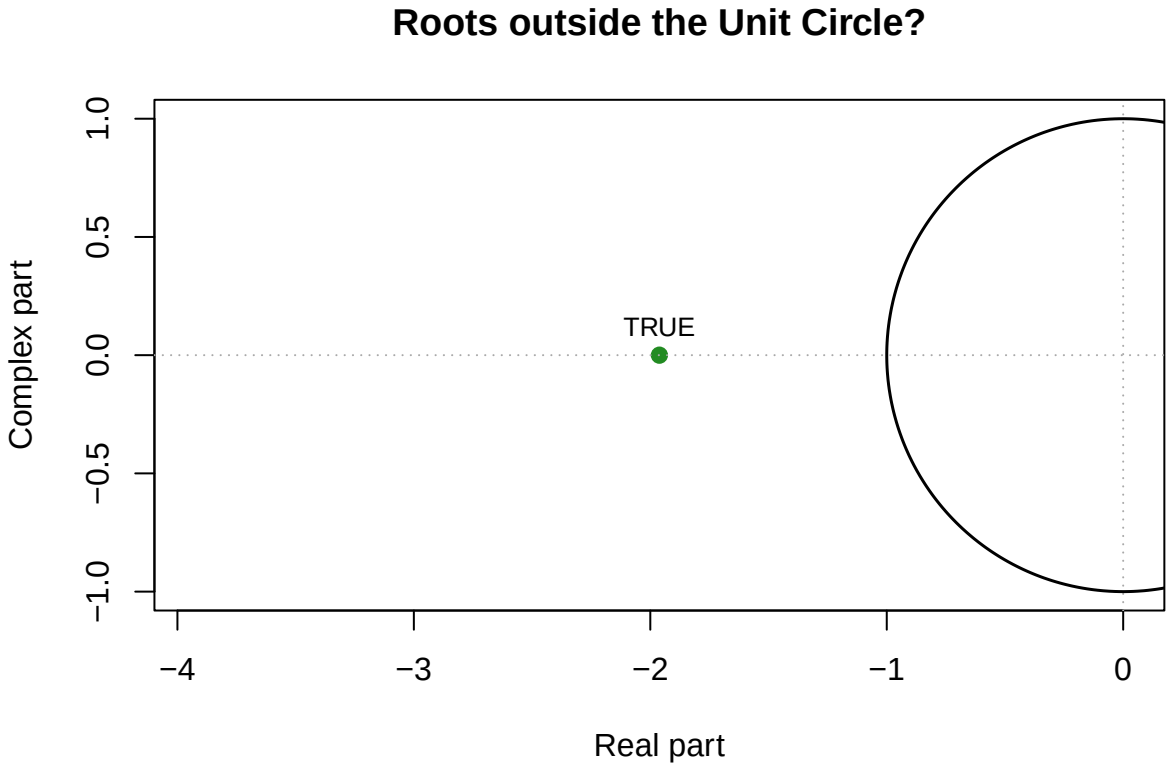
```
## [1] 477.0295
```

iii. Model Expansion with Coefficients

$$(1 + 0.5098B^{12})X_t = Z_t$$

iv. Checking Invertibility/Stationarity

```
##        real complex outside
## 1 -1.961554       0    TRUE
## *Results are rounded to 6 digits.
```



Finally, the root for model C is also outside the unit circle, model C is also stationary and invertible as the original data is AR model.

5d. Summary

1. We will drop model B from here as there is a large absolute value of coefficient after fixed.

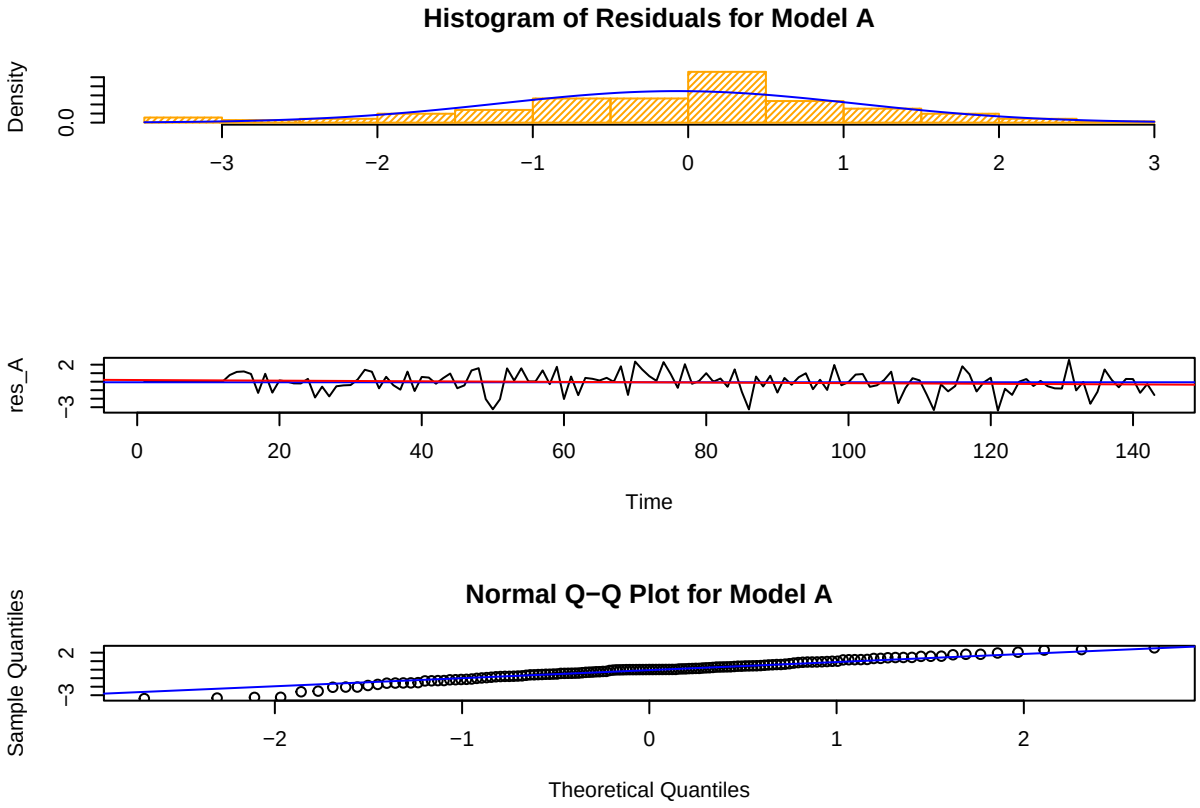
2. Surprisingly, model A and model C have the exact same coefficient after fixed, and their AICcs are very close as model A (477.1165) is just slightly greater than Model C (477.0295). This could be the point to choose the model C as the final model, but we will do diagnostic checking for both model A and C for comparison purpose.

6. Diagnostic Checking

In this step, we will do diagnostic check for model A and model C to choose our final model for forecasting.

6a. Model A

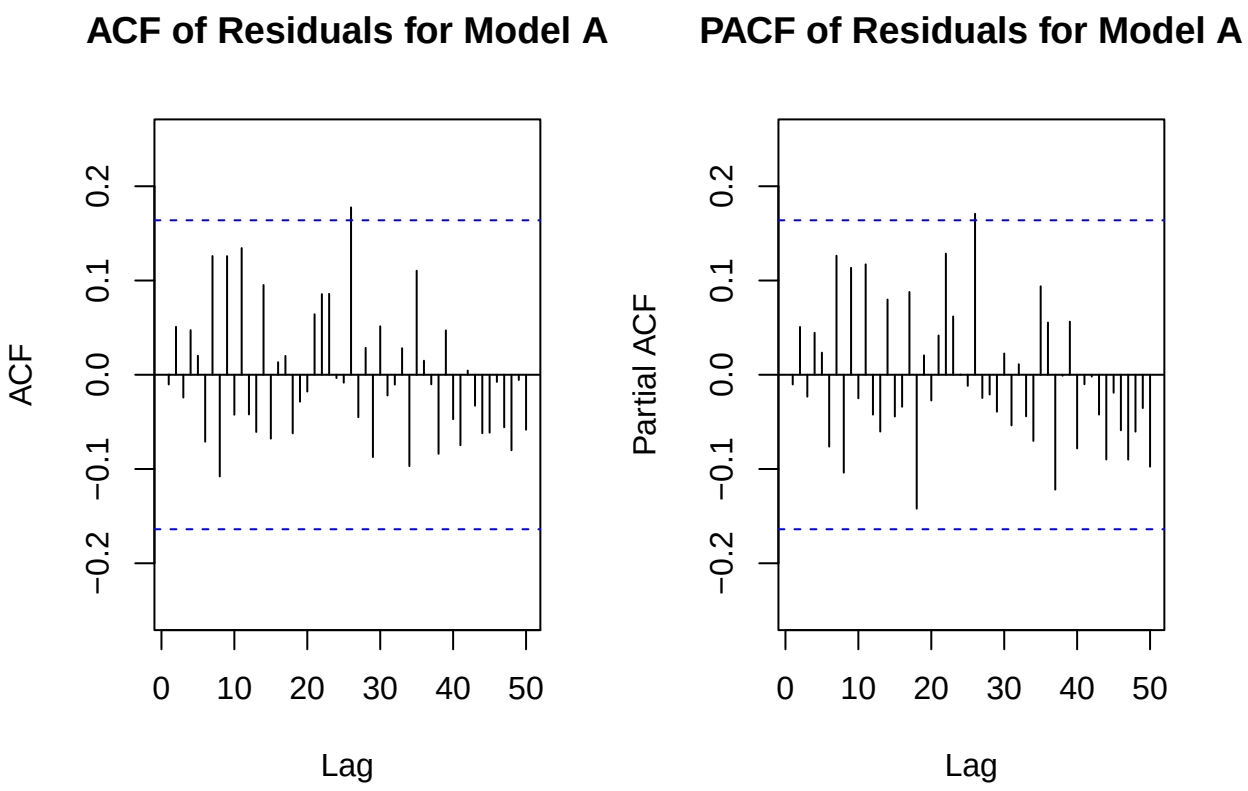
i. Residuals, Histogram, and Q-Q Plot



Interpretation:
1. There is no trend and have a stable mean.
2. The histogram of model A seems approximately normal.
3. The Q-Q plot is not bad, most points fall in the regression line, but there are some outliers on the left trail.

Overall, the residuals of model A looks ok.

ii. ACF/PACF of Residuals



Interpretation:
Although there is a lag in both ACF and PACF plot that crossing the confidence intervals. However, those lags are nearly 0.05, so we are able to count all the spikes as zero. Therefore, the residuals for Model A can be counted as white noise. In the next step we will be doing Normality test, and all the box tests to see if model A passes them all.

iii. Shapiro-Wilk Normality Test, Box-Pierce Test, Box-Ljung Test, McLeod-Li Test

```
##
##  Shapiro-Wilk normality test
##
## data:  res_A
## W = 0.97706, p-value = 0.01668
```

```
##
##  Box-Pierce test
##
## data:  res_A
## X-squared = 10.856, df = 11, p-value = 0.4554
```

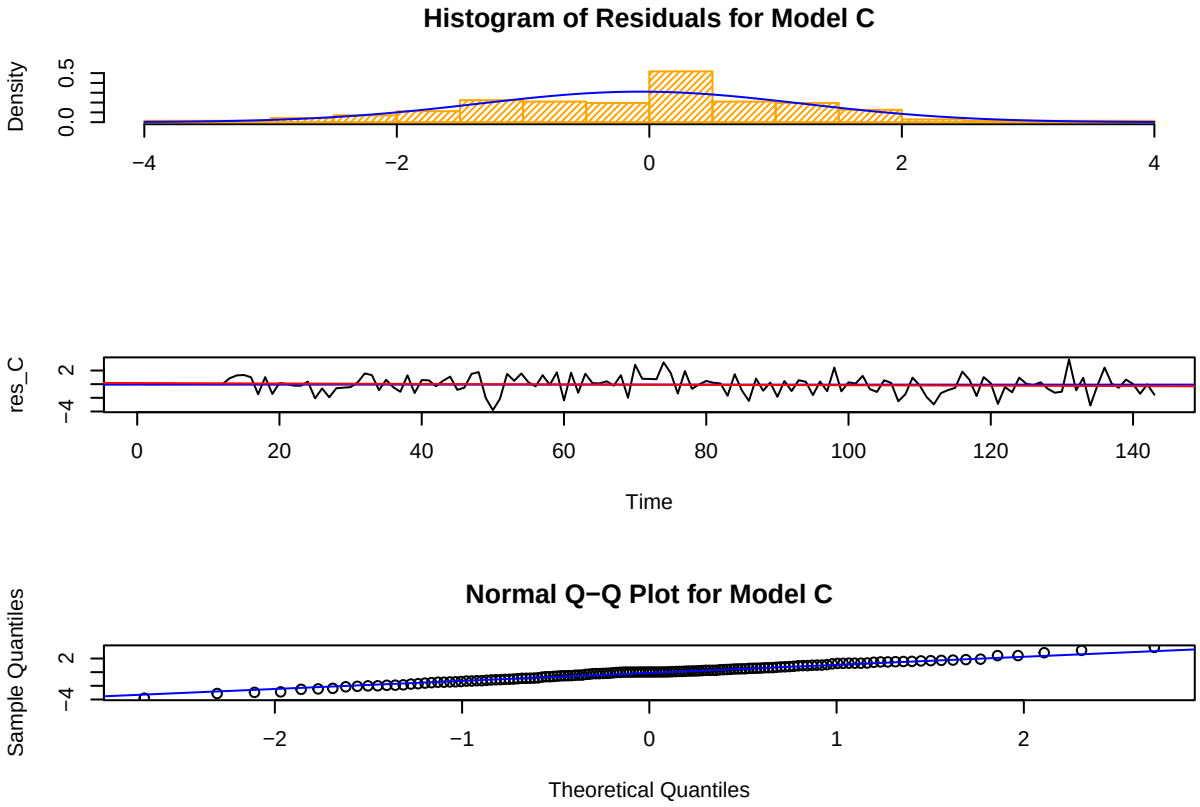
```
##
##  Box-Ljung test
##
## data:  res_A
## X-squared = 11.693, df = 11, p-value = 0.3872
```

```
##
##  Box-Ljung test
##
## data:  (res_A)^2
## X-squared = 17.221, df = 12, p-value = 0.1415
```

Model A passed all the box tests with p-values that are greater than 0.05, but failed to pass Shapiro-Wilk normality test which indicates that its residual does not have normality.

6b. Model C

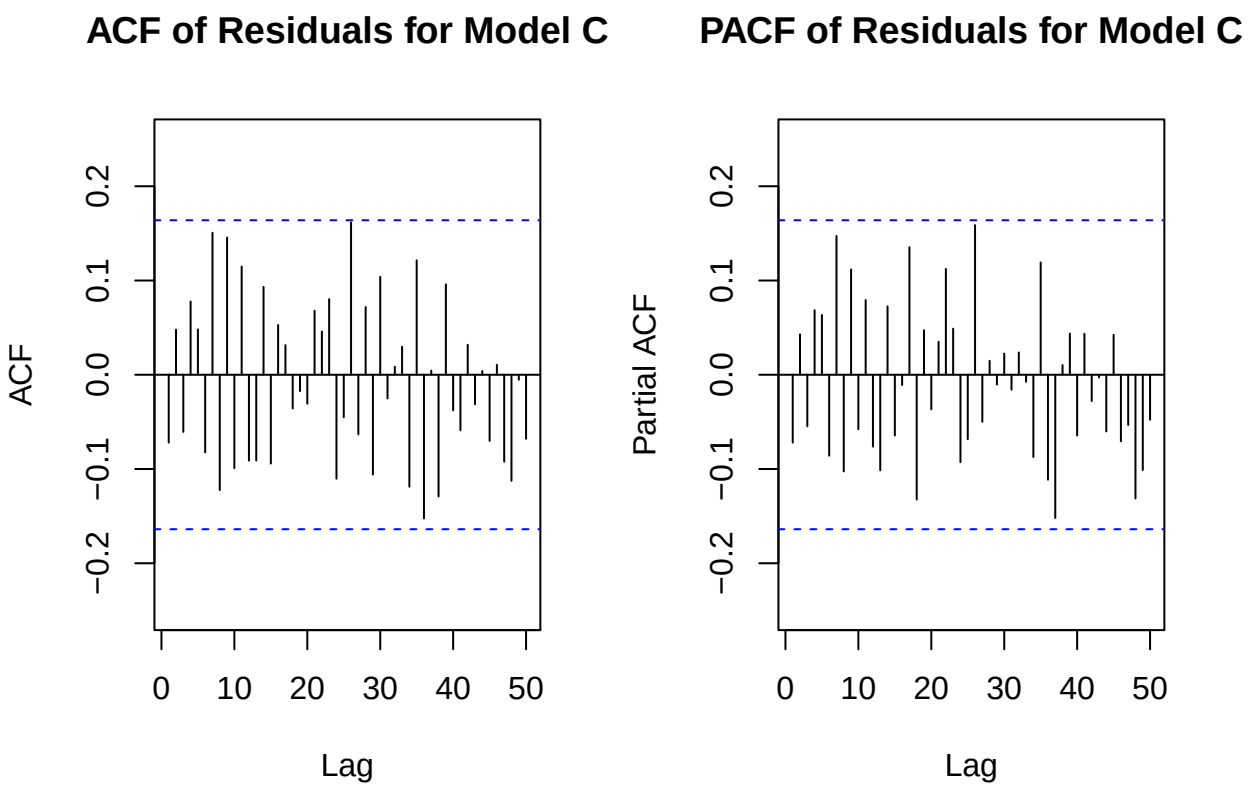
i. Residuals, Histogram, and Q-Q Plot



Interpretation:
1. There is no trend and have a stable mean.
2. The histogram of model C seems approximately normal.
3. The Q-Q plot is great, all points fall in the regression line with no obvious outliers.

Overall, the residuals of model C looks pretty good.

ii. ACF/PACF of Residuals



Interpretation:
All lags are within the confidence interval. The residuals of Model 2 can be counted as a white noise.

iii. Shapiro-Wilk Normality Test, Box-Pierce Test, Box-Ljung Test, McLeod-Li Test

```
##
##  Shapiro-Wilk normality test
##
## data:  res_C
## W = 0.99207, p-value = 0.608
```

```
##
##  Box-Pierce test
##
## data:  res_C
## X-squared = 16.658, df = 11, p-value = 0.1184
```

```
##
##  Box-Ljung test
##
## data:  res_C
## X-squared = 17.875, df = 11, p-value = 0.08453
```

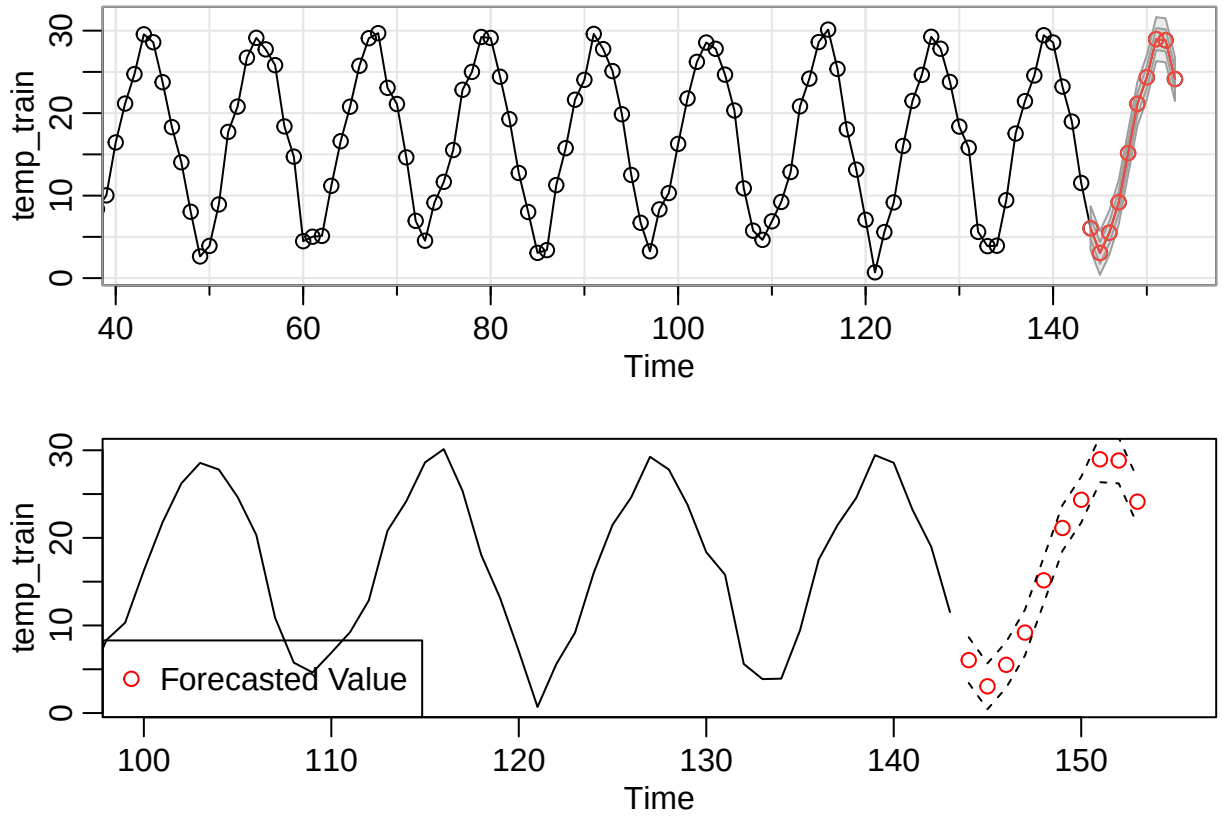
```
##
##  Box-Ljung test
##
## data:  (res_C)^2
## X-squared = 14.83, df = 12, p-value = 0.2509
```

Model C passed all the tests. Passing Shapiro-Wilk Normality Test means the residual of model C has normality. Passing all the box tests indicates that there is no auto-correlation for model C. Therefore, model C will be used to do forecasting in the next step.

7. Forecasting Using Model C

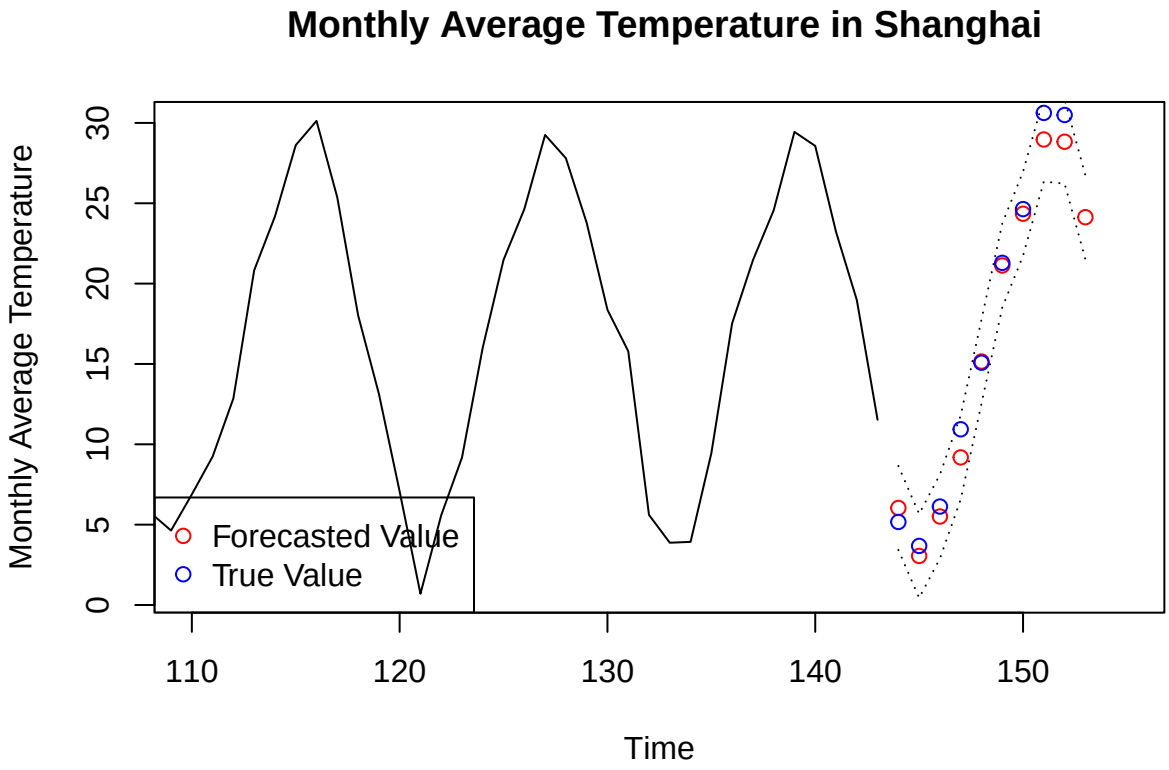
In this step, we will be doing forecasting of the original data using model C and comparing them with true values.

7a. Forecast of Original Data Using Model C



Interpretation:
1. In the first graph, the 10 red circles are forecasted values.
2. To have a better view, we zoom in the plot into the last part which shown in the second graph to see every red circle's locations. And we can clearly see that all forecasted values are within the predicted interval. Therefore, with this strong evidence above, we can say that model C is reasonable and suitable.

7b. Comaring the Original Data Ooints with Forecast Values



Interpretation:
1. As the graph shown above, red circles are forecasted values and blue circles are true values from original data.
2. Again, all forecasted values are with predicted interval, and most of values are very close to true values. Overall, model C did a very good job on forecasting those values that are within predicted interval as well as close to true values mostly.

8. Conclusion

8a. Final Model

After all essential steps, our final model is:

$$SARIMA(0,0,0)(2,1,0)_{12}$$

After fixed the coefficient and expanded, we have:

$$(1 + 0.5098B^{12})X_t = Z_t$$

Now, we have completed all the procedures of this project.

8b. Takeaways

a. At the early stage, we found the data set happended to be an AR model which could save times checking invertibility.

b. In box-cox transformation step, we have 1 within the confidence interval, so we dropped the transformed data and used the original data. Luckily, our final model passed all the test, otherwise we would have to use the transformed data and re-select candidates model to re-do all the tests again.

c. Surprisingly, two of the candidate models are exact same after fixed coefficients and their AICcs are very close too. By comparing their AICcs we could have concluded that the model that has smaller AICc to be the final model. However, we decided to do diagnostic check on both models, and ended up still having the model that has lower AICs to pass all the test and became the final model.

Special thanks to Zongyi Han and Wentao Yu for helping me out for this project. I believe that the result has definitely met the main purpose of this project.

References

1. All lecture slides and notes from PSTAT 174/274 by Professor Raya Feldman.

2. Climate Change: Earth Surface Temperature Data.
   https://www.kaggle.com/datasets/berkeleyearth/climate-change-earth-surface-temperature-data?select=

3. [R] Fixed parameters in an AR (or arima) model.
   https://stat.ethz.ch/pipermail/r-help/2004-January/044449.html

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
#load necessary library
library(forecast)
library(ggplot2)
library(UnitCircle)
library(qpcR)
library(forecast)
library(MASS)
library(astsa)
library(lubridate)
library(dplyr)
library(xts)
#loading data
sh_temp_raw <- read.table("shanghai_temp.csv", sep=",",
                      header = TRUE, skip = 0)
#ts data
sh_temp <- ts(sh_temp_raw$AverageTemperature, frequency = 12)

# splitting the data into training and testing sets
temp_train  <-  sh_temp[c(1:143)]
temp_test  <-  sh_temp[c(144:153)] # leave the last 10 data to be observed
# plot training data set
plot.ts(temp_train, main = "Training Data")

fit <- lm(temp_train ~ as.numeric(1:length(temp_train)))

# add trend line and mean line to training data
abline(fit, col = "red")
abline(h = mean(sh_temp_raw$AverageTemperature), col = "blue")

# plot the histogram
hist(temp_train, freq = F, main = "Histogram of Training Data", density = 40, col = "orange")
m <- mean(temp_train)
std <- sqrt(var(temp_train))
curve(dnorm(x, m, std), col = "blue", add = TRUE, yaxt = "n")

# plot ACf and PACF
par(mfrow = c(1, 2))
acf(temp_train, lag.max = 100, main = "ACF of Training Data")
pacf(temp_train, lag.max = 100, main = "PACF of Training Data")

#variance of original data
var(temp_train)

# using Box-Cox transformation
bc_Transform <- boxcox(temp_train ~ as.numeric(1:length(temp_train)),
                       lambda = seq(-1, 3), plotit=TRUE)

lambda <- bc_Transform$x[which.max(bc_Transform$y)]
lambda
```

```r
temp_train_BC <- (1/lambda)*(temp_train^lambda - 1)


# decomposition of original data
temp_new <- ts(temp_train, frequency = 12)
decomp <- decompose(temp_new)
plot(decomp)
# create a table to keep tracking of variance during differencing process
var.table <- matrix(ncol = 2, nrow = 3)
colnames(var.table) <- c("Differencing", "Variance")

var.table <- as.table(var.table)

# No differencing
fit <- lm(temp_train ~ as.numeric(1:length(temp_train)));
plot.ts(temp_train)
abline(reg = fit, col = "red")
abline(h = mean(temp_train), col = "blue")
# add variance to table
var.table["A", "Differencing"] = "None"
var.table["A", "Variance"] = var(temp_train)

var.table

# difference at 12 to remove the seasonality
temp_train_12 <- diff(temp_train, lag = 12)
plot.ts(temp_train_12, main = "Differencing at Lag 12")
fit_diff_12 <- lm(temp_train_12 ~ as.numeric(1:length(temp_train_12)))

abline(reg = fit_diff_12, col = "red")
abline(h=mean(temp_train_12), col = "blue")
# add the variance after differencing at lag 12 to table
var.table["B", "Differencing"] = "lag_12"
var.table["B", "Variance"] = var(temp_train_12, na.rm = TRUE)

var.table
# difference at 1
temp_train_1 <- diff(temp_train_12, lag=1)
plot.ts(temp_train_1, main = "Differencing at Lag 1")
fit_diff_1 <- lm(temp_train_1 ~ as.numeric(1:length(temp_train_1)))
abline(reg = fit_diff_1, col = "red")
abline(h = mean(temp_train_1), col = "blue")

# add the variance after differencing at lag 1&12 to table
var.table["C", "Differencing"] = "lag_12&1"
var.table["C", "Variance"] = var(temp_train_1,na.rm=TRUE)

var.table
# compare histogram before and after differecing
par(mfrow=c(1, 2))
hist(temp_train, freq = F, main = "Histogram Before Differencing",
     col = "orange", density = 40)
hist(temp_train_12, freq = F, main = "Histogram After Differencing",
```

```
     col = "orange", density = 40)

# plot ACF and PACF
par(mfrow=c(1, 2))
Acf(temp_train_12, lag.max = 40, main = "ACF of Differenced Data")
Pacf(temp_train_12, lag.max = 40, main = "PACF of Differenced Data")

df <- expand.grid(P = 1:2, Q = 0:2 )
df <- cbind(df, AICc = NA)

for (i in 1:nrow(df)) {
  sarima_ob <- NULL
  try(arima_ob <- arima(temp_train, order=c(0, 0, 0),
                        seasonal = list(order = c(df$P[i], 1, df$Q[i]),
                        period = 12), method = "ML"))
  if (!is.null(arima_ob)) { df$AICc[i] <- AICc(arima_ob) }
  }

df[order(df$AICc), ]

#estimating coefficients for model A
model_A <- arima(temp_train, order = c(0, 0, 0),
                 seasonal = list(order = c(2, 1, 1), period = 12), method = "ML")
model_A
#fixing model A
model_A_fixed <- arima(temp_train, order = c(0, 0, 0),
                       seasonal = list(order = c(2, 1, 1), period = 12),
                       fixed = c(NA, 0, 0), method = "ML")
model_A_fixed

#AICc for model A
AICc(arima(temp_train, order = c(0, 0, 0),
           seasonal = list(order = c(2, 1, 1), period = 12),
           fixed = c(NA, 0, 0), method = "ML"))
#check stationarity
uc.check(pol_ = c(1, 0.5098), plot_output = TRUE)

#estimating coefficients for model B
model_B <- arima(temp_train, order = c(0, 0, 0),
                 seasonal = list(order = c(2, 1, 2), period = 12), method = "ML")
model_B

#fixing model B
model_B_fixed <- arima(temp_train, order = c(0, 0, 0),
                       seasonal = list(order = c(2, 1, 2), period = 12),
                       fixed = c(NA, 0, NA, NA), method = "ML")
model_B_fixed

#AICc for model B
AICc(arima(temp_train, order = c(0, 0, 0),
           seasonal = list(order = c(2, 1, 2), period = 12), method = "ML"))

#check stationarity
```

```r
uc.check(pol_ = c(1, 0.1061), plot_output = TRUE)

#estimating coefficients for model C
model_C <- arima(temp_train, order = c(0, 0, 0),
                 seasonal = list(order=c(2, 1, 0), period = 12), method = "ML")
model_C

#fixing model C
model_C_fixed <- arima(temp_train, order = c(0, 0, 0),
                       seasonal = list(order = c(2, 1, 0), period = 12),
                       fixed = c(NA,0), method = "ML")
model_C_fixed

#AICc for model C
AICc(arima(temp_train,order = c(0, 0, 0),
           seasonal = list(order = c(2, 1, 0), period = 12),
           fixed = c(NA, 0), method = "ML"))
#check stationarity
uc.check(pol_ = c(1, 0.5098), plot_output = TRUE)

#plot residuals of Model A
par(mfrow=c(3, 1))
fit_A <- arima(temp_train, order=c(0, 0, 0),
                 seasonal = list(order = c(2, 1, 1), period = 12), method="ML")

res_A <- residuals(fit_A)
m_A <- mean(res_A)
std_A <- sqrt(var(res_A))
nt = length(sh_temp_raw$AverageTemperature)

#plot histogram of Model A
hist(res_A, density = 40, breaks = 20, col = "orange", xlab = " ", prob=TRUE,
     main = "Histogram of Residuals for Model A")
curve(dnorm(x, m_A, std_A), col = "blue", add = TRUE, yaxt = "n")

plot.ts(res_A)
abline(h = mean(res_A), col = "blue")

#plot Q-Q of Model A
fit_AA <- lm(res_A ~ as.numeric(1:length(temp_train)))
abline(fit_AA, col = "red")

qqnorm(res_A, main = "Normal Q-Q Plot for Model A")
qqline(res_A, col = "blue")
# plot ACF and PACF of residuals for Model A
par(mfrow=c(1,2))
Acf(res_A, lag.max = 50, main = "ACF of Residuals for Model A")
Pacf(res_A, lag.max = 50, main = "PACF of Residuals for Model A")

#Shapiro-Wilk Normality Test, Box-Pierce Test, Box-Ljung Test, McLeod-Li Test;
#Model A: 143 obs -> lag =  sqrt(143) = 12; 1 coefficient -> fitdf = 1.
shapiro.test(res_A)
Box.test(res_A, lag=12, type=c("Box-Pierce"), fitdf=1)
```

```r
Box.test(res_A, lag=12, type=c("Ljung-Box"), fitdf=1)
Box.test((res_A)^2, lag=12, type=c("Ljung-Box"), fitdf=0)
# plot residuals of Model C
par(mfrow=c(3, 1))
fit_C <- arima(temp_train, order=c(0, 0, 0),
                seasonal=list(order=c(2, 1, 0), period=12), method="ML")


res_C <- residuals(fit_C)
m_C <- mean(res_C)
std_C <- sqrt(var(res_C))
nt = length(sh_temp_raw$AverageTemperature)

# plot histogram of Model C
hist(res_C, density=40, breaks = 20, col = "orange", xlab = " ", prob=TRUE,
     main = "Histogram of Residuals for Model C")
curve(dnorm(x, m_C, std_C), col = "blue", add = TRUE, yaxt = "n")

plot.ts(res_C)
abline(h = mean(res_C), col = "blue")

# plot Q-Q of Model C
fit_CC <- lm(res_C ~ as.numeric(1:length(temp_train)))
abline(fit_CC,col = "red")

qqnorm(res_C, main = "Normal Q-Q Plot for Model C")
qqline(res_C, col = "blue")
#plot ACF and PACF of residuals for Model C
par(mfrow=c(1,2))
Acf(res_C, lag.max = 50, main = "ACF of Residuals for Model C")
Pacf(res_C, lag.max = 50, main = "PACF of Residuals for Model C")
#Shapiro-Wilk Normality Test, Box-Pierce Test, Box-Ljung Test, McLeod-Li Test for Model C
#Model C: 143 obs -> lag =  sqrt(143) = 12; 1 coefficient -> fitdf = 1.
shapiro.test(res_C)
Box.test(res_C, lag=12, type=c("Box-Pierce"), fitdf=1)
Box.test(res_C, lag=12, type=c("Ljung-Box"), fitdf=1)
Box.test((res_C)^2, lag=12, type=c("Ljung-Box"), fitdf=0)

#plot zoomed in version of forecast of original data
#using model C - SARIMA(0,0,0)x(2,1,0)12
par(mfrow=c(2, 1))
mypred.og = sarima.for(temp_train, n.ahead = 10,
                       p = 0, d = 0, q = 0, P = 2, D = 1, Q = 0, S = 12,
                       no.constant = FALSE, plot.all = F)

ts.plot(temp_train, xlim=c(100, length(temp_train) + 12))

#add points and lines based on prediction
points(144:153, mypred.og$pred,col = "red")
lines(144:153, (mypred.og$pred + 1.96*mypred.og$se), lty = 2)
lines(144:153, (mypred.og$pred - 1.96*mypred.og$se), lty = 2)
legend("bottomleft", fill, pch = 1, col = c("red"),
       legend = c("Forecasted Value"))
```

```r
#forecast of original data with true values on top
ts.plot(temp_train, xlim=c(110, 155),
        ylab = "Monthly Average Temperature",
        main = " Monthly Average Temperature in Shanghai")

#add forecasted values
points(144:153, mypred.og$pred, col = "red")

#add true values
points(144:153, temp_test, col = "blue", pch = 1)

#add legend and prediction intervals
legend("bottomleft", pch=c(1, 1), fill, col = c("red", "blue"),
        legend=c("Forecasted Value", "True Value"))

lines(144:153, (mypred.og$pred + 1.96*mypred.og$se), lty = 3)
lines(144:153, (mypred.og$pred - 1.96*mypred.og$se), lty = 3)
```