Jeffrey Chan Final Project - Weather prediction

Abstract

In this report, we will analyze a dataset called *Daily Minimum Temperatures in Melbourne*[1]. We will use two methods, 1) using 5-day average lowest temperature, and 2) Fourier series to remove the seasonal trend, to fit linear time series models, which can bypass the limitation on the lag number of SARIMA model in R. The results of these two methods are similar, which indicates the validity of both models. In addition, We use both models to forecast the lowest temperature in Melbourne.

Introduction

Weather forecast is one of the most common daily tasks in people's life. It helps people to schedule their daily tasks. There are several factors for weather forecasting, one of which is the daily lowest temperature. Although there are meteorological way for forecasting, we are still interested in learning the possibility of predicting using a mathematics-based method. Fortunately, we have learned time series analysis this quarter, which provides us few useful tools to deal with the problem.

The goal of our project is to use linear time series model to fit and forecast the lowest temperature in Melbourne. We use a dataset *Daily Minimum Temperatures in Melbourne*, which contains 3650 (i.e., 10 years) daily temperature data. A seasonal autoregressive integrated moving average (SARIMA) model is often used to analyze this kind of data. However, the lag of seasonal period cannot be greater than 350 in the R software due to the huge amount of memory it will occupy. Therefore, we propose two methods to avoid this constraint. The first method is to use a 5-day average temperature rather than daily data. Using a 5-day average data will result in a seasonal period of $365/5 = 73$, which is inside the flexible range of R's SARIMA model. Second, we manually remove the seasonal trend and fit the residual using non-seasonal time series model. We will use Fourier series to remove the seasonal trend [2], which is a powerful tool for fitting a curved data and can be easily integrated with ARIMA model. Many libraries have been used in this project: ggplot2, tseries, MASS, knitr, lubridate and forecast. We sincerely thank the authors of these packages. We also thank the R foundation for their contribution to the R community. # Experiments

Load Data

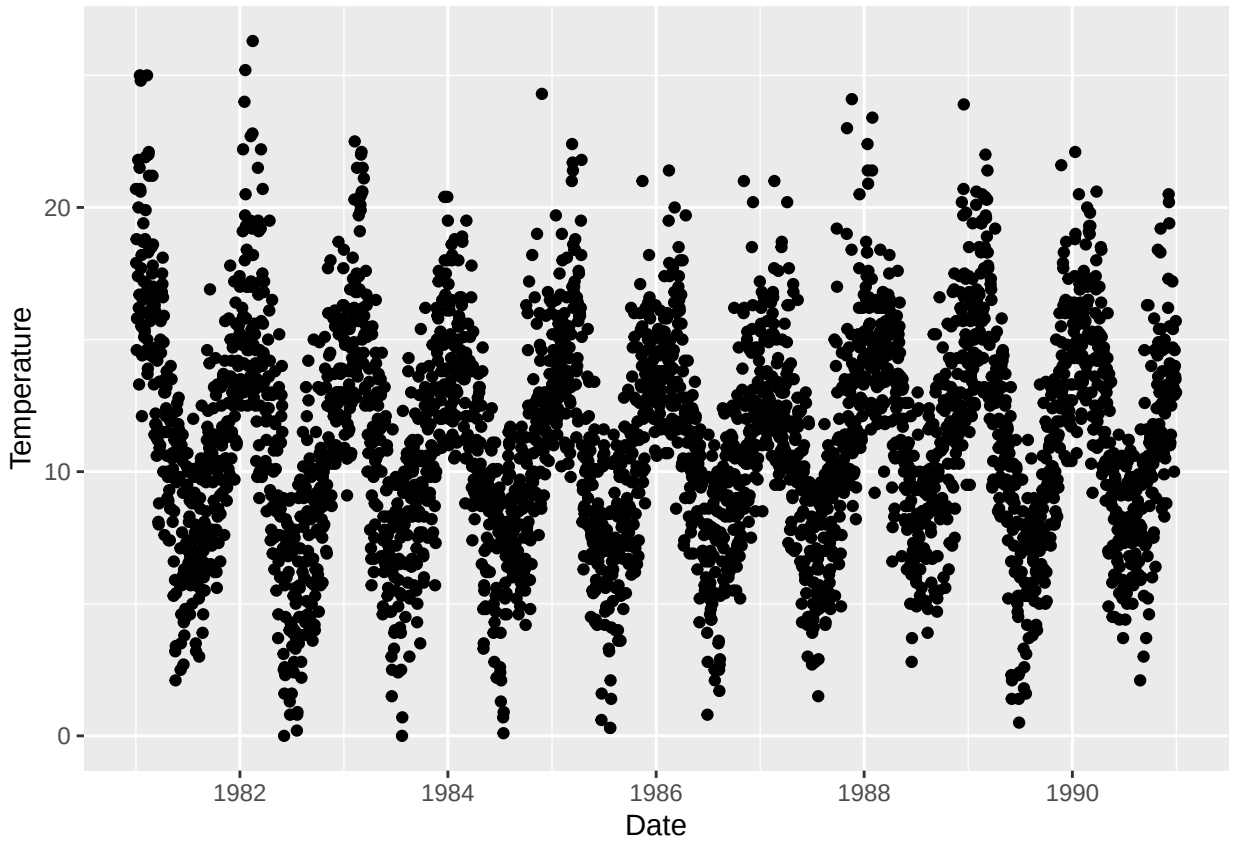
The following code illustrates the procedure to load data.

```
data <- read.csv("~/temp.csv")
colnames(data) <- c("Date", "Temp")
time <- as_date(data$Date)
target <- data$Temp
```

Data Visualization

In order to visually examine the trend and seasonality of the dataset, we first plot the traffic volume against the time.

```
qplot(time, target, ylab="Temperature", xlab="Date")
```



From the plot There is an obvious seasonality in the data, and no obvious trend in the data.

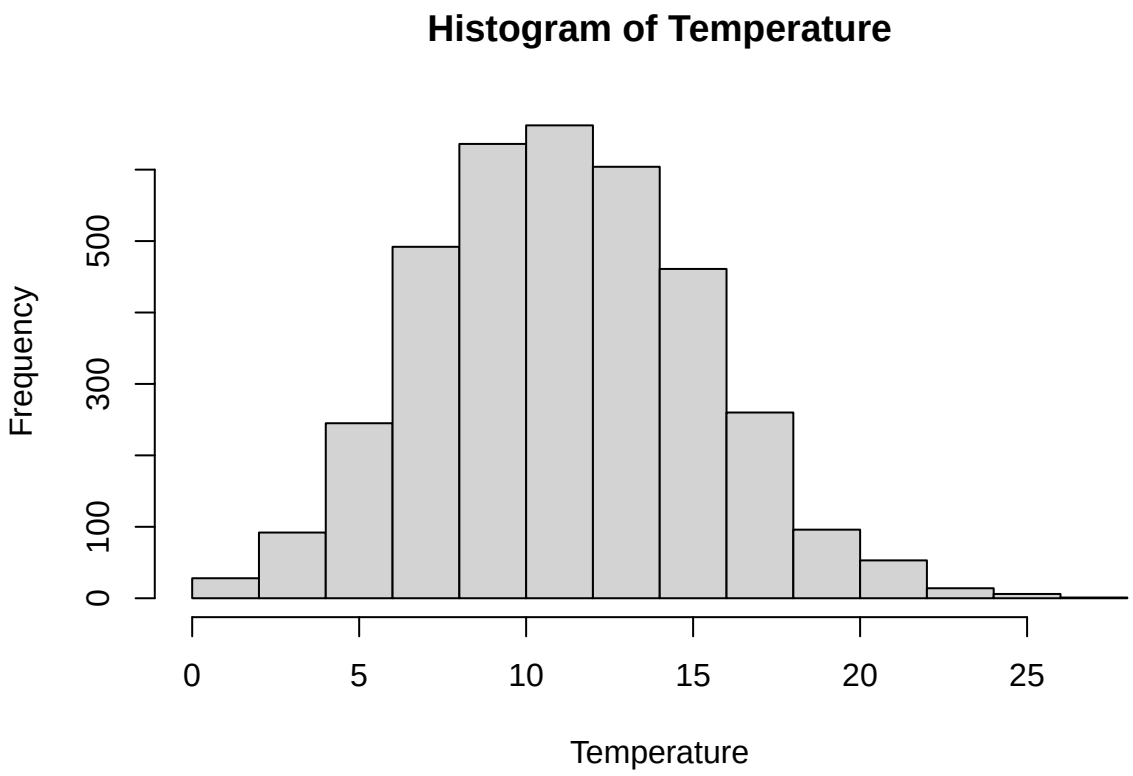
Pre-processing

In this section, we will introduce the pre-processing procedure of the data.

Deciding the Transformation

Firstly, we would like to see if we need a Box-Cox transformation to pre-process the temperature data.

```
options(warn=-1)
ts.target = ts(target, frequency = 365)
hist(ts.target, main = "Histogram of Temperature", xlab = "Temperature")
```



The histogram has showed a great range, also has showed a great variation and the distribution is in an approximate bell shape. According to the plot above, we decide not to transform the data.

Data Aggregation

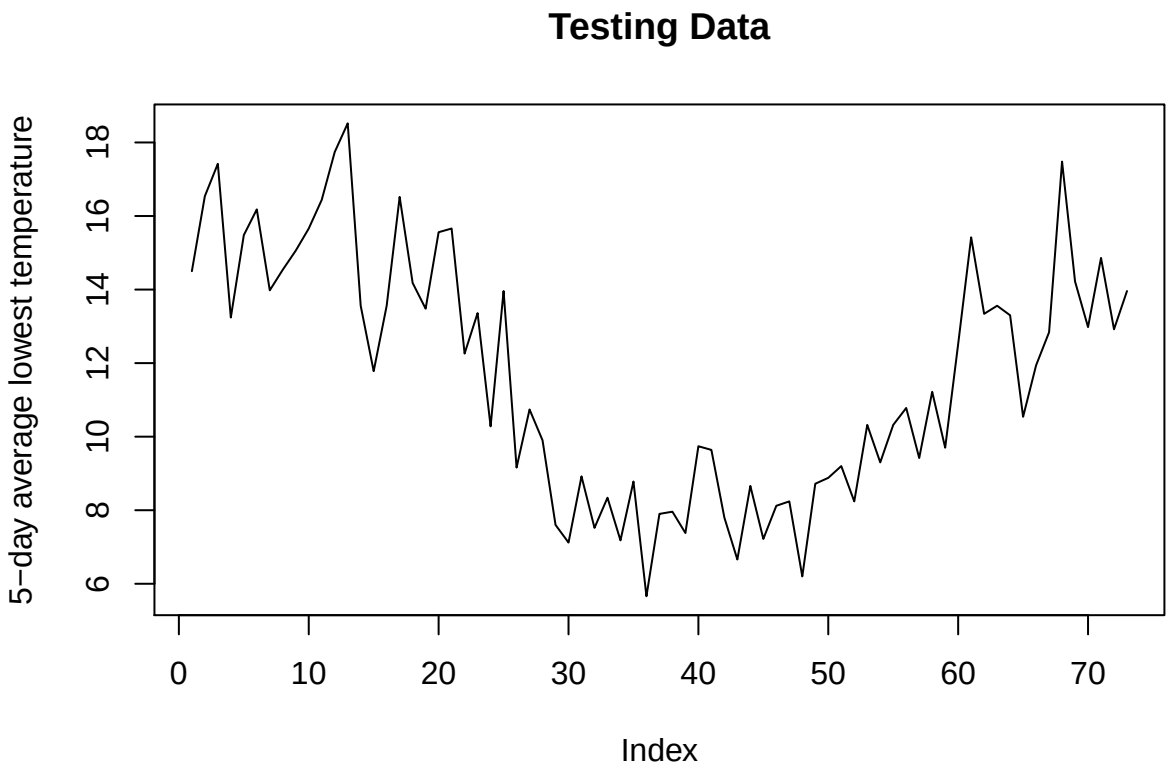
Since the period used in SARIMA model cannot be greater than 365 in R, we proposed two different methods to fit the data: 1) Using a Fourier series to remove the seasonality in the data, and then fit the non-seasonal part with a normal ARIMA model, 2) Combining five days into one day, i.e. $y_t = \frac{(x_{5t}+x_{5t+1}+x_{5t+2}+x_{5t+3}+x_{5t+4})}{5}$, where x_t is the original daily data, and y_t is the newly generated data.

```
aggregated <- (target[seq(1, length(target), 5)] +
                      target[seq(2, length(target), 5)] +
                      target[seq(3, length(target), 5)] +
                      target[seq(4, length(target), 5)] +
                      target[seq(5, length(target), 5)]
                      ) / 5
```

Spliting Dataset

We split the dataset into two parts, a training set and testing set. The testing set is consisted the last 10% data, and the training set are the remaining.

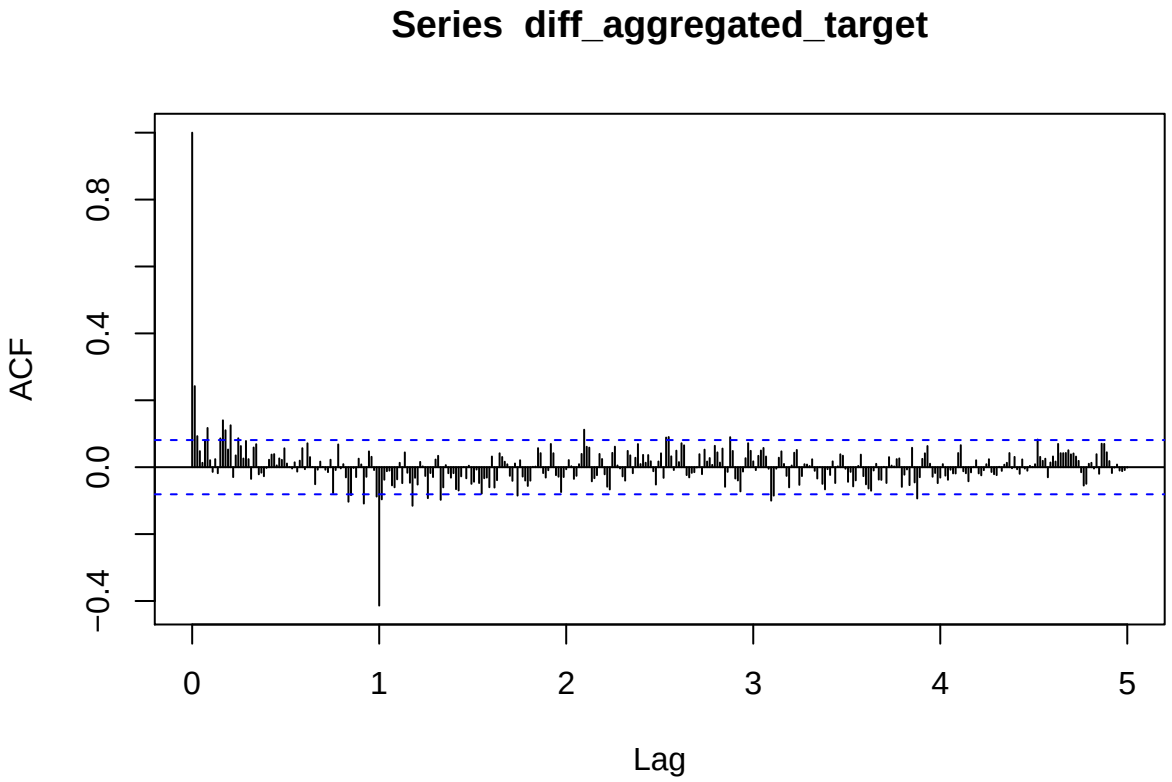
```
aggreated_length <- length(aggregated)
aggregated_target <- aggregated[1:(aggreated_length * 0.9)]
aggregated_test <- aggregated[-(1:(aggreated_length * 0.9))]
plot(aggregated_test, type="l", main="Testing Data", ylab="5-day average lowest temperature")
```



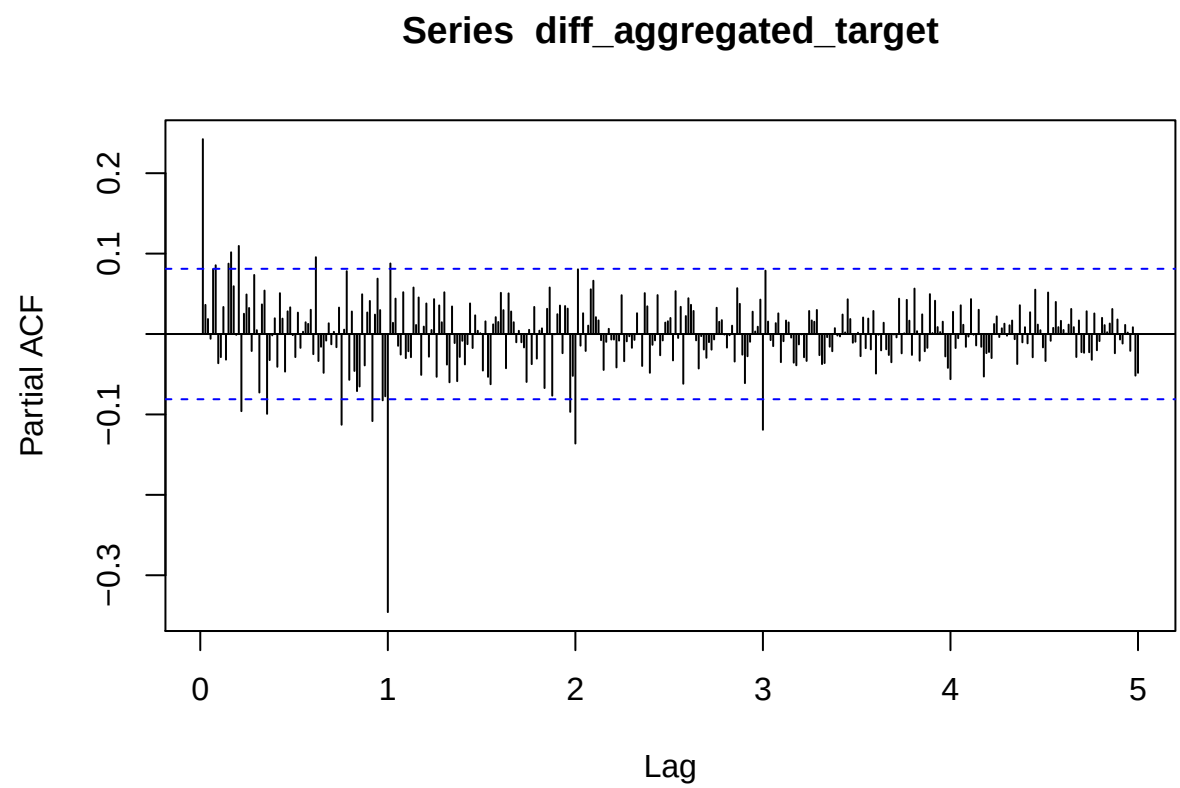
Parameter Identification

In this section, we will find and explain our choice of p, q, d, P, Q, and D. First, we use diff (lag $365/5 = 73$) to remove the seasonal trend.

```
aggregated_target <- ts(aggregated_target, frequency = 73)
diff_aggregated_target <- diff(aggregated_target, 73)
acf(diff_aggregated_target, lag.max=5*73)
```

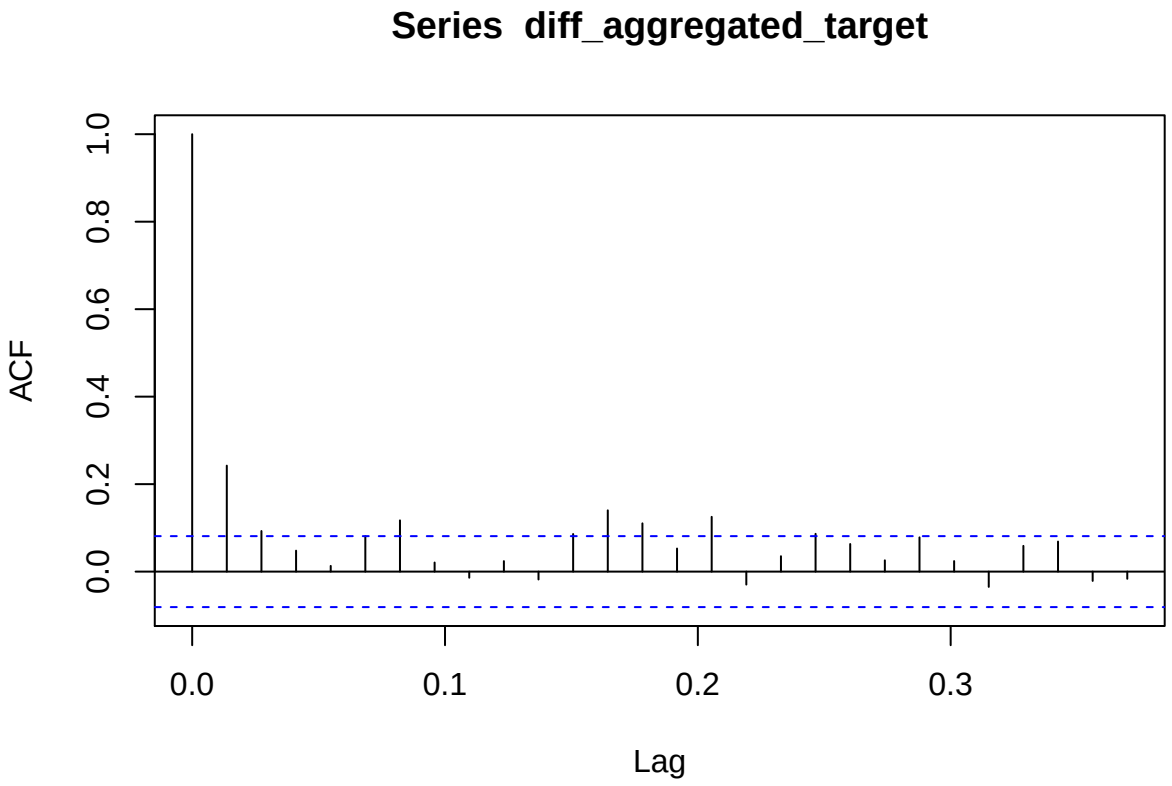



```
pacf(diff_aggregated_target, lag.max=5*73)
```

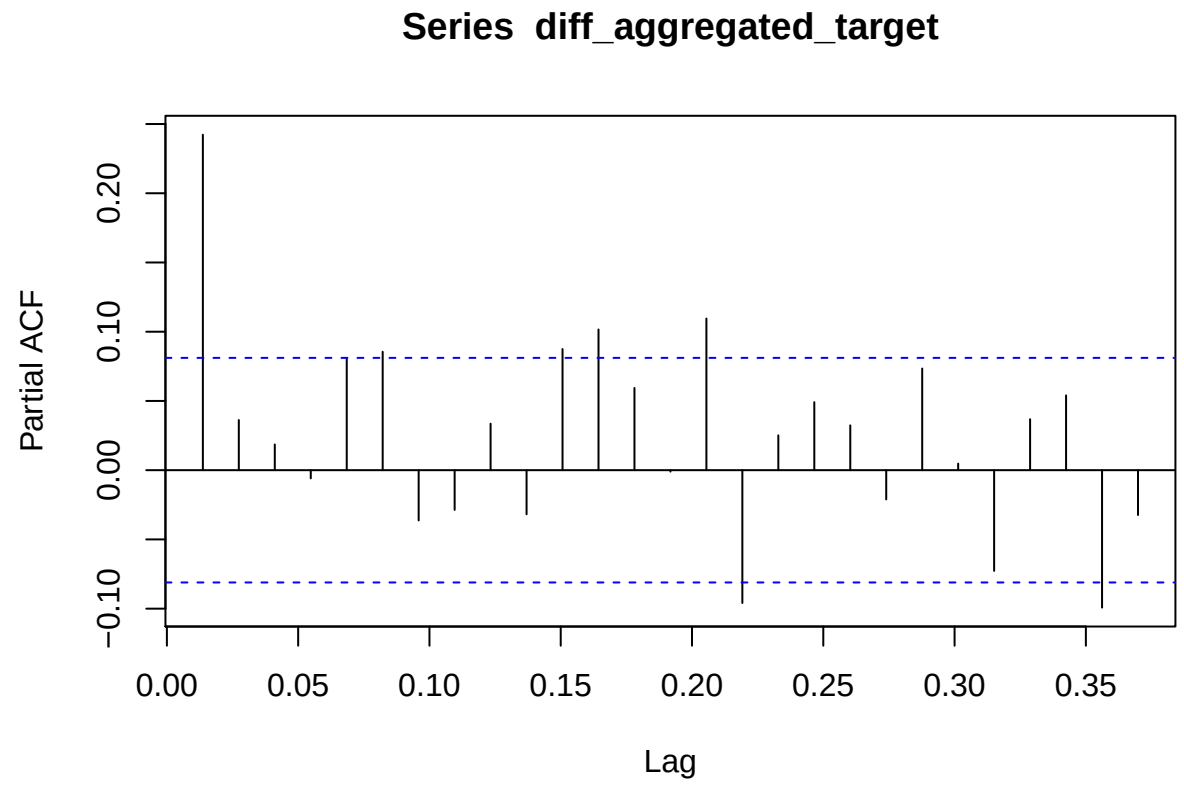


We can see from the graph above that at seasonal level, PACF seems to tail off, while ACF cuts off at lag 2. It suggest that $P = 0$, and $Q = 1$. $D = 1$, because we difference one time at the seasonal level. Now we need to go back to non-seasonal level.

```
acf(diff_aggregated_target)
```




```
pacf(diff_aggregated_target)
```



The ACF seems to cut off at lag 3, and PACF seems to cut off at lag 2. Therefore, we fit a series of models and pick the best one out of them. Based on this, we pick $p = 1$ and $q = 2$.

Model Fitting

In this section, we show two different methods of model fitting: normal ### Normal SARIMA Model Fitting We calculate the value of Akaike's information criterion (AIC) to evaluate the fitness of a model. Specifically, the formula to calculate AIC is

$$\text{AIC} = -2L + 2d,$$

where L is the likelihood and d is the number of parameters to estimate. Based on our choice of p and q, we fit a SARIMA(1,0,2)_(0,0,1)$_7$3 model:

```
options(warn=-1)
model <- arima(aggregated_target, order = c(1, 0, 2), seasonal=list(order=c(0,1,1), period=73))
model
```

```
##
## Call:
## arima(x = aggregated_target, order = c(1, 0, 2), seasonal = list(order = c(0,
##     1, 1), period = 73))
##
## Coefficients:
##          ar1      ma1      ma2     sma1
##       0.9410  -0.6951  -0.1618  -0.9937
## s.e.  0.0463   0.0637   0.0483      NaN
##
## sigma^2 estimated as 3.174:  log likelihood = -1244.39,  aic = 2498.78
```

However, we failed to estimate the standard deviation of seasonal moving average coefficients. Re-fit the model with a smaller moving average order:

```
options(warn=-1)
model <- arima(aggregated_target, order = c(1, 0, 1), seasonal=list(order=c(0,1,1), period=73))
model
```

```
##
## Call:
## arima(x = aggregated_target, order = c(1, 0, 1), seasonal = list(order = c(0,
##     1, 1), period = 73))
##
## Coefficients:
##          ar1      ma1     sma1
##       0.5803  -0.3463  -0.9284
## s.e.  0.2190   0.2596   0.1523
##
## sigma^2 estimated as 3.415:  log likelihood = -1248.45,  aic = 2504.91
```

Although there is an increase in AIC, the results are better than the previous model since the estimated standard deviation of seasonal moving average coefficients is finite.

We compare this model with the model found by auto.arima:

```
auto.model <- auto.arima(aggregated_target)
auto.model
```

```
## Series: aggregated_target
```

```
## ARIMA(1,0,0)(0,1,0)[73]
##
## Coefficients:
##          ar1
##       0.2425
## s.e.  0.0401
##
## sigma^2 estimated as 6.242:  log likelihood=-1362.91
## AIC=2729.82   AICc=2729.84   BIC=2738.56
```

It is clear that our model has a smaller AIC compared the model found by auto.arima. Therefore, our choice of model parameters are reasonable.

The formula of the fitted model is

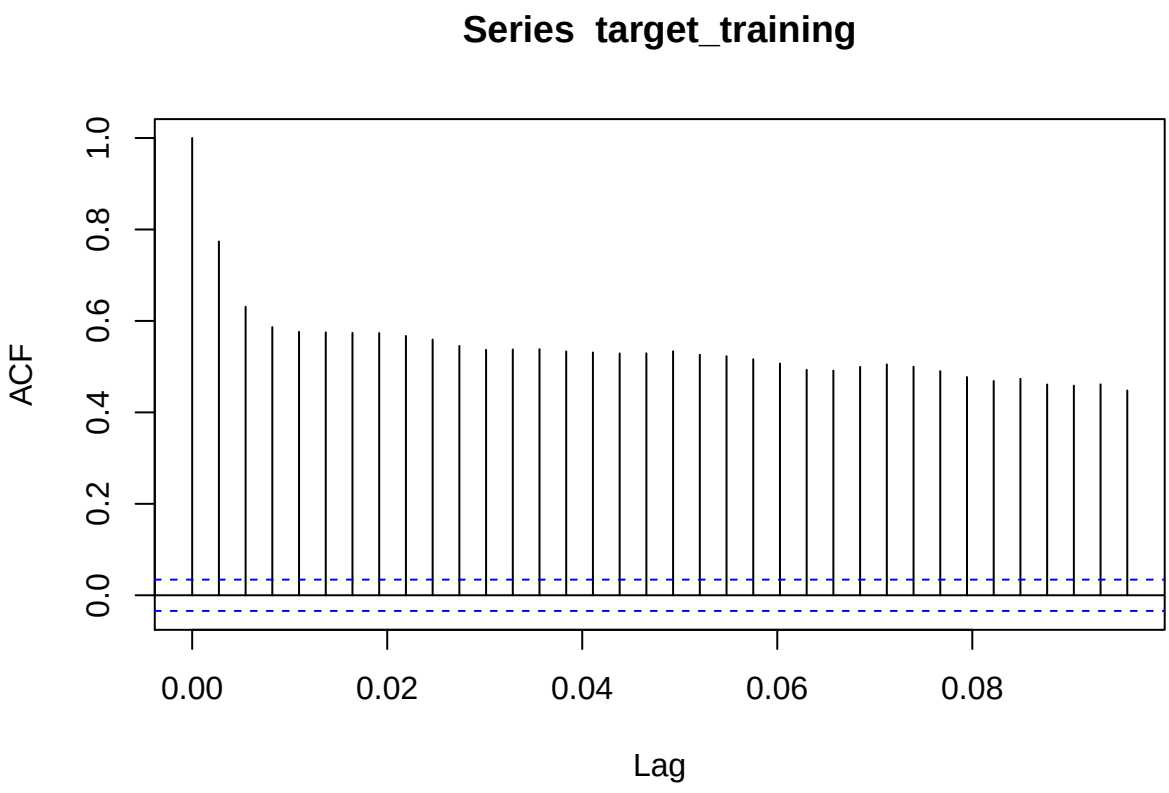
$$(1 - 0.9410B)(1 - B^{73})y_t = (1 + 0.6951B + 0.1618B^2)(1 + 0.9937B^{73})z_t$$

Fourier-series based model

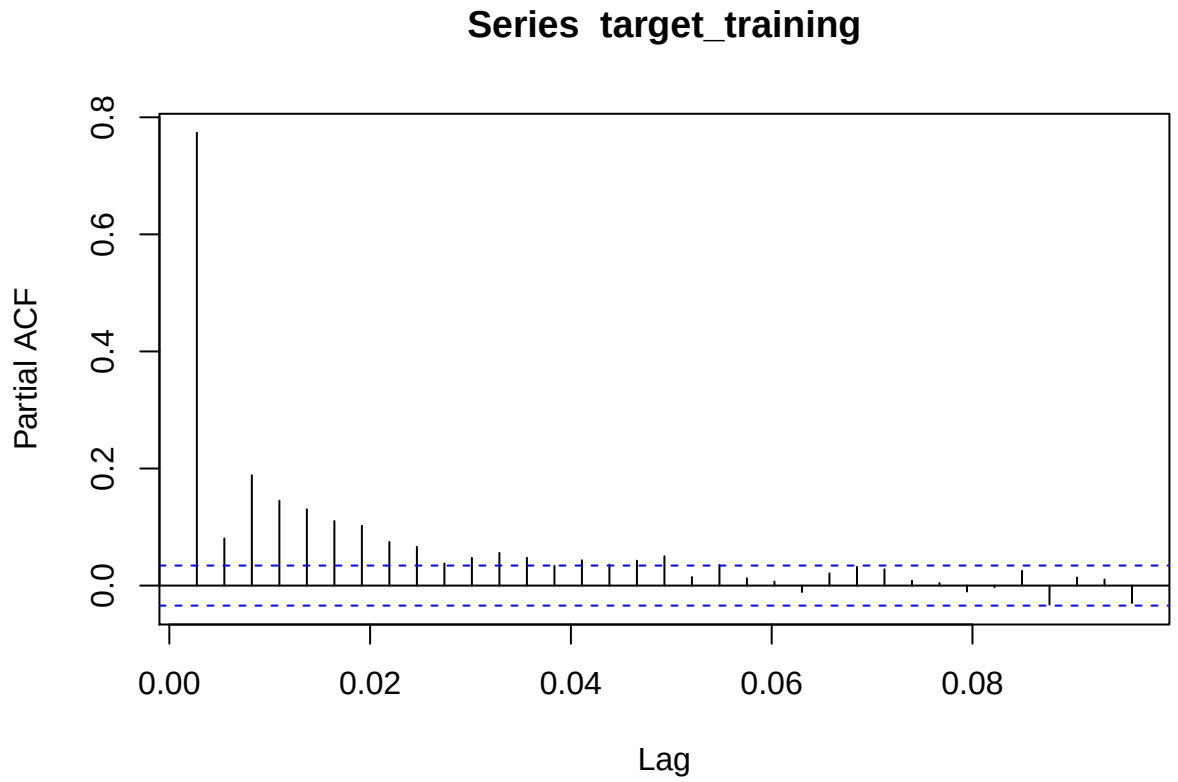
We show in this section that Fourier series can be used to remove the seasonality.

First, we need to pre-process the daily data.

```
target_training <- target[time < "1990-01-01"]
target_training <- ts(target_training, frequency = 365)
acf(target_training)
```

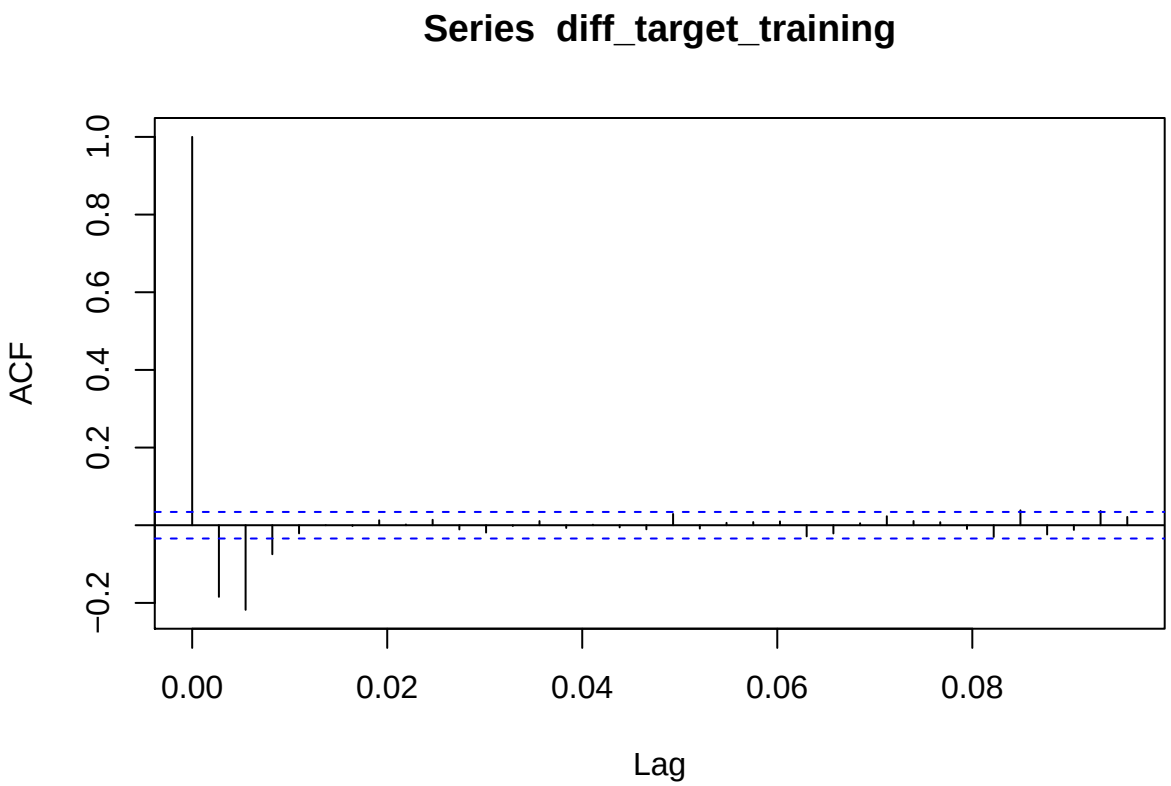



```
pacf(target_training)
```

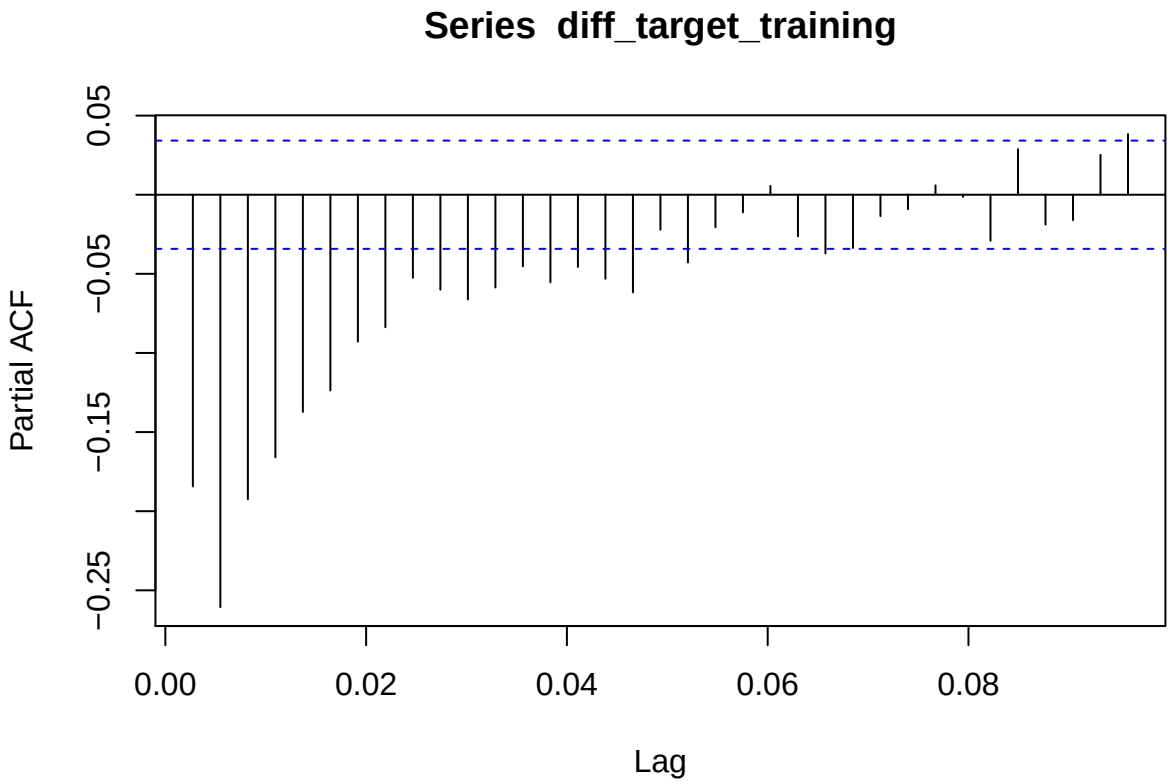


It shows a long memory. Difference the data with lag 1:

```
diff_target_training <- diff(target_training)
acf(diff_target_training)
```



```r
pacf(diff_target_training)
```



ACF cuts off at lag 4, and PACF tails off but peaks at lag 2. So we pick ARMA(3, 2).

```r
fit.fourier <- Arima(target_training, c(3,1,2), xreg=fourier(target_training, K=4))
fit.fourier
```

```
## Series: target_training
## Regression with ARIMA(3,1,2) errors
##
## Coefficients:
##           ar1     ar2     ar3      ma1      ma2  S1-365  C1-365  S2-365
##       -0.1464  0.2559  -0.050  -0.3071  -0.6724  1.7770  3.8055  0.3356
## s.e.   2.5176  1.3665   0.315   2.5116   2.4762  0.1308  0.1275  0.1111
##        C2-365   S3-365  C3-365   S4-365   C4-365
##       -0.4373  -0.0885  0.1045  -0.1590  -0.2186
## s.e.   0.1102   0.1071  0.1066   0.1055   0.1053
##
## sigma^2 estimated as 5.636:  log likelihood=-7494.15
## AIC=15016.3   AICc=15016.43   BIC=15101.65
```

From the expression, we can see that the seasonal part is modeled by Fourier series, while the non-seasonal part is modeled by an ARIMA(3,1,2). The expression for the non-seasonal part is

$$(1 + 0.1464B - 0.2599B^2 + 0.05B^3)(1 - B)y_t = (1 - 0.307B)z_t$$

Model Diagnosis

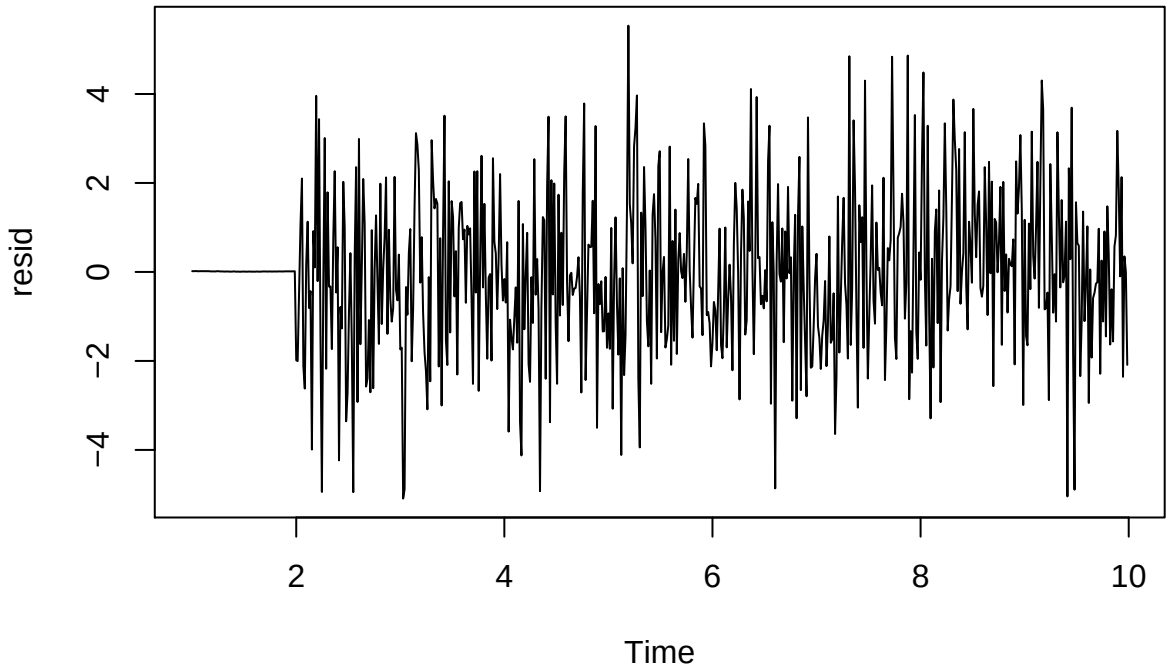
In this section, we will diagnose the model we have just fitted. The first step is to derive the residual of the model.

```
resid <- resid(model)
```

Residual Visualization

We first examine the trend of the residual visually.

```
plot(resid)
```

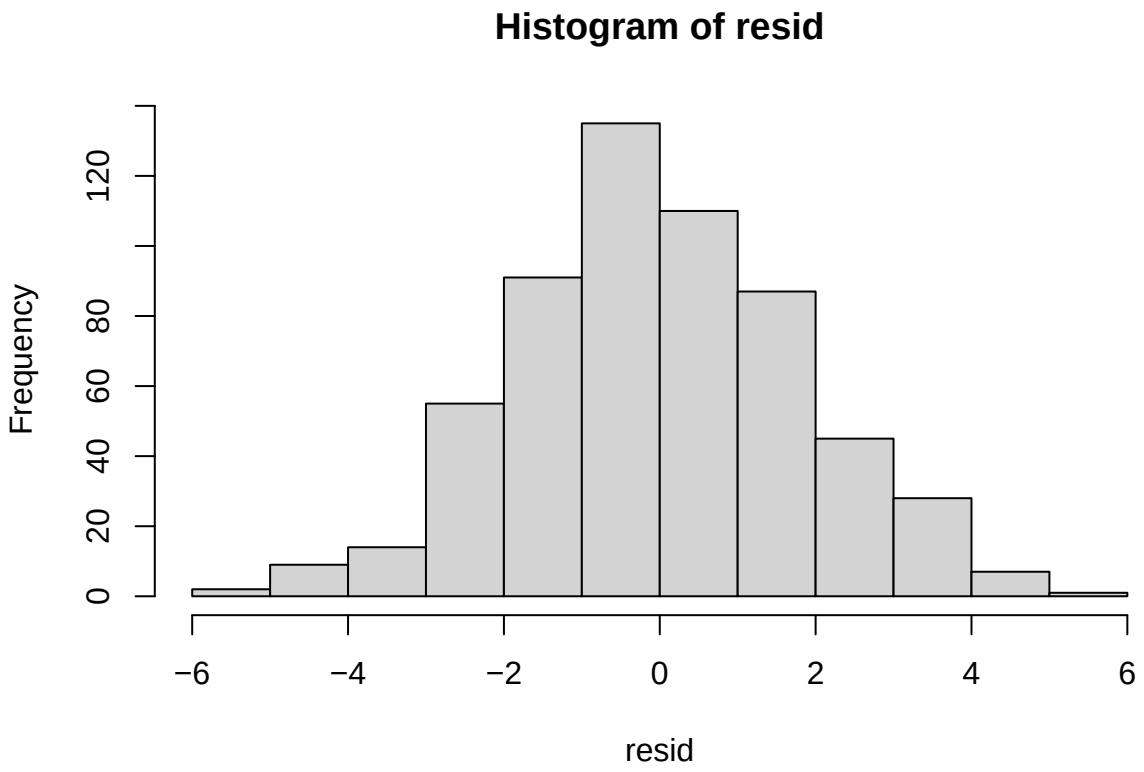


As we can see from the graph, there is no trend in the residual. The variance seems to be uniform all the time. It is also noteworthy that there is a period a time that the residual is very small, since they are basis for prediction. So we remove the residual of the first period from the residual vector.

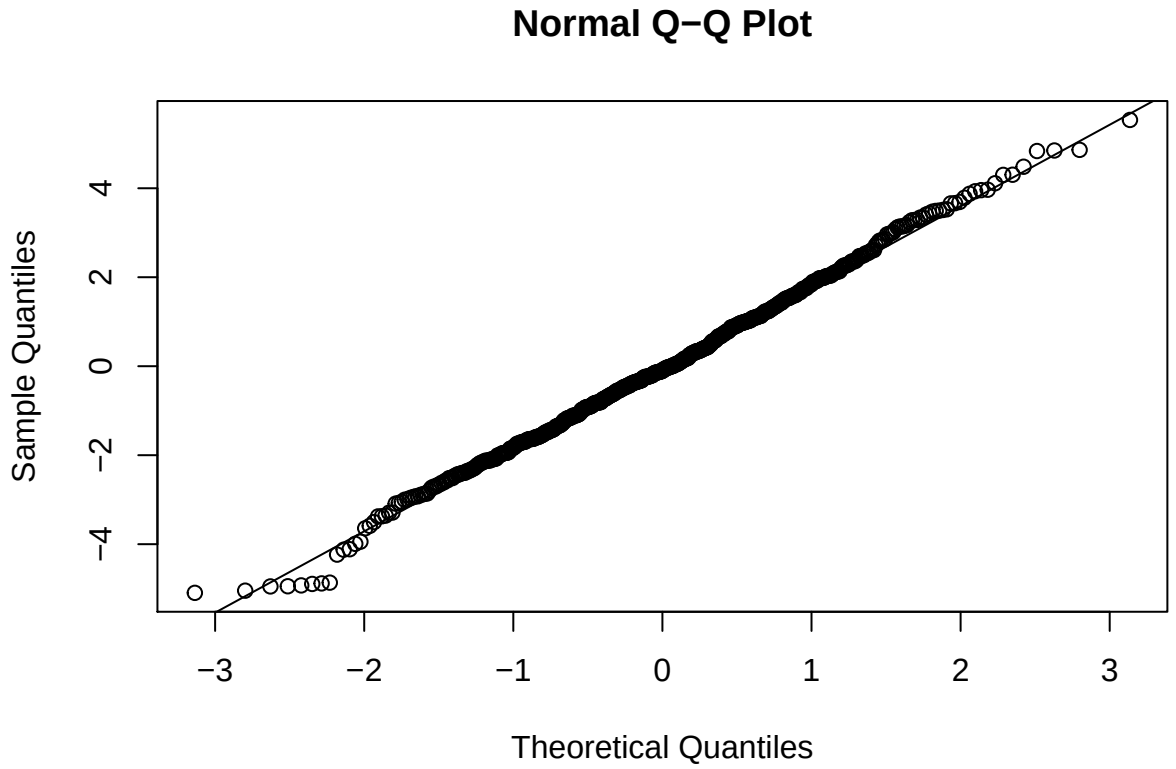
```
resid <- resid[-seq(1,73)]
```

Sequentially, we visually examine the normality of the residual:

```
hist(resid)
```




```
qqnorm(resid)
qqline(resid)
```



The Q-Q plot shows that the data are almost on a straight line.

The distribution of the residual is in a bell shape, potentially meaning that the residual is following a normal

distribution. The Q_Q plot also supports our judgment on the normality of the residual.

To be more accurate, we will also use the Sharpio test to numerically examine the normality of the residual. The Sharpio test is a typical test for normality. The test statistics W is calculated as

$$W = \frac{(\sum_{i=1}^{n} a_i x_{(i)})^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2},$$

where $x_{(i)}$ is the ordered sample values and a_i is a constant generated from the means, variances, and covariances. We will reject normality of the p-value is below 5%.

Testing the normality using Shapiro test:

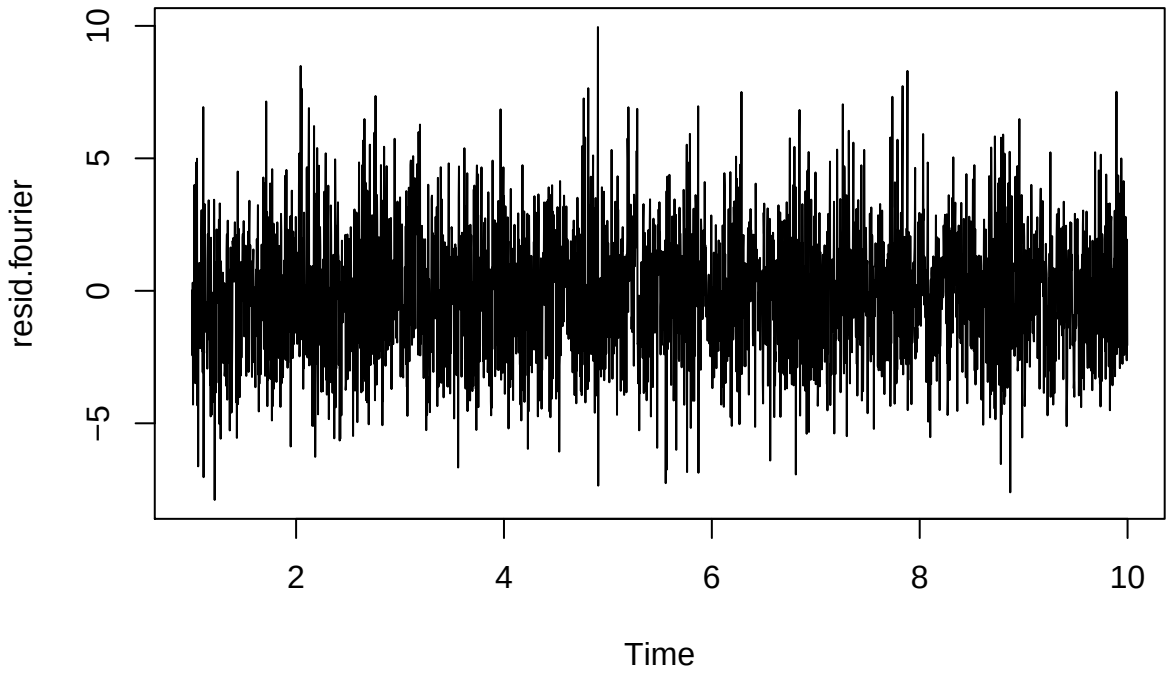
```
shapiro.test(resid)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid
## W = 0.99708, p-value = 0.3838
```

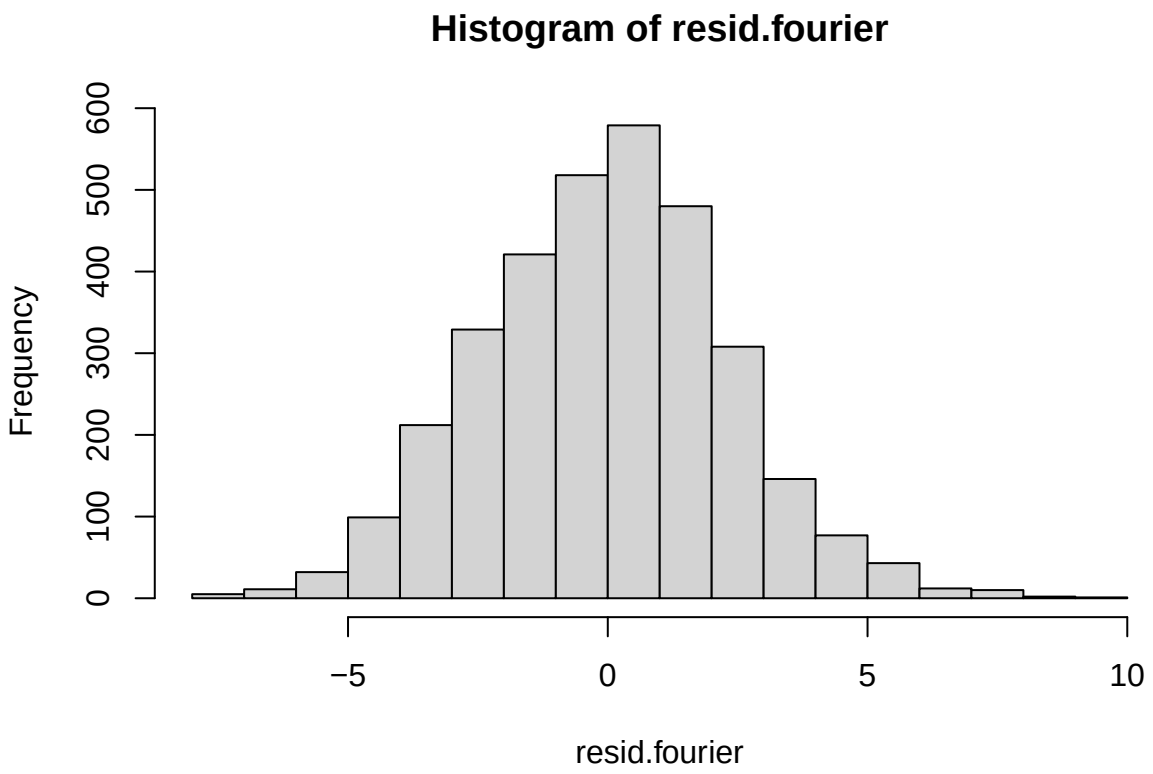
The p-value is greater than 0.05, indicating that the residual is following a normal distribution.

We repeat the same procedure for the Fourier series based model.

```
resid.fourier <- resid(fit.fourier)
plot(resid.fourier)
```

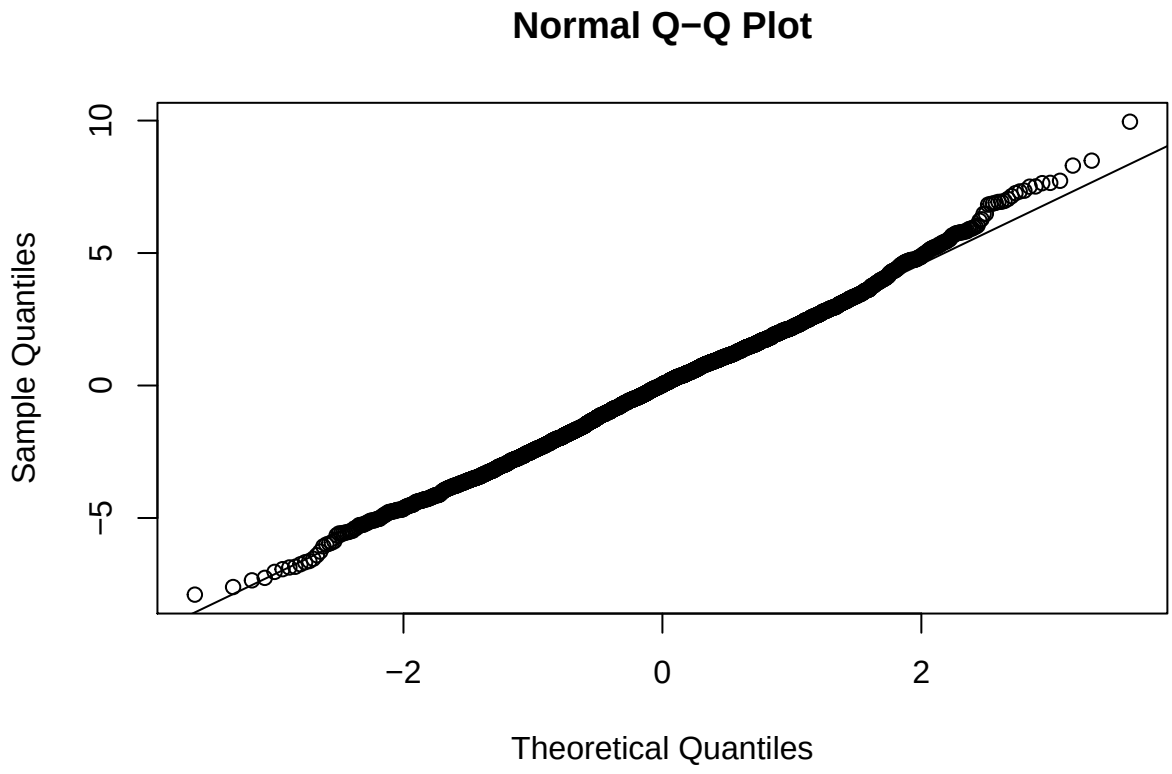



```
resid.fourier <- resid(fit.fourier)
hist(resid.fourier)
```



We do not need to remove the residual of the first period here. The residual is in a bell shape.

```
qqnorm(resid.fourier)
qqline(resid.fourier)
```



Q-Q plot also shows good normality.

```r
shapiro.test(resid.fourier)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid.fourier
## W = 0.99773, p-value = 9.651e-05
```

However, the residual of the Fourier series-based model cannot pass the normality test.

Portmanteau Tests

Next, we will the the independence of residual.

Box-Pierce Test The statistics of the Box-Pierce test is calculated as

$$Q_W = n \sum_{j=1}^{h} \hat{\rho}_{\hat{W}}^2(j) \sim \chi^2(h-d)$$

, where $h = \sqrt{n}$, n is the number of data and d is the number of parameters. It is used to examine whether a series is a white noise. We can use the function *Box.test* to perform Box-Pierce Test:

```r
Box.test(resid, lag = as.integer(sqrt(length(resid))) - (1+2+1+1))
```

```
##
##  Box-Pierce test
##
## data:  resid
## X-squared = 27.958, df = 19, p-value = 0.08423
```

For the Fourier series-based model:

```r
Box.test(resid.fourier, lag = as.integer(sqrt(length(resid.fourier))) - 5)
```

```
##
##  Box-Pierce test
##
## data:  resid.fourier
## X-squared = 41.558, df = 52, p-value = 0.8497
```

Since both of the p-values are greater than 0.05, we fail to reject the null hypothesis. It suggests that the residuals of the two models are potentially white noise.

Ljung-Box Test The statistics of the Ljung-Box test is calculated as

$$Q_W = n(n+2) \sum_{j=1}^{h} \hat{\rho}_{\hat{W}}^2(j)/(n-j) \sim \chi^2(h-d)$$

, where $h = \sqrt{n}$, n is the number of data and d is the number of parameters. It is used to examine whether a series is a white noise. We can use the function *Box.test* to perform Box-Pierce Test:

```r
Box.test(resid, lag = as.integer(sqrt(length(resid))) - 5, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  resid
## X-squared = 28.644, df = 19, p-value = 0.07179
```

For the Fourier series-based model:

```
Box.test(resid.fourier, lag = as.integer(sqrt(length(resid.fourier))) - 5, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  resid.fourier
## X-squared = 41.956, df = 52, p-value = 0.8389
```

Since both of the p-value is greater than 0.05, we fail to reject the null hypothesis. It suggests that the residuals of the two models are potentially white noise.

McLeod Li Test The statistics of the Ljung-Box test is calculated as

$$Q_W = n(n+2)\sum_{j=1}^{h} \hat{\rho}^2_{\hat{W}\hat{W}}(j)/(n-j) \sim \chi^2(h-d),$$

where $h = \sqrt{n}$, n is the number of data and d is the number of parameters. It is used to test the residual for non-linear dependence.

```
Box.test(resid^2, lag = as.integer(sqrt(length(resid))) - 5, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  resid^2
## X-squared = 10.335, df = 19, p-value = 0.9442
```

For the Fourier series-based model:

```
Box.test(resid.fourier^2, lag = as.integer(sqrt(length(resid.fourier))) - 5, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  resid.fourier^2
## X-squared = 120.28, df = 52, p-value = 2.485e-07
```

The residual of the SARIMA model has no non-linear dependence but the residual of the Fourier series-based model has.

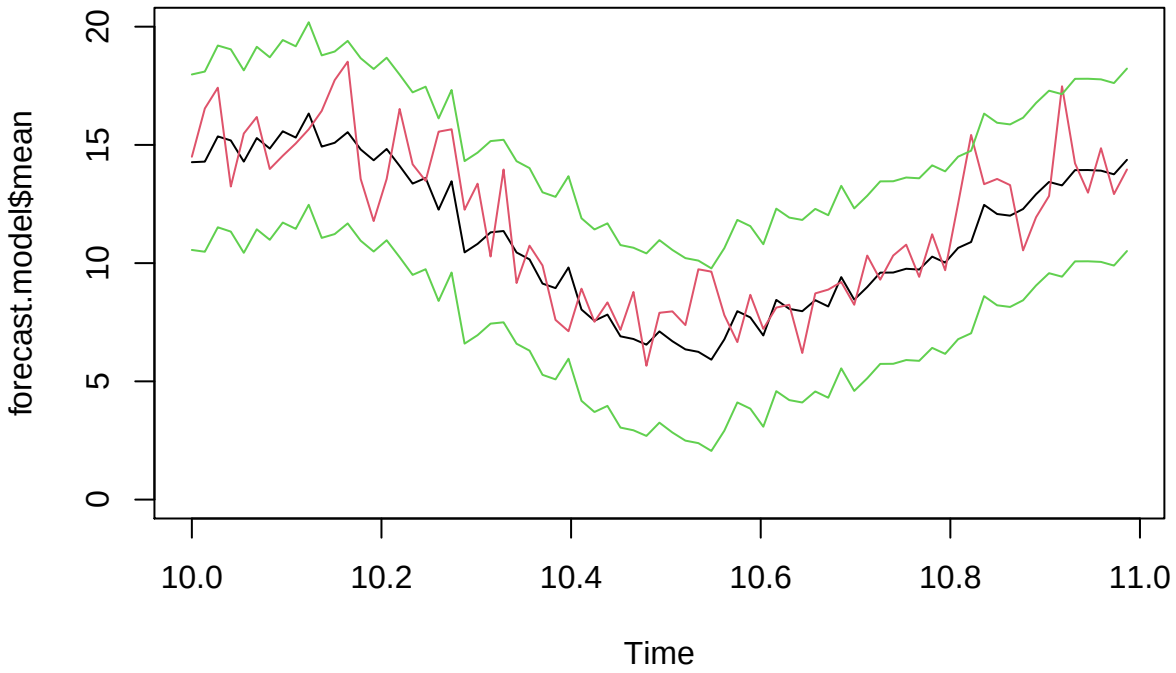
From the above diagnosis, we can see that the SARIMA model has a better fitness result, but the Fourier series-based model can handle the daily level data. So each model has its own advantages.

Prediction

In this section, we predict the lowest temperature using the two models, i.e. the SARIMA model and the Fourier series-based model.

SARIMA Prediction

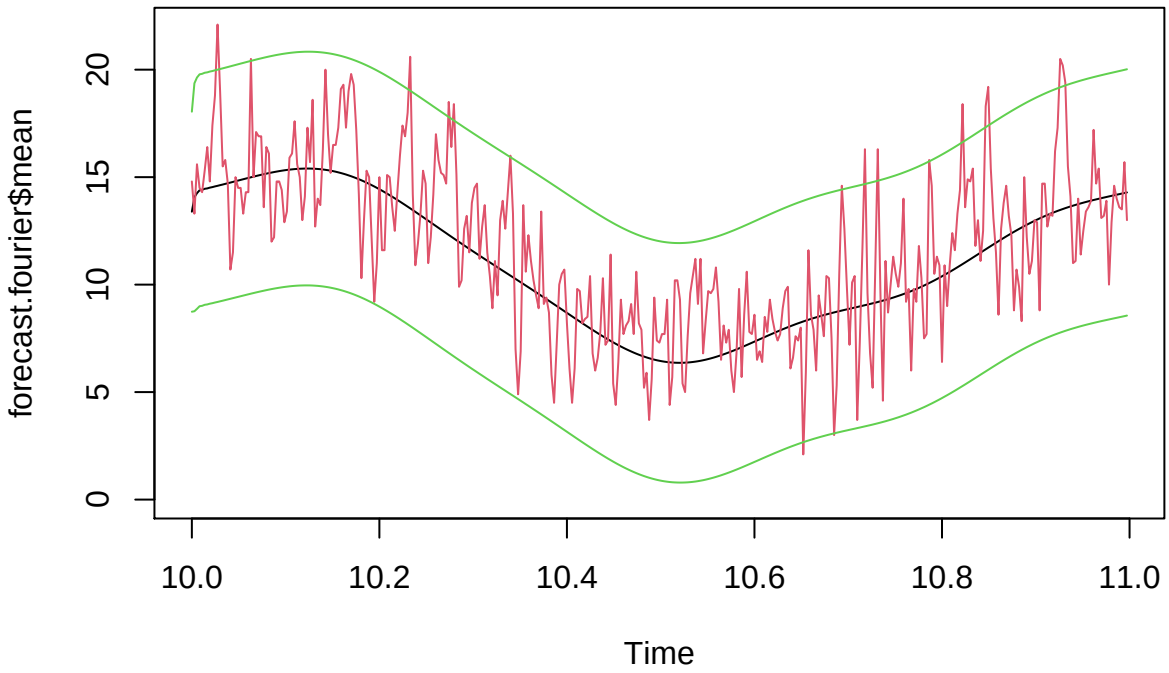
```
forecast.model <- forecast(model, h = length(aggregated_test))
plot(forecast.model$mean,  ylim=c(0, 20))
lines(ts(aggregated_test, start=c(10,1), frequency=73), col=2)
lines(forecast.model$upper[,2], col=3)
lines(forecast.model$lower[,2], col=3)
```



The red lines are the real 5-day average temperature and the black line is the predicted temperature. The green lines indicate the 95% confidence interval. We can see that the prediction are very close to the real data, and the real data are in the prediction interval.

Fourier Prediction

```
target_testing <- ts(target[time >= "1990-01-01"], frequency=365)
forecast.fourier <- forecast(fit.fourier,h = length(target_testing),
        xreg=fourier(target_training, K=4, length(target_testing)))
plot(forecast.fourier$mean,  ylim=c(0, 22))
lines(ts(target_testing, start=c(10,1), frequency=365), col=2)
lines(forecast.fourier$upper[,2], col=3)
lines(forecast.fourier$lower[,2], col=3)
```



The Fourier series-based method would generate very smoothed prediction. The trend of the prediction and the real data are similar; however, the details are different. There are a few days that the real data are outside the 95% confidence interval, showing a inferior performance compared to the SARIMA model. However, given the difficulties of predicting daily temperature due to the high variance, the results are satisfactory.

Conclusion

In this report, we successfully modeled the lowest temperatures in Melbourne. In order to bypass the constraints of R that the lag of SARIMA cannot be greater than 350, we have fitted two models (SARIMA model of 5-day average and Fourier series-based model) and showed that these two models can generate good prediction results, i.e. mostly inside the confidence interval of the prediction.

Reference

[1] https://www.kaggle.com/paulbrabban/daily-minimum-temperatures-in-melbourne

[2] https://www.rdocumentation.org/packages/forecast/versions/8.13/topics/Arima

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(ggplot2)
library(tseries)
library(MASS)
library(knitr)
library(lubridate)
library(forecast)
data <- read.csv("~/temp.csv")
colnames(data) <- c("Date", "Temp")
time <- as_date(data$Date)
target <- data$Temp
```

```
qplot(time, target, ylab="Temperature", xlab="Date")
options(warn=-1)
ts.target = ts(target, frequency = 365)
hist(ts.target, main = "Histogram of Temperature", xlab = "Temperature")
aggregated <- (target[seq(1, length(target), 5)] +
                      target[seq(2, length(target), 5)] +
                      target[seq(3, length(target), 5)] +
                      target[seq(4, length(target), 5)] +
                      target[seq(5, length(target), 5)]
                      ) / 5
aggreated_length <- length(aggregated)
aggregated_target <- aggregated[1:(aggreated_length * 0.9)]
aggregated_test <- aggregated[-(1:(aggreated_length * 0.9))]
plot(aggregated_test, type="l", main="Testing Data", ylab="5-day average lowest temperature")
aggregated_target <- ts(aggregated_target, frequency = 73)
diff_aggregated_target <- diff(aggregated_target, 73)
acf(diff_aggregated_target, lag.max=5*73)
pacf(diff_aggregated_target, lag.max=5*73)
acf(diff_aggregated_target)
pacf(diff_aggregated_target)
options(warn=-1)
model <- arima(aggregated_target, order = c(1, 0, 2), seasonal=list(order=c(0,1,1), period=73))
model
options(warn=-1)
model <- arima(aggregated_target, order = c(1, 0, 1), seasonal=list(order=c(0,1,1), period=73))
model
auto.model <- auto.arima(aggregated_target)
auto.model
target_training <- target[time < "1990-01-01"]
target_training <- ts(target_training, frequency = 365)
acf(target_training)
pacf(target_training)
diff_target_training <- diff(target_training)
acf(diff_target_training)
pacf(diff_target_training)
fit.fourier <- Arima(target_training, c(3,1,2), xreg=fourier(target_training, K=4))
fit.fourier
resid <- resid(model)
plot(resid)
resid <- resid[-seq(1,73)]
hist(resid)
qqnorm(resid)
qqline(resid)
shapiro.test(resid)
resid.fourier <- resid(fit.fourier)
plot(resid.fourier)
resid.fourier <- resid(fit.fourier)
hist(resid.fourier)
qqnorm(resid.fourier)
qqline(resid.fourier)
shapiro.test(resid.fourier)
Box.test(resid, lag = as.integer(sqrt(length(resid))) - (1+2+1+1))
Box.test(resid.fourier, lag = as.integer(sqrt(length(resid.fourier))) - 5)
```

```
Box.test(resid, lag = as.integer(sqrt(length(resid))) - 5, type="Ljung-Box")
Box.test(resid.fourier, lag = as.integer(sqrt(length(resid.fourier))) - 5, type="Ljung-Box")
Box.test(resid^2, lag = as.integer(sqrt(length(resid))) - 5, type="Ljung-Box")
Box.test(resid.fourier^2, lag = as.integer(sqrt(length(resid.fourier))) - 5, type="Ljung-Box")
forecast.model <- forecast(model, h = length(aggregated_test))
plot(forecast.model$mean,  ylim=c(0, 20))
lines(ts(aggregated_test, start=c(10,1), frequency=73), col=2)
lines(forecast.model$upper[,2], col=3)
lines(forecast.model$lower[,2], col=3)
target_testing <- ts(target[time >= "1990-01-01"], frequency=365)
forecast.fourier <- forecast(fit.fourier,h = length(target_testing),
        xreg=fourier(target_training, K=4, length(target_testing)))
plot(forecast.fourier$mean,  ylim=c(0, 22))
lines(ts(target_testing, start=c(10,1), frequency=365), col=2)
lines(forecast.fourier$upper[,2], col=3)
lines(forecast.fourier$lower[,2], col=3)
```