Forcasting Daily Minimum Temperature in Melbourne

Naby Diop

5/21/2020

Contents

Introduction

The Daily minimun temperature in Melbourne is provided by the kaggle website. The data was collected over 10 years period, stating from January 1, 1981 to December 31, 1990. The main purpose of this project is to forecast the daily minimum temperature with Arima model.

Melbourne is an Australian city. It is usually hot there from December to February (summer), it cools down March to May (Fall), chills out June to August (Winter), and finally warm up again September to November (Spring). Usually the hotest pepiods in Melbourne are in Junuary and February and the collest periods are in June and July. Octobers are the wettest months.

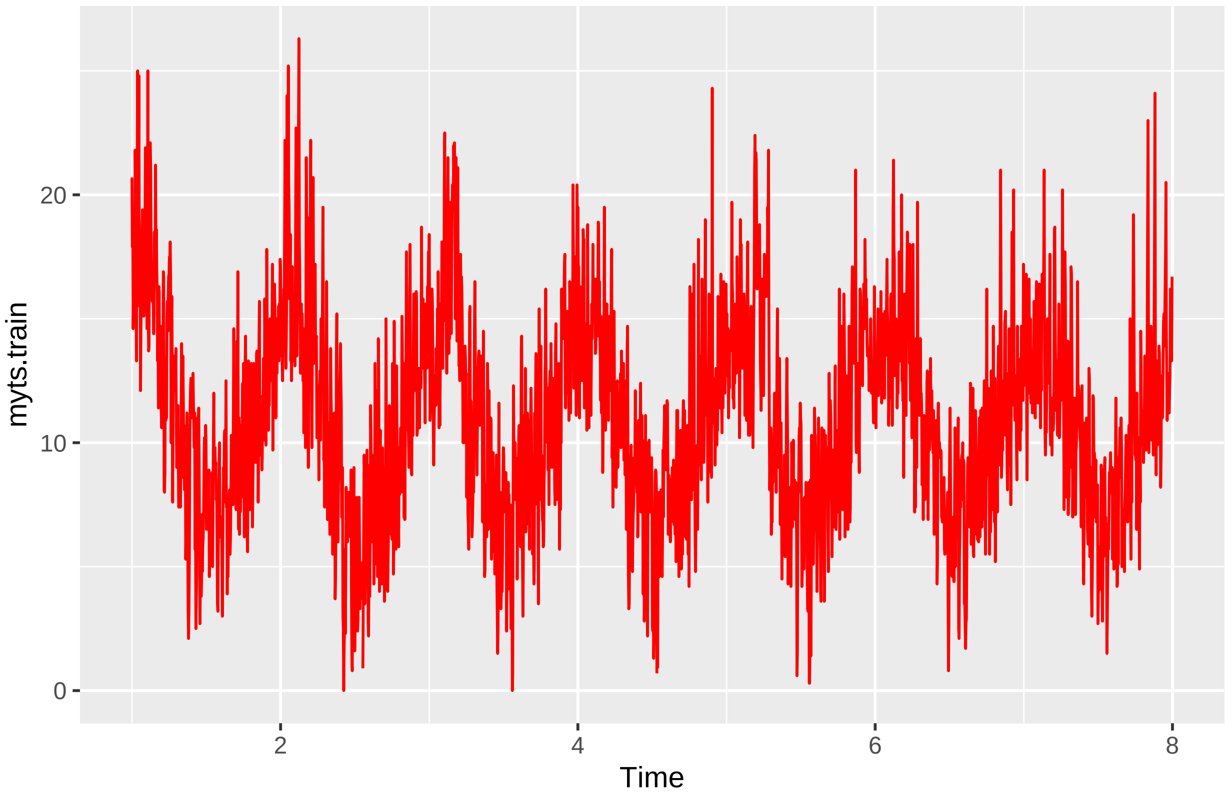
Data preparation

Looking at the structure of the data we find out that this data is not a Time Series (ts) object. Thus, we need to transform it in ts data. To do so, we first transform the time variable recorded in the data as "Date" to a time object, than we can use the *ts()* function to transform the temperature to a ts object. However, this data may present multiple seasonality. Since *ts()* only capture a single seasonality we use *msts()* function in R to capture all the seasonal periods in the data. *msts()* like *ts()* function transform the data to a ts object, but it takes in count all the seasonal periods in the data.

Before we do any forwader analysis we fist split the data into two set: the seven first years as trainning set and the last three years as test set. The R codes can be found on the **R Code** section of this document.

Graphical Analysis.

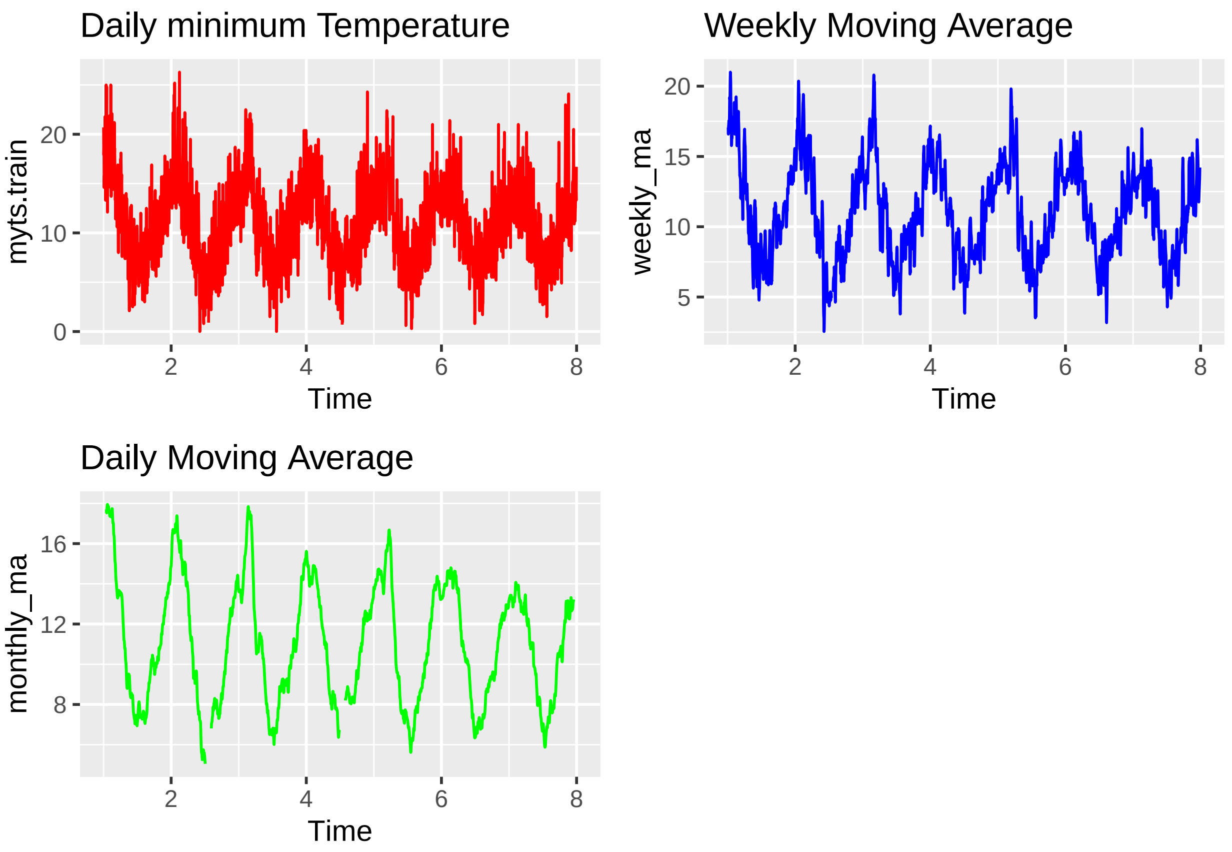
We first plot the Daily Minimum temperature against time for the trainning set. The plot is shown bellow.



The figure clealy shows pattern of saisonality in the data. We can also see that the beginning of the years have the highest temperature and the midle has the lowest temperature. This is not a surprise because usualy Melbourne is hot in January and February cold in June and July. We can also see some variations in the data with a lot of up and down.

One way to reduice these variations is to use the weekly or monthly temperature. We aggregate the temperure for each week or each month. In other words, we can use the weekly or monthly moving average (MA) for forwarder analysis. The function *ma()* is used in R to calculate the MA. The plot of the daily, weekly and monthly temperature is shown bellow.

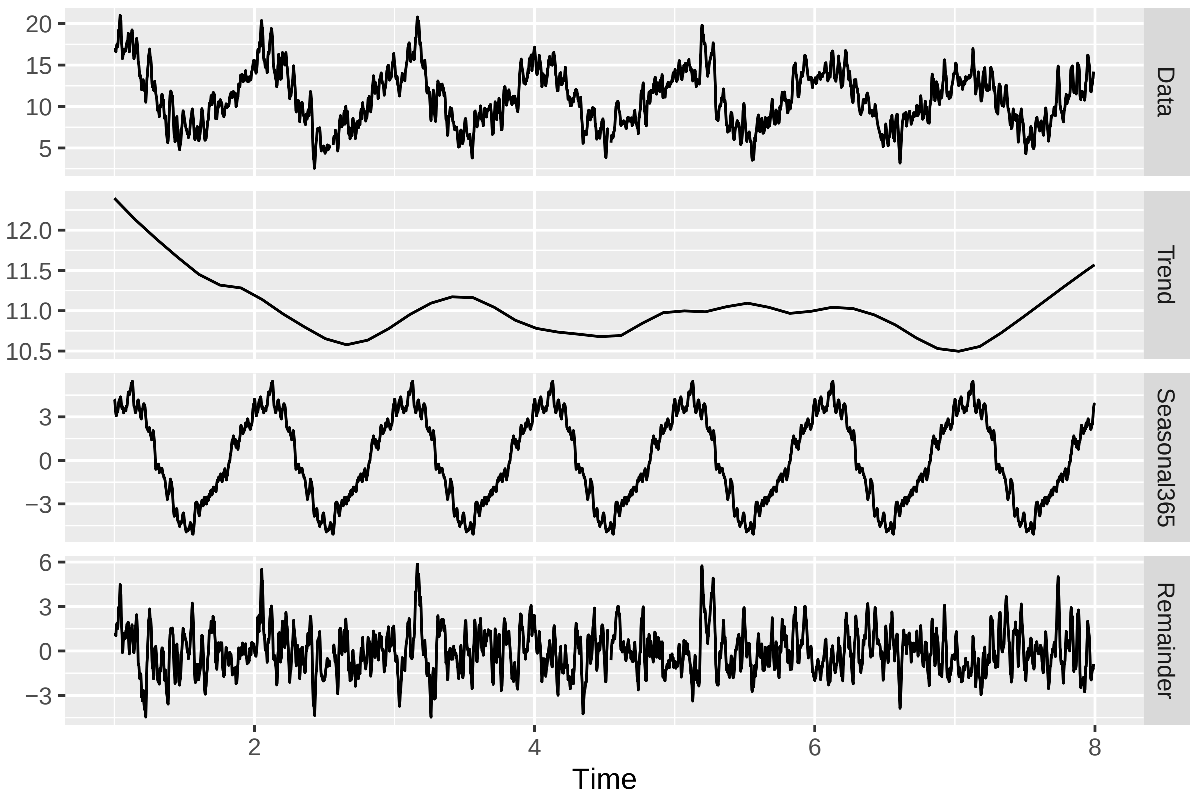
```
## NULL
```



As expected the monthy moving average is cleanner. It is represented in green on the plot. However, using this monthly moving average leads to loosing a lot of information in data. An alternative is to use the weekly MA which is presented in blue on the plot. This weekly MA has less variations than the daily temperature (in red) and capture better the informations in the data than the monthly MA.

Decomposition of the data.

We decompose the data so, we can see the seasonal and trend part if they do exist. The plot bellow is the plot of the decomposed data.



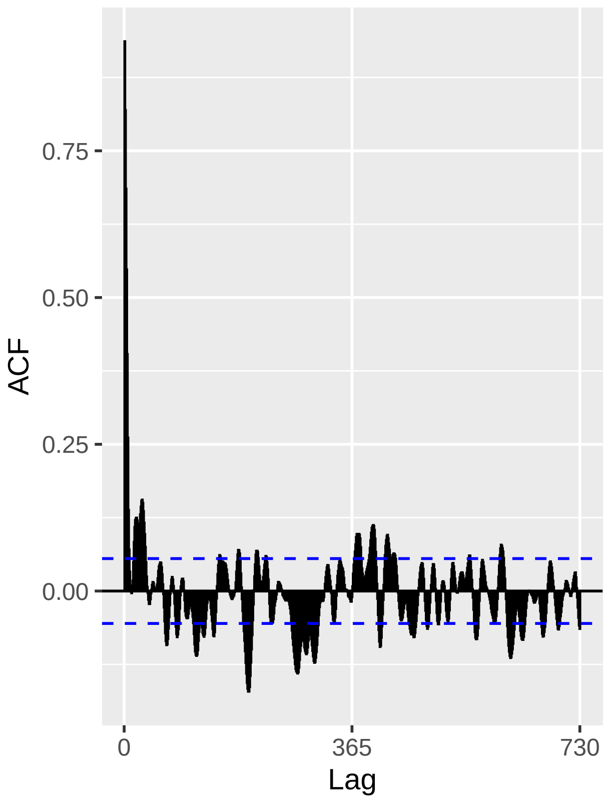
We can clearly see the presence of seasonality and trend. The top plot is the plot of the actual data, this follow by trend component. The third plot represent the seasonal part and the remainder is plot bellow.

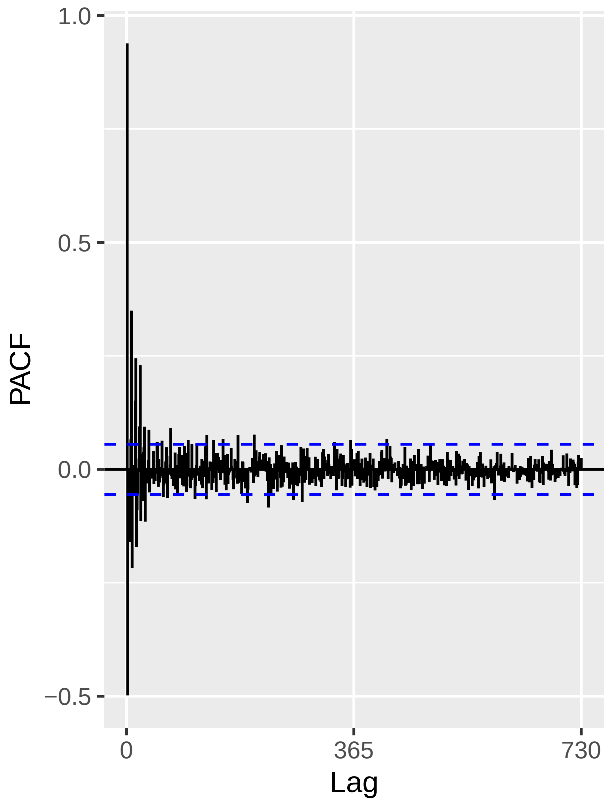
Dickey-Fuller test for stationarity

```
##
##  Augmented Dickey-Fuller Test
##
## data:  wkl_na
## Dickey-Fuller = -3.8617, Lag order = 13, p-value = 0.0159
## alternative hypothesis: stationary
```

We have p-value < 0.05 so we reject the null hypothesis, thus, the data is stationary time series.

ACF and PACF Plot.





We see that many the lags in the ACF and PACF are out of the optimal bandory.

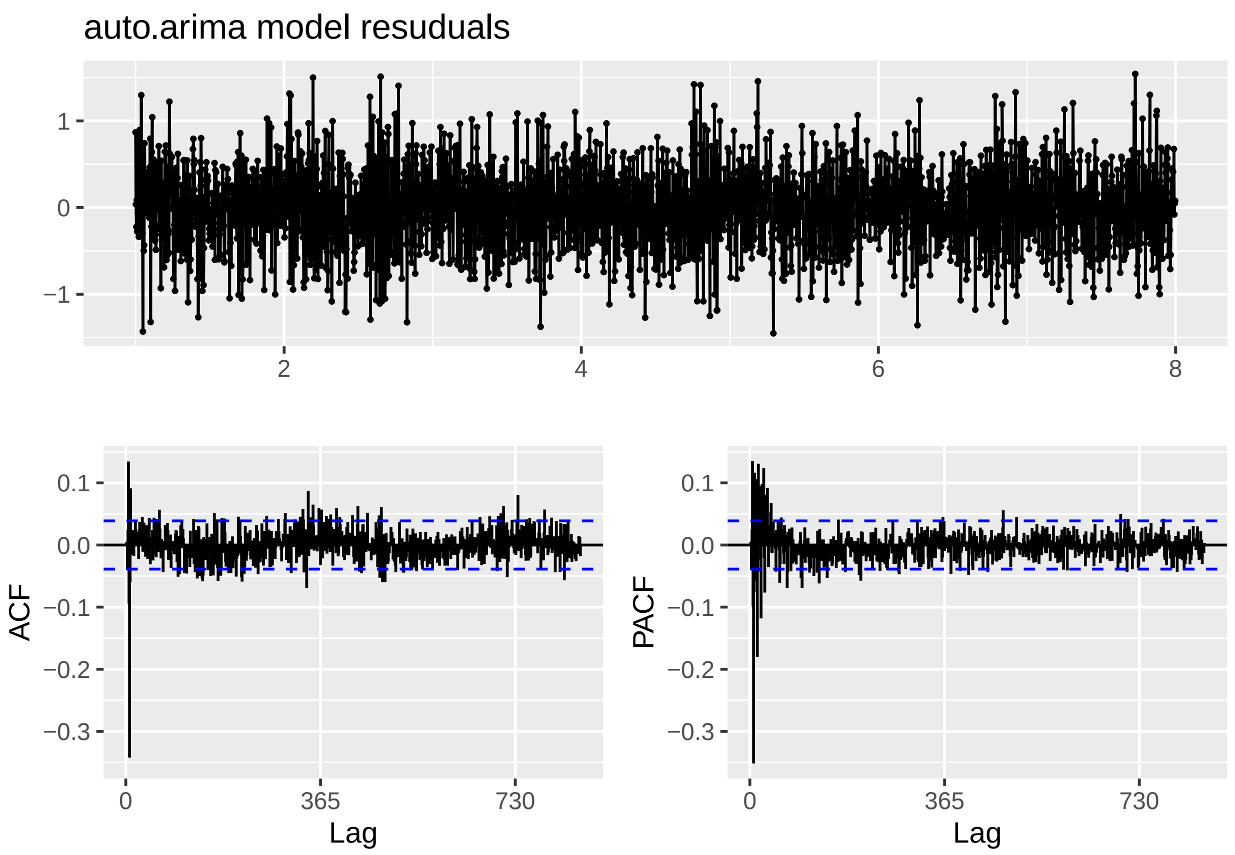
Fitting Arima models an forcasting for each model.

Auto.arima model

To fit the Arima model, we first remove the seanality part from the data. This may increase the accuracy of the model, but it is not alway a good idia. We first use *auto.arima()* on the training set, the model output is shown bellow. We use the argument seasonal=FALSE to prevent it searching for seasonal ARIMA models

```
## Series: wkl_na
## ARIMA(5,0,2) with non-zero mean
##
## Coefficients:
##           ar1     ar2     ar3      ar4      ar5     ma1     ma2     mean
##       -0.1425  0.9890  0.4047  -0.2501  -0.0658  1.7685  0.9607  11.0576
## s.e.   0.0207  0.0204  0.0270   0.0204   0.0204  0.0066  0.0067   0.5014
##
## sigma^2 estimated as 0.1984:  log likelihood=-1558.16
## AIC=3134.31   AICc=3134.38   BIC=3186.93
```

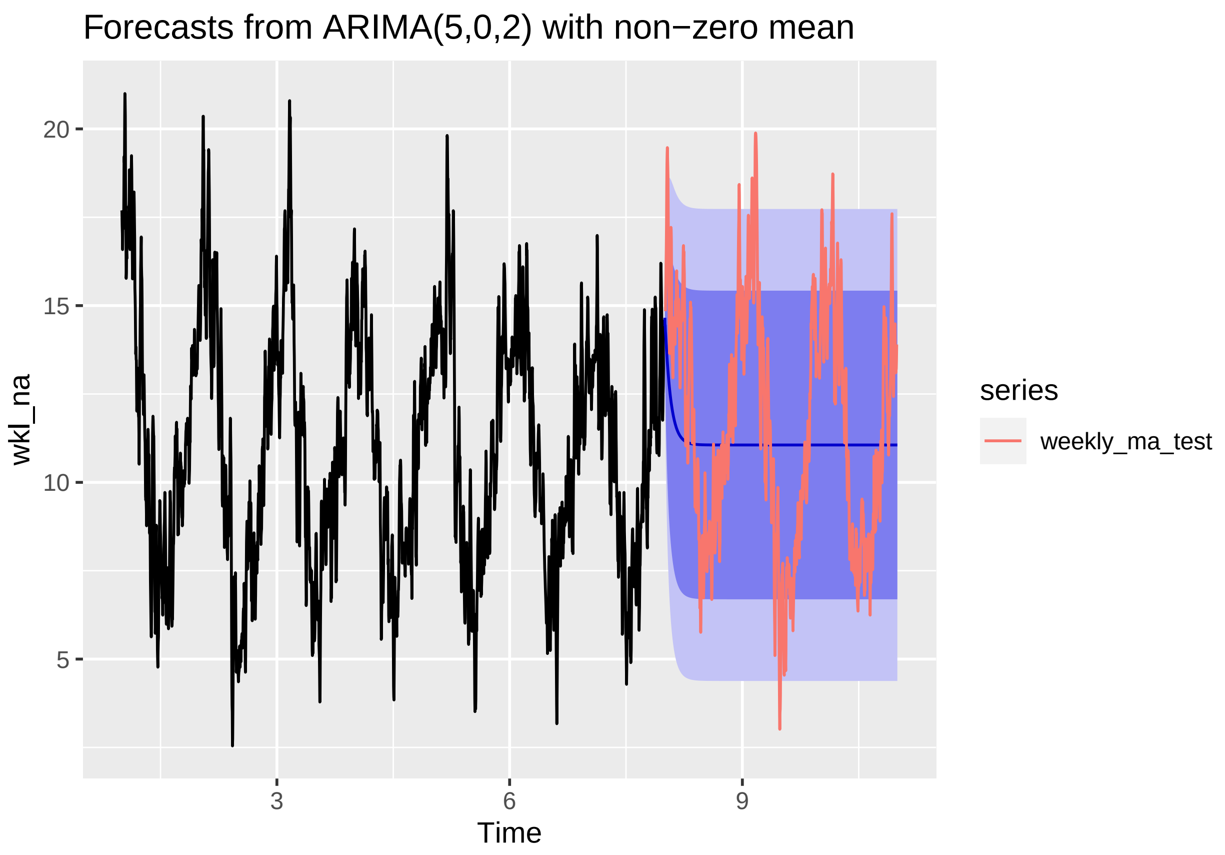
We show the model graphs bellow.



The summary of this model shows that the optimal arima model is the Arima(5,0,2) with AIC=3134.31. In both the ACF and PACF plot the significant spike are within the first lags and we have the most significant spike at lag 1.

Forecasting with the auto.arima model

We forecat for three years of the test set and we plot on the same graph the forecated and the actual values.



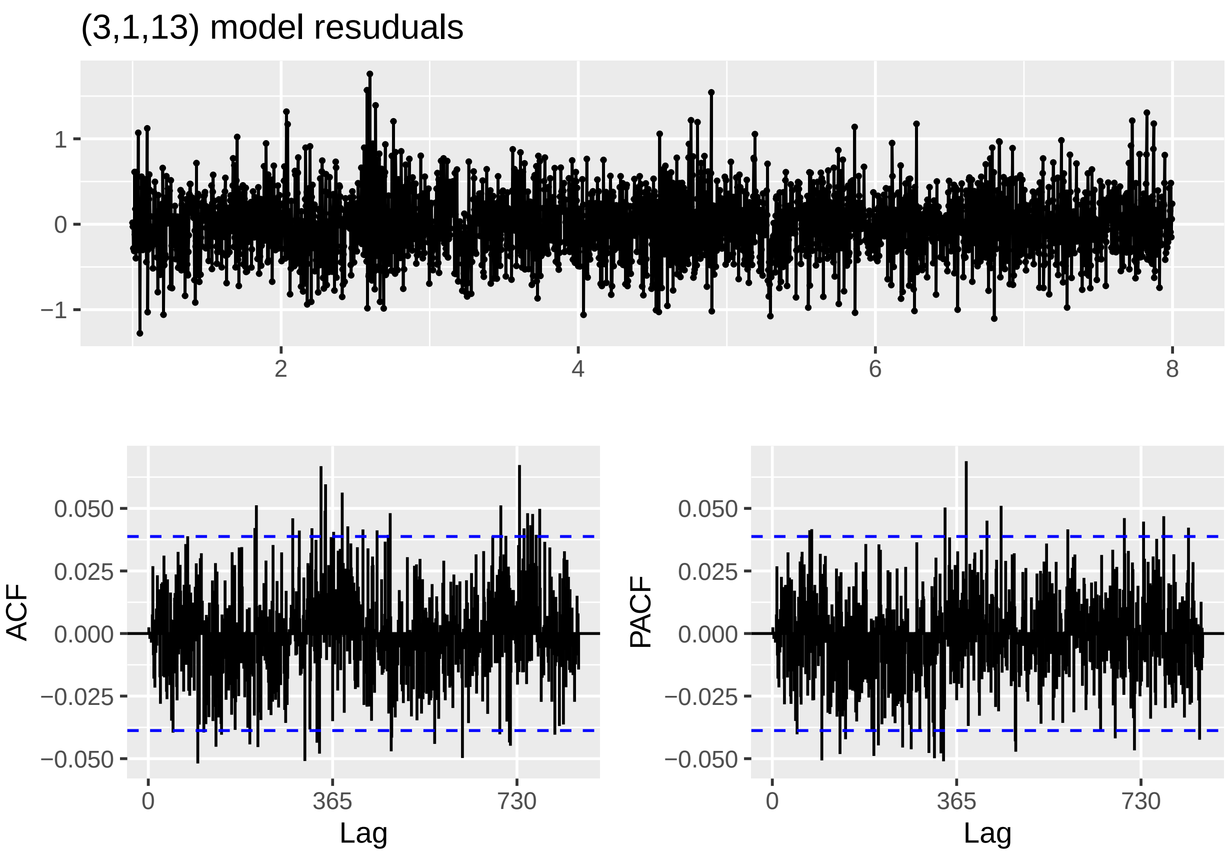
The forcated values (blue line on the plot) are in the 95% confidence interval but it do not fallow the trend of the actual value ploted in red color. The actual temperatures are outside of the 95% of the forcasted area. This means that our model does not perform well.

Non seasonal Arima model

In the Dickey-Fuller Test output we have the lag oder equal to 13, so we will be using q=13.

```
##
## Call:
## arima(x = wkl_na, order = c(3, 1, 13))
##
## Coefficients:
##          ar1     ar2      ar3      ma1      ma2     ma3     ma4    ma5      ma6
##        1.193  0.2886  -0.4883  -0.6544  -0.7217  0.1979  0.1793  0.047  -0.0387
## s.e.     NaN     NaN      NaN      NaN      NaN     NaN     NaN    NaN   0.0229
##          ma7     ma8    ma9     ma10     ma11     ma12    ma13
##      -0.8403  0.5606  0.618  -0.1502  -0.1502  -0.0502  0.0146
## s.e.     NaN     NaN    NaN   0.0084      NaN      NaN  0.0192
##
## sigma^2 estimated as 0.1378:  log likelihood = -1097.27,  aic = 2228.54
```

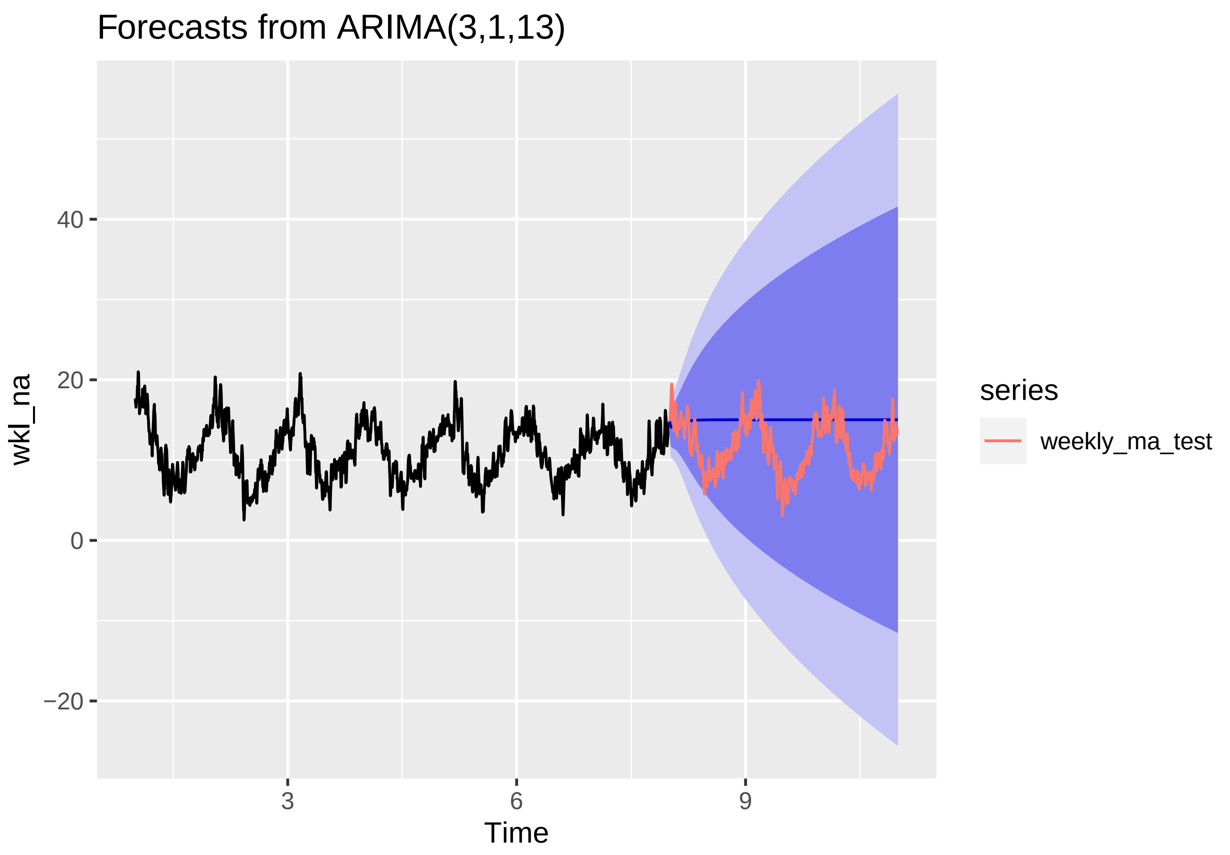
After fitting multiple arima models we choose arima(3,0,13). This model gives the smallest aci=2228.54.



The Acf and Pacf have some significant spikes mostly at the midle of plot.

Forcasting for arima(3,1,13)

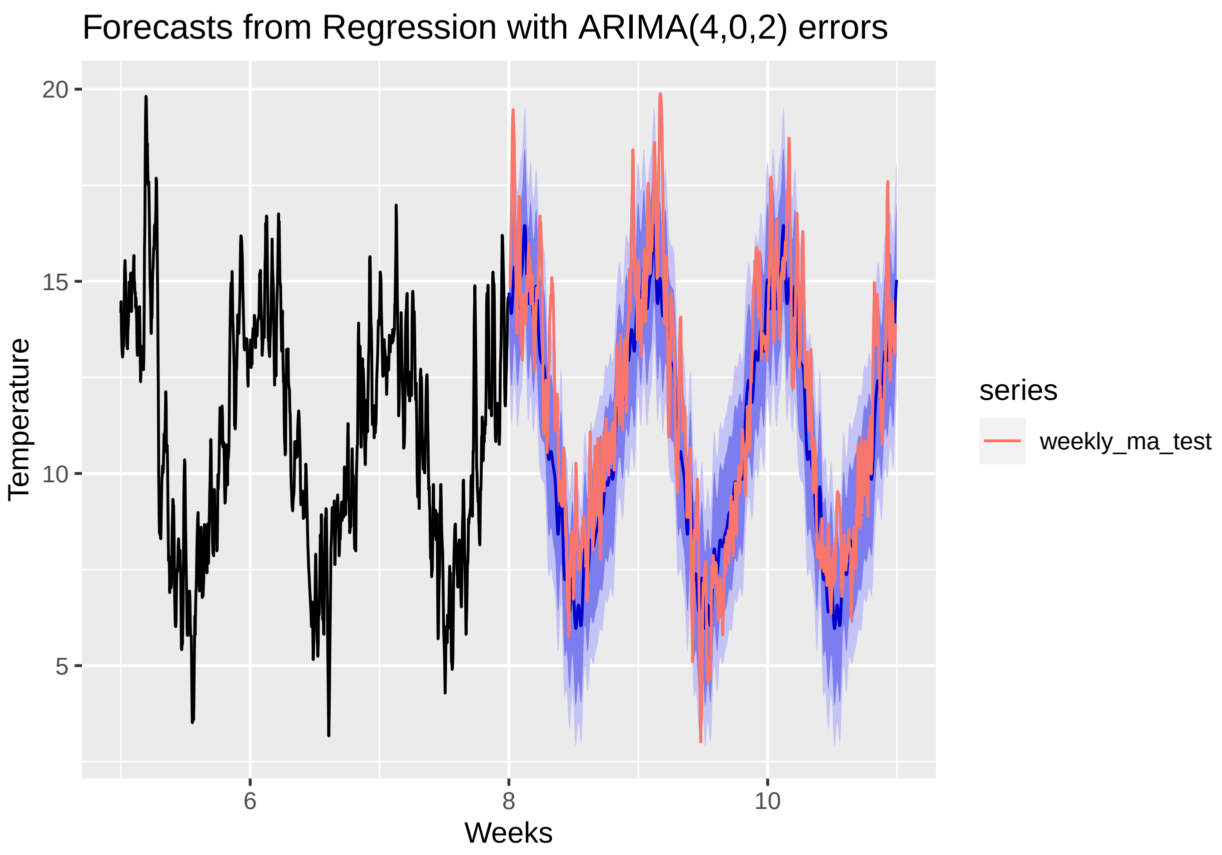
We forecast for the test set period (the three first years) and we plot the actul and the forecasted values on the same plot. The forecasted values (blue) and actual values (in red)are within the 95%. This is a good indicator of the accuracy of the model.



#

Dynamic harmonic regression with multiple seasonal periods using arima.

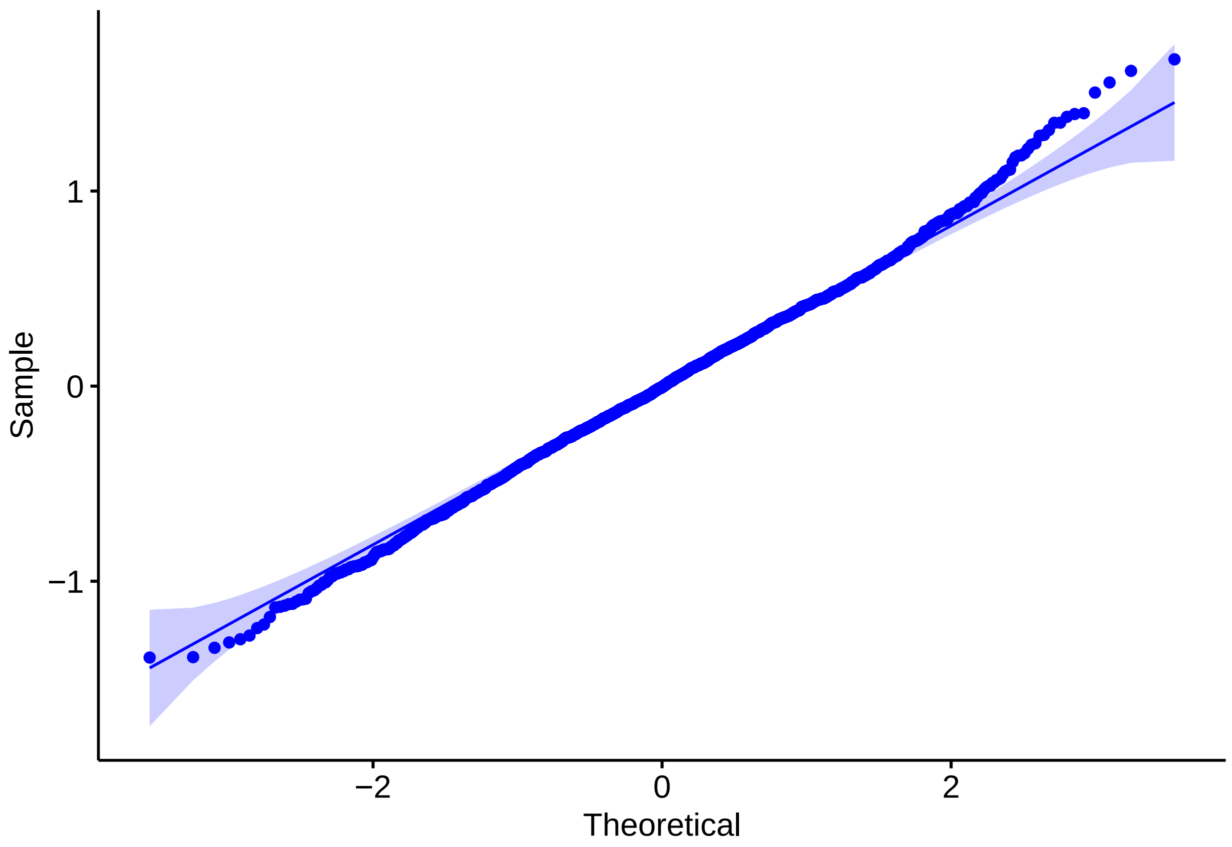
With multiple seasonalities, we can use Fourier terms to fit an arima model. Because there are multiple seasonalities, we need to add Fourier terms for each seasonal period. In this case, the seasonal periods are 7, 12 and 365. However since we using the weekly moving average the seasonal term is 52. The total number of Fourier terms for each seasonal period have been chosen to minimise the AICc, thus the K =52/2. The plot below show the forecasted plot for the 3 years of the test set.



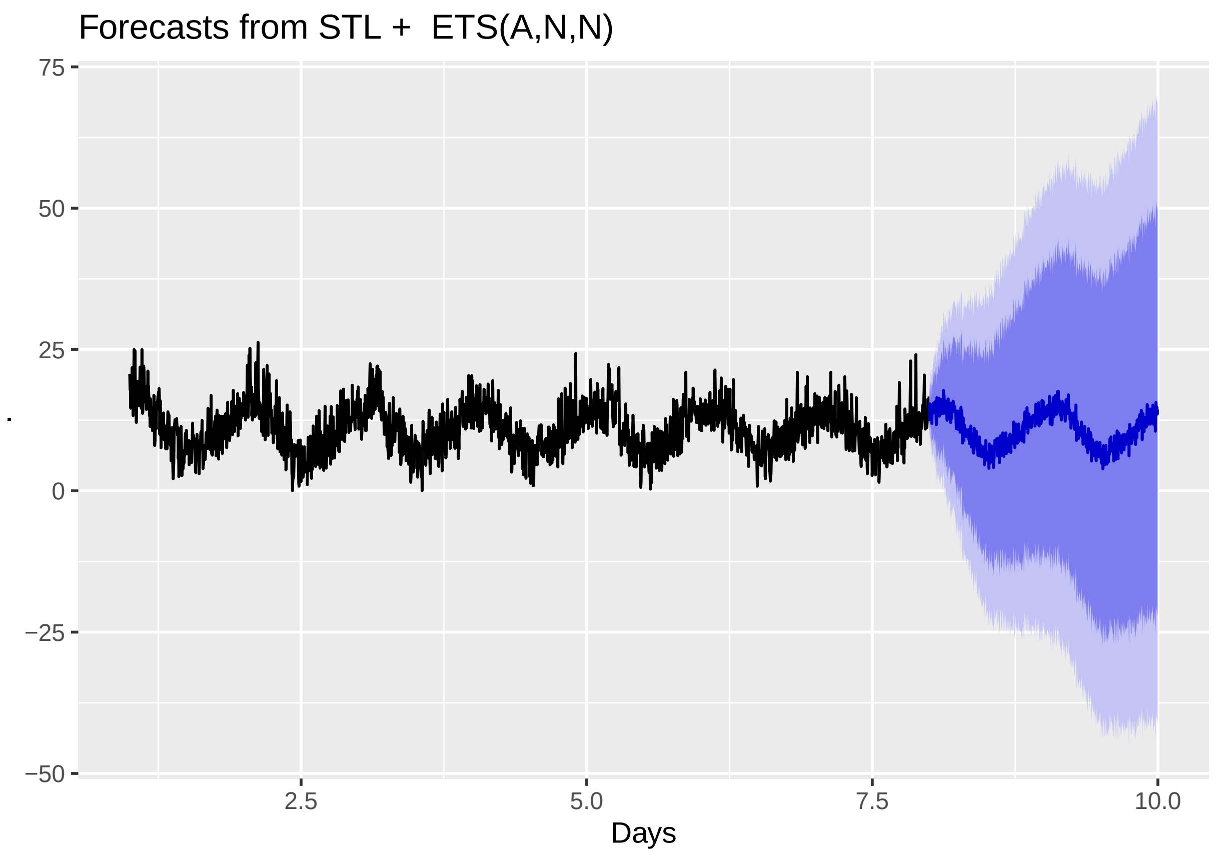
We clearly see on the plot above that the forcating is more accurate. The forcasted values in blue fallow the actual value in blue and they all are within 95% interval.

Conclusion

Arima(4,0,2) with harmonic regression gives the most accurate model. Bellow we plot the QQ-plot of the residual of the model.



The qq normal plot shows that the residuals do not follow the normal distribution. Multiple seasonalities are difficult to forecast with a simple arima model. We can alternatively use a more advence technics such seasonally adjusted data forecasting using ETS. The forcasted plot of such model is given bellow.



This model clearly forecast better in this case.

R Code

```r
library(dplyr)
library(ggplot2)
library(forcats)
library(tseries)
library(tidyverse)
library(rio)
library(forecast)
library(ggpubr)
library(fpp2)
```

```r
dmt=read.csv("daily-minimum-temperatures-in-me.csv")
attach(dmt)
```

```r
dmt$Date = as.Date(dmt$Date, format="%m/%d/%Y")
dmt$Daily.minimum.temperatures = as.numeric(dmt$Daily.minimum.temperatures)
```

```r
ts.dmt = msts(dmt[, c("Daily.minimum.temperatures")],  seasonal.periods  = c(7, 12, 365))
dmt$ts.dmt=ts.dmt
```

```
myts.train <- subset(ts.dmt, end = length(ts.dmt)-1095)
myts.test <- subset(ts.dmt, start= length(ts.dmt)-1094)
```

```
autoplot(myts.train, col = "red")
```

```
par(c(2,2))
weekly_ma = ma(myts.train, order=7)
monthly_ma = ma(myts.train, order=30)
p1= autoplot(myts.train, colour="red") + ggtitle("Daily minimum Temperature")
p2 = autoplot(weekly_ma, colour="blue") + ggtitle("Weekly Moving Average")
p3= autoplot(monthly_ma, colour="green") + ggtitle("Daily Moving Average")
gridExtra::grid.arrange(p1,p2,p3, nrow=2)
```

```
wkl_dcp = mstl(weekly_ma, s.window = "periodic")
wkl_deseasonal = seasadj(wkl_dcp)
autoplot(wkl_dcp)
```

```
weekly_ma_test = ma(myts.test, order=7)
```

```
wkl_dcp_test = mstl(weekly_ma_test, s.window = "periodic")
wkl_deseasonal_test = seasadj(wkl_dcp_test)
```

```
wkl_na = na.interp(weekly_ma)
```

```
adf.test(wkl_na)
```

```
acf = ggAcf(wkl_deseasonal, main="")
pacf = ggPacf(wkl_deseasonal, main="")
gridExtra::grid.arrange(acf, pacf, nrow=1)
```

```
fit1 = auto.arima(wkl_na, seasonal = F)
fit1
```

```
ggtsdisplay(residuals(fit1), main = "auto.arima model resuduals")
```

```
fit1 %>%
  forecast(h=1095) %>%
  autoplot() + autolayer(weekly_ma_test)
```

```
fit2 = arima(wkl_na, order = c(3,1,13))
fit2
```

```
ggtsdisplay(residuals(fit2), main = "(3,1,13) model resuduals")
```

```
fit2 %>%
  forecast(h=1095) %>%
  autoplot() + autolayer(weekly_ma_test)
```

```r
fit3 <- auto.arima(wkl_na, seasonal=FALSE, xreg=fourier(wkl_na, K=26))
fit3 %>%
  forecast(xreg=fourier(wkl_na, K=26, h=1095)) %>% autoplot(include=1095) +
  ylab("Call volume") + xlab("Weeks")
```

References

Athanasopoulos, George and Rob J Hyndman, (2008) *Forecasting: Principles and Practice.* Monash University, Australia.

J. Brockwell, Peter and Richard A. Davis *Introduction to Time Series and Forecasting* Second Edition Springer.

D. Cryer, Jonathan and Kung-Sik Chan (2008) *Time Series Analysis With Applications in R.* Second Edition. Springer.