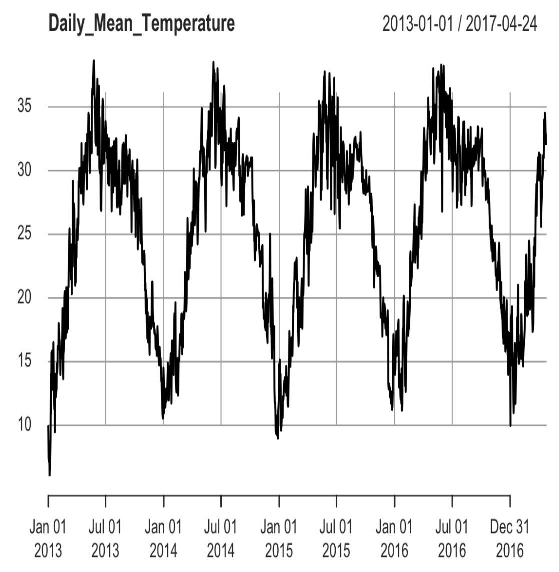
Exploratory Data Analysis of New Delhi Weather

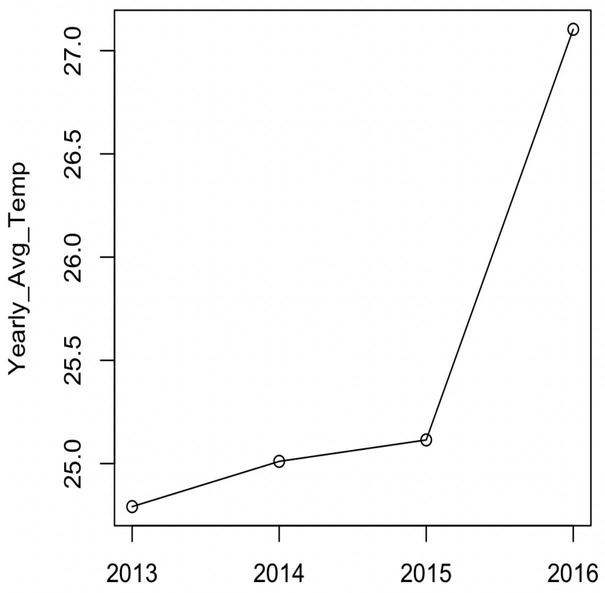
Summary:

In this project, we are working on the time series data of daily temperature at Delhi, India, acquired from Kaggle. The dataset provides Delhi's daily temperature data from 1st January 2013 to 14th April 2017. After comparison, we chose a parametric model plus ARMA(1,1) to fit the data and make predictions. The predicted daily mean temperature clearly has a yearly trend. Though we are only required to predict the next 10 data points, predictions on the next few years offer a sneak peek into global warming.

1 Introduction

The climate of Delhi is humid subtropical and semi-arid which is attributed to high temperature in the summer. According to our graph of time series, there exists seasonality and the change of season leads to significant change in temperature. The undergoing global warming due to human activity is also a variable that affects our prediction. We are going to predict the next 10 days, which is from 04/15/2017 to 04/24/2017, based on the historical data.





The temperature in Delhi fluctuates with a stable pattern across the years, and it looks homoscedastic because variance is stable across the year. At first glance, the yearly mean temperature is stable at around 25 ℃ across years, however, if we take a closer look and actually compute the mean temperature each year, we will actually find Delhi's mean temperature across the year increases in 2013 2017, perhaps due to global warming. There is a strong seasonality pattern (with spikes in summer and winter), with periods being 365 days or 12 months. Since the raw data displays homoscedasticity, we don't need to use variance stabilizing transforms such as log() or sqrt() function. The presence of (relatively weak) linear trend and (strong) seasonality implies we need to remove these linear and seasonal trends to achieve stationarity, before we can use ARMA models to model the residuals.

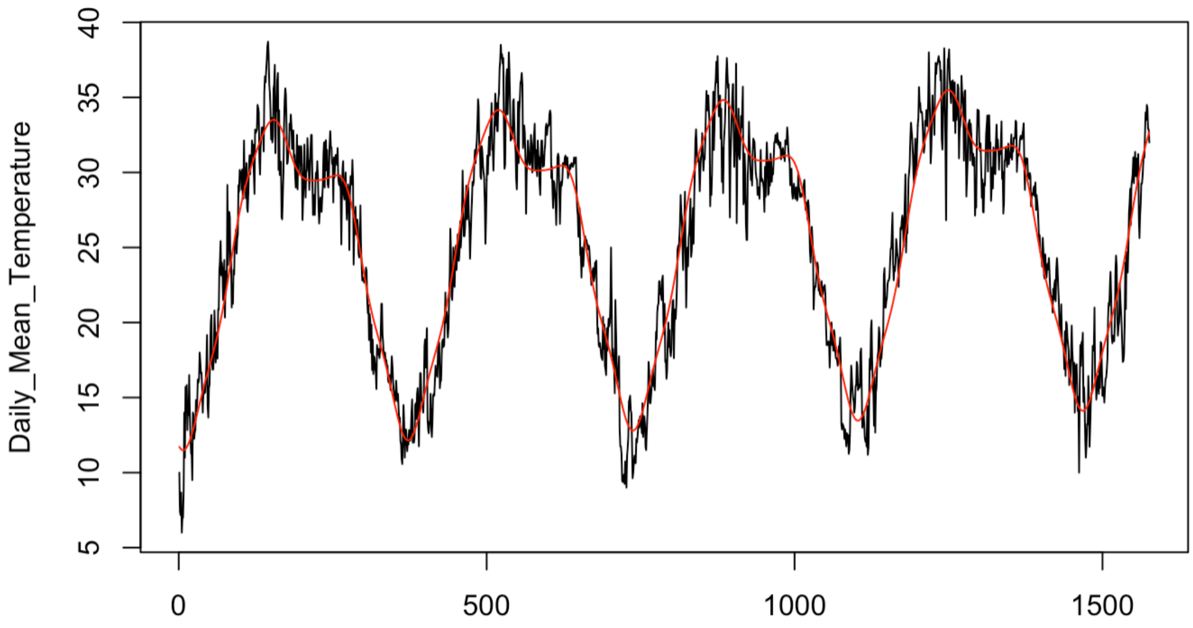
3 Models Considered

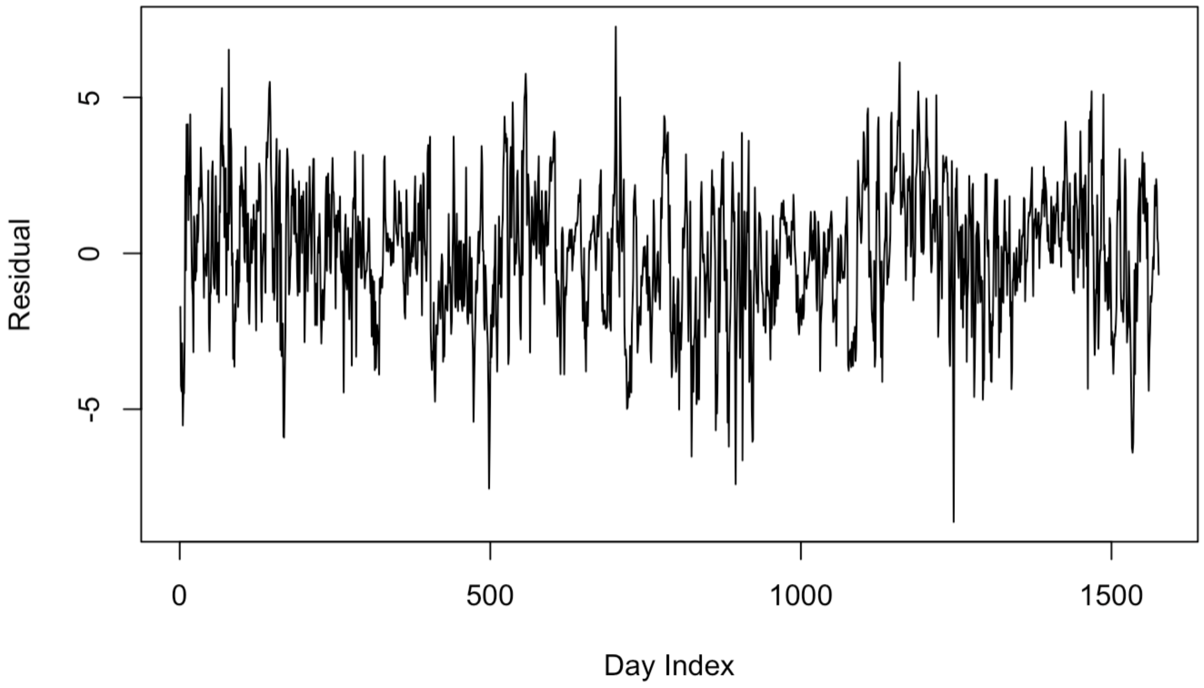
3.1 3.1.1 Parametric model + AR(1)

First, we observed the significant seasonal pattern in our temperature data, so the thought of using sinusoidal regression naturally came to our mind, since sinusoidal functions are good at modeling seasonality. Our first attempt was to regress on sinusoidal functions only, which has satisfactory outcome. However, we managed to improve on it by also incorporating polynomial in the regression, which, by testing on AIC/AICc/BIC values and cross-validation, yields better outcome than only using sinusoidal functions.

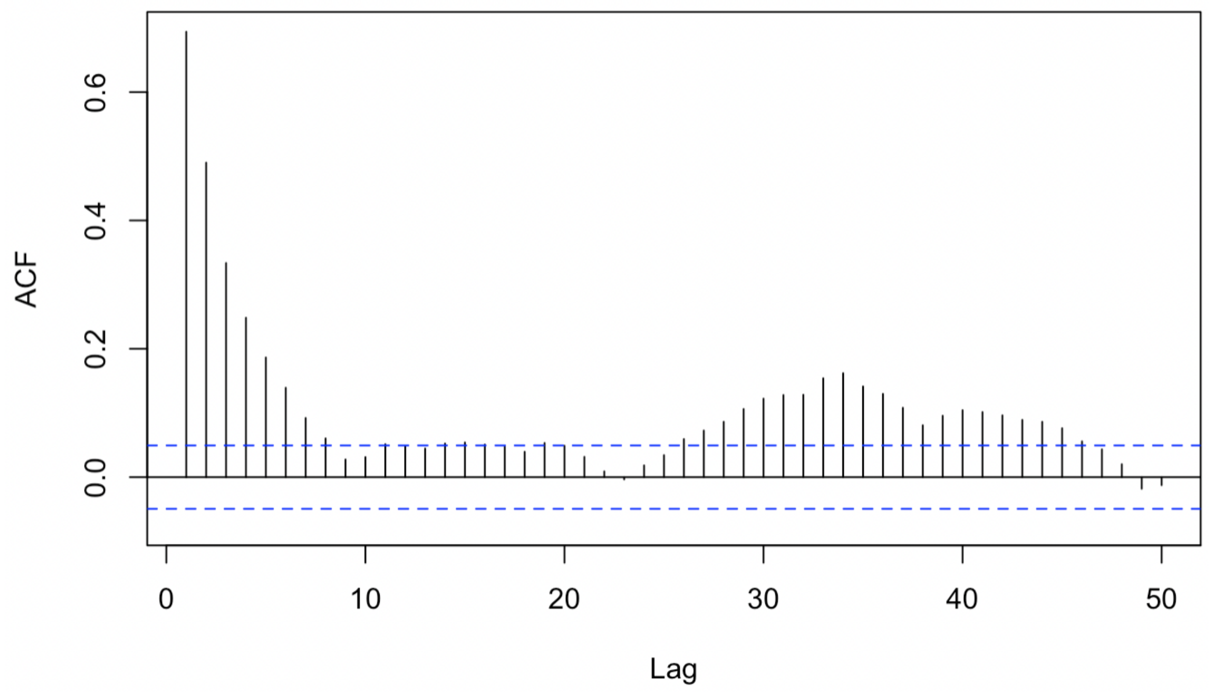
Our proposed paramatric model is as follow:

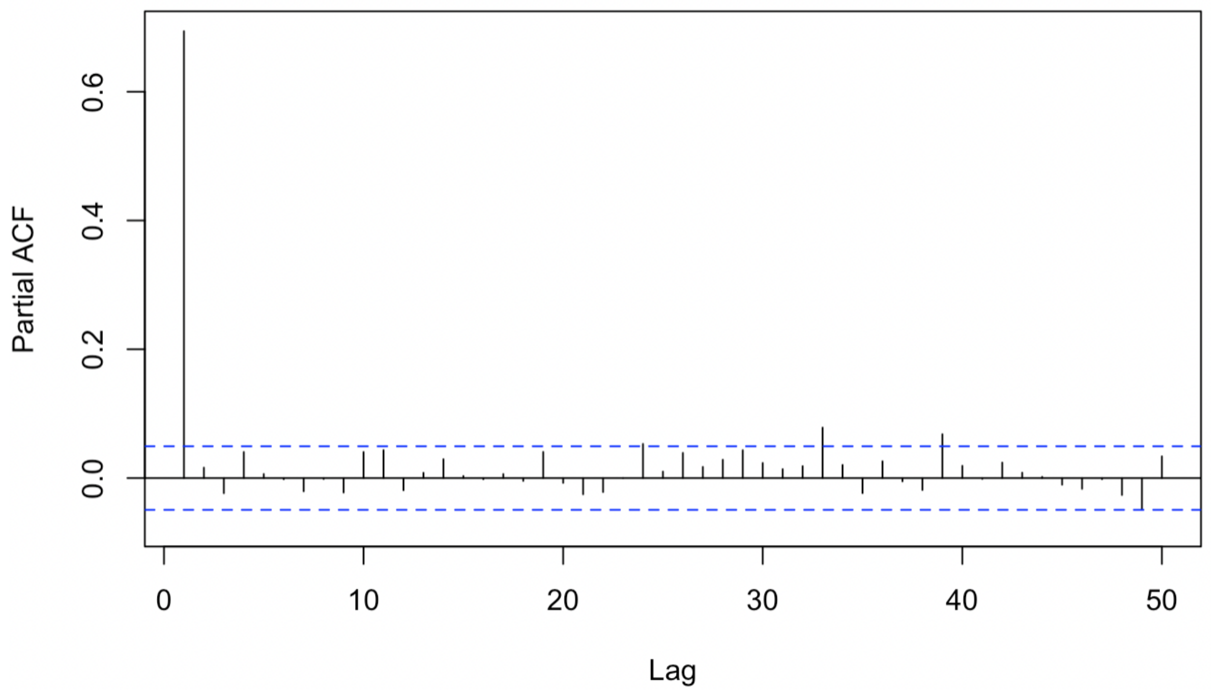
$$y_t = \beta_0 + \beta_1 t + \sum_{j=1}^{6}[\beta_{2j} cos(\frac{2\pi jt}{365.25}) + \beta_{2j+1} sin(\frac{2\pi jt}{365.25})] + X_t \tag{1}$$



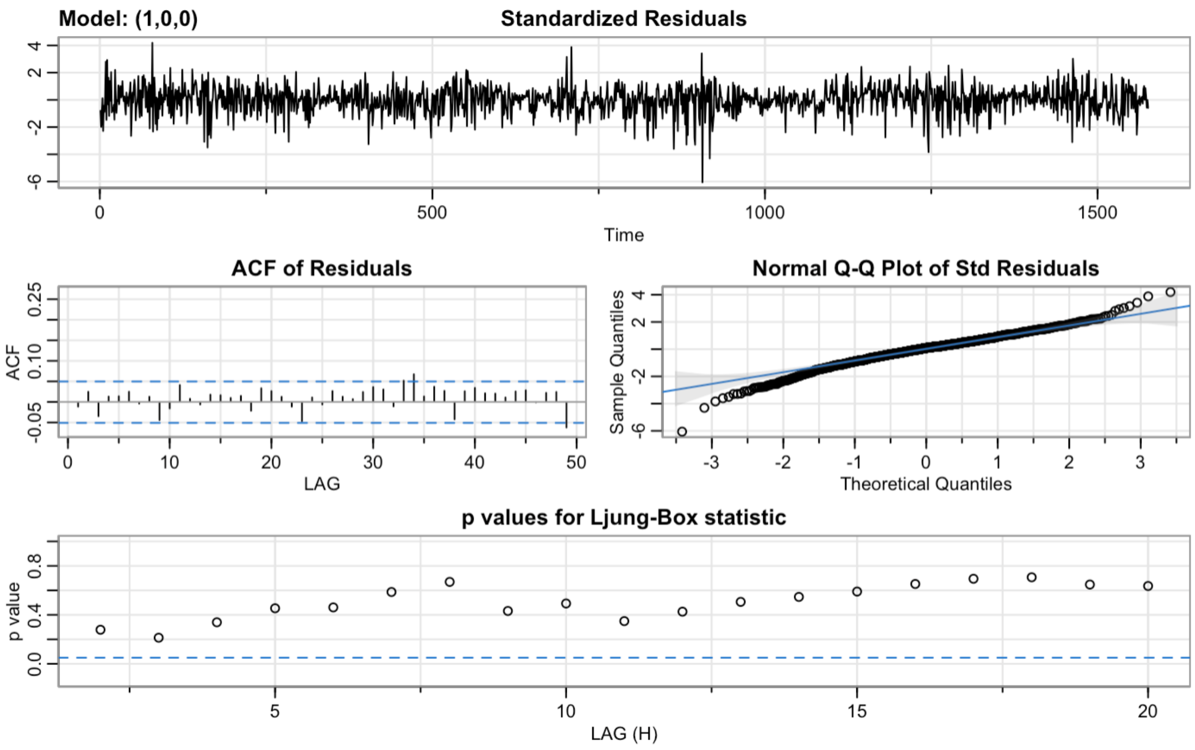


The ACF & PACF plots for the residuals are shown below:





The PACF plots for the residuals supports that we can use an an AR(1) model (because PACF has only one huge spike at lag = 1), we first propose to use AR(1) model to fit the residuals. We include the diagnostic plot and theoretical ACF & PACF plots here. This AR(1) model has AIC: 3.704059, AICc: 3.704064, BIC: 3.714267 for the residuals.



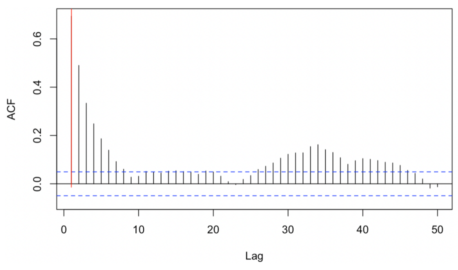


Figure 1: ACF

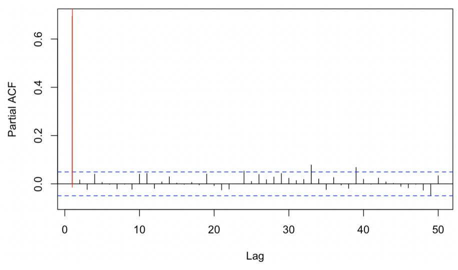
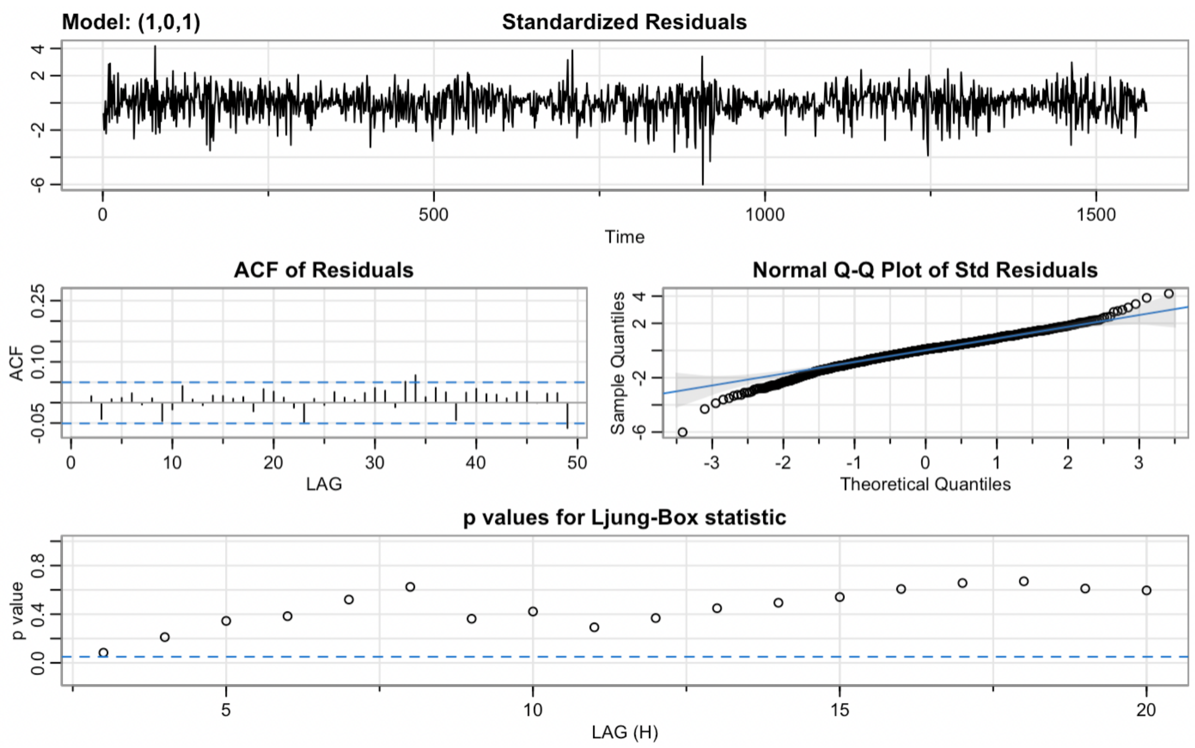


Figure 2: PACF

As we can see from the diagnostic plot, the residuals seem stationary, no significant ACF of residuals is present, and the Ljung-Box statistic shows all p-values are above than 0.05, all these signify that AR(1) is a good model for our residuals.

3.2 3.1.2 Parametric model + ARMA(1,1)

After exploring the previous AR(1) model, we also explored ARMA(1,1) model as another candidate, because the ACF and PACF plot for the residuals can also come from an ARMA(1,1) model, and the the ACF and PACF plot for the residuals still display some remaining significant ACF and PACF values after lag = 1. We wish to try an ARMA(1,1) model to see if it can better model the residuals.



edition

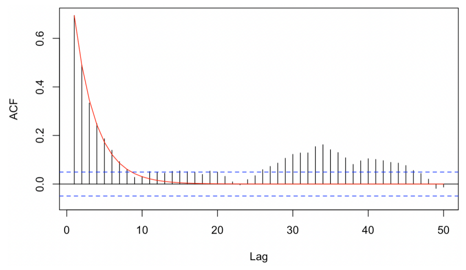


Figure 3: ACF

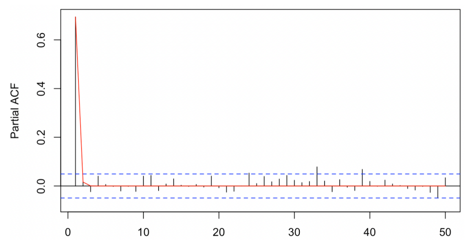


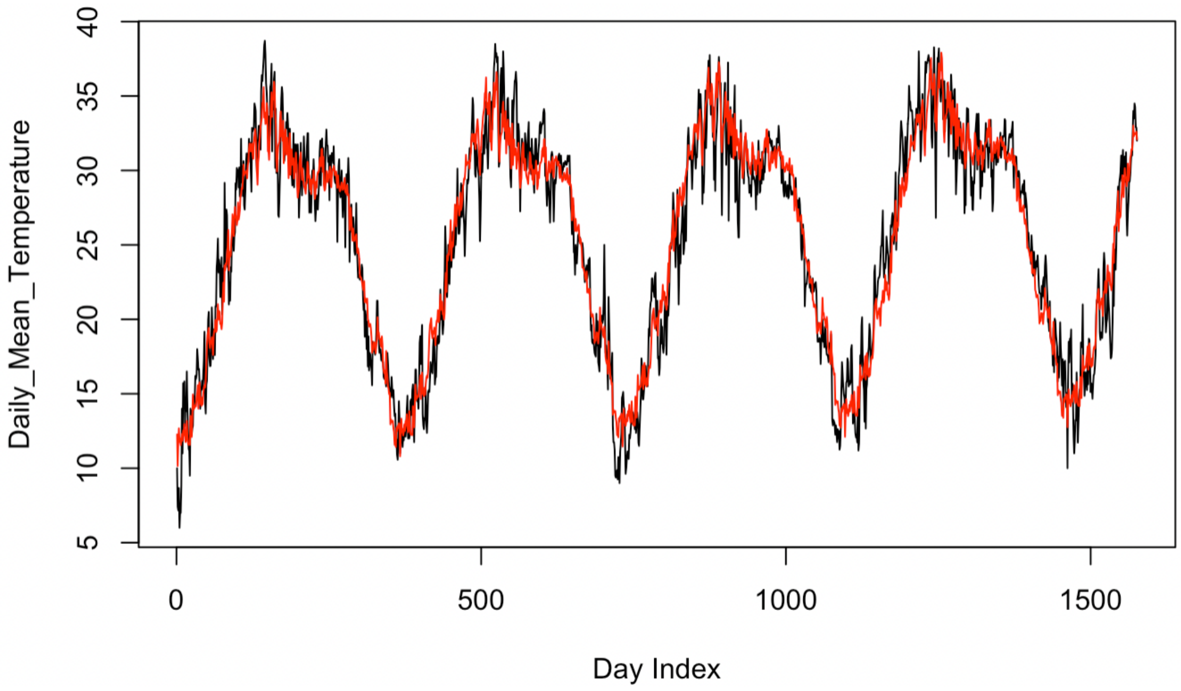
Figure 4: PACF

The residuals seem stationary, no significant ACF of residuals is present, and the Ljung-Box statistic shows all p-values are above than 0.05, all these signify that ARMA(1,1) is a good model for our residuals.

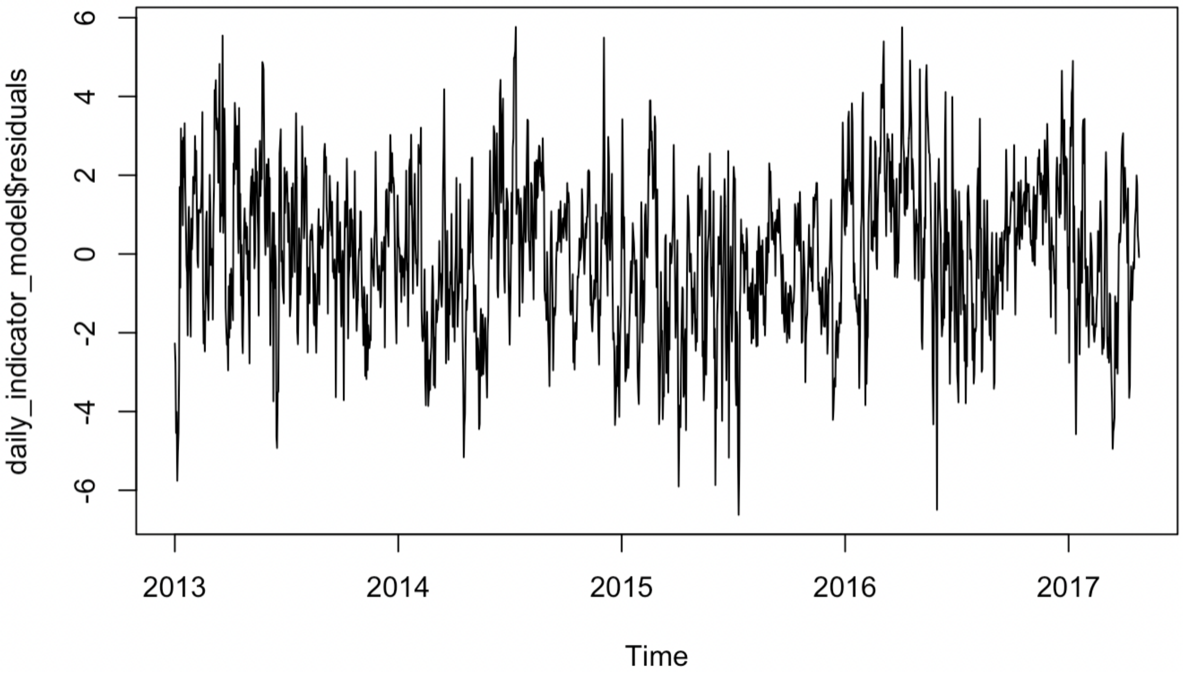
3.3 3.2.1 (Linear Trend + Daily Indicator) Model + AR(1)

Since the data has very significant seasonal trend with seasonality of 365 days, the thought of using SARIMA models with S=365 naturally came to our mind. However, since our seasonality is large (S=365), it will result in very large number of parameters to calculate, which R doesn't support (in the sarima() function, is the seasonal ARMA parameters P and Q are non-zero, that is, if we wish to include the seasonal ARMA part, then the maximum lag supported is 350, but our data has seasonality S=365). Hence, we can't fit our model using SARIMA in this way. Then, the best we could do is to set the seasonal differencing parameter D=1 but leave seasonal ARMA parameters P and Q to both be 0. However, the resulting residual would have significant ACF at lag = 365. If would be good if we could set Q = 1 to use seasonal MA to resolve this issue, however, since our lag = 365 ¿ 350, we can't use R to process it. Therefore, we had to come up with other ways that capture seasonality well, but also being computationally viable, to model our raw data.

Luckily, we do have another powerful tool at hand, that is, to use indicator variables that indicate each day in a year. The justification is that, since our data exhibits extremely significant seasonality of one year, with a weak linear trend, then it implies that after removing the linear trend, the same date in each year (such as Feb. 1st in each year) should have roughly the same temperature, and the pattern of temperature change inside a year should be roughly the same acroos years. This also makes sense empirically, since we expect the temperature in Delhi in July 1st should be higher than that in Feb 1st each year.



Upon analyzing the residuals after fitting the raw data using our (Linear Trend + Daily Indicator) Model, the ACF and PACF plot suggests we use an AR(1) model to fit the residual. All the relevant graphs are shown below:



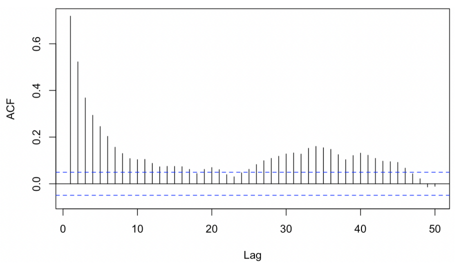


Figure 5: ACF

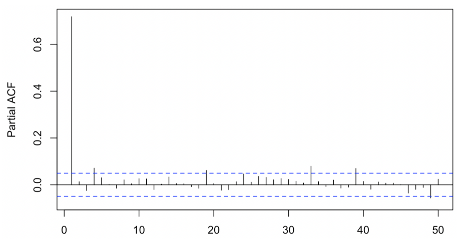
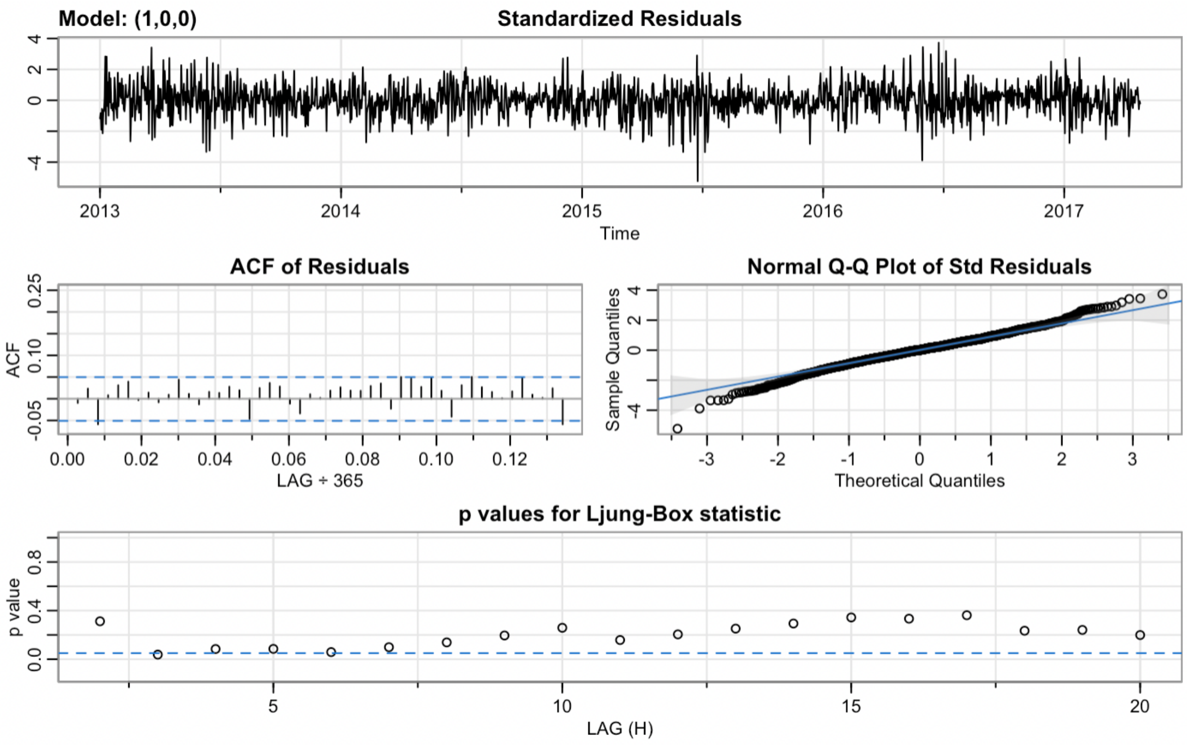


Figure 6: PACF



The residuals seem stationary, no significant ACF of residuals is present, and the Ljung-Box statistic shows most p-values are above than 0.05, all these signify that ARMA(1,1) is a good model for our residuals.

3.4 3.2.2 (Linear Trend + Daily Indicator) Model + ARIMA(3,1,2)

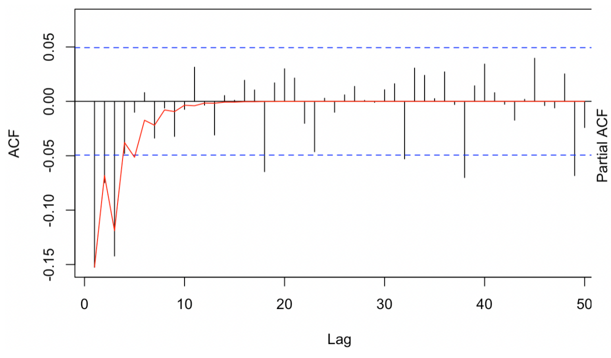


Figure 7: ACF

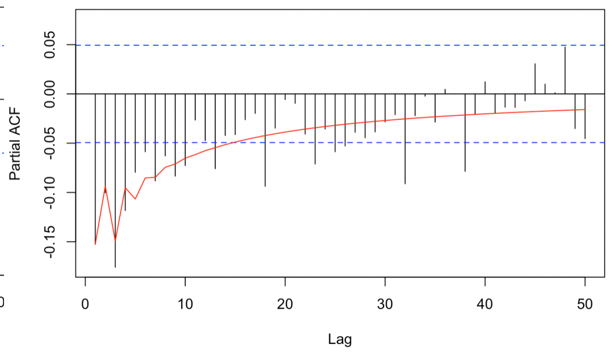
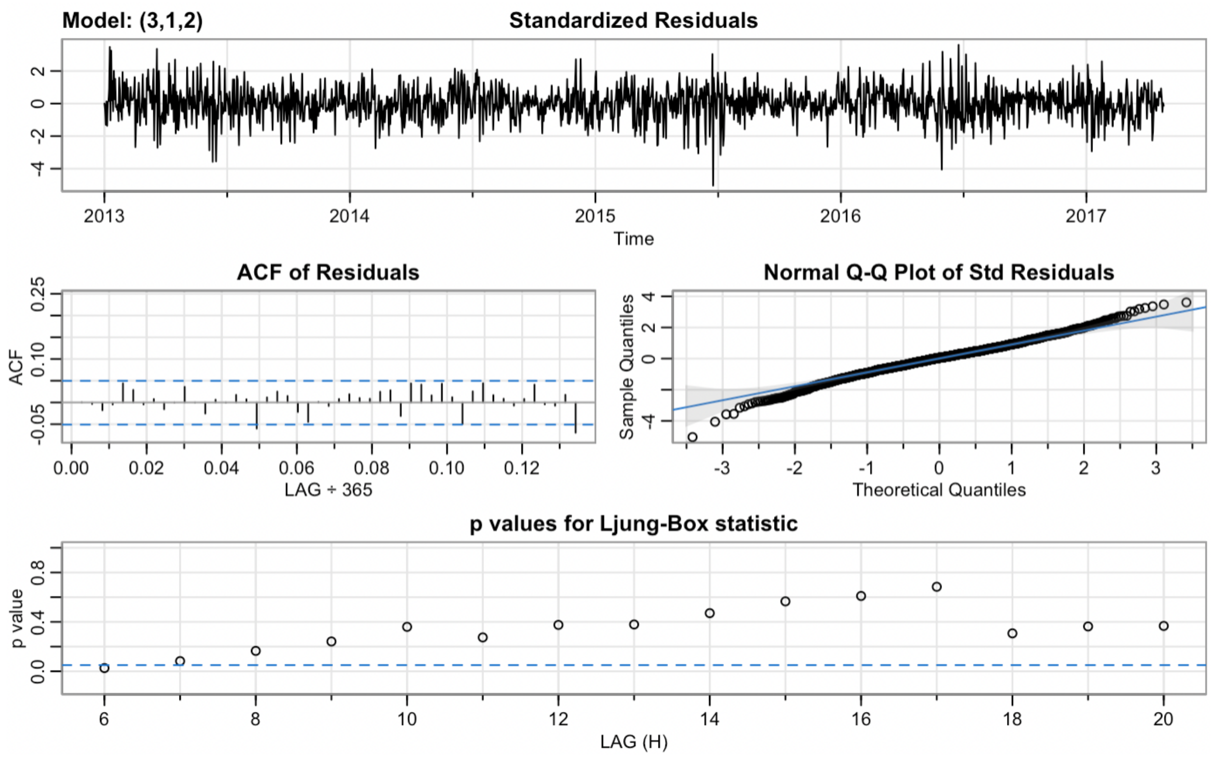


Figure 8: PACF



The residuals seem stationary, no significant ACF of residuals is present, and the Ljung-Box statistic shows most p-values are above than 0.05, all these signify that ARMA(1,1) is a good model for our residuals.

4 Model Comparison and Selection

In this section, you derive an ARIMA (or SARIMA, MSARIMA, etc.) model, which you use in Section 5 to make predictions. On the way, you should compare different candidate models and do diagnostics for them (which can include different time functions from the previous section too). For example, perhaps you compare a parametric trend function with MA(1) noise to a second-differences with AR(1) noise model, etc. Go through the lecture materials and try your best to apply all the different approaches to find models, evaluate their accuracy, and make comparison between different models. For example, there should certainly be ACF and PACF plots shown here! Sometimes, it may be helpful to introduce subsections.

4.1 In-sample AIC/AICc/BIC Scores Comparison

Model	AIC	AICc	BIC
Parametric model	7291.278	7514.881	9259.369
AR(1) for Parametric Model's Residuals	3.704059	3.704064	3.714267
ARMA(1,1) for Parametric model's Residuals	3.705054	3.705063	3.718665
(Linear Trend + Daily Indicator) Model	6897.454	6897.762	6977.894
AR(1) for (Linear Trend + Daily Indicator) model's Residuals	3.437964	3.437969	3.448172
ARMA(1,1) for (Linear Trend + Daily Indicator) model's Residuals	3.441707	3.441741	3.465539

Comments: These are the in-sample AIC/AICc/BIC scores for our four candidate models. We see that the parametric model (with two different ARIMA models for the residuals) has higher AIC/AICc/BIC scores compared to that of the (Linear Trend + Daily Indicator) model. This makes sense because our parametric model aim to capture the trend or "general shape" of the temperature pattern, and it produces a smooth fitted curve, while our indicator model try to capture the "characteristic" for each day in a year, which produces more "jagged" fitted curve. Our indicator model may be better at fitting in-sample data, which explains the lower AIC/AICc/BIC scores for indicator models. For fitting the residuals, we notice that the "simpler" ARMA model, i.e., ARMA models with fewer parameters, have lower AIC/AICc/BIC scores, because while candidate ARMA models both fit the residuals relatively well, the simpler ARMA models have less parameters, contributing to lower AIC/AICc/BIC scores. We still need cross-validation to test out-of-sample performance for our data.

4.2 Cross Validation Model Comparison

For example, you may want to compare different models by how well the predict future values (as your predictions will also be evaluated in this final project).

Model	MSE
Parametric model + AR(1)	33.15451
Parametric model + ARMA(1,0,1)	33.12917
(Linear Trend + Daily Indicator) Model + AR(1)	47.35897
(Linear Trend + Daily Indicator) Model + ARIMA(3,1,2)	49.52325

These are the out-of-sample MSE values for our four candidate models. As we can see, our parametric models actually perform better than (Linear Trend + Daily Indicator) models. To explain why, we believe the reason is that the parametric model (which has fewer parameters than daily indicator model) provide a smoother fitted curve that better generalize for out-of-sample data, while the (Linear Trend + Daily Indicator) captures the characteristic for each day within a year, which produces a more jagged fitted curve. Hence, the parametric model may be better at avoiding over-fitting issue, which may present in our (Linear Trend + Daily Indicator) model. For fitting the residuals of the parametric model, the cross-validation is in favor of ARMA(1,0,1) model. To summarize, our cross-validation suggests Parametric model + ARMA(1,0,1) to be the best model. Hence, we will use this as our chosen best model for prediction.

5 Results

We've chosen the best model from our four candidate models to be the Parametric + ARMA(1,0,1) model. To write it out explicitly:

$$y_t = \beta_0 + \beta_1 t + \sum_{j=1}^{6} [\beta_{2j} cos(\frac{2\pi jt}{365.25}) + \beta_{2j+1} sin(\frac{2\pi jt}{365.25})] + X_t \qquad (2)$$

Where X_t is modeled by ARMA(1,1) model:

$$X_t = \phi_1 X_{t-1} + W_t + \theta W_{t-1} \qquad (3)$$

5.1 Estimation of model parameters

(Intercept)	24.17*** (0.11)
c1	-9.33*** (0.08)
c2	-2.51*** (0.08)
c3	0.05 (0.08)
c4	-0.05 (0.08)
c5	-0.21** (0.08)
c6	-0.32*** (0.08)
s1	-0.62*** (0.08)
s2	-1.68*** (0.08)
s3	0.69*** (0.08)
s4	-0.33*** (0.08)
s5	-0.11 (0.08)
s6	-0.19* (0.08)
time	0.00*** (0.00)
R^2	0.92
Adj. R^2	0.91
Num. obs.	1576

$^{***}p < 0.001$; $^{**}p < 0.01$; $^{*}p < 0.05$

Table 1: Polynomial model coefficients

Parameter	Estimate (s.e)
ϕ_1	0.7059 (0.0253)
θ_1	-0.0232 (0.0353)

Table 2: ARMA(1,1) model coefficients

5.2 Prediction (With a technical caveat)

Here, we'll provide two ways for prediction for the daily mean temperature for Delhi for the next 10 days (from 04/15/2017 to 04/24/2017). The first one is done by naively adding the signal model's forecast for this time period, plus the ARMA(1,1) model's forecast for the same time period. However, this "naive" method has several drawbacks. Firstly, it makes calculating the prediction confidence interval (standard errors for the prediction) for the joint model extremely difficult, if not impossible, because while we have the summation of prediction from the signal model and the ARMA(1,1) model, we can't calculate the standard errors for the joint of these two models (we can't just add the s.e. from these two models, and the correct way to calculate the joint s.e. may require more information than we have, such as the exact distribution of the two models). Secondly, to be more rigorous and more theoretically correct, in real life, it's better to fit the signal model at the same time while accounting the residuals using the ARMA(1,1) model at the beginning, because this estimates the joint-model by maximum-likelihood, as opposed to fitting the signal model and the ARMA model for residuals separately. Hence, to be more correct and acquire better (again, more correct) statistics for our prediction, we also include the prediction from the model that's jointly-fitted using the signal model and the ARMA(1,1) model, and we include the prediction confidence interval and prediction standard errors for that model here. While these two methods differ in theory, we examined the prediction in future 10 days, and found them to be very close with negligible differences, as can be seen in the 10-future-points prediction table below.

Figure 9: Prediction using the "naive" sum-up method

Date	Temperature
04/15/2017	32.35274
04/16/2017	32.61673
04/17/2017	32.83874
04/18/2017	33.03065
04/19/2017	33.20091
04/20/2017	33.35553
04/21/2017	33.49880
04/22/2017	33.63375
04/23/2017	33.76255
04/24/2017	33.88675

Table 3: Predicted Values from Naive Method

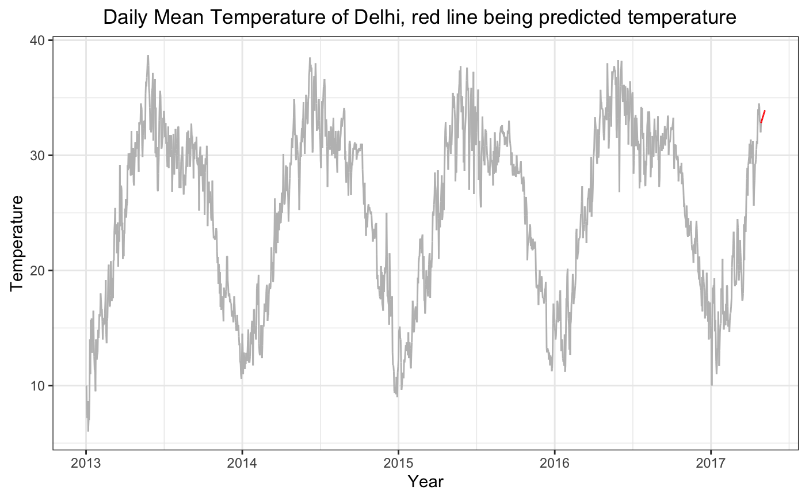
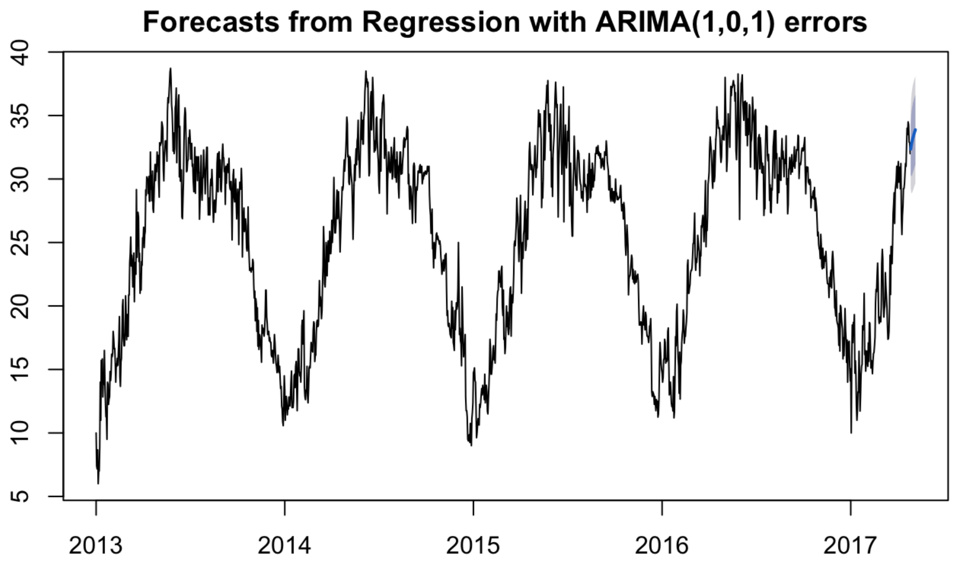


Figure 10: Prediction using the "more correct" jointly-fitted model, with correct prediction confidence interval included

Date	Temperature (s.e.)	95% CI
04/15/2017	32.35274 (1.55)	29.32 ~ 35.38
04/16/2017	32.61673 (1.87)	28.94 ~ 36.28
04/17/2017	32.83874 (2.02)	28.88 ~ 36.78
04/18/2017	33.03065 (2.08)	28.94 ~ 37.11
04/19/2017	33.20091 (2.12)	29.05 ~ 37.34
04/20/2017	33.35553 (2.13)	29.17 ~ 37.53
04/21/2017	33.49880 (2.14)	29.30 ~ 37.68
04/22/2017	33.63375 (2.14)	29.42 ~ 37.83
04/23/2017	33.76255 (2.15)	29.55 ~ 37.96
04/24/2017	33.88675 (2.15)	29.67 ~ 38.09

Table 4: Predicted Values, Standard Errors, and 95% CI, from Revised Method



6 Conclusion

Our prediction for the next 10 days seem satisfactory and reasonable, and we've also tried to predict into a further future, which also yielded reasonable outcomes, because they observe the seasonality and the slowly uprising linear trend observed in the training set. From this project, we learned the entire procedure of time-series analysis on real-life data, and we horned our skills in R programming, as well as LaTex! We also learned about the difference between fitting signal model and ARMA model for residual, and fitting them simultaneously, which we haven't learned in class. It's an extremely meaningful journey to finish this project!

~~~~
---
title: "Stat 153 Project"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

```{r}
library(AICcmodavg)
```

```{r cars}
library(zoo)
library(forecast)
library(Metrics)
library(xts)
library(astsa)

library(TSA)
library(texreg)
library(lmtest)
library(ggplot2)


# EDA
temp <- read.csv(file = 'data.csv')
df <- data.frame(timestamp=temp$date,temperature=temp$meantemp)
ZOO <- zoo(df$temperature,
order.by=as.Date(as.character(df$timestamp),format='%Y-%m-%d'))
print(ZOO)
idx <- seq(as.Date("2013-01-01"), as.Date("2017-04-14"), by = "day")
Daily_Mean_Temperature <- ts(ZOO,frequency=365,start = 2013)
plot.ts(Daily_Mean_Temperature)
time = c(1:length(temp$date))
```

```{r}
daily_indicator_model <- tslm(Daily_Mean_Temperature ~ trend + season)
summary(daily_indicator_model)
plot(time, Daily_Mean_Temperature, type = "l", xlab = "Day Index")
lines(time, daily_indicator_model$fitted.values, col='red')
plot(daily_indicator_model$residuals)
```
```{r}
AIC(daily_indicator_model)
AICc(daily_indicator_model)
BIC(daily_indicator_model)
```

```{r}
res <- daily_indicator_model$residuals
ar1 <- sarima(res, p=1,d=0,q=0)
arma312 <- sarima(res, p=3,d=1,q=2)
```

```{r}
~~~~

~~~
ar1 <- sarima(daily_indicator_model$residuals, p=1,d=0,q=0,)
forecast::Acf(daily_indicator_model$residuals,lag.max=50)
theo_acf <- ARMAacf(ar = c(ar1$ttable[1,1],lag.max=50))


forecast::Pacf(daily_indicator_model$residuals,lag.max=50)
theo_pacf <- ARMAacf(ar = c(ar1$ttable[1,1],lag.max=50),pacf=T)
lines(1:50,theo_pacf[1:50],col="red")
```

```{r}
for1 <- forecast(daily_indicator_model, h=365)$mean
plot(for1)
```
```{r}
for2 <- sarima.for(daily_indicator_model$residuals, p=1,d=0,q=0,
n.ahead=365)
plot(for2$pred)
```

```{r}
arma <- sarima(daily_indicator_model$residuals, p=3,d=1,q=2)
forecast::Acf(diff(daily_indicator_model$residuals),lag.max=50)
theo_acf1 <- ARMAacf(ar =
c(arma$ttable[1,1],arma$ttable[2,1],arma$ttable[3,1]),ma =
c(arma$ttable[4,1],arma$ttable[5,1]),lag.max=50)
lines(1:50,theo_acf1[2:51],col="red")

forecast::Pacf(diff(daily_indicator_model$residuals),lag.max=50)
theo_pacf1 <- ARMAacf(ar =
c(arma$ttable[1,1],arma$ttable[2,1],arma$ttable[3,1]),ma =
c(arma$ttable[4,1],arma$ttable[5,1]),lag.max=50, pacf=T)
lines(1:50,theo_pacf1[1:50],col="red")
```
```{r}
r <- Daily_Mean_Temperature[366:730]
idx <- seq(as.Date("2013-01-01"), as.Date("2017-04-14"), by = "day")
r <- ts(r, frequency=365,start = c(2013+366%/%365))
plot(r)
```

```{r}
#Special case when using only the first year's data to forecast the next
year's temperatures, we need at least 368 days instead of 365 days to
ensure the matrix being invertible, otherwise only the 365 data points
aren't enough to fit the model.
#          1   2    3    4    5     6
points = c(1, 368, 730, 1095, 1461, 1576)
MSE <- c(Indicator_AR1=0, Indicator_ARMA312=0)

for(i in 2:5){
  train = Daily_Mean_Temperature[1:points[i]]
  test = Daily_Mean_Temperature[(points[i]+1):points[i+1]]
  train_time = 1:points[i]
  test_time = (points[i]+1):points[i+1]

  #Fit parametric model
  idx <- seq(as.Date("2013-01-01"), as.Date("2017-04-14"), by = "day")
~~~

```
  train <- ts(train, frequency=365,start = 2013)

  daily_indicator_model <- tslm(train ~ trend + season)
  residuals <- daily_indicator_model$residuals

  #Predict
  forcast <- forecast(daily_indicator_model, h=points[i+1]-points[i])$mean
  ar1 = sarima.for(residuals,n.ahead=(points[i+1]-points[i]),p=1,d=0,q=0)
  arma312 = sarima.for(residuals,n.ahead=(points[i+1]-
points[i]),p=3,d=1,q=2)

  MSE[1] <- MSE[1] + sum((test - forcast - ar1$pred)^2) / (points[i+1]-
points[i])
  MSE[2] <- MSE[2] + sum((test - forcast - arma312$pred)^2) /
(points[i+1]-points[i])
}
print(MSE)
```

#Question 3.1 Parametric Model
```{r}
d = 365.25
c1 = cos(2*pi*time/d)
c2 = cos(2*pi*2*time/d)
c3 = cos(2*pi*3*time/d)
c4 = cos(2*pi*4*time/d)
c5 = cos(2*pi*5*time/d)
c6 = cos(2*pi*6*time/d)
s1 = sin(2*pi*time/d)
s2 = sin(2*pi*2*time/d)
s3 = sin(2*pi*3*time/d)
s4 = sin(2*pi*4*time/d)
s5 = sin(2*pi*5*time/d)
s6 = sin(2*pi*6*time/d)
lin.mod = lm(Daily_Mean_Temperature ~ 1 + c1 + c2 + c3 + c4 + c5 + c6 + s1
+ s2 + s3 + s4 + s5 + s6 + poly(time,1,raw=T))

plot(time, Daily_Mean_Temperature, type = "l", xlab = "Day Index")
lines(time, lin.mod$fitted.values, col='red')

plot(lin.mod$residuals, type = "l", xlab = "Day Index", ylab = "Residual")
forecast::Acf(lin.mod$residuals,lag.max=50)
```

```{r}
texreg(lin.mod)
```

```{r}
AIC(lin.mod)
AICc(lin.mod)
BIC(lin.mod)
```

```{r}
res <- lin.mod$residuals
```

```
ar1 <- sarima(res, p=1,d=0,q=0)
arma11 <- sarima(res, p=1,d=0,q=1)
```

#Cross-validation
#1: 1st year predict 2nd year, 2: 1st + 2nd year predict 3rd year, 3:
1st+2nd+3rd year predict 4th year, 4:1+2+3+4 year predict the rest days
```{r}
#FOR indicator model, there is a special case when using only the first
year's data to forecast the next year's temperatures, we need at least 368
days instead of 365 days to ensure the matrix being invertible, otherwise
only the 365 data points aren't enough to fit the model.
#To Keep constant between parametric model with indicator model, we also
set "first year" to be 368 days
#          1    2    3     4     5     6
points = c(0, 368, 730, 1095, 1461, 1576)
MSE <- c(Poly_Model_AR1=0, Poly_Model_ARMA11=0)

for(i in 2:5){
  train = Daily_Mean_Temperature[1:points[i]]
  test = Daily_Mean_Temperature[(points[i]+1):points[i+1]]
  train_time = 1:points[i]
  test_time = (points[i]+1):points[i+1]

  #Fit parametric model
  d = 365.25
  c1 = cos(2*pi*train_time/d)
  c2 = cos(2*pi*2*train_time/d)
  c3 = cos(2*pi*3*train_time/d)
  c4 = cos(2*pi*4*train_time/d)
  c5 = cos(2*pi*5*train_time/d)
  c6 = cos(2*pi*6*train_time/d)
  s1 = sin(2*pi*train_time/d)
  s2 = sin(2*pi*2*train_time/d)
  s3 = sin(2*pi*3*train_time/d)
  s4 = sin(2*pi*4*train_time/d)
  s5 = sin(2*pi*5*train_time/d)
  s6 = sin(2*pi*6*train_time/d)
  lin.mod = lm(train ~ 1 + c1 + c2 + c3 + c4 + c5 + c6 + s1 + s2 + s3 + s4
+ s5 + s6 + poly(train_time,1,raw=T))

  #Fit ARMA model
  residual = lin.mod$residuals
  ar1 = sarima.for(residual,n.ahead=(points[i+1]-points[i]),p=1,d=0,q=0)
  arma11 = sarima.for(residual,n.ahead=(points[i+1]-
points[i]),p=1,d=0,q=1)

  #Predict
  c1 = cos(2*pi*test_time/d)
  c2 = cos(2*pi*2*test_time/d)
  c3 = cos(2*pi*3*test_time/d)
  c4 = cos(2*pi*4*test_time/d)
  c5 = cos(2*pi*5*test_time/d)
  c6 = cos(2*pi*6*test_time/d)
```

~~~
  s1 = sin(2*pi*test_time/d)
  s2 = sin(2*pi*2*test_time/d)
  s3 = sin(2*pi*3*test_time/d)
  s4 = sin(2*pi*4*test_time/d)
  s5 = sin(2*pi*5*test_time/d)
  s6 = sin(2*pi*6*test_time/d)

  beta = lin.mod$coefficients

  design_matrix = model.matrix( ~ 1 + c1 + c2 + c3 + c4 + c5 + c6 + s1 +
s2 + s3 + s4 + s5 + s6 + poly(test_time,1,raw=T))


  lin_forecast = (design_matrix %*% beta)
  MSE[1] <- MSE[1] + sum((test - lin_forecast - ar1$pred)^2) /
(points[i+1]-points[i])
  MSE[2] <- MSE[2] + sum((test - lin_forecast - arma11$pred)^2) /
(points[i+1]-points[i])
}
print(MSE)
```

#prediction
##forcast the next 10 data points
```{r}
train = Daily_Mean_Temperature
train_time = 1:1576
test_time = 1577:1586

#Fit parametric model
d = 365.25
c1 = cos(2*pi*train_time/d)
c2 = cos(2*pi*2*train_time/d)
c3 = cos(2*pi*3*train_time/d)
c4 = cos(2*pi*4*train_time/d)
c5 = cos(2*pi*5*train_time/d)
c6 = cos(2*pi*6*train_time/d)
s1 = sin(2*pi*train_time/d)
s2 = sin(2*pi*2*train_time/d)
s3 = sin(2*pi*3*train_time/d)
s4 = sin(2*pi*4*train_time/d)
s5 = sin(2*pi*5*train_time/d)
s6 = sin(2*pi*6*train_time/d)
lin.mod = lm(train ~ 1 + c1 + c2 + c3 + c4 + c5 + c6 + s1 + s2 + s3 + s4 +
s5 + s6 + poly(train_time,1,raw=T))

#Fit ARMA(1,1) model
residual = lin.mod$residuals
arma11 = sarima.for(residual,n.ahead=10,p=1,d=0,q=1)

#Predict
c1 = cos(2*pi*test_time/d)
c2 = cos(2*pi*2*test_time/d)
c3 = cos(2*pi*3*test_time/d)
c4 = cos(2*pi*4*test_time/d)
c5 = cos(2*pi*5*test_time/d)
c6 = cos(2*pi*6*test_time/d)
s1 = sin(2*pi*test_time/d)
~~~

~~~
s2 = sin(2*pi*2*test_time/d)
s3 = sin(2*pi*3*test_time/d)
s4 = sin(2*pi*4*test_time/d)
s5 = sin(2*pi*5*test_time/d)
s6 = sin(2*pi*6*test_time/d)

beta = lin.mod$coefficients

design_matrix = model.matrix( ~ 1 + c1 + c2 + c3 + c4 + c5 + c6 + s1 + s2
+ s3 + s4 + s5 + s6 + poly(test_time,1,raw=T))


lin_forecast = (design_matrix %*% beta)
prediction = lin_forecast + arma11$pred
```






## predicted temperature
```{r}
prediction
```
##time series plot of prediction
```{r}
combined = c(Daily_Mean_Temperature, prediction)
df <- data.frame(vals = combined)
plot <- ggplot(df, aes(x = seq_along(vals), y = vals, color =
I(ifelse(seq_along(vals) > 1576, "Blue", "Black")))) +
geom_line(aes(group = 1))
plot <- plot + ggtitle("         Daily Mean Temperature of Delhi, blue
line being predicted temperature")
plot <- plot + labs(y = "Temperature", x = "Year")
plot <- plot + scale_x_continuous(breaks = c(0, 365, 730, 1095, 1460),
labels = c(2013, 2014, 2015, 2016, 2017))
print(plot)
```
```{r}
combined = c(Daily_Mean_Temperature, lin_forecast)
df <- data.frame(vals = combined)
plot <- ggplot(df, aes(x = seq_along(vals), y = vals, color =
I(ifelse(seq_along(vals) > 1576, "Red", "Grey")))) +   geom_line(aes(group
= 1))
plot <- plot + ggtitle("         Daily Mean Temperature of Delhi, red line
being predicted temperature")
plot <- plot + labs(y = "Temperature", x = "Year")
plot <- plot + scale_x_continuous(breaks = c(0, 365, 730, 1095, 1460),
labels = c(2013, 2014, 2015, 2016, 2017))
print(plot)
```
~~~

~~~
```{r}
train = Daily_Mean_Temperature
train_time = 1:1576
test_time = 1577:1586

#Fit parametric model
d = 365.25
c1 = cos(2*pi*train_time/d)
c2 = cos(2*pi*2*train_time/d)
c3 = cos(2*pi*3*train_time/d)
c4 = cos(2*pi*4*train_time/d)
c5 = cos(2*pi*5*train_time/d)
c6 = cos(2*pi*6*train_time/d)
s1 = sin(2*pi*train_time/d)
s2 = sin(2*pi*2*train_time/d)
s3 = sin(2*pi*3*train_time/d)
s4 = sin(2*pi*4*train_time/d)
s5 = sin(2*pi*5*train_time/d)
s6 = sin(2*pi*6*train_time/d)
lin.mod = lm(train ~ 1 + c1 + c2 + c3 + c4 + c5 + c6 + s1 + s2 + s3 + s4 +
s5 + s6 + poly(train_time,1,raw=T))

#Fit ARMA(1,1) model
residual = lin.mod$residuals
arma11 = sarima.for(residual,n.ahead=10,p=1,d=0,q=1)

#Predict
c1 = cos(2*pi*test_time/d)
c2 = cos(2*pi*2*test_time/d)
c3 = cos(2*pi*3*test_time/d)
c4 = cos(2*pi*4*test_time/d)
c5 = cos(2*pi*5*test_time/d)
c6 = cos(2*pi*6*test_time/d)
s1 = sin(2*pi*test_time/d)
s2 = sin(2*pi*2*test_time/d)
s3 = sin(2*pi*3*test_time/d)
s4 = sin(2*pi*4*test_time/d)
s5 = sin(2*pi*5*test_time/d)
s6 = sin(2*pi*6*test_time/d)

beta = lin.mod$coefficients

design_matrix = model.matrix( ~ 1 + c1 + c2 + c3 + c4 + c5 + c6 + s1 + s2
+ s3 + s4 + s5 + s6 + poly(test_time,1,raw=T))


lin_forecast = (design_matrix %*% beta)
prediction = lin_forecast + arma11$pred
```
~~~

~~~
```{r}
#Fit parametric model
d = 365.25
~~~

```
c1 = cos(2*pi*train_time/d)
c2 = cos(2*pi*2*train_time/d)
c3 = cos(2*pi*3*train_time/d)
c4 = cos(2*pi*4*train_time/d)
c5 = cos(2*pi*5*train_time/d)
c6 = cos(2*pi*6*train_time/d)
s1 = sin(2*pi*train_time/d)
s2 = sin(2*pi*2*train_time/d)
s3 = sin(2*pi*3*train_time/d)
s4 = sin(2*pi*4*train_time/d)
s5 = sin(2*pi*5*train_time/d)
s6 = sin(2*pi*6*train_time/d)

t <- 1:length(Daily_Mean_Temperature)
m = model.matrix( ~  c1 + c2 + c3 + c4 + c5 + c6 + s1 + s2 + s3 + s4 + s5
+ s6 + poly(t,1,raw=T))
fit = Arima(Daily_Mean_Temperature, xreg=m,
order=c(1,0,1),include.mean=FALSE)
```

```{r}
test_time <- (length(Daily_Mean_Temperature)+1):
(length(Daily_Mean_Temperature)+10)
idx <- seq(as.Date("2017-04-14"), as.Date("2019-04-14"), by = "day")
test_time <- ts(test_time,frequency=365,start = 2017)
c1 = cos(2*pi*test_time/d)
c2 = cos(2*pi*2*test_time/d)
c3 = cos(2*pi*3*test_time/d)
c4 = cos(2*pi*4*test_time/d)
c5 = cos(2*pi*5*test_time/d)
c6 = cos(2*pi*6*test_time/d)
s1 = sin(2*pi*test_time/d)
s2 = sin(2*pi*2*test_time/d)
s3 = sin(2*pi*3*test_time/d)
s4 = sin(2*pi*4*test_time/d)
s5 = sin(2*pi*5*test_time/d)
s6 = sin(2*pi*6*test_time/d)
design_matrix = model.matrix( ~ 1 + c1 + c2 + c3 + c4 + c5 + c6 + s1 + s2
+ s3 + s4 + s5 + s6 + poly(test_time,1,raw=T))
fc <-
forecast(fit,h=10,xreg=design_matrix,col="Grey",fcol="Red",shadecols="Red")
plot(forecast(fit,h=10,xreg=design_matrix,col="Grey",fcol="Red",shadecols="Red"))

```
```{r}
summary(fc)
```

```{r}
fc <- forecast(fit,h=10,xreg=design_matrix)
fsd <- (fc$upper[,1] - fc$lower[,1]) / (2 * qnorm(.5 + fc$level[1] / 200))
fsd
mean <- fc$mean
combined = c(Daily_Mean_Temperature, lin_forecast)
df <- data.frame(vals = combined)
```

```r
plot <- ggplot(df, aes(x = seq_along(vals), y = vals, color =
I(ifelse(seq_along(vals) > 1576, "Red", "Grey")))) +   geom_line(aes(group
= 1))
plot <- plot + ggtitle("         Daily Mean Temperature of Delhi, red line
being predicted temperature")
plot <- plot + labs(y = "Temperature", x = "Year")
plot <- plot + scale_x_continuous(breaks = c(0, 365, 730, 1095, 1460),
labels = c(2013, 2014, 2015, 2016, 2017))
plot <- plot + theme_bw()
print(plot)
```