Forecasting Weekly PM2.5 Levels in Delhi

2025-06-07

Abstract

This project seeks to forecast weekly PM2.5 levels using time series techniques. We aggregated the data to weekly instead of the original hourly data. After preliminary testing, we applied a Box-Cox transformation and seasonal/trend differencing to address non-stationarity and heteroskedasticity. ARMA models were evaluated using ACF/PACF plots, AICc, and residual diagnostics. While our first model ARMA(2,1) had a slightly better AICc, the second model ARMA(5,1) performed significantly better during diagnostic checking, and was therefore selected for forecasting.

Using the ARMA(5,1) model, we forecasted 14 weeks ahead with confidence intervals. However, reversing the Box-Cox and differencing steps caused the upper bounds to inflate unrealistically. To address this, we adjusted the z-value used in the intervals and constrained extreme values in the inverse Box-Cox transformation. This yielded more reasonable prediction intervals while maintaining forecast interpretability, but also lowered confidence in our predictions.

Introduction

Air pollution poses a risk to public health in India, with Delhi often ranked among the most polluted. PM2.5 is the leading pollutant in Delhi, consisting of particles smaller than 2.5 micrometers in diameter that can enter the lungs and bloodstream, resulting in various health effects including even fatality after long periods of exposure. Major sources of PM2.5 include vehicular emissions, industrial activities, and other common activities.

A recent study published in The Lancet Planetary Health found that long-term exposure to air pollution increases deaths by 1.5 million per year in India, compared to conditions meeting the World Health Organization's reccomendations for safe exposure[1]. This indicates a nationawide health crisis for the Indian citizens that needs to be monitered and mitigated.

In this project, I analyzed weekly PM2.5 measurements from Delhi using data from the Kaggle "Time Series Air Quality Data of India (2010-2023)"[2]: specifically the file DL019.csv, which contains time series data recorded hourly from February 5th, 2018 to March 31, 2023. This dataset provides the concentration of certain pollutants in the air, like PM2.5. After investigating the stationarity of our series, we applied a Box-Cox transformation and both trend and seasonal differencing to address non-stationarity and heteroskedasticity in the series. However, the Box-Cox transformation yielded a negative lambda value close to zero, indicating that the data required more complex treatments to stabilize the variance. I then applied ARMA modeling, selecting between competing models based on AICc values and residual diagnostics. While ARMA(2, 1) had a slightly lower AICc, ARMA(5, 1) better adhered to normality when checking diagnostics and was ultimately chosen for forecasting.

The main objective was to generate a 14-week forecast of PM2.5 levels with confidence intervals, evaluating how well out model's predictions capture seasonal behavior without producing unrealistic CI bounds. One

[1] https://hsph.harvard.edu/environmental-health/news/air-pollution-in-india-linked-to-millions-of-deaths/.

[2] https://www.kaggle.com/datasets/abhisheksjha/time-series-air-quality-data-of-india-2010-2023?resource=download&select=DL019.csv

important result is that the model correctly forecasted a seasonal decline in PM2.5 levels during the late-winter to early-spring period, and right after the spiking occurring around mid-winter. This pattern is consistent with the original data and likely reflects seasonal factors such as holiday activities: pollution often spikes around this time due to holidays like Diwali and New Years that commonly light fireworks, contributing more to air pollution.

This analysis was conducted using R markdown and associated packages. Additionally, I consulted my close friend Arnav Gokhale to better understand Delhi's severe air quality challenges and the broader dynamics of air pollution throughout India. Using aggregated weekly data, this time series analysis will hopefully reveal seasonal or systemic patterns with PM2.5 levels that policy planning can address.

Modeling and Forecasting

Importing and Preparing the Data:

```r
library(xts)
library(lubridate)
library(dplyr)

# importing the dataset
Delhi <- read.csv('/Users/nickcohn24/Desktop/Career - in full/Resume-Projects/Time Series Project/india

# cleaning the dataset
Delhi <- Delhi[, c(1, 3)]
names(Delhi) <- c('Time', 'PM2.5')
Delhi <- na.omit(Delhi)
Delhi <- Delhi[-(1:86), ]

# ensuring timestamp format is standardized
Delhi$Time <- sub(" PDT$", "", Delhi$Time)
Delhi$Time <- as.POSIXct(Delhi$Time, format="%Y-%m-%d")

# aggregating to monthly
weekly_xts <- xts(Delhi$PM2.5, order.by=Delhi$Time)
weekly_delhi <- apply.weekly(weekly_xts, mean, na.rm=TRUE)

# creating dataframe and changing time format
weekly_df <- data.frame(Time=index(weekly_delhi),
                        PM2.5=as.numeric(weekly_delhi))
weekly_df <- weekly_df[1:(nrow(weekly_df) - 14), ]
weekly_df$Time <- as.Date(weekly_df$Time)
weekly_df$Time <- ifelse(weekdays(weekly_df$Time)=="Saturday",
                         weekly_df$Time + 1,
                         weekly_df$Time)
weekly_df$Time <- as.Date(weekly_df$Time, origin = "1970-01-01")
```

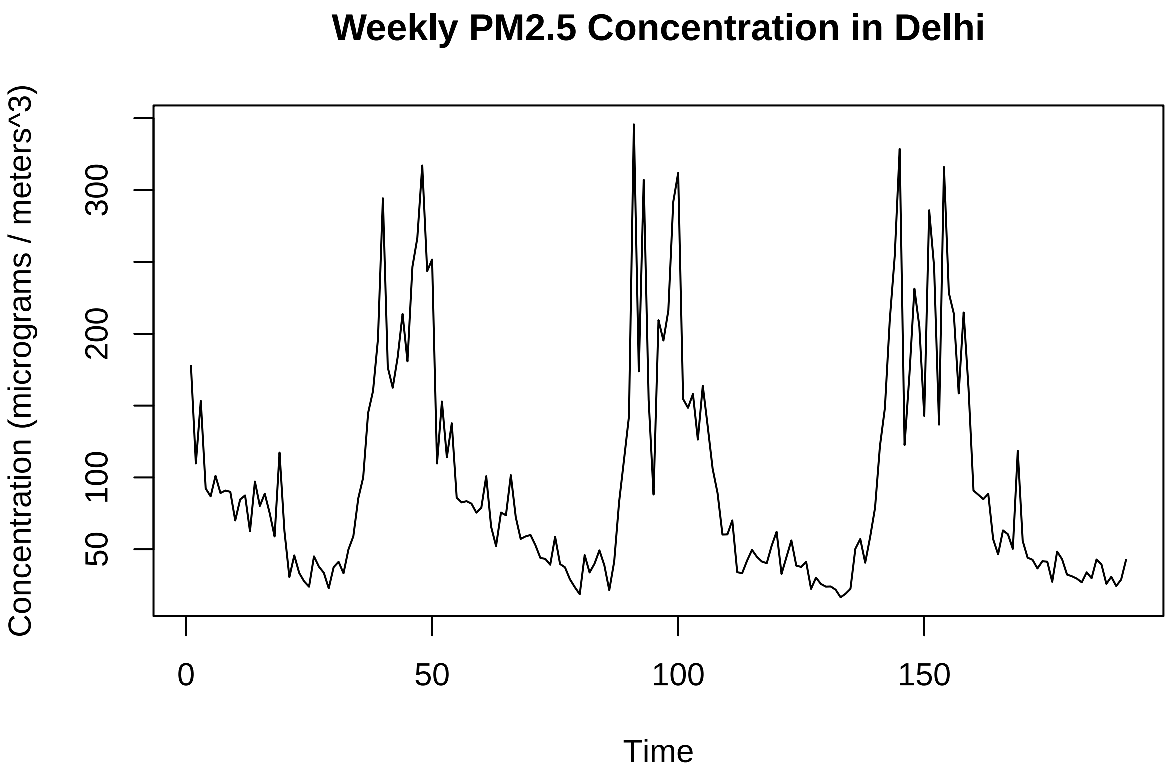
Splitting into Train/Test Sets:

```r
# converting to time series
pollution_ts <- ts(weekly_df$PM2.5, start=c(2018, 6), freq=52)
```

```r
# splitting into sets
train_set <- pollution_ts[1:191]
test_set <- pollution_ts[192:255]
```

Plot and Analyze the Training Set:

```r
ts.plot(train_set, ylab='Concentration (micrograms / meters^3)',
        main='Weekly PM2.5 Concentration in Delhi')
```

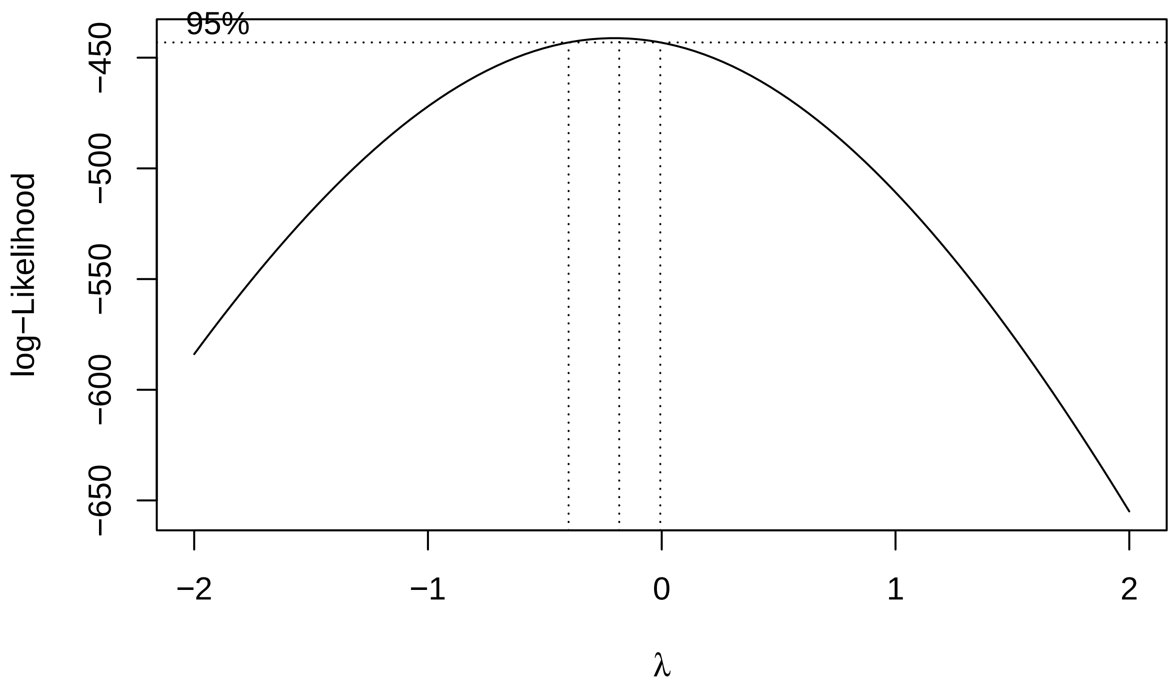


- There was no dominant trend to the plot, but there was a slight positive linear trend to the plot. This suggests we difference at lag 1 to remove trend.
- There was a clear seasonal component, where peaks and drops occur in a repeating M-shaped pattern. This led to 2 peaks per year: 1 in late-fall and 1 in mid-winter. Troughs occur arround summer to early fall. This suggests we difference at lag52
- While there are sharp changes in behavior (spiking/dropping), they adhere to a seasonal pattern, with the only break in structure being how the second spike being higher than the other peaks. However, the variance is very extreme, so a BoxCox transformation seems necessary to stabilize the variance.

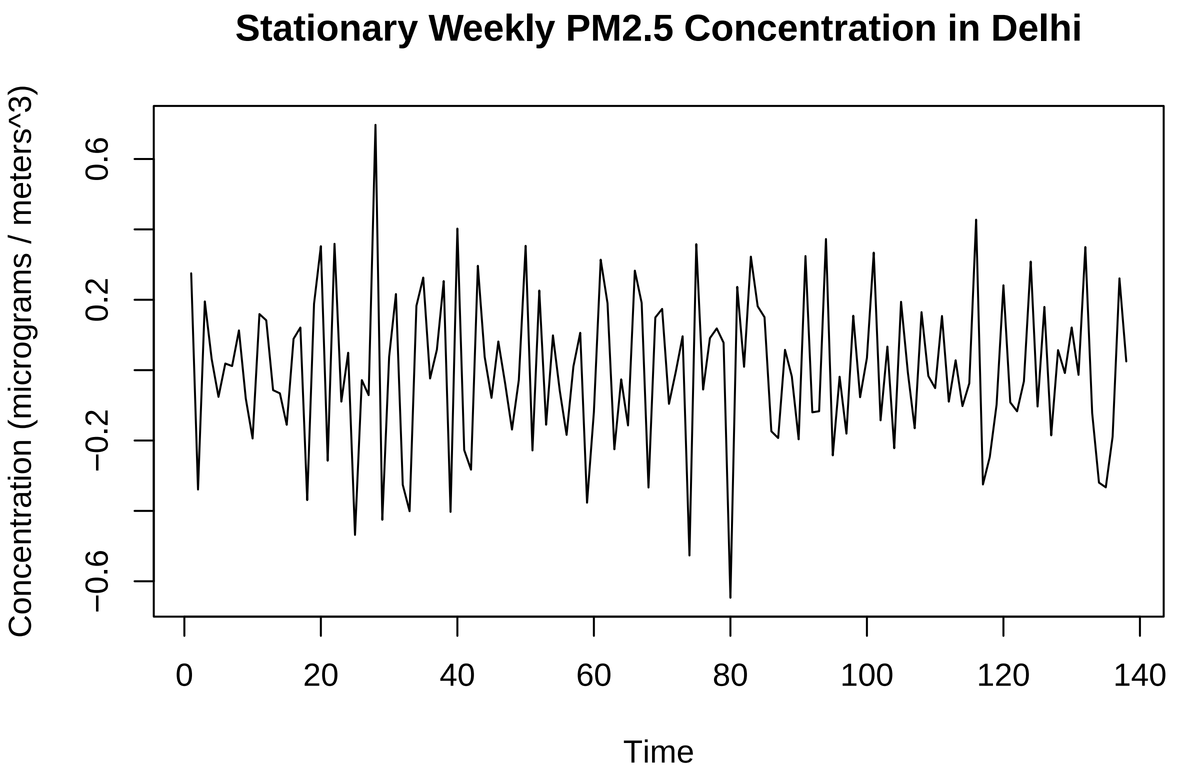
Checking Stationarity + Applying Necessary Transformations:

```r
library(tseries)
library(forecast)
library(MASS)
```

```r
# applying boxcox transformation
bcTransform <- boxcox(as.numeric(train_set)~ as.numeric(1:length(train_set)))
```



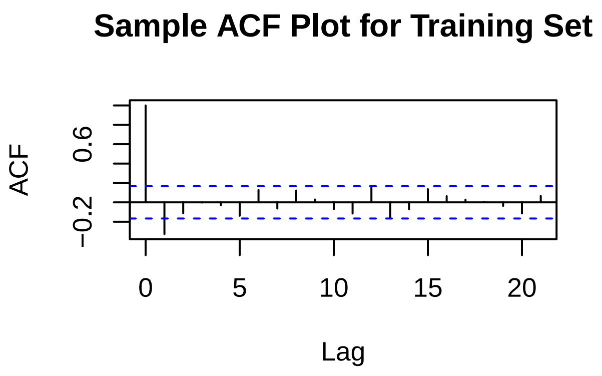
```r
trans_lambda <- bcTransform$x[which(bcTransform$y == max(bcTransform$y))]
train_set <- (train_set^trans_lambda - 1) / trans_lambda

# to undo transformations later
bc_data <- (pollution_ts^trans_lambda - 1) / trans_lambda
bc_data52 <- diff(bc_data, lag=52)
last_1 <- tail(bc_data52, 1)
last_52 <- tail(bc_data, 52)

# differencing at lags 1 and 52 to remove trend and seasonality
train_set <- diff(train_set, 52)
train_set <- diff(train_set, 1)

ts.plot(train_set, ylab='Concentration (micrograms / meters^3)',
        main='Stationary Weekly PM2.5 Concentration in Delhi')
```

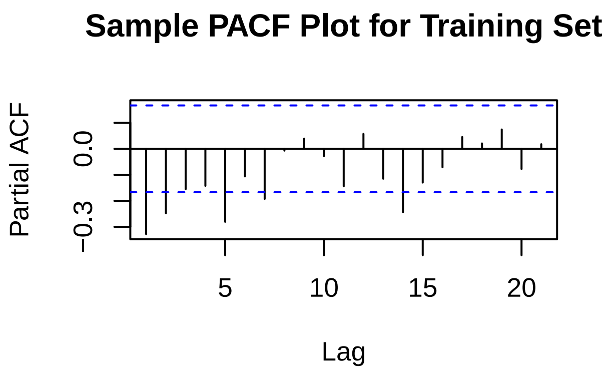


We used a boxcox transformation to stabilize the variance, and differencing to remove seasonality and trend. The time series plot now shows that we have a stationary series, with the values having a constant variance fluctating around the mean.

ACF and PACF plots to identify model(s):

```r
par(mfrow=c(2, 2))
acf(train_set, main='Sample ACF Plot for Training Set')
pacf(train_set, main='Sample PACF Plot for Training Set')
```





There was significance only at lags 0 and 1 for the ACF plot, suggesting an MA(1) component. The significance at lags 1 and 2 for the PACF plot suggests an AR(2) component, but the signfiicance at high lags (5 and 14) warns us from including too many AR terms. Knowing this, we propose the following models:

- Model 1: $ARMA(2,1)$ −> non-seasonal AR(2) + MA(1)
- Model 2: $ARMA(5,1)$ −> non-seasonal AR(5) + MA(1), but is more complex

Fitting the models, Estimating coefficients, Diagnostics:

```
# fitting ARMA models
arma21 <- arima(train_set, order=c(2, 0, 1))
arma51 <- arima(train_set, order=c(5, 0, 1))

# estimating the coefficients
arma21
```

```
##
## Call:
## arima(x = train_set, order = c(2, 0, 1))
##
## Coefficients:
##          ar1     ar2      ma1  intercept
##       0.3548  0.0112  -1.0000     0.0012
## s.e.  0.0854  0.0860   0.0321     0.0006
##
## sigma^2 estimated as 0.03351:  log likelihood = 36.41,  aic = -62.82
```

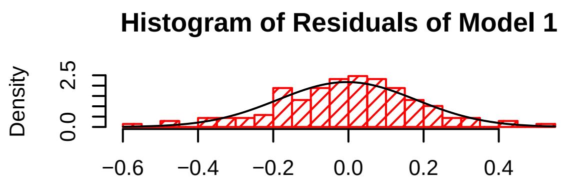
```
arma51
```

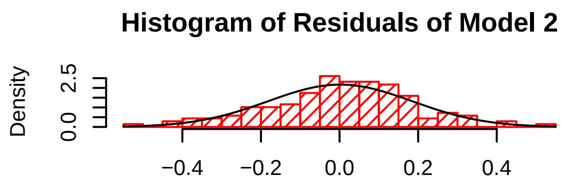
```
##
## Call:
## arima(x = train_set, order = c(5, 0, 1))
##
## Coefficients:
##          ar1      ar2      ar3      ar4      ar5      ma1  intercept
##       0.2467  -0.0479  -0.0435  -0.1024  -0.1322  -0.8828     0.0009
## s.e.  0.1091   0.0949   0.0960   0.0950   0.0995   0.0764     0.0018
##
## sigma^2 estimated as 0.03356:  log likelihood = 37.53,  aic = -59.07
```

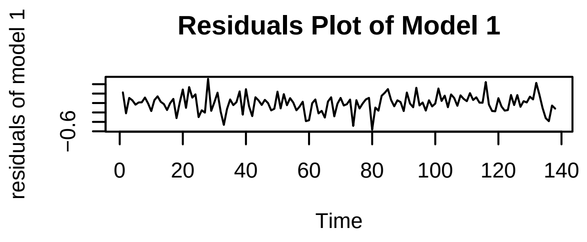
Now we check for normality of residuals:

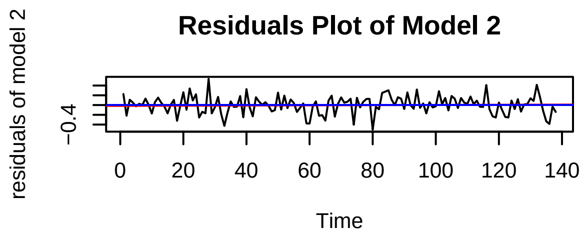
```
res21 <- residuals(arma21)
res51 <- residuals(arma51)

par(mfrow=c(3, 2))

# Histograms of residuals
hist(res21, density=20, breaks=20, col='red', xlab='', prob=TRUE,
     main='Histogram of Residuals of Model 1')
m <- mean(res21)
stdev <- sqrt(var(res21))
curve(dnorm(x, m, stdev), add=TRUE)
hist(res51, density=20, breaks=20, col='red', xlab='', prob=TRUE,
     main='Histogram of Residuals of Model 2')
m <- mean(res51)
stdev <- sqrt(var(res51))
curve(dnorm(x, m, stdev), add=TRUE)
```

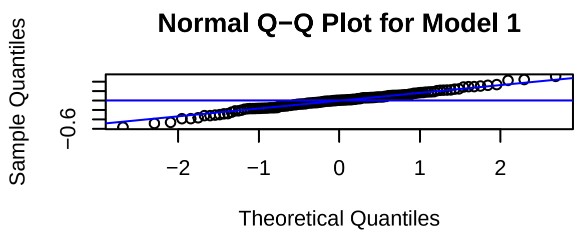
```r
# Residual plots
plot.ts(res21, ylab='residuals of model 1', main='Residuals Plot of Model 1')
plot.ts(res51, ylab='residuals of model 2', main='Residuals Plot of Model 2')

# QQ-plot
qq1 <- lm(res21 ~ as.numeric(1:length(res21)))
abline(qq1, col='red')
abline(h=m, col='blue')
qqnorm(res21, main='Normal Q-Q Plot for Model 1')
qqline(res21, col='blue')
qq2 <- lm(res51 ~ as.numeric(1:length(res51)))
abline(qq2, col='red')
abline(h=m, col='blue')
qqnorm(res51, main='Normal Q-Q Plot for Model 2')
qqline(res51, col='blue')
```

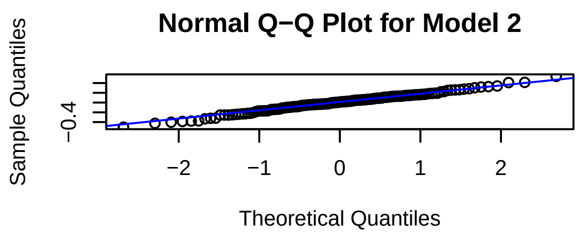












- While both models' histograms are approximately normal, model 1 had more outliers affecting the normality, as seen by the histograms, residual plots, and qq-plots. It's clear that model 2 adhered to normality exceptionally better than model 1.

Portmanteau statistical tests:

```r
# Shapiro-Wilk Test
shapiro.test(res21)
```

```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  res21
## W = 0.98933, p-value = 0.3715
```

```
shapiro.test(res51)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res51
## W = 0.99363, p-value = 0.7983
```

```
# since n=138, sqrt(n) is approximately 12, so lag=12
length(res21)
```

```
## [1] 138
```

```
# Box-Ljung Test
Box.test(res21, lag=12, type=c('Ljung-Box'), fitdf=3)
```

```
##
##  Box-Ljung test
##
## data:  res21
## X-squared = 10.695, df = 9, p-value = 0.2972
```

```
Box.test(res51, lag=12, type=c('Ljung-Box'), fitdf=6)
```

```
##
##  Box-Ljung test
##
## data:  res51
## X-squared = 6.5103, df = 6, p-value = 0.3685
```

- The Shapiro-Wilk tests gave us the p-values greater than 0.05, so we fail to reject the null hypothesis. This supports both models' residuals not violating normality. In addition, the Box-Ljung test for both models produced p-values greater than 0.05, so we fail to reject the null hypothesis. This supports both models having independence in their residuals.

Comparing AICc:

```
# AICc = AIC + (2*k * (k + 1)) / (n - k - 1)
n21 <- length(arma21$residuals)
k21 <- length(arma21$coef) + 1
aic21 <- AIC(arma21)
aicc21 <- aic21 + (2 * k21 * (k21 + 1)) / (n21 - k21 - 1)
aicc21
```

```
## [1] -62.36868
```

```r
n51 <- length(arma51$residuals)
k51 <- length(arma51$coef) + 1
aic51 <- AIC(arma51)
aicc51 <- aic51 + (2 * k51 * (k51 + 1)) / (n51 - k51 - 1)
aicc51
```

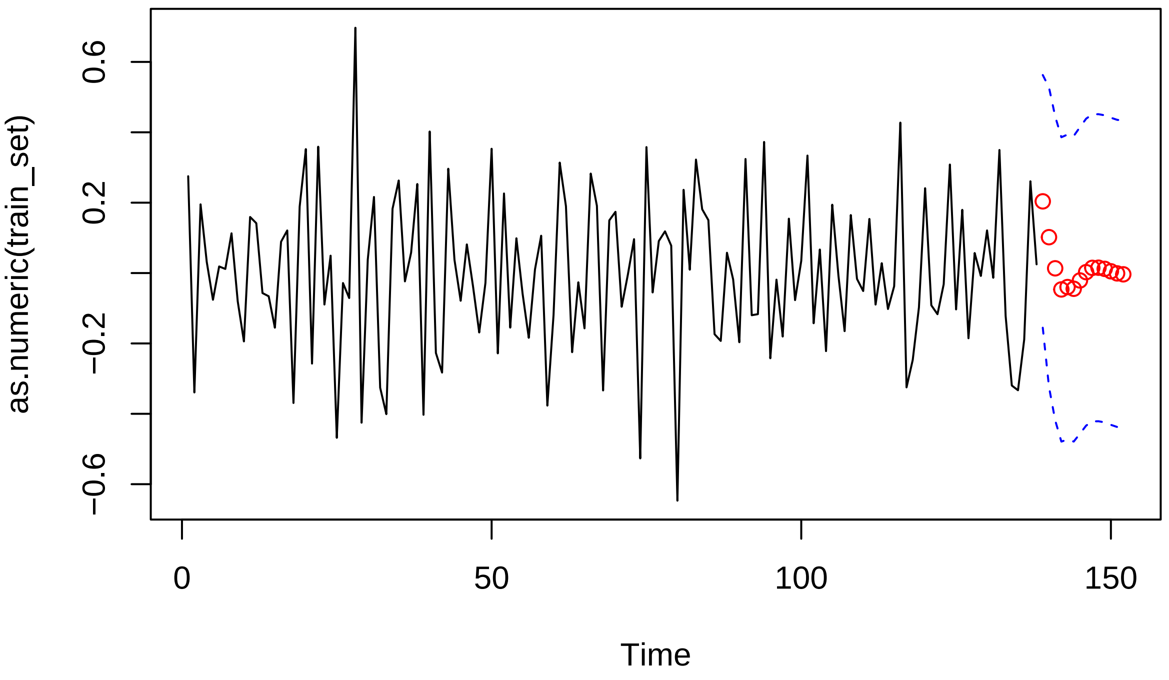
```
## [1] -57.95268
```

Model 1 has a lower AICc (-62.38) than model 2 (-57.95), suggesting model 1 is better due to AICc. While this is true, model 1 performed significantly better during diagnostics and this is AICc difference is small, so we should ultimately choose model 2 to represent the series. This gives a fitted model (ARMA(5, 1)) with algebraic form:

- $(1 - 0.2467B + 0.0479B^2 + 0.0435B^3 + 0.1024B^4 + 0.1322B^5)X_t = (1 - 0.8828B)Z_t + 0.00085$

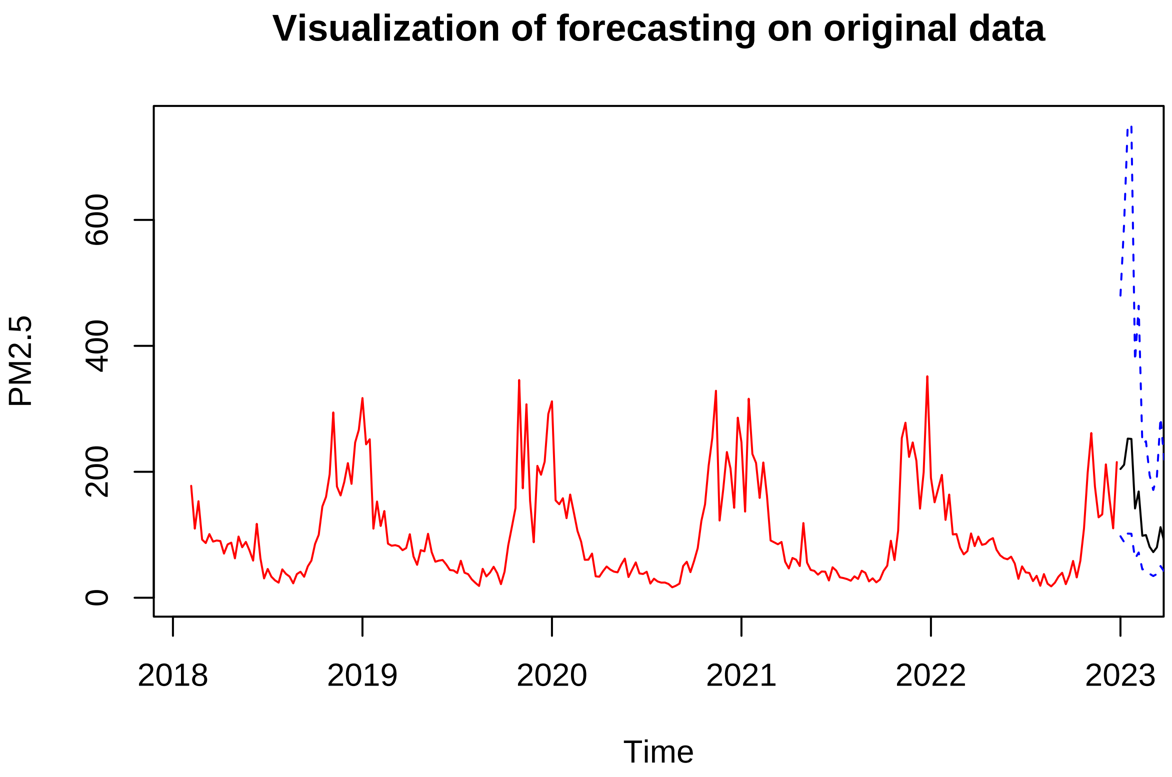
Forecasting and CI's:

```r
# forecasting on transformed data, the next 2 weeks
prediction <- predict(arma51, n.ahead=14)
U <- prediction$pred + 1.96*prediction$se
L <- prediction$pred - 1.96*prediction$se
ts.plot(as.numeric(train_set), xlim=c(1, length(train_set)+14),
        ylim=c(min(c(train_set, L)), max(c(train_set, U))))
lines(U, col='blue', lty='dashed')
lines(L, col='blue', lty='dashed')
points((length(train_set)+1):(length(train_set)+14), prediction$pred, col='red')
```

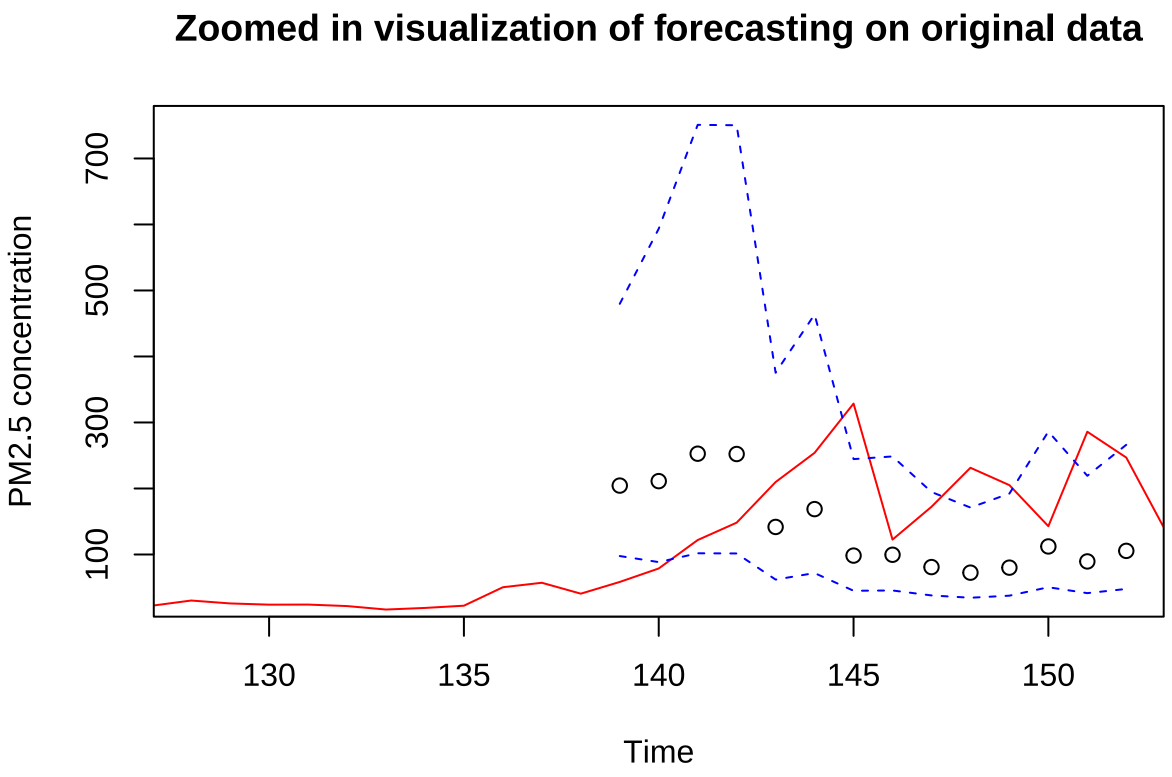


We can't reverse the box-cox transformation directly, since we differenced twice before the transformation. Due to out lambda=-0.18, being negative and close to 0, we need to find another way of reversing the transformation without inflating the CI bounds or the forecasted predictions themselves.

```r
prediction <- predict(arma51, n.ahead=14)
pred_vals <- prediction$pred
pred_se <- prediction$se

# undoing differencing
last_1_scalar <- as.numeric(last_1)
h <- length(pred_vals)
pred_cumsum_1 <- cumsum(pred_vals) + last_1_scalar
pred_cumsum_seasonal <- pred_cumsum_1 + last_52[1:h]

# 90% Confidence intervals
adj_pred_se <- pred_se
U_z <- pred_cumsum_seasonal + 1.64 * adj_pred_se
L_z <- pred_cumsum_seasonal - 1.64 * adj_pred_se

# defining an inverse box-cox function
inv_boxcox <- function(x, lambda) {
  if (lambda == 0) {
    return(exp(x))
  } else {
    arg <- lambda * x + 1
    arg[arg <= 0] <- 1e-6  # prevent invalid power roots
    return(arg^(1 / lambda))
  }
}

# applying the inverse box-cox to undo the transformation on the CI bounds and predictions
pred_origin <- inv_boxcox(pred_cumsum_seasonal, trans_lambda)
U_origin <- inv_boxcox(U_z, trans_lambda)
L_origin <- inv_boxcox(L_z, trans_lambda)
# since the upper/lower bound keep getting inflated, we put a cap on the values
max_upper <- max(pollution_ts, na.rm=TRUE) * 2.5
U_origin[U_origin > max_upper | is.infinite(U_origin)] <- max_upper
L_origin[L_origin < 0 | is.infinite(L_origin)] <- 0
last_train_time <- time(pollution_ts)[length(pollution_ts)]
freq <- frequency(pollution_ts)
start_time <- last_train_time + 1 / freq
pred_origin_ts <- ts(pred_origin, start=start_time, frequency=freq)
U_origin_ts <- ts(U_origin, start=start_time, frequency=freq)
L_origin_ts <- ts(L_origin, start=start_time, frequency=freq)

# plotting the forecast on original data
plot(pollution_ts, col='red', ylab='PM2.5',
     xlim = c(time(pollution_ts)[1], time(pollution_ts)[length(pollution_ts)] + 0.05),
     ylim = c(0, max(U_origin_ts, na.rm = TRUE)),
     main = 'Visualization of forecasting on original data')
lines(U_origin_ts, col='blue', lty='dashed')
lines(L_origin_ts, col='blue', lty='dashed')
lines(pred_origin_ts, col = 'black')
```



```r
# zoomed in plot
time_points <- time(pollution_ts)
ts.plot(as.numeric(pollution_ts),
        xlim = c(length(train_set) - 10, length(train_set) + 14),
        ylim = c(min(L_origin, na.rm=TRUE), max(U_origin, na.rm=TRUE)),
        col = "red",
        ylab = "PM2.5 concentration",
        main = "Zoomed in visualization of forecasting on original data")
lines(U_origin, col="blue", lty="dashed")
lines(L_origin, col="blue", lty="dashed")
points((length(train_set) + 1):(length(train_set) + length(pred_origin)), pred_origin, col = "black")
```



We had to lower the confidence to only 90% (1.64 multiplier on se) since our CI bounds were sensitive to exploding after undoing the transformations, since lambda was negative and close to 0 (-0.18).

Conclusion

The primary objective of this project was to model and forecast PM2.5 levels in Delhi using time series methods that accoutned for both trend and seasonal variation. Through Box-Cox transformation, differecing, and ARMA modeling, I produced a 14-week forecast of weekly PM2.5 concentrations. The selected ARMA(5, 1) model was chosen based on its superior residual diagnostics, despite the ARMA(2, 1) model holding a slightly lower AICc.

The model successfully captured important seasonal behavior, including a predicted decline in PM2.5 levels during the late-winter to early-spring transition: a pattern visible in historical data. This seasonal rise and followed mid-winter pollution spikes, likely tied to major holidays such as Diwali and New Year's, during which increased emissions from fireworks and other festival activites raising the concentration of pollutants in the air.

This project demonstrates the value of time series forecasting in understanding recurring pollution trends and informing policy regarding AQI. While the model performed well on diagnostics and produced reasonable forecasts, we ran into issues regarding the confidence interval. We had to reduce the confidence level to 90% (z=1.64) to make the bounds more reasonable, reducing the usefulness of our forecasts. Although, despite the inflated upper bound, the forecast still successfully captured key seasonal patterns, indicating that the model performed well in identifying underlying trends.

As such, our final ARMA(5, 1) model can be written as:

- $(1 - 0.2467B + 0.0479B^2 + 0.0435B^3 + 0.1024B^4 + 0.1322B^5)X_t = (1 - 0.8828B)Z_t + 0.00085$

I would like to thank my close friend Arnav Gokhale for discussing the broader implications of air pollution in Delhi and offering insight into saesonal behaviors and local practices.

References

1. Balakrishnan, K., et al. (2024). Long-term exposure to air pollution and mortality in India: A nation-wide cohort study. The Lancet Planetary Health, 8(4), e215–e226. https://doi.org/10.1016/S2542-5196(24)00248-1

2. Jha, A. (2023). Time series air quality data of India (2010-2023). Kaggle. https://www.kaggle.com/datasets/abhisheksjha/time-series-air-quality-data-of-india-2010-2023

/Users/nickcohn24/Desktop/UCSB_Courses/PSTAT174-TImeSeries/Final Project/india_AQI/DL016.csv