Forecasting Metal Oxide Chemicals in an Italian City

Nava Roohi, Juliet Sieland-Harris, Andrew Zazueta

University of San Diego

Master of Science, Applied Data Science

ADS506 Applied Time Series Analysis

Section 2

2021

Author Note

We have no known conflict of interest to disclose.

Correspondence concerning this article should be addressed to Nava Roohi, Juliet Sieland-Harris, or Andrew Zazueta, at University of San Diego, 5998 Alcala Park, San Diego, CA 92110.

Emails: nroohi@sandiego.edu, jsielandharris@sandiego.edu, azazueta@sandiego.edu

Project Research

Background

Air pollution has been increasing tremendously since the Industrial Revolution. Currently, it is one of the major concerns impacting the lives of humans, animals, and plants. There are many short-term and long-term consequences of air pollution, unfortunately all of them harmful, which can cause permanent damages in the ecosystem if it is not controlled and effectively prevented. A huge number of hazardous compounds have been identified in the environment. However, the most common pollutants are certain concentrations of substances such as particulate matter, radioactive materials and most importantly gases such as carbon monoxide (CO), nitrogen oxides (NOx and NO2), and benzene (C6H6)(Naveen and Anu, 2017). These turn out to be the most harmful pollutants threatening the living ecosystem.

Due to the significantly increased issues surrounding air pollution, developed or developing nations have started to increase public awareness about the matter. Officials in these countries are regularly monitoring and predicting the particulars about different air pollutants. Predicting or forecasting of the concentration of these air pollutants species used for air pollution assessment and management is mainly done using time series approaches. The predicting methods analyze the sequence of historical data in a period of time in order to build an accurate forecasting model.

Problem and Motivation

The purpose of this project is to analyze, compare, and forecast air quality measurements using the time series approach in an extremely polluted Italian city. For air quality forecasting, this study looks at the daily average of four metal oxide chemicals: CO, C6H6, NOx, and NO2. The model of Seasonal Auto Regressive Integrated Moving Average (SARIMA) method was implemented in order to forecast their concentration over a month period. Accurate air quality prediction could be benefecial in terms of having important theoretical and practical values for the public; it could facilate prevention of the health damage caused by air pollution or improve the emergency response capability of heavy pollution days.

Description of Source and Raw Data Set

In an Italian city, air quality was recorded using a chemical multisensor device at road level. The device collected the hourly concentrations of an array of 5 metal oxide chemicals, while a co-located reference certified analyzer recorded hourly average concentrations of carbon monoxide, Non Metanic HydroCarbons, Benzene, Nitrogen Oxide, and Nitrogen Dioxide. Additionally, the device recorded the temperature and humidity of the time of data gathering. In total, there are 15 variables in the dataset. The dataset for this project was provided by the National Agency for New Technologies, Energy and Sustainable Economic Development (ENEA) and contains 9,358 instances between March 2004 and February 2005. The ENEA is an Italian "public body aimed at research, technological innovation and the provision of advanced services to enterprises, public administration and citizens in the sectors of energy, the environment and sustainable economic development." (ENEA, 2015) At the time of the completed one year of data recording, this dataset represented the longest freely available recording of this type**.**

Literature Review

As pollution continues to threaten the environment and the health of the population, it has become increasingly important to monitor and track air quality. For this reason, extensive literature studies have examined air quality and pollution, targeting various harmful pollutants, to better forecast and prevent air pollution. Nitrogen Dioxide, and additional Nitrogen Oxides, are one of the target pollutants in most studies, and is included in the data for this project. In Wroclaw, Poland, Espinosa et al. (2021) created a 24-hours ahead forecast model for nitrogen oxides. Nitrogen Dioxide in the air can cause respiratory diseases and primarily pollute the air from emissions from cars, trucks, and power plants. (EPA, 2021) Traffic data, used by Espinosa et al., can prove to be a valuable variable in predicting accurate results for this pollutant. For future iterations of this study, a similar methodology may be used to optimize results.

An early method of interest for this study, Autoregressive Integrated Moving Average (ARIMA) or Seasonal Autoregressive Integrated Moving Average (SARIMA) was identified. A similar study by Naveen and Anu (2017) trained both ARIMA and SARIMA methods to forecast the Air Quality Index, and key pollutants. The study in Kerala, India found the ARIMA method to successfully predict future air quality, indicating this methodology can be effective for air quality time series data. Nimesh et al. (2014) collected data for four pollutants, Suspended Particulate Matter, Respirable Suspended Particulate Matter, Nitrogen Dioxide, and Sulfur Dioxide in the city of Chandigarh, India. A similar method, Autoregressive Fractionally Integrated Moving Average (ARFIMA) was determined as the most appropriate model, further solidifying relevance for ARIMA, SARIMA, and ARFIMA effectiveness for this study.

Future iterations of this study may be able to use newer applications to optimize results. Freeman et al. (2018) predominantly monitored Tropospheric Ozone in Kuwait to apply to a deep learning model. This technique was one of the first applications of this model on air pollution time series, and proved to be more accurate than previously discussed models.

Project Overview

Data Cleaning

The first step was to read in the data set, which came in an excel worksheet, and to load in the necessary libraries needed for the project. After looking at the data set, it was noted that both the date and time were in separate columns. To reduce the number of features in the data set, it was decided that it was best to combine these columns. When looking at the time column, R automatically added a date to the feature when the data set was read in, so the first step was to remove this date and only take the first hour. The hour was then read into a new feature, and the original time feature was removed. The last step needed for this process was merging the correct date feature with the new time feature and using the library 'Lubridate' to make this column a Date Time data type with correct formatting.

This led to one of the most important steps of data cleaning, which was looking for missing values. The authors of the data set indicated that any missing values were replaced with -200, but it was also noticed that there were rows with NA's. Every column was missing the same percentage of values, and they were all in order, so these rows were removed. An additional two columns were removed which contained 100% missing values.

The next step was to check for outliers, and this was done by examining the box plot of each feature. Because the authors imputed -200 for legitimate missing values, the bounds for each plot had to be modified so that all the plots would not be negatively skewed. No outliers were found that could be attributed to miss inputs; the outliers all appeared to come about naturally, and no further action was needed. However, it was found that the feature measuring Non Metanic Hydrocarbons had over 90% of its values as -200, so this feature was also removed.

The other -200 values from each feature were dealt with by imputing an NA for every value less than -10 (all measurements other than temperature could not possibly take negative values, and -10 was decided due to some negative temperatures which appeared in the data).

Each feature had a different percentage of missing values ranging from about 4% to about 18%, so this was handled by using a function which imputed a moving average of the data into the missing values. After this step, the cleaned data was read into a new CSV file so that the preprocessing steps would not have to be ran every time R was started.

Exploratory Data Analysis (EDA)

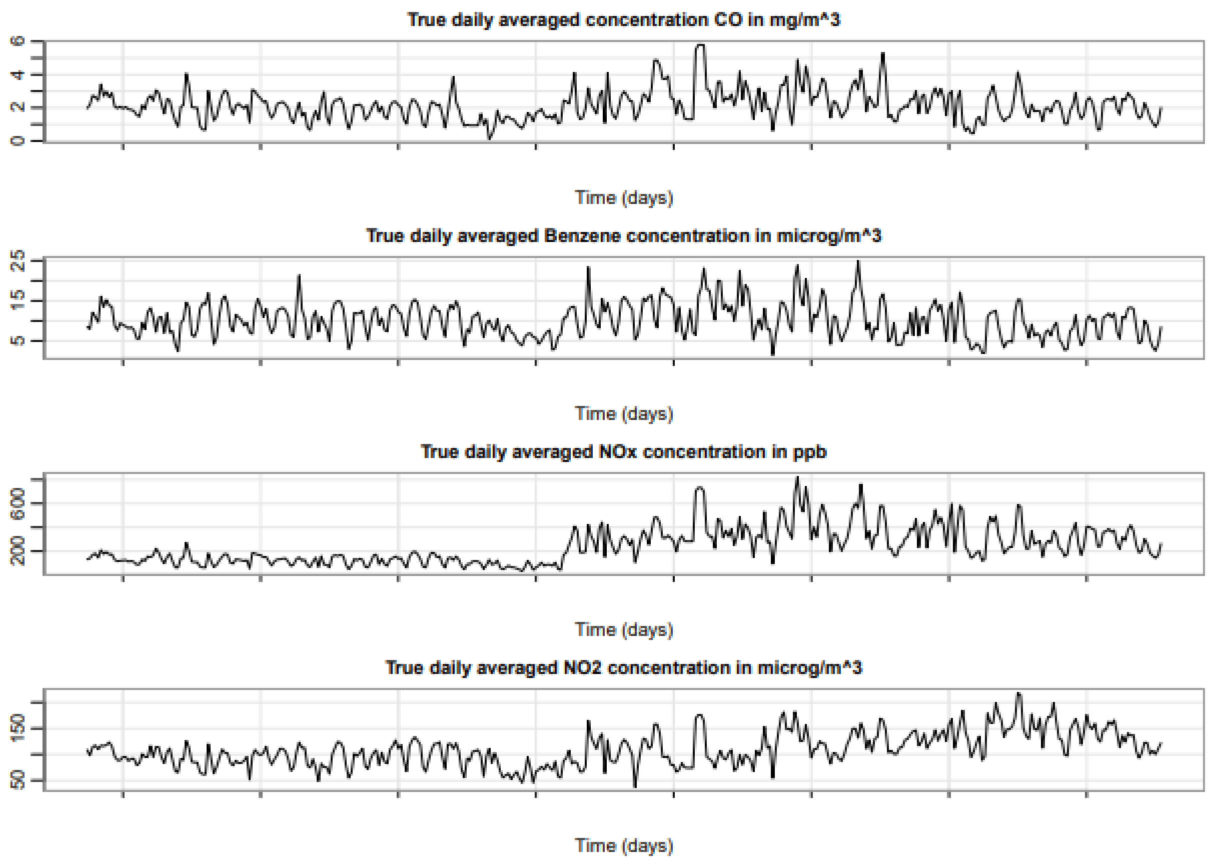
The first step of EDA was to examine the time series of the four metal oxide chemicals. Since the data was taken hourly, the graphs appeared noisy and hard to read. It was decided that the best fix to this issue was to use the daily average over the hourly average for the seasonal autoregressive moving average (SARMA) models, and these time series were plotted as seen in Figure 1.

The next step for EDA was to make our data stationary, which was done by differencing the data. This gave the features a constant mean of 0, with a relatively constant variance, which is required of a stationary time series.

The last EDA step was to examine the autocorrelation function (ACF) and partial autocorrelation function (PACF) for each feature to help determine what parameters the SARMA models would take. It was found that each P/ACF graph benefited from adding a 7-day seasonal component, which indicates that there is a weekly cycle to the data. Each plot indicated an autoregressive model due to the PACF cutting off after lag 1 and the ACF tailing off. More information about each plot can be found in the appendix code under the "Looking at P/ACF for Different Features" section.

Figure 1

Daily Averaged Time Series for each Metal Oxide Chemical



Note. The time scale is over 391 days, starting March 10, 2004, and ending April 5, 2005.

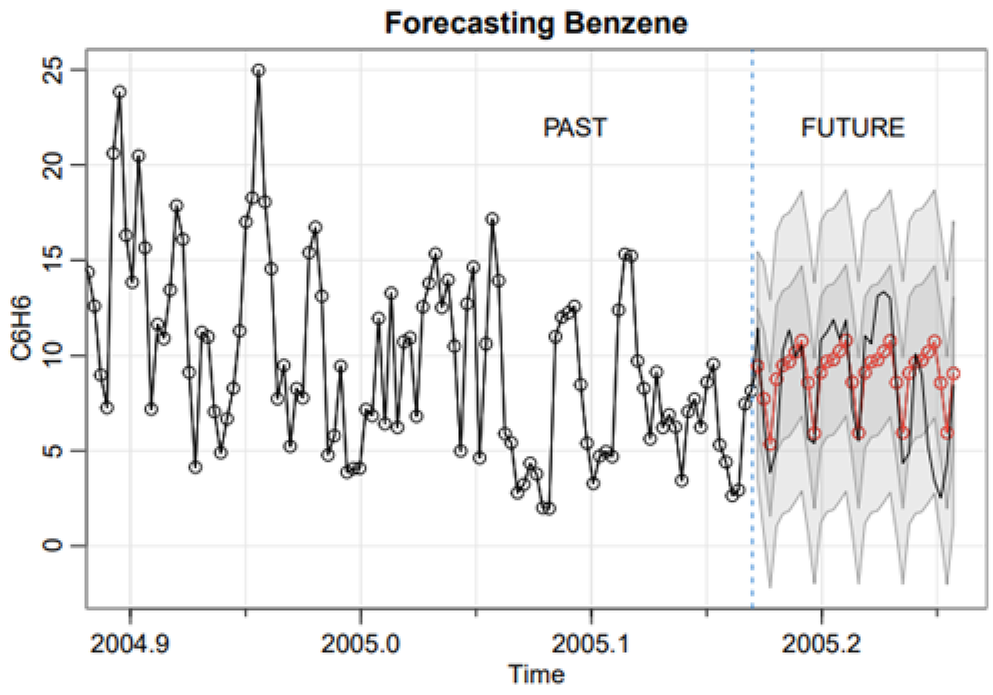
Modeling and Forecasting

Now that a starting point of which parameters to use for the SARMA models were found, the next step was to build a model for each of the four chemicals to examine various statistics using the function “sarima” from the “astsa” library. For carbon monoxide (NO), the parameters that yielded the best results were p = 1, d = 1, q = 1, P = 1, D = 0, Q = 1, and S = 7 (for the weekly cyclic pattern). “p” and “P” are the orders of the autoregressive portion of our model, with “P” being the seasonal autoregressive component. “d” and “D” are our differencing parameters, which is why “d” was equal to one (the data had to be differenced once to be stationary). “q” and “Q” are the orders of the moving average portion of our model, with “Q” being the seasonal moving average component. Initially, our P/ACF graphs indicated a q = Q = 0 model, but these values were changed to one due to it improving the Ljung−Box statistic p-values. It is important that these values are all over 0.05 because then it is not necessary to reject the null hypothesis that the noise is white.

Benzene (C6H6) was found to have the same parameters as CO for the same reasons listed above. Different parameters came from Total Nitrogen Oxides (NOx) and Nitrogen Dioxide (NO2), which both had p = 1, d = 1, q = 1, P = 2, D = 0, Q = 1, and S = 7. Initially, P = 1 was tried, but this gave back errors due to P = Q = 1, so another seasonal autoregressive component was needed. Once these models were made, the next step was to forecast.

Figure 2

Forecasting Benzene over a 32-day Period



Note. Benzene is measured in microg/m^3. This graph only shows the last 100 points of the entire time series. The red lines are the predictions with the circles indicating the prediction values. The gray areas are error bounds.

The forecasting was done by building our model up to the final month of observations. Then, our model forecasted the following 32 days. This forecast was compared to the original observations by superimposing the forecast graph on top of the original data, which can be seen in Figure 2. Each forecast lined up with the original observations well, with the error bounds of each almost lining up perfectly with the actual data. The only time the forecast did not follow well with the data was with NO2, which took a sudden and unpredictable dip in the last month.

Model Evaluation

The last step of the project was to see how each of the models performed using AIC, AICc, and BIC. The model with the best values for these statistics was for CO, with an AIC and AICc equal to 2.13 and a BIC equal to 2.19. The worst performing model was for NOx, which had an AIC and AICc of 11.77 and a BIC of 11.84. During the modeling stage, these values were minimized for each feature by using different hyperparameters in the "sarima" function, as discussed earlier.

Discussion and Conclusion

In conclusion, the models did well at forecasting the different metal oxides found in the atmosphere in an Italian city. Other insights that were not performed for this project but would be interesting to see are: 1) determining what days of the week correlate to each day of the graph and plot them (this would show what days of the week produced the most chemical compounds), 2) how the features are correlated with each other, 3) building a linear regression model from these features on top of building a SARMA model to predict the temperature feature in the data set, and 4) to use our models to forecast the data which has not been seen before. All of these could have been done within the scope of the Applied Time Series class and would be informative side projects to perform in the future.

Other tasks that are outside the scope of this course but would also be interesting to see are: 1) how the data can be used to make improvements to the air quality in the Italian city and 2) measure how much these changes alter the modeling of the data. In the end, this type of research is used by climate scientists to figure out which areas are the biggest polluters. This research also gives insights on how the air quality will change over time. Both are extremely important to figure out to reduce our ecological footprint to improve our atmosphere before humans are no longer able to change it.

References

National Agency for New Technologies, Energy and Sustainable Economic Development. (2015, December 28). *The institutional mission (Law no. 221/2015).* https://www.enea.it/en/enea/about-us

Espinosa, R., Palma, J., Jiménez, F., Kamińska, J., Sciavicco, G., & Lucena-Sánchez, E. (2021, September 7). A time series forecasting based multi-criteria methodology for air quality prediction. *Elsevier B.V*. https://doi.org/10.1016/j.asoc.2021.107850

United States Environmental Protection Agency. (2021, June 7). *Nitrogen Dioxide (NO2) Pollution*. https://www.epa.gov/no2-pollution/basic-information-about-no2

Naveen, V. & Anu, N. (2017, June). Time Series Analysis to Forecast Air Quality Indices in Thiruvananthapuram District, Kerala, India. *Naveen V. Int. Journal of Engineering Research and Application*, *7*(6), 66-84. https://doi.org/10.9790/9622-0706036684

Nimesh, R., Arora, S., Mahajan, K. K., & Gill, A. N. (2014) Predicting air quality using ARIMA, ARFIMA and HW smoothing. *Model Assisted Statistics and Applications 9,* 137–149. https://doi.org/10.3233/MAS-130285

Freeman, B. S., Taylor, G., Gharabaghi, B., & Thé, J. (2018) Forecasting air quality time series using deep learning. *Journal of the Air & Waste Management Association, 68*(8), 866-886. https://doi.org/10.1080/10962247.2018.1459956

Appendix

Andrew Zazueta Nava Roohi Juliet Sieland-Harris

12/3/2021

Setting Work Directory, Loading Libraries, and Loading Data

```
setwd("C:/Users/mzazu/OneDrive/Documents/USD papers/506/AirQualityUCI")
library('astsa')
library('xlsx')
library('tidyverse')
library('lubridate')
library('imputeTS')
library('Rcpp')
aq <- read.xlsx('AirQualityUCI.xlsx', sheetIndex = 1, header = TRUE)
```

Data Cleaning

Editing the Date and Time Columns

```
# The time column is reading in random dates, so we are going to fix this issue first
head(aq[2], 10)
```

```
##                   Time
## 1  1899-12-30 18:00:00
## 2  1899-12-30 19:00:00
## 3  1899-12-30 20:00:00
## 4  1899-12-30 21:00:00
## 5  1899-12-30 22:00:00
## 6  1899-12-30 23:00:00
## 7  1899-12-30 00:00:00
## 8  1899-12-30 01:00:00
## 9  1899-12-30 02:00:00
## 10 1899-12-30 03:00:00
```

```
## Splitting Time and retrieving hour
hour <- aq %>%
  separate(Time , c("1", "2", "3", "Hour"), extra='drop') %>%
  select("Hour")

head(hour, 20)
```

```
##    Hour
## 1    18
## 2    19
## 3    20
## 4    21
## 5    22
## 6    23
## 7    00
## 8    01
## 9    02
## 10   03
## 11   04
## 12   05
## 13   06
## 14   07
## 15   08
## 16   09
## 17   10
## 18   11
## 19   12
## 20   13
```

```
## Adding Hour to data frame and getting rid of Time
aq$Hour <- as.numeric(unlist(hour))
aq <- aq[-2]

## Merging Date and Hour and getting rid of Date and Hour columns
aq$DateTime <- paste(aq$Date, aq$Hour)
aq <- aq[-c(1,17)]

## Making DateTime into Date and Time data type
aq$DateTime <- as.POSIXct(aq$DateTime,format="%Y-%m-%d %H", tz= "CET")
```

Checking for missing data

```
# Checking percentage of missing values in each column
cols <- colnames(aq)
for(i in 1:length(aq)){
  missing <- round(sum(is.na(aq[i]))/dim(aq)[1], 5) * 100
  print(c(cols[i], missing))
}
```

```
## [1] "CO.GT." "1.204"
## [1] "PT08.S1.CO." "1.204"
## [1] "NMHC.GT." "1.204"
## [1] "C6H6.GT." "1.204"
## [1] "PT08.S2.NMHC." "1.204"
## [1] "NOx.GT." "1.204"
## [1] "PT08.S3.NOx." "1.204"
## [1] "NO2.GT." "1.204"
## [1] "PT08.S4.NO2." "1.204"
## [1] "PT08.S5.O3." "1.204"
## [1] "T"      "1.204"
## [1] "RH"     "1.204"
## [1] "AH"     "1.204"
## [1] "NA." "100"
## [1] "NA..1" "100"
## [1] "DateTime" "1.225"
```

```
# Removing NA. and NA..1
aq <- aq[-c(14, 15)]

# Finding out where the missing values are
which(is.na(aq[1]))
```

```
##   [1] 9358 9359 9360 9361 9362 9363 9364 9365 9366 9367 9368 9369 9370 9371 9372
##  [16] 9373 9374 9375 9376 9377 9378 9379 9380 9381 9382 9383 9384 9385 9386 9387
##  [31] 9388 9389 9390 9391 9392 9393 9394 9395 9396 9397 9398 9399 9400 9401 9402
##  [46] 9403 9404 9405 9406 9407 9408 9409 9410 9411 9412 9413 9414 9415 9416 9417
##  [61] 9418 9419 9420 9421 9422 9423 9424 9425 9426 9427 9428 9429 9430 9431 9432
##  [76] 9433 9434 9435 9436 9437 9438 9439 9440 9441 9442 9443 9444 9445 9446 9447
##  [91] 9448 9449 9450 9451 9452 9453 9454 9455 9456 9457 9458 9459 9460 9461 9462
## [106] 9463 9464 9465 9466 9467 9468 9469 9470 9471
```

```
# Since each column is missing the same percentage of values, and aq[1]'s missing
# values are all in an order, lets see what happens to the missing values when we
# remove these rows.
aqNew <- aq[complete.cases(aq[1]), ]
cols <- colnames(aqNew)
for(i in 1:length(aqNew)){
  missing <- round(sum(is.na(aqNew[i]))/dim(aqNew)[1], 5) * 100
  print(c(cols[i], missing))
}
```

```
## [1] "CO.GT." "0"
## [1] "PT08.S1.CO." "0"
## [1] "NMHC.GT." "0"
## [1] "C6H6.GT." "0"
## [1] "PT08.S2.NMHC." "0"
## [1] "NOx.GT." "0"
## [1] "PT08.S3.NOx." "0"
## [1] "NO2.GT." "0"
## [1] "PT08.S4.NO2." "0"
## [1] "PT08.S5.O3." "0"
## [1] "T" "0"
## [1] "RH" "0"
## [1] "AH" "0"
## [1] "DateTime" "0.021"
```

```
# Now we are only missing values in DateTime
which(is.na(aqNew$DateTime))
```

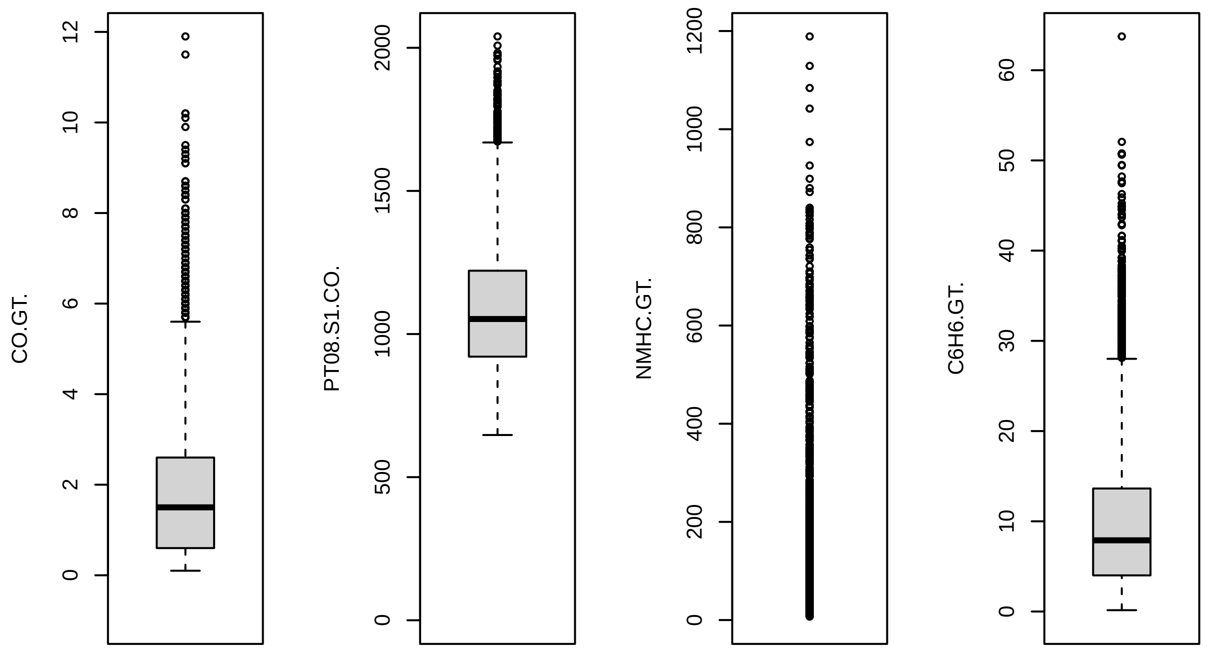
```
## [1]  417 9153
```

```r
# Since this is only two rows, it is simplest to just remove them
aqNew2 <- aqNew[complete.cases(aqNew$DateTime), ]
```

Checking for outliers

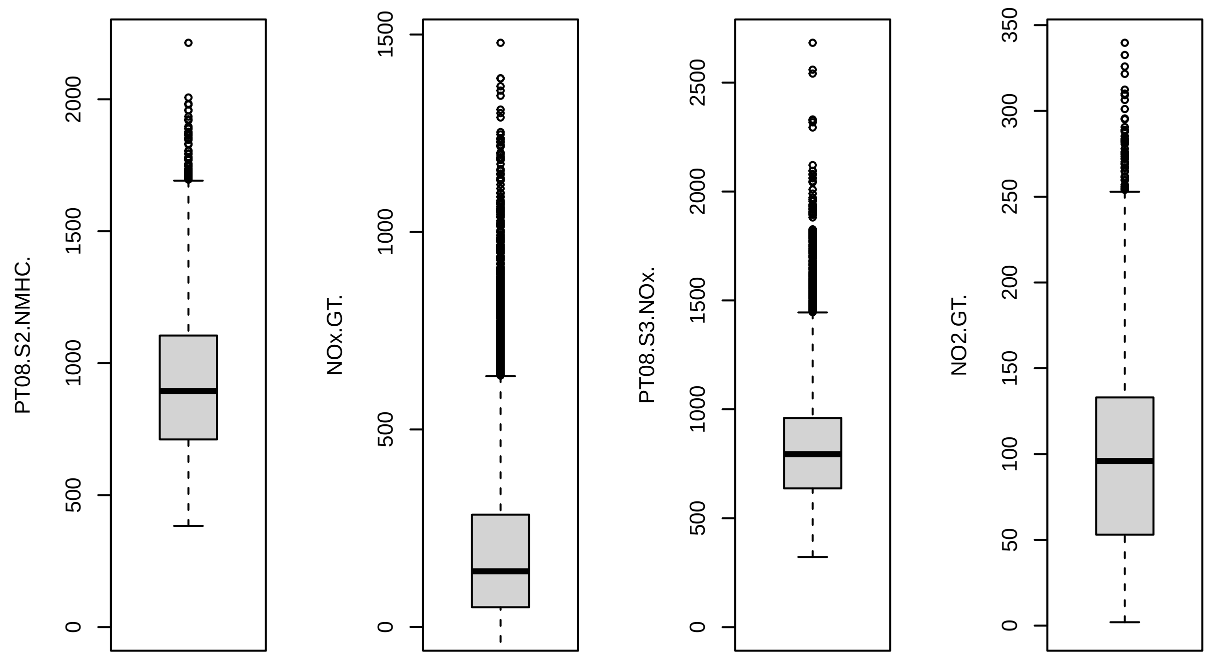
Note: -200 was imputed for any missing values by the creators of the data set, which is hy these plots all have their lower bounds changed.

```r
par(mfrow = c(1,4))
cols <- colnames(aqNew2)
for(i in 1:4){
  ylim <- c(-1, max(aqNew2[i]))
  boxplot(aqNew2[i], ylab = cols[i], ylim = ylim)
}
```

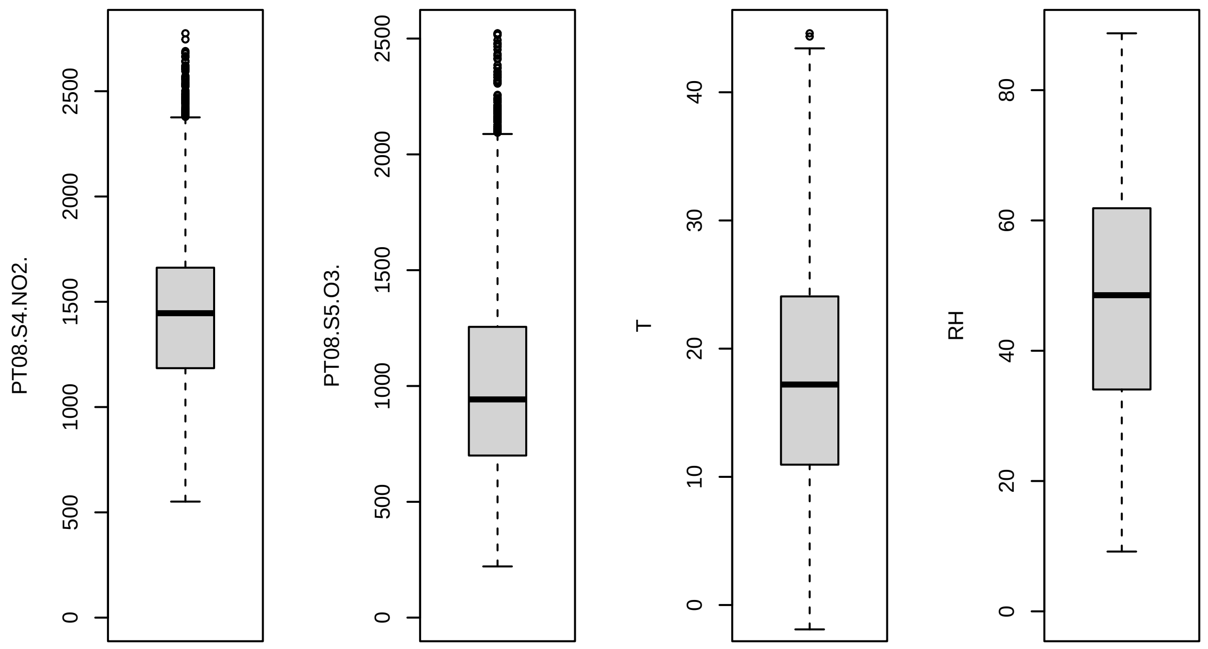


NMHC is missing many values and has -200 imputed a lot, and that is the reason for the box plot not showing up. This feature will be removed later.

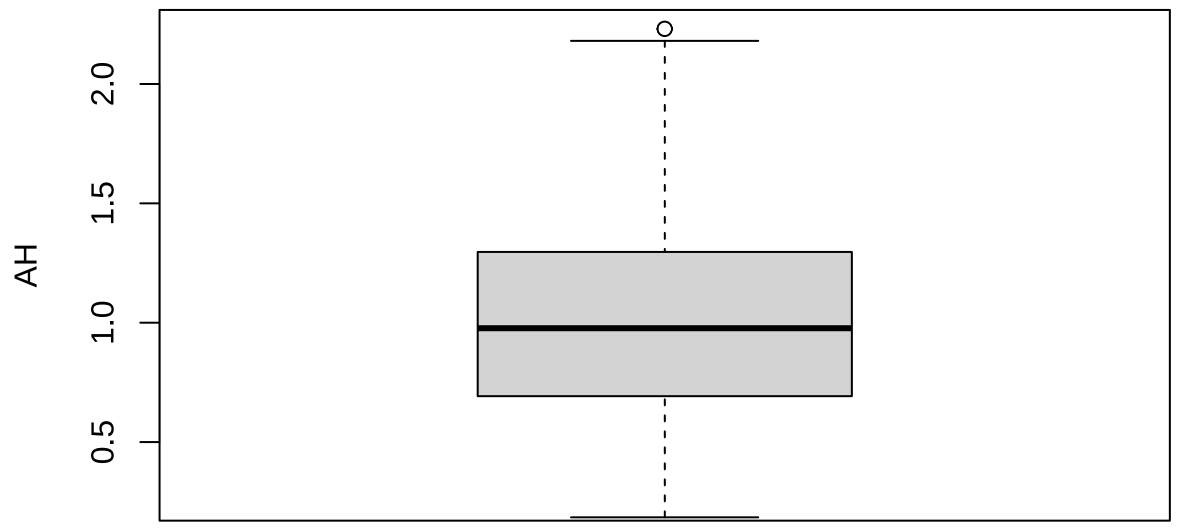
```r
par(mfrow = c(1,4))
for(i in 5:8){
  ylim <- c(-1, max(aqNew2[i]))
  boxplot(aqNew2[i], ylab = cols[i], ylim = ylim)
}
```



```r
par(mfrow = c(1,4))
for(i in 9:12){
  ylim <- c(-1, max(aqNew2[i]))
  boxplot(aqNew2[i], ylab = cols[i], ylim = ylim)
}
```



```r
ylim <- c(0.25, max(aqNew2[13]))
boxplot(aqNew2[13], ylab = cols[13], ylim = ylim)
```



A lot of these boxplots have values of -200, which is impossible for the measurements. So, the next step is to remove any negative values from the data set. The rest of the outliers are impossible to distinguish if they are from misinputs or not, so they will be kept.

```
aqNew3 <- aqNew2
aqNew3[aqNew3 < -10] <- NA
```

We should now inspect the columns and see what percentage of values are missing from each column.

```
cols <- colnames(aqNew3)
for(i in 1:length(aqNew3)){
  missing <- round(sum(is.na(aqNew3[i]))/dim(aqNew3)[1], 5) * 100
  print(c(cols[i], missing))
}
```

```
## [1] "CO.GT." "17.99"
## [1] "PT08.S1.CO." "3.912"
## [1] "NMHC.GT." "90.241"
## [1] "C6H6.GT." "3.912"
## [1] "PT08.S2.NMHC." "3.912"
## [1] "NOx.GT." "17.52"
## [1] "PT08.S3.NOx." "3.912"
## [1] "NO2.GT." "17.552"
## [1] "PT08.S4.NO2." "3.912"
## [1] "PT08.S5.O3." "3.912"
## [1] "T"     "3.912"
## [1] "RH"       "3.912"
## [1] "AH"       "3.912"
## [1] "DateTime" "0"
```

```
# NMHC.GT. is missing 90% of its values, so this feature will be removed
aqNew3 <- aqNew3[-3]
```

Now, we should impute values into the missing values to avoid gaps in our time series.

```
# We are using a method which finds the moving average to impute missing values
aqNew4 <- aqNew3
for(i in 1:length(aqNew4)){
  aqNew4[i] <- na_ma(aqNew4[i], k = 4, weighting = "exponential", maxgap = Inf)
}
```

```
# Examining the new dimensions of our data set
dim(aqNew4)
```

```
## [1] 9355   13
```

Reading cleaned data frame into new CSV file

```
#write.csv(aqNew4, file = 'AirQualityCleaned.csv', row.names = FALSE)
```

Exploritory Data Analysis

Reading in new Data Set

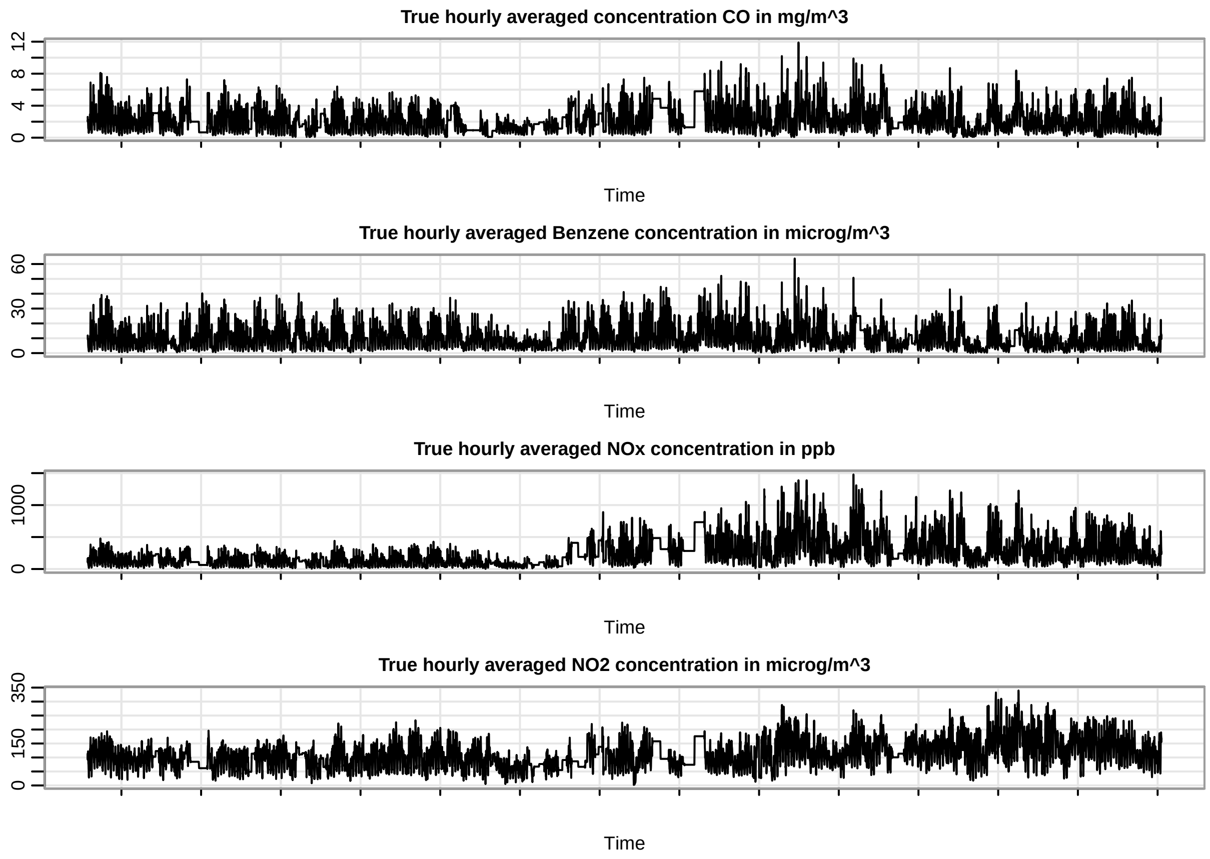
```
aqClean <- read_csv('AirQualityCleaned.csv')
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   CO.GT. = col_double(),
##   PT08.S1.CO. = col_double(),
##   C6H6.GT. = col_double(),
##   PT08.S2.NMHC. = col_double(),
##   NOx.GT. = col_double(),
##   PT08.S3.NOx. = col_double(),
##   NO2.GT. = col_double(),
##   PT08.S4.NO2. = col_double(),
##   PT08.S5.O3. = col_double(),
##   T = col_double(),
##   RH = col_double(),
##   AH = col_double(),
##   DateTime = col_datetime(format = "")
## )
```

Now, lets look at some of the time series plots.

Time Series Plots

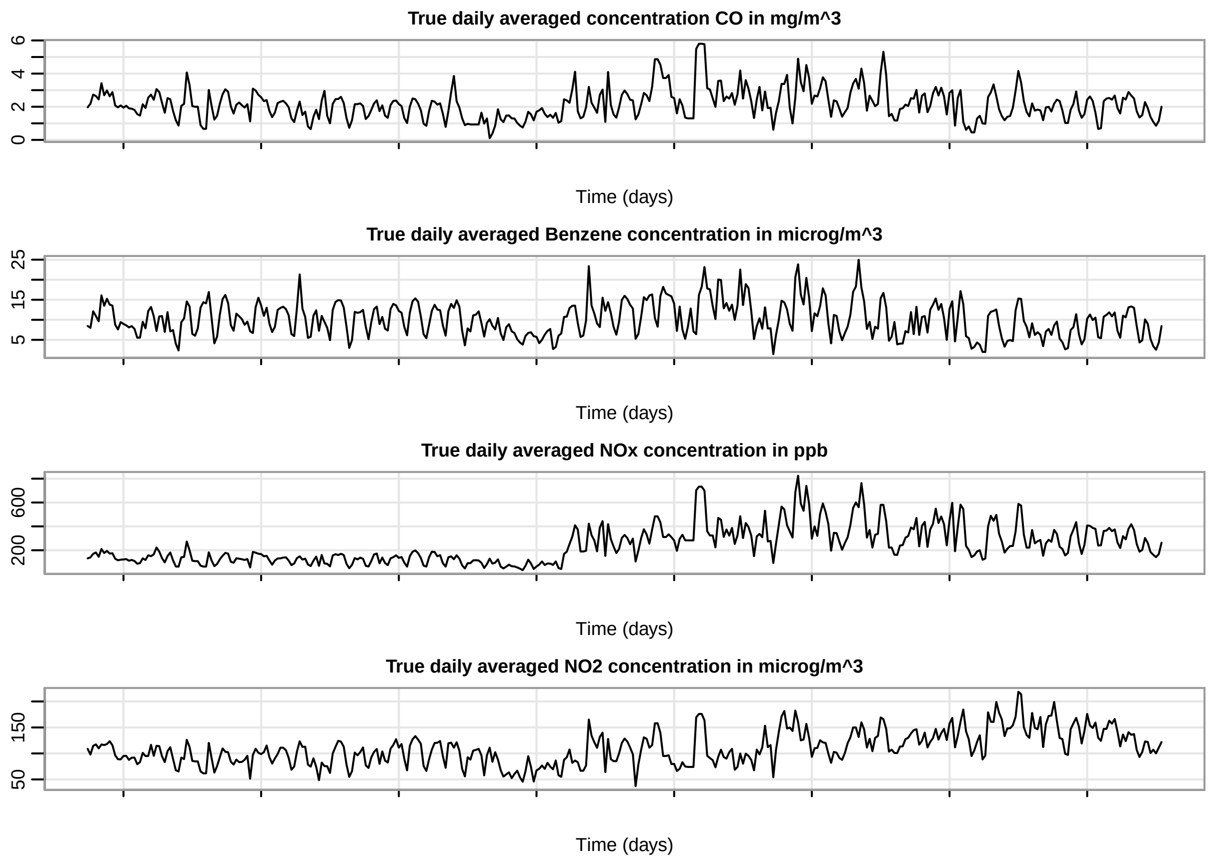
```r
par(mfrow = c(4,1))
tsplot(x = aqClean$DateTime, y = aqClean$CO.GT.,
       main = "True hourly averaged concentration CO in mg/m^3", ylab = "", xaxt="none",
       cex.main = 0.85, cex.axis = 0.85, cex.lab = 0.85)
tsplot(x = aqClean$DateTime, y = aqClean$C6H6.GT.,
       main = "True hourly averaged Benzene concentration in microg/m^3", ylab = "",
       xaxt="none", cex.main = 0.85, cex.axis = 0.85, cex.lab = 0.85)
tsplot(x = aqClean$DateTime, y = aqClean$NOx.GT.,
       main = "True hourly averaged NOx concentration in ppb", ylab = "", xaxt="none",
       cex.main = 0.85, cex.axis = 0.85, cex.lab = 0.85)
tsplot(x = aqClean$DateTime, y = aqClean$NO2.GT.,
       main = "True hourly averaged NO2 concentration in microg/m^3", ylab = "",
       xaxt="none", cex.main = 0.85, cex.axis = 0.85, cex.lab = 0.85)
```



After taking a look at a few time series in the data set, it is apparent that it would be better to look at daily average than to look at hourly average. Doing this would make the data more clear and less noisy, and also allow for better forecasting.

```r
aqClean$Day <- format(aqClean$DateTime, format = "%Y-%m-%d")
aqDaily <- aggregate(aqClean, list(as.Date(aqClean$Day)), FUN=mean)
aqDaily <- aqDaily[-c(14, 15)]
names(aqDaily)[1] <- "Day"
```

```r
# Now lets take a look at the same graphs now
par(mfrow = c(4,1))
tsplot(x = aqDaily$Day, y = aqDaily$CO.GT.,
       main = "True daily averaged concentration CO in mg/m^3", ylab = "", xlab = "Time (days)",
       cex.main = 0.85, cex.axis = 0.85, cex.lab = 0.85, xaxt="n")
tsplot(x = aqDaily$Day, y = aqDaily$C6H6.GT.,
       main = "True daily averaged Benzene concentration in microg/m^3", ylab = "",
       cex.main = 0.85, cex.axis = 0.85, cex.lab = 0.85, xlab = "Time (days)", xaxt="none")
tsplot(x = aqDaily$Day, y = aqDaily$NOx.GT.,
       main = "True daily averaged NOx concentration in ppb", ylab = "", xlab = "Time (days)",
       cex.main = 0.85, cex.axis = 0.85, cex.lab = 0.85, xaxt="none")
tsplot(x = aqDaily$Day, y = aqDaily$NO2.GT.,
       main = "True daily averaged NO2 concentration in microg/m^3", ylab = "",
       xlab = "Time (days)", cex.main = 0.85, cex.axis = 0.85, cex.lab = 0.85, xaxt="none")
```

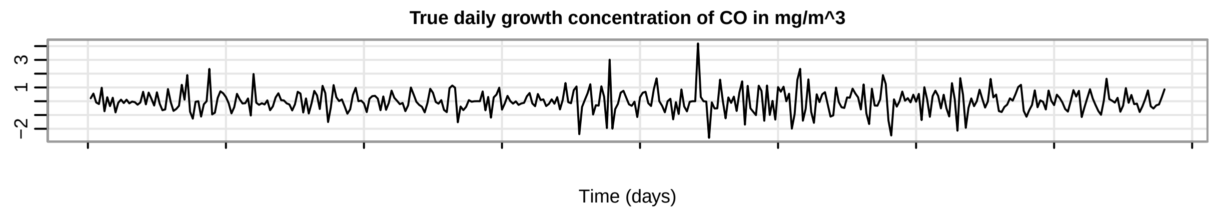


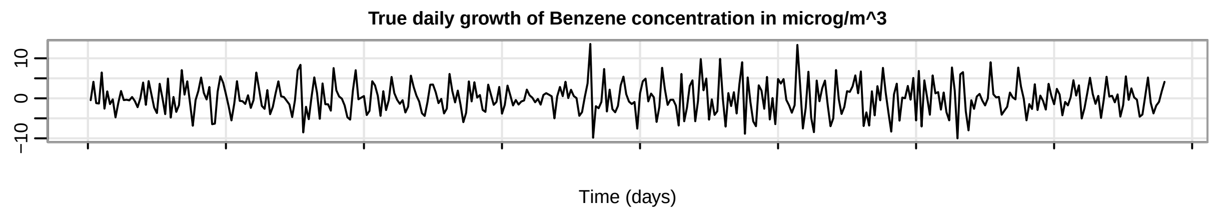
These graphs are much easier to read, so now we can move on to making the time series stationary.

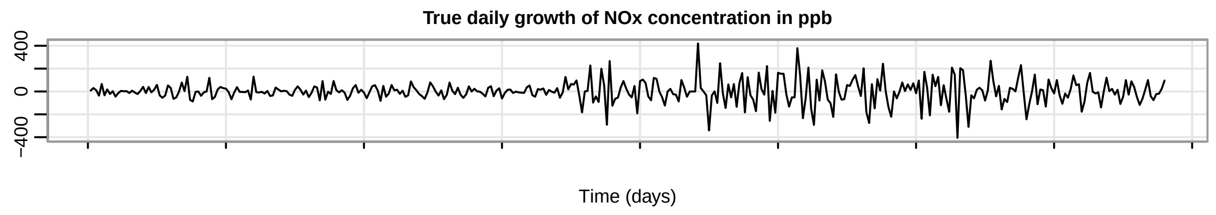
Differencing Data

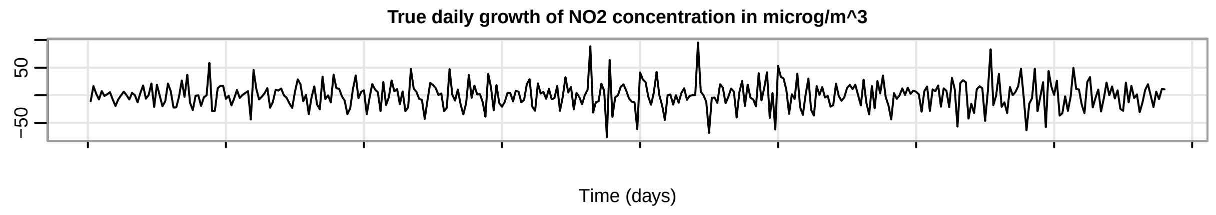
```r
# Same graphs for reference
par(mfrow = c(4,1))
tsplot(diff(aqDaily$CO.GT.),
       main = "True daily growth concentration of CO in mg/m^3", ylab = "", xlab = "Time (days)",
       cex.main = 0.85, cex.axis = 0.85, cex.lab = 0.85, xaxt="n")
```

```r
tsplot(diff(aqDaily$C6H6.GT.),
       main = "True daily growth of Benzene concentration in microg/m^3", ylab = "",
       cex.main = 0.85, cex.axis = 0.85, cex.lab = 0.85, xlab = "Time (days)", xaxt="none")
tsplot(diff(aqDaily$NOx.GT.),
       main = "True daily growth of NOx concentration in ppb", ylab = "", xlab = "Time (days)",
       cex.main = 0.85, cex.axis = 0.85, cex.lab = 0.85, xaxt="none")
tsplot(diff(aqDaily$NO2.GT.),
       main = "True daily growth of NO2 concentration in microg/m^3", ylab = "",
       xlab = "Time (days)", cex.main = 0.85, cex.axis = 0.85, cex.lab = 0.85, xaxt="none")
```





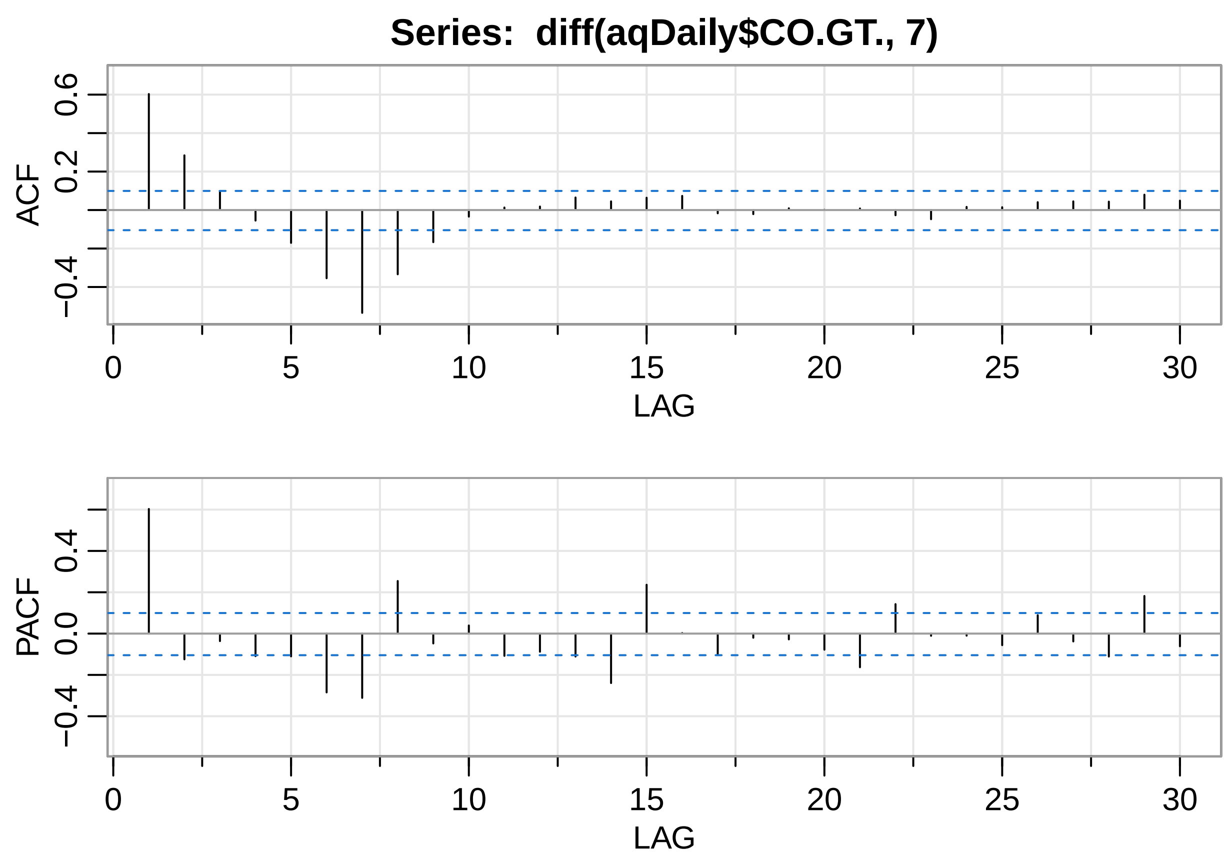




The differencing appeared to make our data stationary, so now we can move on to building an ARIMA model from our data. To do this, we will need to examine the ACF and PACF for each to help determine which model to make.

Looking at P/ACF for Different Features

```r
# CO in mg/m^3
acf2(diff(aqDaily$CO.GT., 7)) # Weekly fluctuations
```

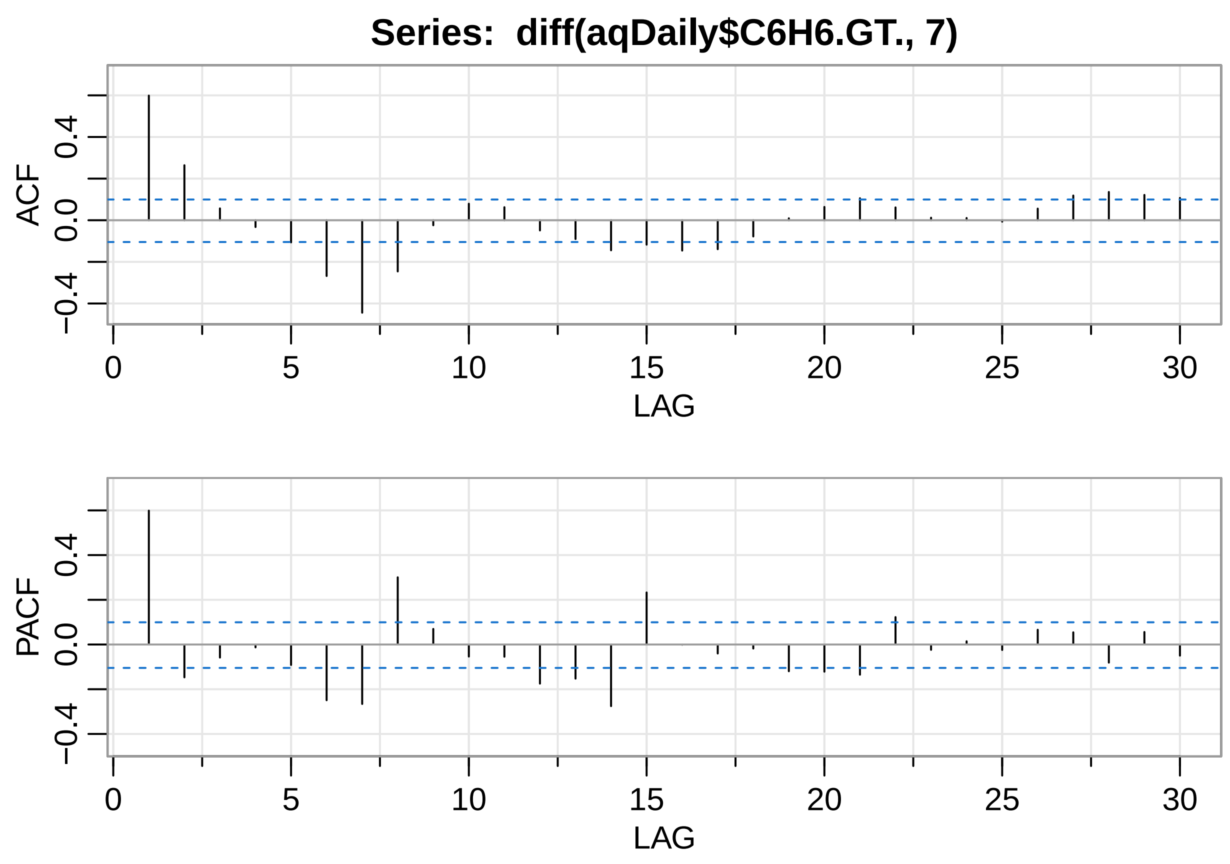


```
##       [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12]
## ACF   0.6  0.28  0.09 -0.05 -0.17 -0.35 -0.53 -0.33 -0.17 -0.03  0.01  0.02
## PACF  0.6 -0.12 -0.04 -0.11 -0.11 -0.28 -0.31  0.25 -0.05  0.04 -0.11 -0.09
##      [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## ACF   0.07  0.05  0.06  0.07 -0.02 -0.02  0.01  0.00  0.01 -0.03 -0.05  0.02
## PACF -0.11 -0.24  0.24  0.00 -0.10 -0.02 -0.03 -0.08 -0.16  0.14 -0.01 -0.01
##      [,25] [,26] [,27] [,28] [,29] [,30]
## ACF   0.01  0.04  0.05  0.04  0.08  0.05
## PACF -0.06  0.09 -0.04 -0.11  0.18 -0.06
```

Seasonal: The PACF is cutting off a lag 1s (s = 7), whereas the ACF is tailing off at lags 1s, 7s. These results imply an SAR(1), P = 1, Q = 0.

Non-Seasonal: It appears that the PACF cuts off at lag 1, whereas the PACF tails off. This suggests an AR(1) with p = 1 and q = 0.

```
# Benzene in microg/m^3
acf2(diff(aqDaily$C6H6.GT., 7)) # Appears to have a weekly spike
```

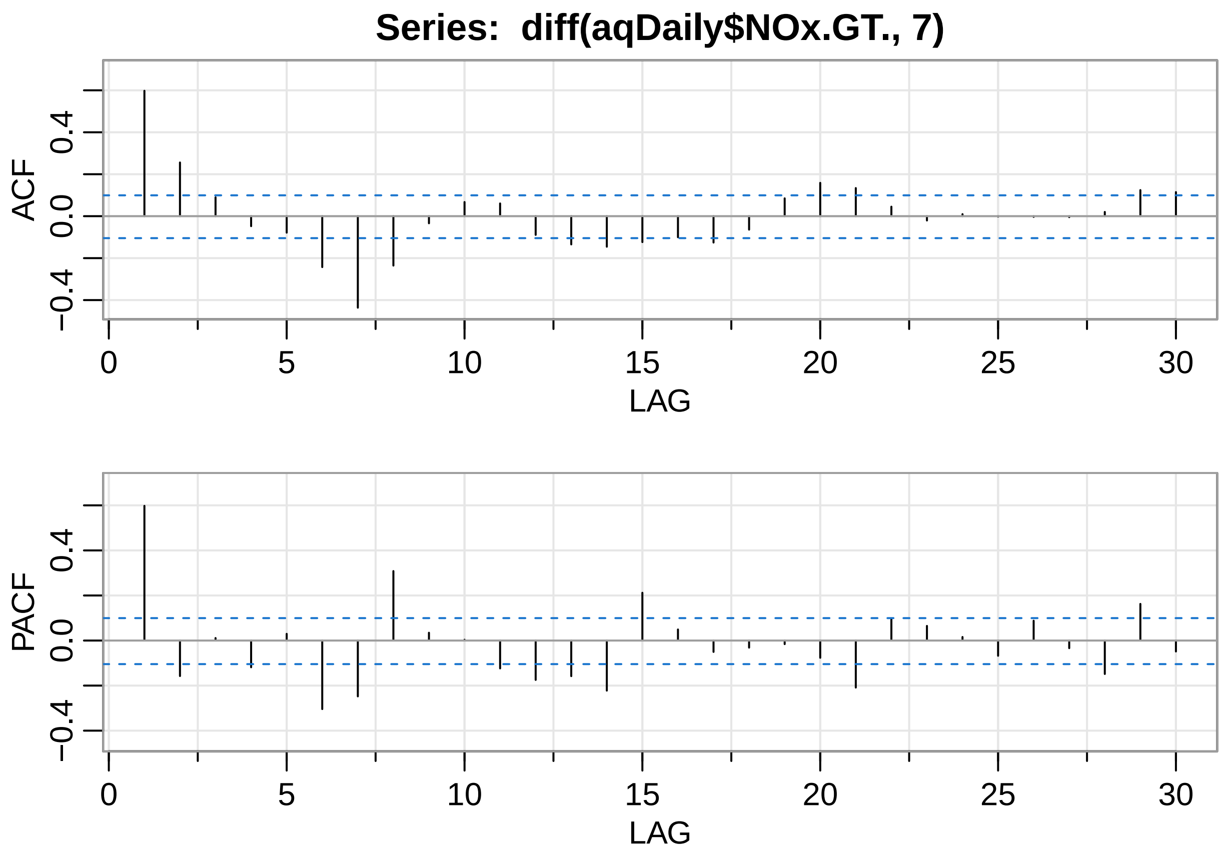


```
##      [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12]
## ACF   0.6  0.26  0.06 -0.03 -0.11 -0.27 -0.44 -0.25 -0.02  0.08  0.06 -0.05
## PACF  0.6 -0.15 -0.06 -0.01 -0.09 -0.25 -0.27  0.30  0.07 -0.05 -0.05 -0.17
##      [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## ACF  -0.09 -0.14 -0.12 -0.15 -0.14 -0.08  0.01  0.06  0.11  0.06  0.01  0.01
## PACF -0.15 -0.28  0.23  0.00 -0.04 -0.02 -0.12 -0.12 -0.14  0.12 -0.02  0.01
##      [,25] [,26] [,27] [,28] [,29] [,30]
## ACF  -0.01  0.06  0.12  0.14  0.12  0.11
## PACF -0.02  0.07  0.05 -0.08  0.06 -0.05
```

Seasonal: The PACF is cutting off a lag 1s (s = 7), whereas the ACF is tailing off at lags 1s, 7s, and 14s. These results imply an SAR(1), P = 1, Q = 0.

Non-Seasonal: It appears that the PACF cuts off at lag 1, whereas the ACF tails off. This suggests an AR(1) with p = 1 and q = 0.

```
# NOx in ppb
acf2(diff(aqDaily$NOx.GT., 7))
```

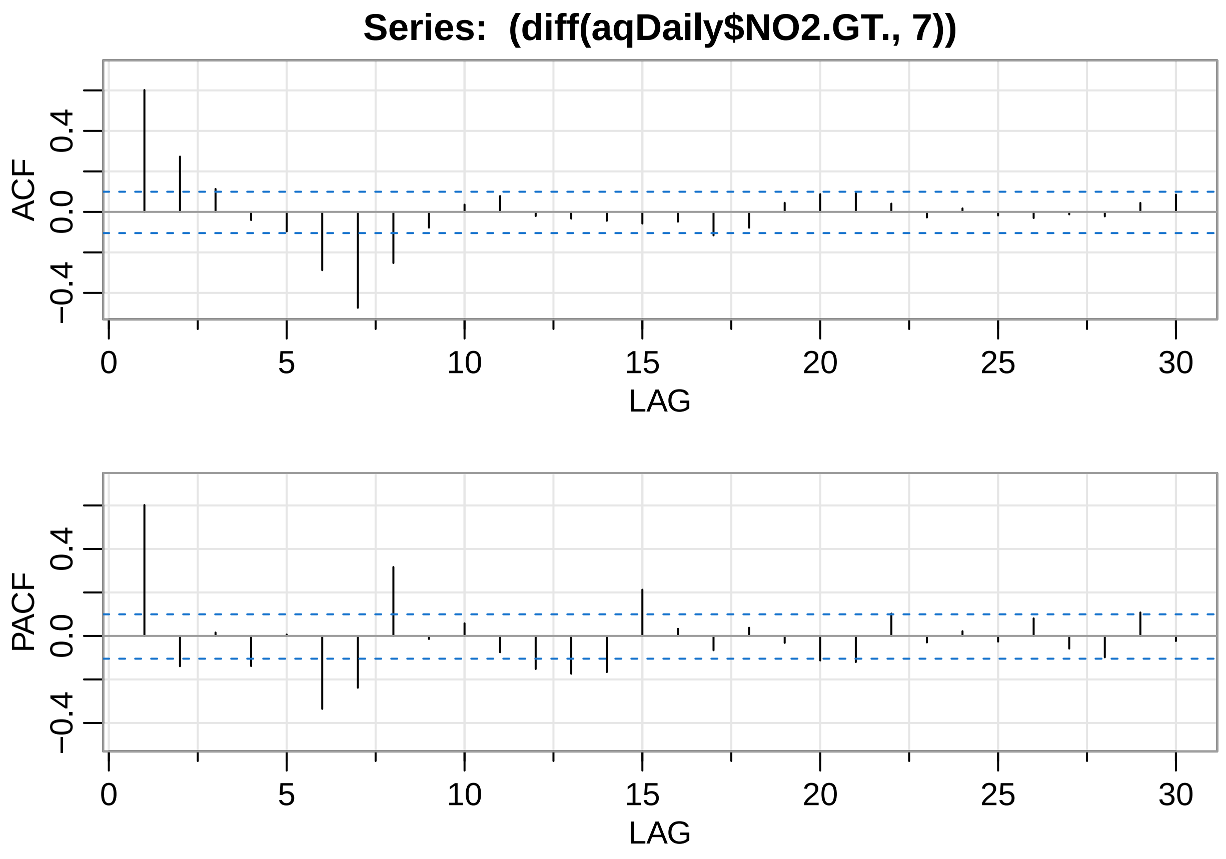


```
##       [,1]  [,2] [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12]
## ACF   0.6  0.26 0.09 -0.05 -0.08 -0.24 -0.44 -0.24 -0.03  0.07  0.06 -0.09
## PACF  0.6 -0.16 0.01 -0.12  0.03 -0.30 -0.25  0.31  0.03  0.00 -0.12 -0.17
##      [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## ACF  -0.13 -0.15 -0.12 -0.10 -0.13 -0.06  0.09  0.16  0.13  0.05 -0.02  0.01
## PACF -0.16 -0.22  0.21  0.05 -0.05 -0.03 -0.02 -0.08 -0.21  0.09  0.06  0.02
##      [,25] [,26] [,27] [,28] [,29] [,30]
## ACF   0.00  0.00  0.00  0.02  0.12  0.11
## PACF -0.07  0.09 -0.03 -0.15  0.16 -0.05
```

Seasonal: The PACF is cutting off a lag 1s (s = 7), whereas the ACF is tailing off at lags 1s, 7s, 14s. These results imply an SAR(1), P = 1, Q = 0.

Non-Seasonal: It appears that the PACF cuts off at lag 1, whereas the ACF tails off. This suggests an AR(1) with p = 1 and q = 0.

```
# NO2 in microg/m^3
acf2((diff(aqDaily$NO2.GT., 7)))
```



```
##       [,1]  [,2] [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12]
## ACF   0.6  0.27 0.11 -0.04 -0.10 -0.29 -0.47 -0.25 -0.08  0.04  0.08 -0.02
## PACF  0.6 -0.14 0.02 -0.14  0.01 -0.34 -0.24  0.32 -0.01  0.06 -0.08 -0.15
##      [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## ACF  -0.03 -0.04 -0.06 -0.05 -0.12 -0.08  0.04  0.09  0.09  0.04 -0.03  0.02
## PACF -0.17 -0.17  0.21  0.03 -0.07  0.04 -0.03 -0.11 -0.12  0.10 -0.03  0.02
##      [,25] [,26] [,27] [,28] [,29] [,30]
## ACF  -0.02 -0.03 -0.01 -0.02  0.04  0.09
## PACF -0.03  0.08 -0.06 -0.10  0.11 -0.02
```

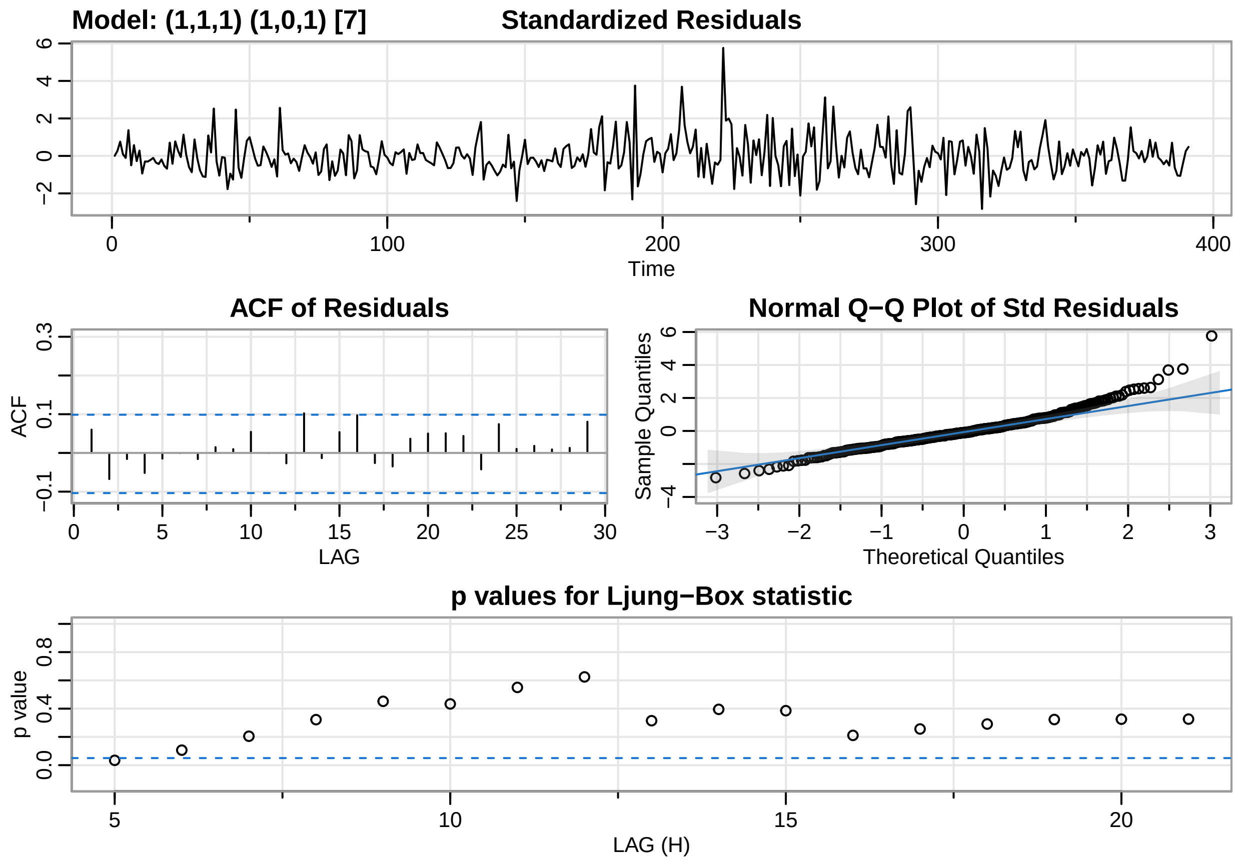
Note that both NOx and NO2 are identical. The only difference between the two are the absolute values of their size. For example, NOx and NO2 will peek and dip at the same time, but their actual values differ. This is due probably to the different units of measurement. However, NOx refers to the total nitrogen oxides while NO2 is nitrogen dioxide, so there is a difference to be noted.

Building Models and Forecasting

Determining the ARMA models

```
# For CO
coMod <- sarima(aqDaily$CO.GT., p = 1, d = 1, q = 1, P = 1, D = 0, Q = 1, S = 7)
```

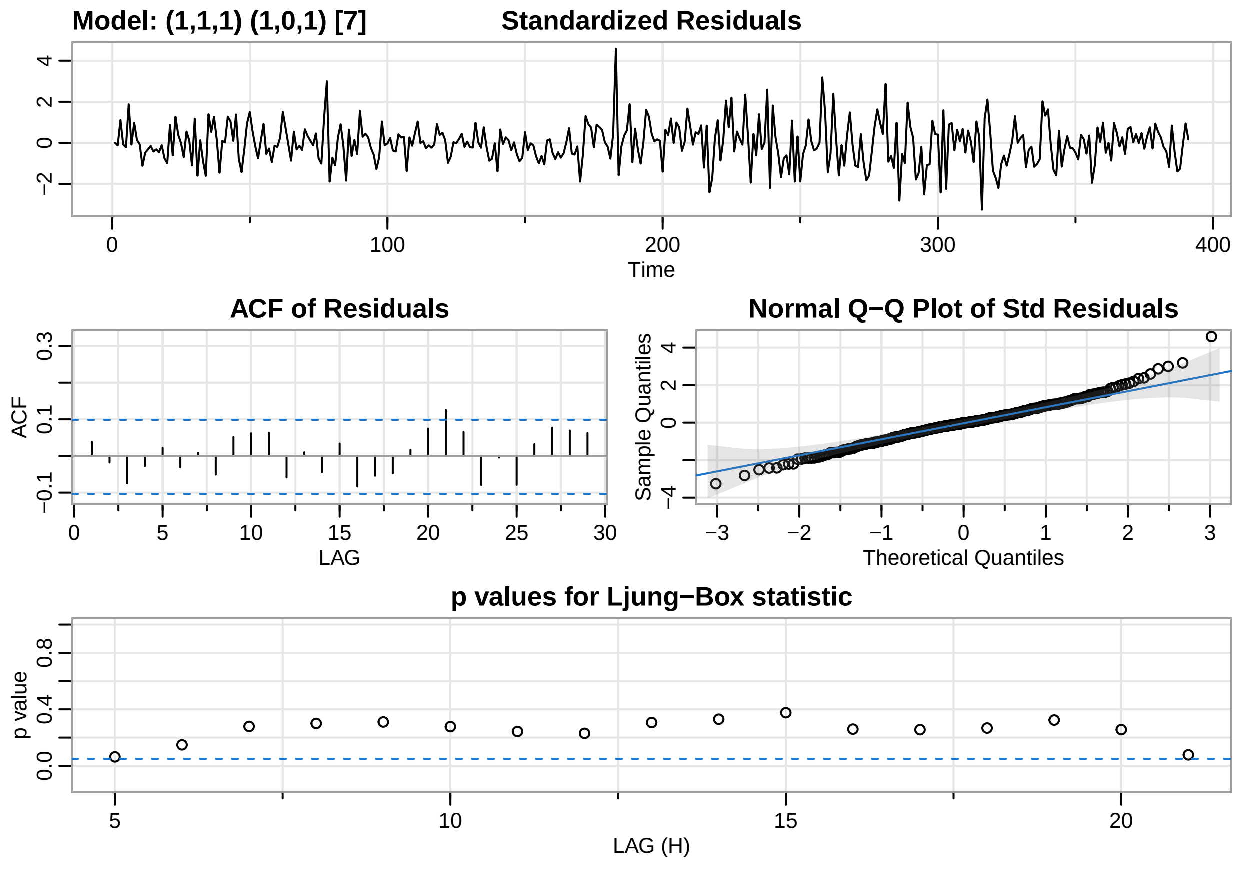
```
## initial  value -0.218162
## iter   2 value -0.227753
## iter   3 value -0.233694
## iter   4 value -0.234299
## iter   5 value -0.235556
## iter   6 value -0.265699
## iter   7 value -0.268079
## iter   8 value -0.290150
## iter   9 value -0.310106
## iter  10 value -0.326514
## iter  11 value -0.340144
## iter  12 value -0.341297
## iter  13 value -0.342001
## iter  14 value -0.344215
## iter  15 value -0.345326
## iter  16 value -0.346491
## iter  17 value -0.347237
## iter  18 value -0.349924
## iter  19 value -0.350709
## iter  20 value -0.352156
## iter  21 value -0.352311
## iter  22 value -0.352421
## iter  23 value -0.352475
## iter  24 value -0.352537
## iter  25 value -0.352542
## iter  26 value -0.352548
## iter  27 value -0.352554
## iter  28 value -0.352559
## iter  29 value -0.352561
## iter  30 value -0.352562
## iter  31 value -0.352563
## iter  31 value -0.352563
## iter  31 value -0.352563
## final  value -0.352563
## converged
## initial  value -0.363177
## iter   2 value -0.364367
## iter   3 value -0.364694
## iter   4 value -0.366110
## iter   5 value -0.366426
## iter   6 value -0.366861
## iter   7 value -0.367158
## iter   8 value -0.367957
## iter   9 value -0.368051
## iter  10 value -0.368310
## iter  11 value -0.368467
## iter  12 value -0.368694
## iter  13 value -0.368722
## iter  14 value -0.368737
## iter  15 value -0.368740
## iter  16 value -0.368804
## iter  17 value -0.368854
## iter  18 value -0.369023
## iter  19 value -0.369154
## iter  20 value -0.369280
## iter  21 value -0.369309
## iter  22 value -0.369311
## iter  23 value -0.369311
## iter  23 value -0.369311
## final  value -0.369311
## converged
```



It appears to be decent. 1 of our Q-statistic p-values are less than 0.05, but the majority are not. It is important to have values greater than 0.05, because then we do not need to reject the null hypothesis that the noise is white. Adding q = 1 and Q = 1 helps with model evaluation metrics.

```
# For Benzene
benzeneMod <- sarima(aqDaily$C6H6.GT., p = 1, d = 1, q = 1, P = 1, D = 0, Q = 1, S = 7)
```

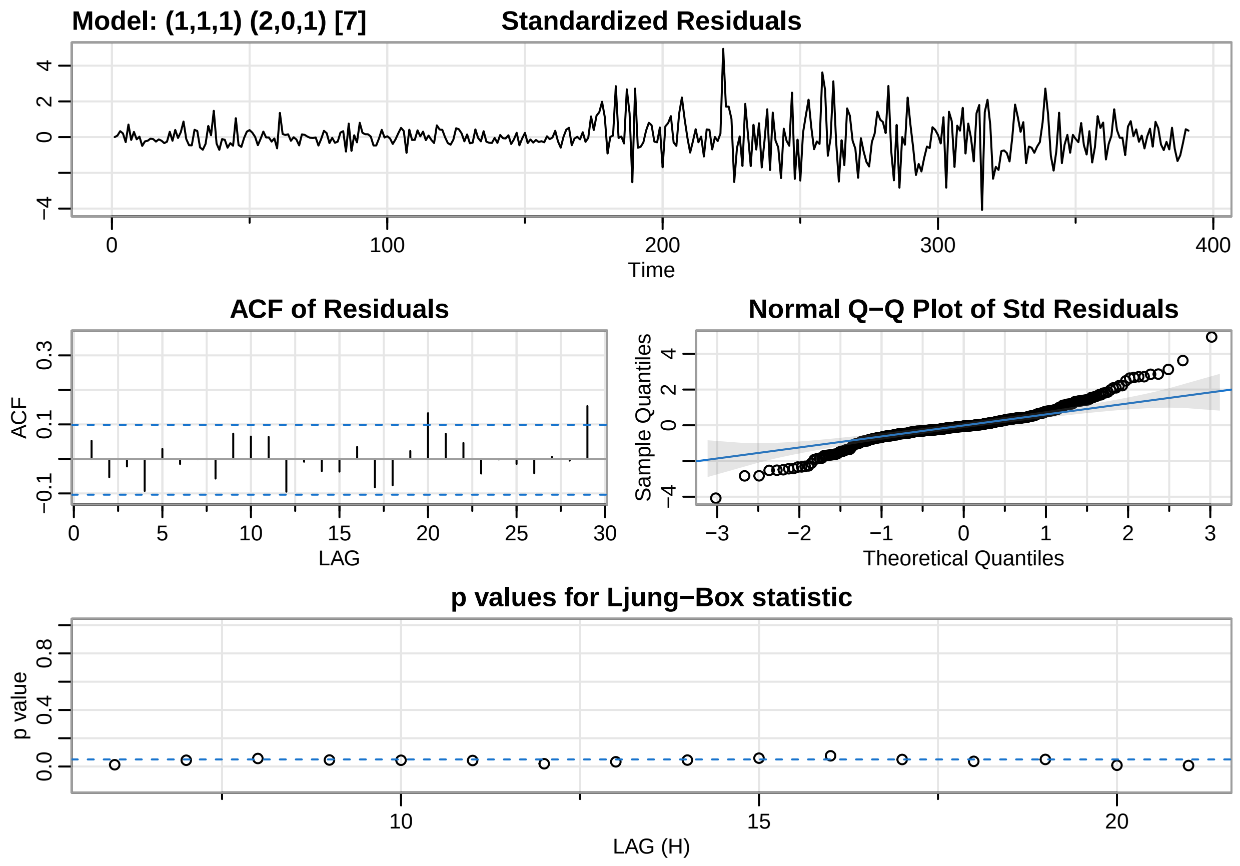
```
## initial  value 1.343675
## iter   2 value 1.310250
## iter   3 value 1.293150
## iter   4 value 1.291192
## iter   5 value 1.272600
## iter   6 value 1.266102
## iter   7 value 1.259624
## iter   8 value 1.250126
## iter   9 value 1.235392
## iter  10 value 1.227723
## iter  11 value 1.220317
## iter  12 value 1.212361
## iter  13 value 1.195409
## iter  14 value 1.191142
## iter  15 value 1.177596
## iter  16 value 1.175049
## iter  17 value 1.164856
## iter  18 value 1.160993
## iter  19 value 1.157911
## iter  20 value 1.152780
## iter  21 value 1.147654
## iter  22 value 1.142913
## iter  23 value 1.141232
## iter  24 value 1.139310
## iter  25 value 1.139285
## iter  26 value 1.138945
## iter  27 value 1.138919
## iter  28 value 1.138904
## iter  29 value 1.138898
## iter  30 value 1.138896
## iter  31 value 1.138896
## iter  31 value 1.138896
## iter  31 value 1.138896
## final  value 1.138896
## converged
## initial  value 1.113958
## iter   2 value 1.113361
## iter   3 value 1.098533
## iter   4 value 1.095053
## iter   5 value 1.093871
## iter   6 value 1.093229
## iter   7 value 1.091243
## iter   8 value 1.089739
## iter   9 value 1.086984
## iter  10 value 1.086089
## iter  11 value 1.085774
## iter  12 value 1.085533
## iter  13 value 1.085437
## iter  14 value 1.085362
## iter  15 value 1.085228
## iter  16 value 1.085194
## iter  17 value 1.085163
## iter  18 value 1.085154
## iter  19 value 1.085076
## iter  20 value 1.085020
## iter  21 value 1.084825
## iter  22 value 1.084569
## iter  23 value 1.084221
## iter  24 value 1.084215
## iter  25 value 1.084171
## iter  26 value 1.084152
## iter  27 value 1.084146
## iter  28 value 1.084140
## iter  29 value 1.084139
## iter  30 value 1.084139
## iter  31 value 1.084136
## iter  32 value 1.084131
## iter  33 value 1.084118
## iter  34 value 1.084094
## iter  35 value 1.084093
## iter  36 value 1.084086
## iter  37 value 1.084080
## iter  38 value 1.084074
## iter  39 value 1.084073
## iter  40 value 1.084073
## iter  41 value 1.084070
## iter  42 value 1.084066
## iter  43 value 1.084056
## iter  44 value 1.084037
## iter  45 value 1.084009
## iter  46 value 1.084007
## iter  47 value 1.084004
## iter  48 value 1.084000
## iter  49 value 1.083996
## iter  50 value 1.083994
## iter  51 value 1.083994
## iter  52 value 1.083993
## iter  53 value 1.083993
## iter  54 value 1.083991
## iter  55 value 1.083988
## iter  56 value 1.083982
## iter  57 value 1.083981
## iter  58 value 1.083981
## iter  59 value 1.083981
## iter  60 value 1.083981
## iter  61 value 1.083980
## iter  61 value 1.083980
## iter  62 value 1.083980
## iter  62 value 1.083980
## iter  62 value 1.083980
## final  value 1.083980
## converged
```



Adding q = 1 and q = 1 to our model improves the Q-statistic p-values.

```
# For NOx
noxMod <- sarima(aqDaily$NOx.GT., p = 1, d = 1, q = 1, P = 2, D = 0, Q = 1, S = 7)
```

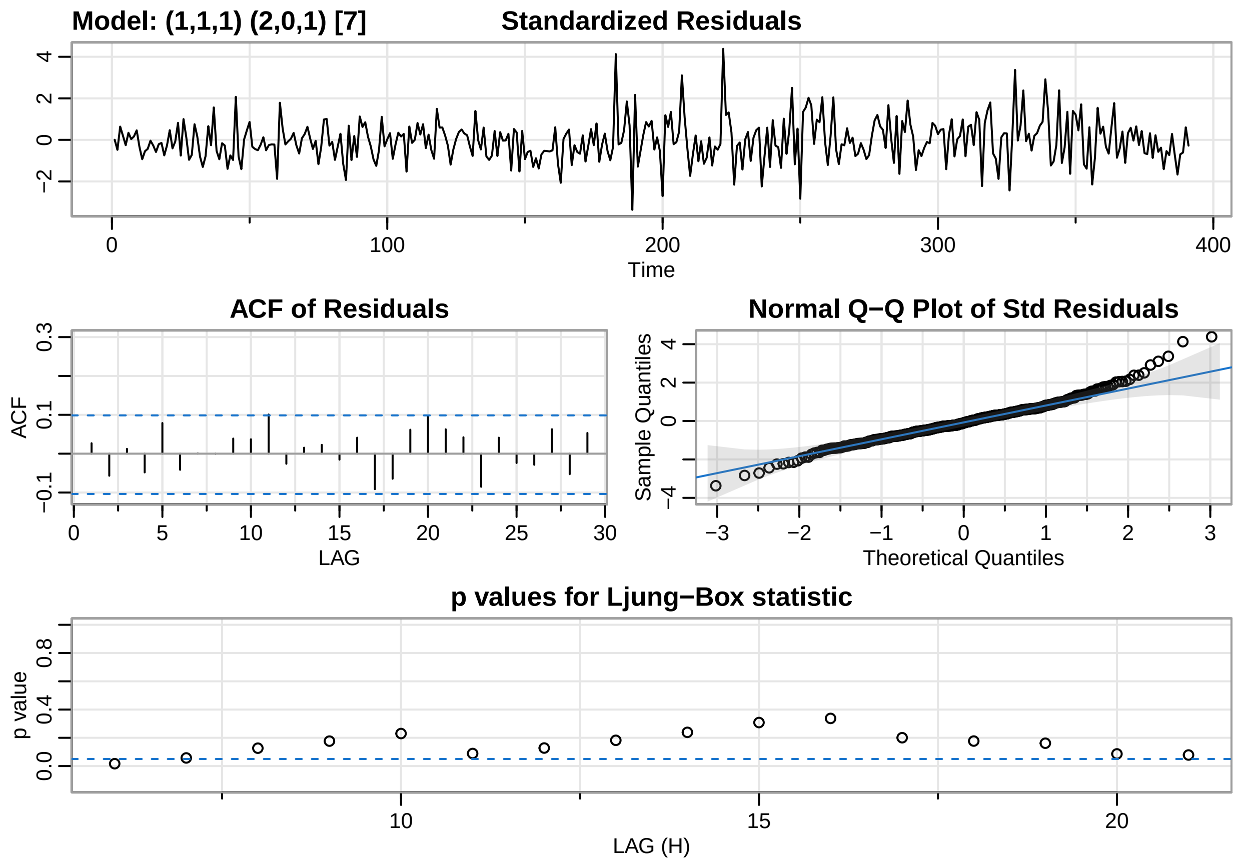
```
## initial  value 4.600811
## iter   2 value 4.596160
## iter   3 value 4.578285
## iter   4 value 4.577468
## iter   5 value 4.575434
## iter   6 value 4.555569
## iter   7 value 4.553105
## iter   8 value 4.547404
## iter   9 value 4.545436
## iter  10 value 4.539536
## iter  11 value 4.532841
## iter  12 value 4.524433
## iter  13 value 4.500208
## iter  14 value 4.477782
## iter  15 value 4.474291
## iter  16 value 4.473235
## iter  17 value 4.470487
## iter  18 value 4.470169
## iter  19 value 4.469722
## iter  20 value 4.469580
## iter  21 value 4.469412
## iter  22 value 4.469338
## iter  23 value 4.469322
## iter  24 value 4.469318
## iter  25 value 4.469317
## iter  26 value 4.469316
## iter  26 value 4.469316
## iter  26 value 4.469316
## final  value 4.469316
## converged
## initial  value 4.451445
## iter   2 value 4.449730
## iter   3 value 4.449607
## iter   4 value 4.448895
## iter   5 value 4.448655
## iter   6 value 4.448183
## iter   7 value 4.448160
## iter   8 value 4.448153
## iter   9 value 4.448129
## iter  10 value 4.448105
## iter  11 value 4.448043
## iter  12 value 4.447975
## iter  13 value 4.447909
## iter  14 value 4.447890
## iter  14 value 4.447890
## final  value 4.447890
## converged
```



Adding q = 1, Q = 1, and P = 2 helps with our model. Since the first half of the data is different from the second half, we are getting bad results from the different statistical measures that cannot be changed. If P = 1, it does not allow for Q = 1, so this is why P = 2 instead.

```
# For NO2
no2Mod <- sarima(aqDaily$NO2.GT., p = 1, d = 1, q = 1, P = 2, D = 0, Q = 1, S = 7)
```

```
## initial  value 3.152529
## iter   2 value 3.143735
## iter   3 value 3.112233
## iter   4 value 3.110394
## iter   5 value 3.107882
## iter   6 value 3.092818
## iter   7 value 3.087324
## iter   8 value 3.080252
## iter   9 value 3.068423
## iter  10 value 3.062030
## iter  11 value 3.052657
## iter  12 value 3.038814
## iter  13 value 3.030925
## iter  14 value 3.028363
## iter  15 value 3.025532
## iter  16 value 3.025374
## iter  17 value 3.025239
## iter  18 value 3.024923
## iter  19 value 3.024534
## iter  20 value 3.024515
## iter  21 value 3.024390
## iter  22 value 3.024257
## iter  23 value 3.024135
## iter  24 value 3.023737
## iter  25 value 3.023012
## iter  26 value 3.021884
## iter  27 value 3.021797
## iter  28 value 3.021678
## iter  29 value 3.021662
## iter  30 value 3.021257
## iter  31 value 3.020941
## iter  32 value 3.019217
## iter  33 value 3.018176
## iter  34 value 3.016903
## iter  35 value 3.016879
## iter  36 value 3.016649
## iter  37 value 3.016380
## iter  38 value 3.016352
## iter  39 value 3.016335
## iter  40 value 3.016321
## iter  41 value 3.015296
## iter  41 value 3.015296
## iter  41 value 3.015296
## final  value 3.015296
## converged
## initial  value 3.016589
## iter   2 value 3.011961
## iter   3 value 3.011415
## iter   4 value 3.010931
## iter   5 value 3.009911
## iter   6 value 3.007866
## iter   7 value 3.007734
## iter   8 value 3.007497
## iter   9 value 3.006881
## iter  10 value 3.006084
## iter  11 value 3.002350
## iter  12 value 2.994954
## iter  13 value 2.992929
## iter  14 value 2.991712
## iter  15 value 2.989070
## iter  16 value 2.988721
## iter  17 value 2.988377
## iter  18 value 2.987618
## iter  19 value 2.987333
## iter  20 value 2.987272
## iter  21 value 2.987200
## iter  22 value 2.987155
## iter  23 value 2.987089
## iter  24 value 2.987048
## iter  25 value 2.986840
## iter  26 value 2.986558
## iter  27 value 2.986304
## iter  28 value 2.986171
## iter  29 value 2.986155
## iter  30 value 2.986153
## iter  31 value 2.986153
## iter  31 value 2.986153
## iter  31 value 2.986153
## final  value 2.986153
## converged
```



This will share the same parameters as NOx for the seasonal ARMA model.

Forecasting

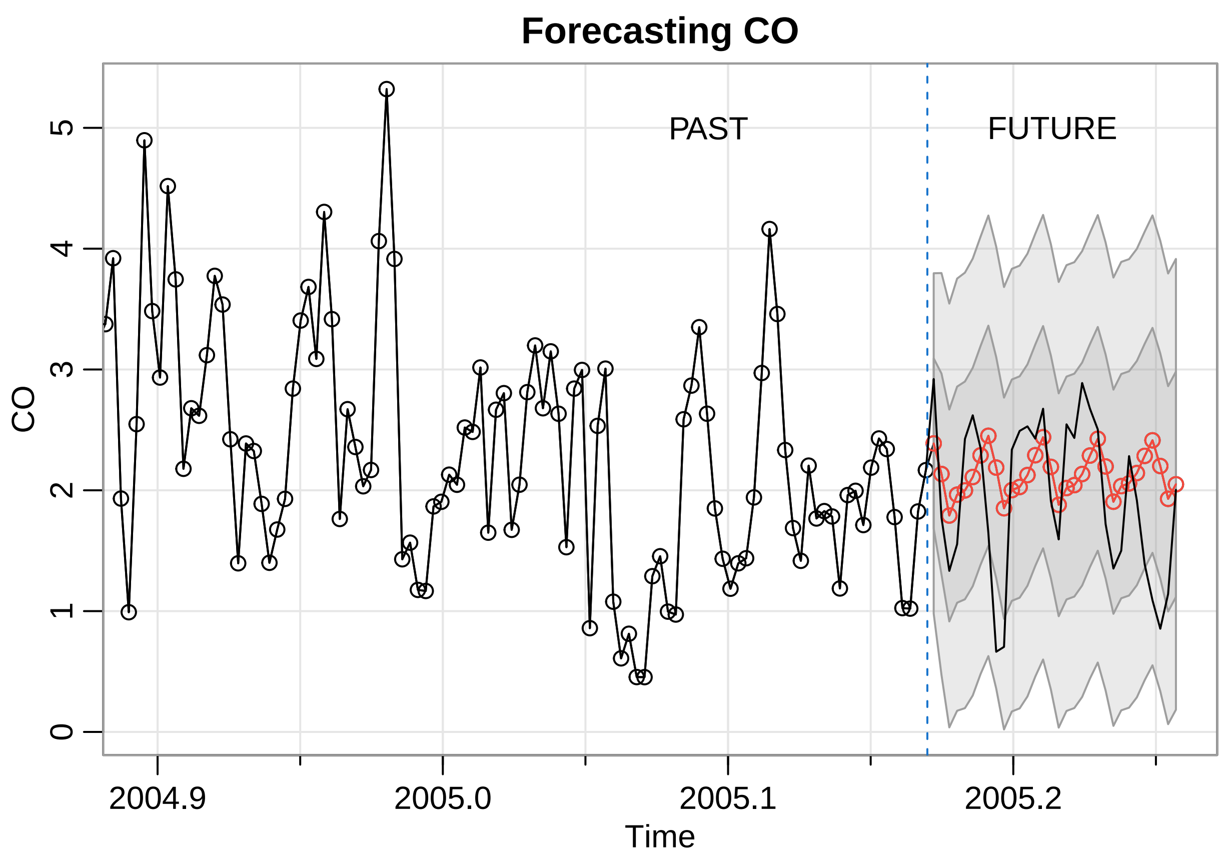
The 'Future' portion of our graph is where the forecasting is made. The black line represents the actual data collected while the red line shows our ARMA model prediction. The grayed out areas are confidence intervals.

```r
# Plotting the final month of time series for CO
x <- ts(aqDaily$CO.GT., start = decimal_date(as.Date("2004-03-10")), frequency = 365)
CO <- window(x, start=decimal_date(as.Date("2004-03-10")),
            end=decimal_date(as.Date("2005-03-04")))
sarima.for(CO, n.ahead = 32, p = 1, d = 1, q = 1, P = 1, D = 0, Q = 1, S = 7, plot.all=FALSE,
           main = "Forecasting CO")
```

```
## $pred
## Time Series:
## Start = 2005.172086234
## End = 2005.25701774085
## Frequency = 365
##  [1] 2.387969 2.134849 1.791367 1.963855 1.999050 2.109885 2.289176 2.451192
##  [9] 2.188225 1.851467 2.001216 2.027251 2.126183 2.290385 2.439501 2.193817
## [17] 1.879861 2.018809 2.042800 2.134712 2.287412 2.426113 2.197503 1.905386
## [25] 2.034673 2.057006 2.142538 2.284632 2.413702 2.201015 1.929240 2.049553
##
```

```
## $se
## Time Series:
## Start = 2005.172086234
## End = 2005.25701774085
## Frequency = 365
##  [1] 0.7043185 0.8316126 0.8768882 0.8942709 0.9011591 0.9039500 0.9051079
##  [8] 0.9116763 0.9143515 0.9154694 0.9159522 0.9161695 0.9162724 0.9163238
## [15] 0.9199632 0.9214770 0.9221270 0.9224176 0.9225540 0.9226216 0.9226570
## [22] 0.9257779 0.9270886 0.9276585 0.9279174 0.9280411 0.9281036 0.9281368
## [29] 0.9308712 0.9320302 0.9325402 0.9327752
```

```
text(decimal_date(as.Date("2005-02-04")), 5, "PAST")
text(decimal_date(as.Date("2005-03-20")), 5, "FUTURE")
abline(v=decimal_date(as.Date("2005-03-04")), lty=2, col=4)
lines(x)
```

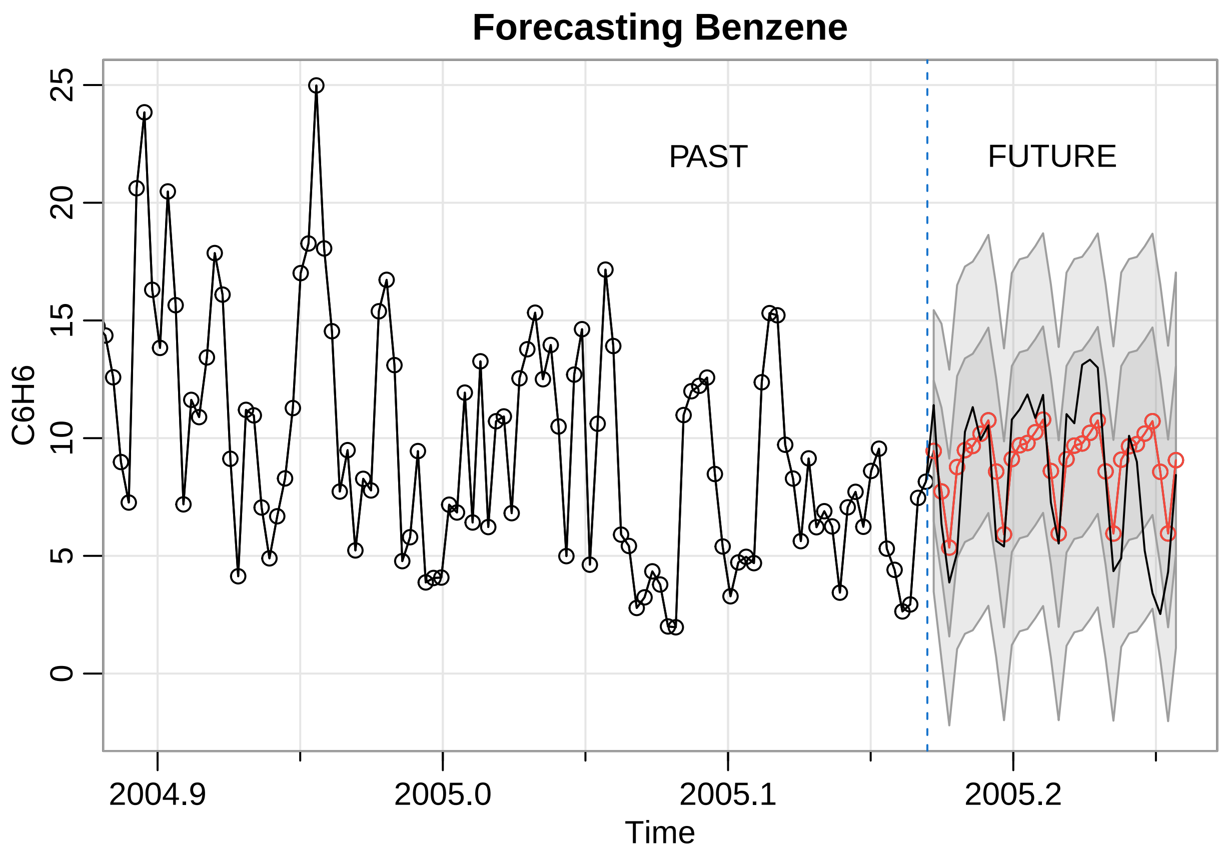


```
# Plotting the final month of time series for Benzene
x <- ts(aqDaily$C6H6.GT., start = decimal_date(as.Date("2004-03-10")), frequency = 365)
C6H6 <- window(x, start=decimal_date(as.Date("2004-03-10")),
               end=decimal_date(as.Date("2005-03-04")))
sarima.for(C6H6, n.ahead = 32, p = 1, d = 1, q = 1, P = 1, D = 0, Q = 1, S = 7,
           plot.all=FALSE, main = "Forecasting Benzene")
```

```
## $pred
## Time Series:
```

```
## Start = 2005.172086234
## End = 2005.25701774085
## Frequency = 365
##  [1]  9.457919  7.734957  5.351241  8.771272  9.488782  9.668681 10.182794
##  [8] 10.757834  8.580876  5.913347  9.110985  9.696287  9.792866 10.249956
## [15] 10.786203  8.604052  5.942992  9.106341  9.681039  9.772953 10.223933
## [22] 10.754339  8.588159  5.947184  9.085352  9.655250  9.746236 10.193548
## [29] 10.719707  8.570375  5.949976  9.063580
##
## $se
## Time Series:
## Start = 2005.172086234
## End = 2005.25701774085
## Frequency = 365
##  [1] 2.990859 3.563064 3.777437 3.863962 3.899815 3.914853 3.921221 3.938398
##  [9] 3.945665 3.948766 3.950103 3.950686 3.950940 3.951061 3.958436 3.961588
## [17] 3.962950 3.963546 3.963810 3.963925 3.963985 3.970894 3.973852 3.975134
## [25] 3.975698 3.975949 3.976059 3.976116 3.982901 3.985811 3.987075 3.987632
```

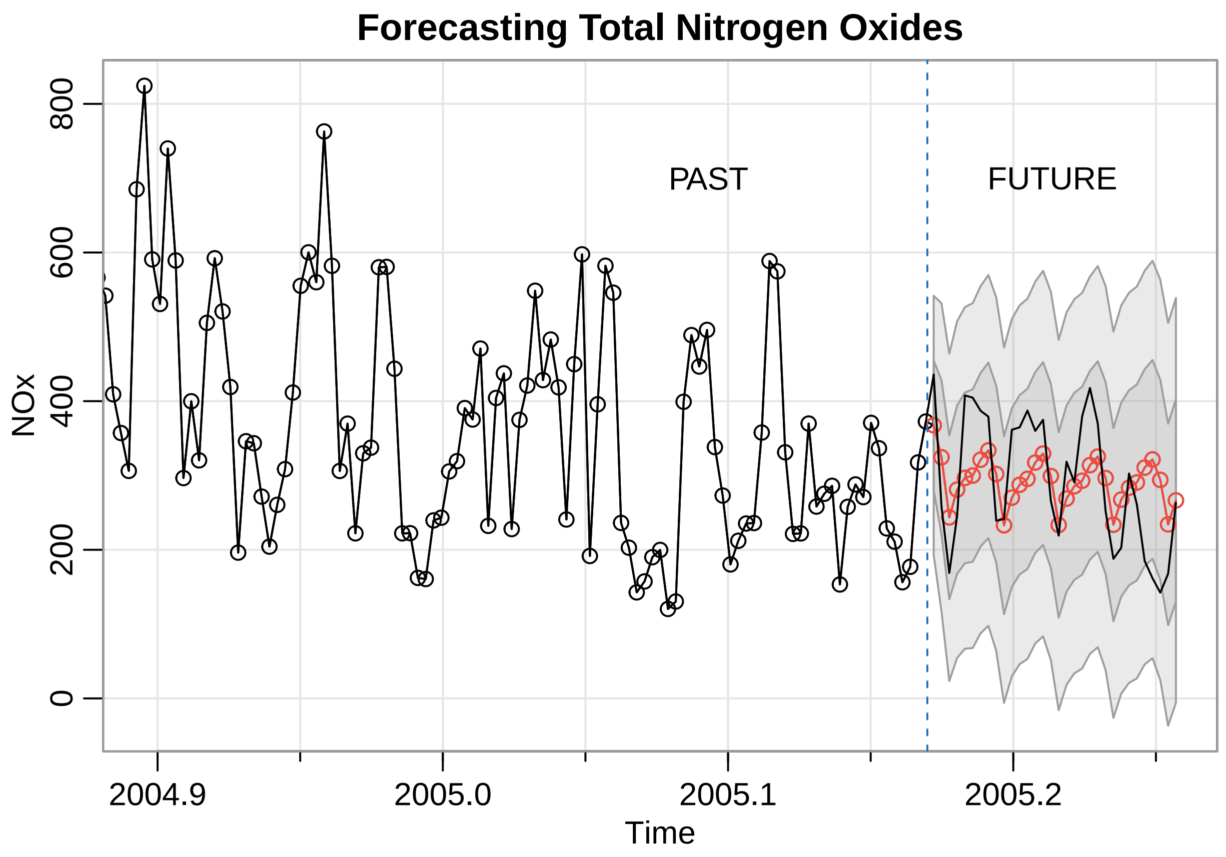
```r
text(decimal_date(as.Date("2005-02-04")), 22, "PAST")
text(decimal_date(as.Date("2005-03-20")), 22, "FUTURE")
abline(v=decimal_date(as.Date("2005-03-04")), lty=2, col=4)
lines(x)
```



```r
# Plotting the final month of time series for NOx
x <- ts(aqDaily$NOx.GT., start = decimal_date(as.Date("2004-03-10")), frequency = 365)
NOx<- window(x, start=decimal_date(as.Date("2004-03-10")),
            end=decimal_date(as.Date("2005-03-04")))
sarima.for(NOx, n.ahead = 32, p = 1, d = 1, q = 1, P = 2, D = 0, Q = 1, S = 7,
           plot.all=FALSE, main = "Forecasting Total Nitrogen Oxides")
```

```
## $pred
## Time Series:
## Start = 2005.172086234
## End = 2005.25701774085
## Frequency = 365
##  [1] 367.5042 324.4783 243.6187 280.9625 296.7014 299.8619 321.1545 333.7442
##  [9] 301.9457 232.9976 270.0982 287.5370 295.4433 317.4236 329.4645 299.1799
## [17] 233.3639 268.8307 285.4720 292.9278 313.8822 325.3889 296.5102 233.7329
## [25] 267.5770 283.4619 290.5853 310.5843 321.5689 294.0248 234.1406 266.4355
##
## $se
## Time Series:
## Start = 2005.172086234
## End = 2005.25701774085
## Frequency = 365
##  [1]  87.16773 103.60285 110.14205 113.23358 114.92567 115.99189 116.75518
##  [8] 118.01557 118.89944 119.59077 120.17813 120.70694 121.20166 121.67593
## [15] 122.96200 123.88746 124.62613 125.26288 125.84173 126.38661 126.91095
## [22] 128.20231 129.15613 129.93281 130.61182 131.23486 131.82481 132.39455
## [29] 133.69377 134.67463 135.48655 136.20451
```

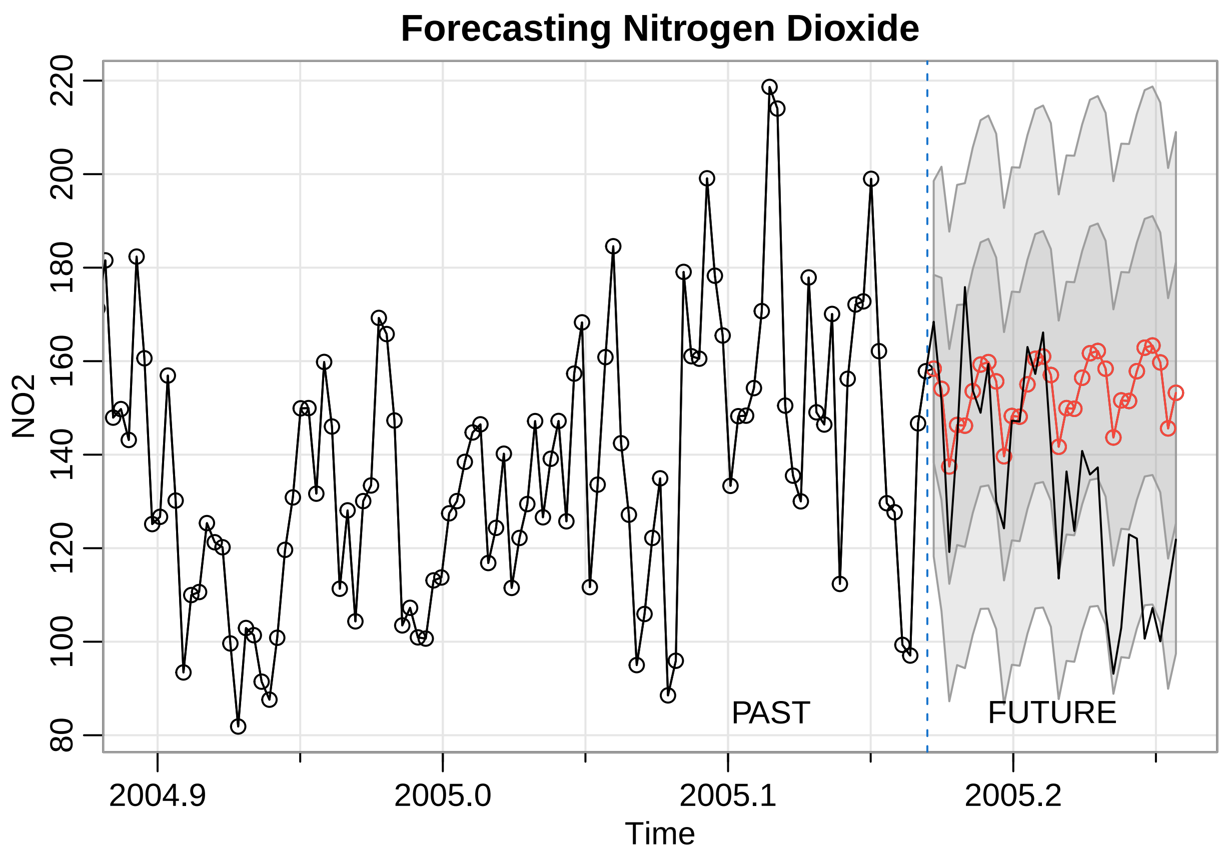
```r
text(decimal_date(as.Date("2005-02-04")), 700, "PAST")
text(decimal_date(as.Date("2005-03-20")), 700, "FUTURE")
abline(v=decimal_date(as.Date("2005-03-04")), lty=2, col=4)
lines(x)
```



```r
# Plotting the final month of time series for NO2
x <- ts(aqDaily$NO2.GT., start = decimal_date(as.Date("2004-03-10")), frequency = 365)
NO2 <- window(x, start=decimal_date(as.Date("2004-03-10")),
              end=decimal_date(as.Date("2005-03-04")))
sarima.for(NO2, n.ahead = 32, p = 1, d = 1, q = 1, P = 2, D = 0, Q = 1, S = 7,
           plot.all=FALSE, main = "Forecasting Nitrogen Dioxide")
```

```
## $pred
## Time Series:
## Start = 2005.172086234
## End = 2005.25701774085
## Frequency = 365
##  [1] 158.3959 154.0786 137.4812 146.3565 146.2074 153.5760 159.2810 159.7990
##  [9] 155.6774 139.6607 148.2771 148.1280 155.0442 160.4965 160.9948 157.0453
## [17] 141.6880 149.9572 149.8196 156.4579 161.6929 162.1767 158.3948 143.6720
## [25] 151.6088 151.4829 157.8557 162.8825 163.3526 159.7316 145.6174 153.2355
##
## $se
## Time Series:
## Start = 2005.172086234
## End = 2005.25701774085
## Frequency = 365
##  [1] 20.08138 23.75429 25.11146 25.67489 25.93040 26.05793 26.12897 26.36211
##  [9] 26.48293 26.55313 26.59905 26.63261 26.65953 26.68273 26.83868 26.92735
## [17] 26.98413 27.02484 27.05700 27.08441 27.10907 27.26078 27.34964 27.40825
## [25] 27.45140 27.48621 27.51634 27.54376 27.69337 27.78327 27.84405 27.88977
```

```r
text(decimal_date(as.Date("2005-02-12")), 85, "PAST")
text(decimal_date(as.Date("2005-03-20")), 85, "FUTURE")
abline(v=decimal_date(as.Date("2005-03-04")), lty=2, col=4)
lines(x)
```



This model fits the least well at the end. This is because it takes a sudden dove down which is not predictable by seeing how the time series behaves before.

Model Evaluation

Table of AIC, AICc, and BIC for each model

```r
# Inputting the metrics previously saved
metrics <- c(coMod$AIC, coMod$AICc, coMod$BIC, benzeneMod$AIC, benzeneMod$AICc,
             benzeneMod$BIC, noxMod$AIC, noxMod$AICc, noxMod$BIC, no2Mod$AIC,
             no2Mod$AICc, no2Mod$BIC)
metrics <- round(metrics, 2)
tab <- matrix(metrics, ncol=3, byrow=TRUE)
colnames(tab) <- c('AIC','AICc','BIC')
rownames(tab) <- c('CO','Benzene', 'NOx', 'NO2')
tab <- as.table(tab)
tab
```

```
##            AIC  AICc   BIC
## CO        2.13  2.13  2.19
## Benzene  5.04  5.04  5.10
## NOx     11.77 11.77 11.84
## NO2      8.85  8.85  8.92
```

From this table, we can see that our SARMA model best fit for CO. In the end, the parameters were the same for CO and Benzene, while NOx and NO2 were identical. This can infer that CO and Benzene are similar in how they come to be in the air, but statements like this are best left to climate scientists.