Necessary Libraries

[43]:
```
# Install necessary packages if not already installed
#install.packages(c("dplyr", "ggplot2", "plotly", "shiny", "leaflet", "forecast", "zoo", "dygraphs"))
#install.packages("reshape2")
# Load libraries
library(dplyr)
library(ggplot2)
library(plotly)
library(shiny)
library(leaflet)
library(forecast)
library(zoo)
library(dygraphs)
library(reshape2)
```

Load The Dataset

[44]:
```
# Load the data
data1 <- read.csv("/kaggle/input/air-quality-data-in-india/city_day.csv")
```

[45]:
```
# View the first 10 rows of the dataset
head(data1, 10)
```

A data.frame: 10 × 16

	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket
	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
1	Ahmedabad	2015-01-01	NA	NA	0.92	18.22	17.15	NA	0.92	27.64	133.36	0.00	0.02	0.00	NA	
2	Ahmedabad	2015-01-02	NA	NA	0.97	15.69	16.46	NA	0.97	24.55	34.06	3.68	5.50	3.77	NA	
3	Ahmedabad	2015-01-03	NA	NA	17.40	19.30	29.70	NA	17.40	29.07	30.70	6.80	16.40	2.25	NA	
4	Ahmedabad	2015-01-04	NA	NA	1.70	18.48	17.97	NA	1.70	18.59	36.08	4.43	10.14	1.00	NA	
5	Ahmedabad	2015-01-05	NA	NA	22.10	21.42	37.76	NA	22.10	39.33	39.31	7.01	18.89	2.78	NA	
6	Ahmedabad	2015-01-06	NA	NA	45.41	38.48	81.50	NA	45.41	45.76	46.51	5.42	10.83	1.93	NA	
7	Ahmedabad	2015-01-07	NA	NA	112.16	40.62	130.77	NA	112.16	32.28	33.47	0.00	0.00	0.00	NA	
8	Ahmedabad	2015-01-08	NA	NA	80.87	36.74	96.75	NA	80.87	38.54	31.89	0.00	0.00	0.00	NA	
9	Ahmedabad	2015-01-09	NA	NA	29.16	31.00	48.00	NA	29.16	58.68	25.75	0.00	0.00	0.00	NA	
10	Ahmedabad	2015-01-10	NA	NA	NA	7.04	0.00	NA	NA	8.29	4.55	0.00	0.00	0.00	NA	

[46]:
```
tail(data1)
```

A data.frame: 6 × 16

	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket
	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
29526	Visakhapatnam	2020-06-26	7.63	32.27	5.91	23.27	17.19	11.15	0.46	6.87	19.90	1.45	5.37	1.45	47	Good
29527	Visakhapatnam	2020-06-27	15.02	50.94	7.68	25.06	19.54	12.47	0.47	8.55	23.30	2.24	12.07	0.73	41	Good
29528	Visakhapatnam	2020-06-28	24.38	74.09	3.42	26.06	16.53	11.99	0.52	12.72	30.14	0.74	2.21	0.38	70	Satisfactory
29529	Visakhapatnam	2020-06-29	22.91	65.73	3.45	29.53	18.33	10.71	0.48	8.42	30.96	0.01	0.01	0.00	68	Satisfactory
29530	Visakhapatnam	2020-06-30	16.64	49.97	4.05	29.26	18.80	10.03	0.52	9.84	28.30	0.00	0.00	0.00	54	Satisfactory
29531	Visakhapatnam	2020-07-01	15.00	66.00	0.40	26.85	14.05	5.20	0.59	2.10	17.05	NA	NA	NA	50	Good

+ Code + Markdown

Calculate the average AQI for each city

[47]:
```
# Calculate the average AQI for each city
avg_aqi_by_city <- data1 %>%
  group_by(City) %>%
  summarize(Average_AQI = mean(AQI, na.rm = TRUE))

# View the resulting dataset
print(avg_aqi_by_city)
```

```
# A tibble: 26 × 2
   City         Average_AQI
   <chr>              <dbl>
 1 Ahmedabad          452.
 2 Aizawl              34.8
 3 Amaravati           95.3
 4 Amritsar           120.
 5 Bengaluru           94.3
 6 Bhopal             133.
 7 Brajrajnagar       150.
 8 Chandigarh          96.5
 9 Chennai            115.
10 Coimbatore          73.0
# ℹ 16 more rows
```

[48]:
```
# Check the structure of the dataset
str(data1)
```

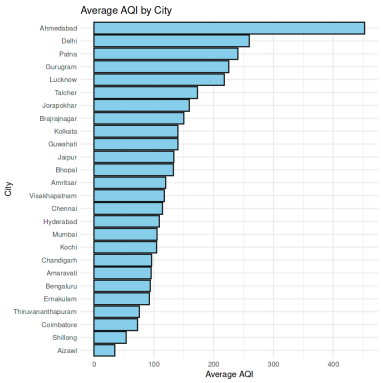
```
'data.frame':   29531 obs. of  16 variables:
 $ City      : chr  "Ahmedabad" "Ahmedabad" "Ahmedabad" "Ahmedabad" ...
 $ Date      : chr  "2015-01-01" "2015-01-02" "2015-01-03" "2015-01-04" ...
 $ PM2.5     : num  NA NA NA NA NA NA NA NA NA NA ...
 $ PM10      : num  NA NA NA NA NA NA NA NA NA NA ...
 $ NO        : num  0.92 0.97 17.4 1.7 22.1 ...
 $ NO2       : num  18.2 15.7 19.3 18.5 21.4 ...
 $ NOx       : num  17.1 16.5 29.7 18 37.8 ...
 $ NH3       : num  NA NA NA NA NA NA NA NA NA NA ...
 $ CO        : num  0.92 0.97 17.4 1.7 22.1 ...
 $ SO2       : num  27.6 24.6 29.1 18.6 39.3 ...
 $ O3        : num  133.4 34.1 30.7 36.1 39.3 ...
 $ Benzene   : num  0 3.68 6.8 4.43 7.01 5.42 0 0 0 0 ...
 $ Toluene   : num  0.02 5.5 16.4 10.14 18.89 ...
 $ Xylene    : num  0 3.77 2.25 1 2.78 1.93 0 0 0 0 ...
 $ AQI       : num  NA NA NA NA NA NA NA NA NA NA ...
 $ AQI_Bucket: chr  "" "" "" "" "" ...
```

Descriptive Statistics

[49]:
```
# Create the updated dataframe
average_aqi_data <- data.frame(
  City = c("Ahmedabad", "Aizawl", "Amaravati", "Amritsar", "Bengaluru", "Bhopal",
           "Brajrajnagar", "Chandigarh", "Chennai", "Coimbatore", "Delhi", "Ernakulam",
           "Gurugram", "Guwahati", "Hyderabad", "Jaipur", "Jorapokhar", "Kochi",
           "Kolkata", "Lucknow", "Mumbai", "Patna", "Shillong", "Talcher",
           "Thiruvananthapuram", "Visakhapatnam"),
  Average_AQI = c(452.12294, 34.76577, 95.29964, 119.92096, 94.31832, 132.82734,
                  150.28050, 96.49833, 114.50265, 73.02326, 259.48774, 92.35948,
                  225.12388, 140.11111, 109.20745, 133.67916, 159.25162, 104.28481,
                  140.56631, 217.97306, 105.35226, 240.78204, 53.79512, 172.88682,
                  75.87833, 117.2698)
)

# Display the updated dataframe
print(average_aqi_data)
```

```
                City Average_AQI
1          Ahmedabad   452.12294
2             Aizawl    34.76577
3          Amaravati    95.29964
4           Amritsar   119.92096
5          Bengaluru    94.31832
6             Bhopal   132.82734
7        Brajrajnagar   150.28050
8          Chandigarh    96.49833
9             Chennai   114.50265
10         Coimbatore    73.02326
11              Delhi   259.48774
12          Ernakulam    92.35948
13           Gurugram   225.12388
14           Guwahati   140.11111
15          Hyderabad   109.20745
16             Jaipur   133.67916
17         Jorapokhar   159.25162
18              Kochi   104.28481
19            Kolkata   140.56631
20            Lucknow   217.97306
21             Mumbai   105.35226
22              Patna   240.78204
23           Shillong    53.79512
24            Talcher   172.88682
25 Thiruvananthapuram    75.87833
26      Visakhapatnam   117.26980
```

[50]:
```
# Load necessary library
library(ggplot2)

# Create a bar plot for Average_AQI by City
ggplot(average_aqi_data, aes(x = reorder(City, Average_AQI), y = Average_AQI)) +
  geom_bar(stat = "identity", fill = "skyblue", color = "black") +
  coord_flip() +  # Flip coordinates to make it horizontal
  labs(title = "Average AQI by City", x = "City", y = "Average AQI") +
  theme_minimal()
```



[51]:
```
# Convert Date column to Date type
data1$Date <- as.Date(data1$Date, format = "%Y-%m-%d")
```

[52]:
```
# Check for null values in each column of the dataset
colSums(is.na(data1))
```

City: 0 **Date:** 0 **PM2.5:** 4598 **PM10:** 11140 **NO:** 3582 **NO2:** 3585 **NOx:** 4185 **NH3:** 10328 **CO:** 2059 **SO2:** 3854 **O3:** 4022 **Benzene:** 5623 **Toluene:** 8041 **Xylene:** 18109 **AQI:** 4681 **AQI_Bucket:** 0

[53]:
```
#install.packages("imputeTS")
library(forecast)
library(imputeTS)
data <- na_kalman(data1)  # Fills missing values with a state-space model (requires `imputeTS` package)
```

[54]:
```
# Check for null values in each column of the dataset
colSums(is.na(data))
```

City: 0 **Date:** 0 **PM2.5:** 0 **PM10:** 0 **NO:** 0 **NO2:** 0 **NOx:** 0 **NH3:** 0 **CO:** 0 **SO2:** 0 **O3:** 0 **Benzene:** 0 **Toluene:** 0 **Xylene:** 0 **AQI:** 0 **AQI_Bucket:** 0

[55]:
```
head(data)
```

A data.frame: 6 × 16

	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket
	<chr>	<date>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
1	Ahmedabad	2015-01-01	73.37076	141.5376	0.92	18.22	17.15	26.63971	0.92	27.64	133.36	0.00	0.02	0.00	209.7513	
2	Ahmedabad	2015-01-02	73.60067	141.5339	0.97	15.69	16.46	26.63973	0.97	24.55	34.06	3.68	5.50	3.77	211.7754	
3	Ahmedabad	2015-01-03	73.83058	141.5301	17.40	19.30	29.70	26.63975	17.40	29.07	30.70	6.80	16.40	2.25	213.7995	
4	Ahmedabad	2015-01-04	74.06050	141.5264	1.70	18.48	17.97	26.63977	1.70	18.59	36.08	4.43	10.14	1.00	215.8236	
5	Ahmedabad	2015-01-05	74.29041	141.5227	22.10	21.42	37.76	26.63979	22.10	39.33	39.31	7.01	18.89	2.78	217.8477	
6	Ahmedabad	2015-01-06	74.52032	141.5190	45.41	38.48	81.50	26.63981	45.41	45.76	46.51	5.42	10.83	1.93	219.8718	

Extracting Observatios Of Delhi For Further Analysis

[56]:
```
library(dplyr)

data <- data %>%
  filter(City == "Delhi")
```

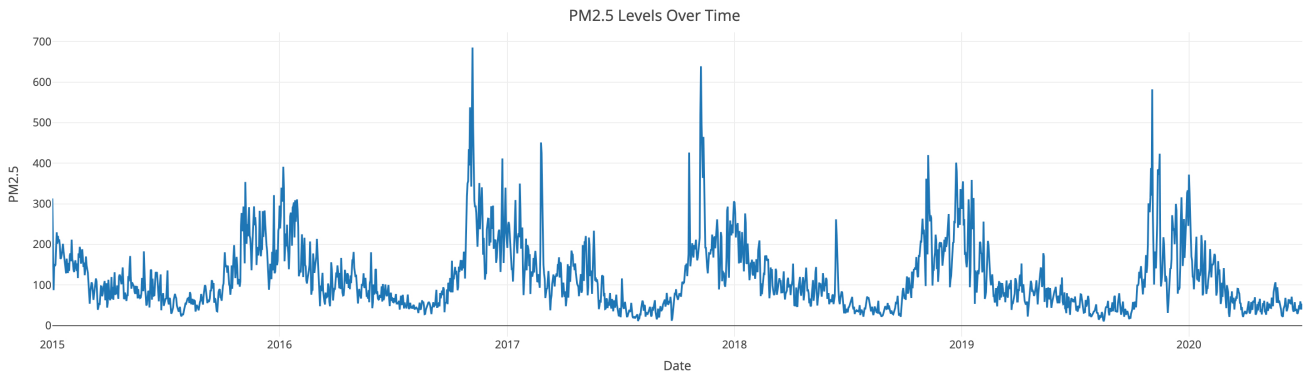
[57]:
```
head(data)
```

A data.frame: 6 × 16

	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket
	<chr>	<date>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
1	Delhi	2015-01-01	313.22	607.98	69.16	36.39	110.59	33.85	15.20	9.25	41.68	14.36	24.86	9.84	472	Severe
2	Delhi	2015-01-02	186.18	269.55	62.09	32.87	88.14	31.83	9.54	6.65	29.97	10.55	20.09	4.29	454	Severe
3	Delhi	2015-01-03	87.18	131.90	25.73	30.31	47.95	69.55	10.61	2.65	19.71	3.91	10.23	1.99	143	Moderate
4	Delhi	2015-01-04	151.84	241.84	25.01	36.91	48.62	130.36	11.54	4.63	25.36	4.26	9.71	3.34	319	Very Poor
5	Delhi	2015-01-05	146.60	219.13	14.01	34.92	38.25	122.88	9.20	3.33	23.20	2.80	6.21	2.96	325	Very Poor
6	Delhi	2015-01-06	149.58	252.10	17.21	37.84	42.46	134.97	9.44	3.66	26.83	3.63	7.35	3.47	318	Very Poor

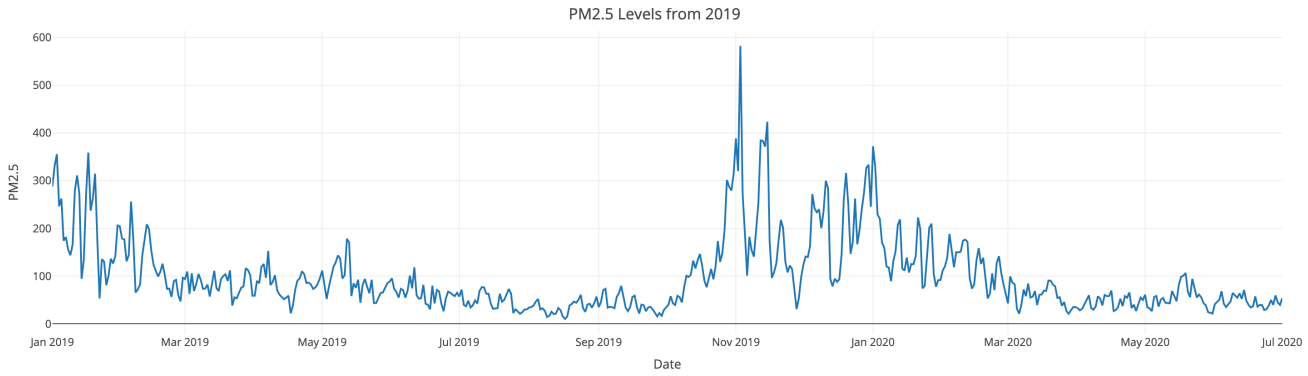
+ Code + Markdown

[58]:
```
# Assuming your data frame is named `delhi_data` and has columns "Date" and "PM2.5"
fig <- plot_ly(data, x = ~Date, y = ~PM2.5, type = 'scatter', mode = 'lines') %>%
  layout(title = "PM2.5 Levels Over Time",
         xaxis = list(title = "Date"),
         yaxis = list(title = "PM2.5"))

# Display the plot
fig
```



+ Code + Markdown

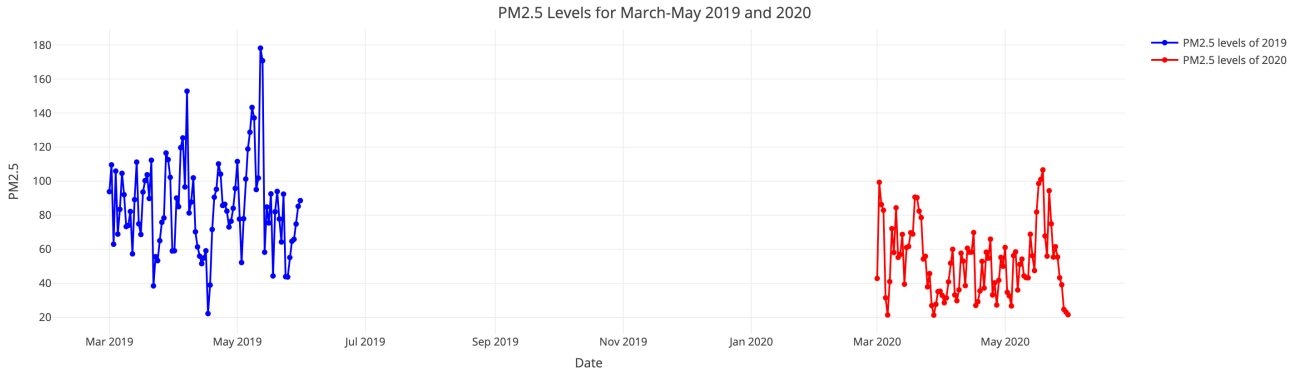
[59]:
```
# Assuming your data frame is named `delhi_data` and has columns "Date" and "PM2.5"
# Convert `Date` to Date format if it isn't already
data$Date <- as.Date(data$Date)

# Filter data for dates starting from 2019
df_year_19_20 <- data %>% filter(Date >= as.Date("2019-01-01"))

# Create an interactive line plot for PM2.5
fig <- plot_ly(df_year_19_20, x = ~Date, y = ~PM2.5, type = 'scatter', mode = 'lines') %>%
  layout(title = "PM2.5 Levels from 2019",
         xaxis = list(title = "Date"),
         yaxis = list(title = "PM2.5"))

# Display the plot
fig
```



Comparison of PM 2.5 between 2019 and 2020

[60]:
```
# Filter data for March to May 2019
Mar_may_2019 <- data %>%
  filter(Date >= as.Date("2019-03-01") & Date <= as.Date("2019-05-31"))

# Filter data for March to May 2020
Mar_may_2020 <- data %>%
  filter(Date >= as.Date("2020-03-01") & Date <= as.Date("2020-05-31"))
```

[61]:
```
# Create the interactive plot
fig <- plot_ly()

# Add trace for PM2.5 levels in 2019
fig <- fig %>%
  add_trace(x = ~Mar_may_2019$Date, y = ~Mar_may_2019$PM2.5,
            type = 'scatter', mode = 'lines+markers',
            name = 'PM2.5 levels of 2019',
            line = list(color = 'blue'),
            marker = list(color = 'blue'))

# Add trace for PM2.5 levels in 2020
fig <- fig %>%
  add_trace(x = ~Mar_may_2020$Date, y = ~Mar_may_2020$PM2.5,
            type = 'scatter', mode = 'lines+markers',
            name = 'PM2.5 levels of 2020',
            line = list(color = 'red'),
            marker = list(color = 'red'))

# Set layout for titles and axes
fig <- fig %>%
  layout(title = "PM2.5 Levels for March-May 2019 and 2020",
         xaxis = list(title = "Date"),
         yaxis = list(title = "PM2.5"),
         hovermode = "compare") # Enable hover comparison

# Display the plot
fig
```



[62]:
```
mean_PM2.5 <- mean(Mar_may_2019$PM2.5, na.rm = TRUE)
mean_PM2.5
```

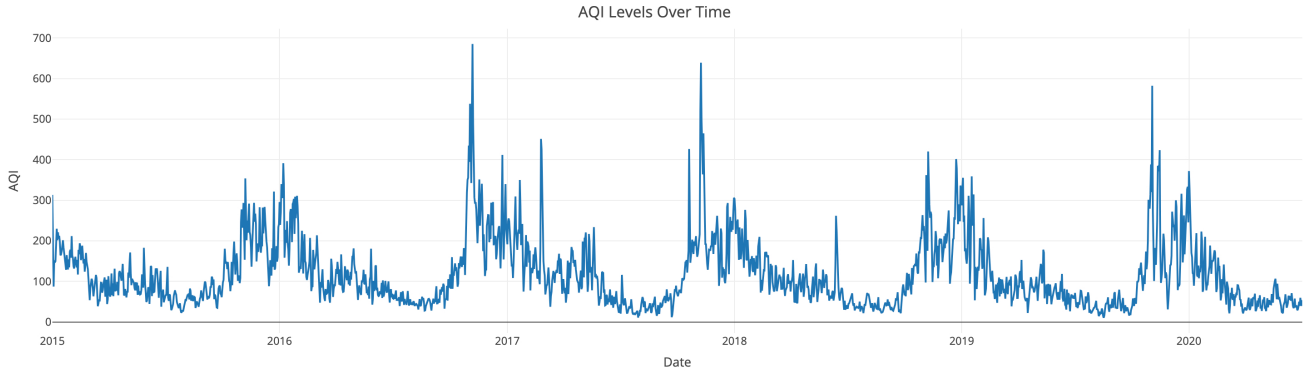
85.4369565217391

[63]:
```
mean_PM2.5 <- mean(Mar_may_2020$PM2.5, na.rm = TRUE)
mean_PM2.5
```

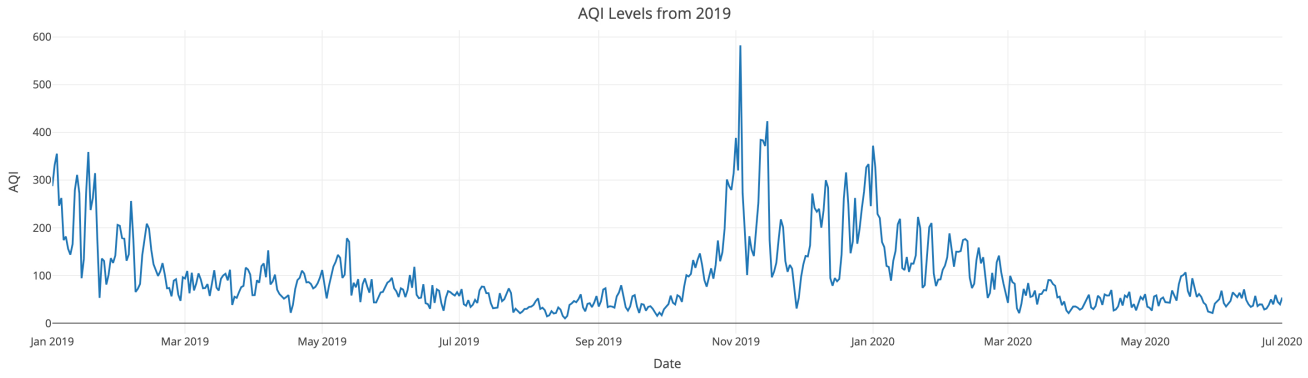
52.7153260869565

Comparison of AQI between 2019 and 2020

[64]:
```
# Assuming your data frame is named `delhi_data` and has columns "Date" and "PM2.5"
fig <- plot_ly(data, x = ~Date, y = ~PM2.5, type = 'scatter', mode = 'lines') %>%
  layout(title = "AQI Levels Over Time",
         xaxis = list(title = "Date"),
         yaxis = list(title = "AQI"))

# Display the plot
fig
```



[65]:
```
# Assuming your data frame is named `delhi_data` and has columns "Date" and "PM2.5"
# Convert `Date` to Date format if it isn't already
data$Date <- as.Date(data$Date)

# Filter data for dates starting from 2019
df_year_19_20 <- data %>% filter(Date >= as.Date("2019-01-01"))

# Create an interactive line plot for PM2.5
fig <- plot_ly(df_year_19_20, x = ~Date, y = ~PM2.5, type = 'scatter', mode = 'lines') %>%
  layout(title = "AQI Levels from 2019",
         xaxis = list(title = "Date"),
         yaxis = list(title = "AQI"))

# Display the plot
fig
```



[66]:
```
# Create the interactive plot
fig <- plot_ly()

# Add trace for PM2.5 levels in 2019
fig <- fig %>%
  add_trace(x = ~Mar_may_2019$Date, y = ~Mar_may_2019$PM2.5,
            type = 'scatter', mode = 'lines+markers',
            name = 'AQI levels of 2019',
            line = list(color = 'blue'),
            marker = list(color = 'blue'))

# Add trace for PM2.5 levels in 2020
fig <- fig %>%
  add_trace(x = ~Mar_may_2020$Date, y = ~Mar_may_2020$PM2.5,
            type = 'scatter', mode = 'lines+markers',
```

```
            name = 'AQI levels of 2020',
            line = list(color = 'red'),
            marker = list(color = 'red'))

# Set layout for titles and axes
fig <- fig %>%
  layout(title = "AQI Levels for March-May 2019 and 2020",
         xaxis = list(title = "Date"),
         yaxis = list(title = "AQI"),
         hovermode = "compare") # Enable hover comparison

# Display the plot
fig
```

[67]:
```
mean_AQI <- mean(Mar_may_2019$AQI, na.rm = TRUE)
mean_AQI
```

217.510869565217

[68]:
```
mean_AQI <- mean(Mar_may_2020$PM2.5, na.rm = TRUE)
mean_AQI
```

52.7153260869565

Find The Correlations Among All Variables

[69]:
```
data2 <- data[, !(colnames(data) %in% c("City", "Date","AQI_Bucket","NH3"))]
```

[70]:
```
head(data2)
```

A data.frame: 6 × 12

	PM2.5	PM10	NO	NO2	NOx	CO	SO2	O3	Benzene	Toluene	Xylene	AQI
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	313.22	607.98	69.16	36.39	110.59	15.20	9.25	41.68	14.36	24.86	9.84	472
2	186.18	269.55	62.09	32.87	88.14	9.54	6.65	29.97	10.55	20.09	4.29	454
3	87.18	131.90	25.73	30.31	47.95	10.61	2.65	19.71	3.91	10.23	1.99	143
4	151.84	241.84	25.01	36.91	48.62	11.54	4.63	25.36	4.26	9.71	3.34	319
5	146.60	219.13	14.01	34.92	38.25	9.20	3.33	23.20	2.80	6.21	2.96	325
6	149.58	252.10	17.21	37.84	42.46	9.44	3.66	26.83	3.63	7.35	3.47	318

[71]:
```
# Calculate the correlation matrix
cor_matrix <- cor(data2)
cor_matrix
```

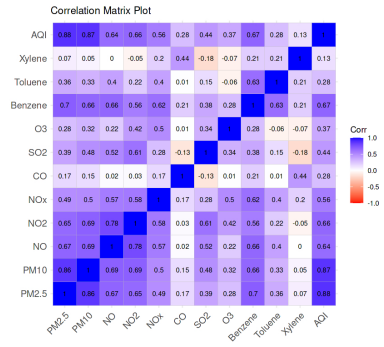
A matrix: 12 × 12 of type dbl

	PM2.5	PM10	NO	NO2	NOx	CO	SO2	O3	Benzene	Toluene	Xylene	AQI
PM2.5	1.00000000	0.85502340	0.6681459706	0.64571591	0.4876480	0.171657023	0.3871260	0.282945351	0.6983862	0.359506189	0.0725874690	0.8822743
PM10	0.85502340	1.00000000	0.6903862275	0.68886995	0.4952298	0.147262136	0.4812983	0.319320300	0.6611345	0.325873378	0.0527239292	0.8705569
NO	0.66814597	0.69038623	1.0000000000	0.77518837	0.5734969	0.022071364	0.5223910	0.216666432	0.6558356	0.402664233	-0.0006610109	0.6396242
NO2	0.64571591	0.68886995	0.7751883750	1.00000000	0.5790411	0.030115343	0.6050619	0.418415434	0.5618120	0.217717131	-0.0514072376	0.6579999
NOx	0.48764797	0.49522983	0.5734969140	0.57904114	1.0000000	0.172149441	0.2800731	0.501284763	0.6204333	0.397076836	0.2000284676	0.5601198
CO	0.17165702	0.14726214	0.0220713644	0.03011534	0.1721494	1.000000000	-0.1260051	0.005695311	0.2063620	0.007876802	0.4369447616	0.2833045
SO2	0.38712597	0.48129829	0.5223909936	0.60506193	0.2800731	-0.126005072	1.0000000	0.343295146	0.3789430	0.149636084	-0.1804604544	0.4363885
O3	0.28294535	0.31932030	0.2166664325	0.41841543	0.5012848	0.005695311	0.3432951	1.000000000	0.2791174	-0.056842164	-0.0679973282	0.3703464
Benzene	0.69838621	0.66113445	0.6558356201	0.56181203	0.6204333	0.206362022	0.3789430	0.279117395	1.0000000	0.632059671	0.2104793849	0.6729325
Toluene	0.35950619	0.32587338	0.4026642327	0.21771713	0.3970768	0.007876802	0.1496361	-0.056842164	0.6320597	1.000000000	0.2147109779	0.2845041
Xylene	0.07258747	0.05272393	-0.0006610109	-0.05140724	0.2000285	0.436944762	-0.1804605	-0.067997328	0.2104794	0.214710978	1.0000000000	0.1268156
AQI	0.88227433	0.87055688	0.6396242301	0.65799995	0.5601198	0.283304534	0.4363885	0.370346396	0.6729325	0.284504098	0.1268156194	1.0000000

[72]:
```
# Install the package if you haven't already
#install.packages("ggcorrplot")
```

```r
# Load the library
library(ggcorrplot)
```

[73]:
```r
# Plot the correlation matrix
ggcorrplot(cor_matrix,
           method = "square",        # Choose "circle" or "square"
           lab = TRUE,               # Display the correlation coefficients
           lab_size = 3,             # Adjust label size
           colors = c("red", "white", "blue"),  # Color gradient
           title = "Correlation Matrix Plot",
           ggtheme = theme_minimal())
```



[74]:
```r
# Set row and column names
colnames(cor_matrix) <- rownames(cor_matrix) <- c("PM2.5", "PM10", "NO", "NO2", "NOx", "CO", "SO2", "O3", "Benzene", "Toluene", "Xylene", "AQI")

# Convert to long format and filter for high correlations
cor_df <- as.data.frame(melt(cor_matrix))
colnames(cor_df) <- c("Var1", "Var2", "Correlation")

# Filter for high correlations (e.g., abs > 0.7) and remove self-correlations
high_corr <- cor_df %>%
  filter(Var1 != Var2) %>%               # Exclude self-correlations
  filter(abs(Correlation) > 0.7) %>%     # Filter for high correlations
  arrange(desc(abs(Correlation)))        # Sort by absolute correlation value

# Display the result
high_corr
```

A data.frame: 8 × 3

Var1	Var2	Correlation
<fct>	<fct>	<dbl>
AQI	PM2.5	0.8822743
PM2.5	AQI	0.8822743
AQI	PM10	0.8705569
PM10	AQI	0.8705569
PM10	PM2.5	0.8550234
PM2.5	PM10	0.8550234
NO2	NO	0.7751884
NO	NO2	0.7751884

Taking PM-2.5 for Forecasting

[75]:
```r
y = data[,3] #PM2.5
```

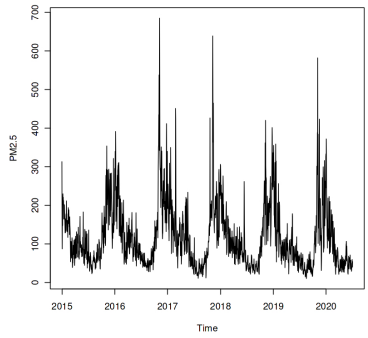
[76]:
```r
y
```

313.22 · 186.18 · 87.18 · 151.84 · 146.6 · 149.58 · 217.87 · 229.9 · 201.66 · 221.02 · 205.41 · 212.41 · 197.61 · 164.39 · 166.19 · 174.98 · 196.46 · 201.51 · 183.35 · 165.63 · 159.54 · 143.68 · 128.73 · 164.98 · 148.59 · 129.34 · 154.2 · 134.6 · 173.13 · 178.04 · 158.59 · 211.59 · 163.64 · 157.71 · 137.26 · 131.76 · 151.6 · 160.75 · 145.34 · 134.67 · 153.07 · 116.91 · 177.98 · 167.01 · 193.96 · 184.06 · 152.62 · 161.27 · 188.2 · 159.97 · 144.12 · 155.38 · 124.1 · 141.39 · 170.16 · 148.19 · 140.78 · 130.42 · 105.86 · 71.83 · 53.71 · 83.49 · 91.07 · 106.57 · 101.75 · 77.57 · 63.24 · 76.62 · 86.7 · 104.29 · 115.93 · 108.66 · 67.94 · 38.89 · 45.39 · 55.55 · 50.59 · 69.51 · 96.62 · 96.28 · 72.2 · 78.47 · 91.87 · 107.26 · 109.78 · 62.27 · 102.22 · 104.36 · 44.19 · 55.66 · 112.86 · 78.51 · 66.52 · 62.56 · 78.61 · 119.7 · 66.09 · 95.34 · 63.57 · 96.56 · 131.39 · 75.35 · 86.49 · 77.07 · 98.79 · 81.81 · 65.78 · 119.82 · 125.53 · 121.51 · 114.5 · 101.83 · 118.57 · 143.22 · 106.76 · 65.36 · 67.08 · 81.37 · 63.76 · 60.56 · 76.1 · 72.66 · 121.94 · 108.98 · 145.99 · 171.33 · 117.76 · 90.66 · 106.4 · 101.21 · 97.57 · 96.34 · 83.42 · 78.06 · 97.87 · 77.52 · 97.16 · 66.85 · 89.85 · 67.41 · 75.54 · 67.91 · 69.02 · 79.1 · 117.27 · 96.09 · 84.71 · 182.76 · 106.61 · 92.68 · 73.62 · 49.42 · 52.51 · 65.28 · 85.45 · 83.02 · 44.54 · 85.03 · 137.7 · 136.05 · 63.77 · 115.57 · 83.29 · 73.7 · 74.56 · 106.11 · 132.21 · 123.56 · 132.21 · 110.65 · 80.31 · 59.73 · 85.4 · 101.99 · 126.07 · 37.59 · 76 · 69.46 · 79.84 · 48.5 · 53.79 · 57.99 · 67.12 · 65.25 · 88.75 · 135.92 · 60.27 · 52.2 · 67.92 · 65.95 · 47.68 · 29.08 · 32.29 · 36.68 · 44.48 · 63.79 · 77.2 · 77.99 · 71.11 · 67.39 · ⋯ · 94.67 · 88.05 · 93.07 · 144 · 261.16 · 316.09 · 249.87 · 146.81 · 171.47 · 262.69 · 166.71 · 196.97 · 240.07 · 275.6 · 326.79 · 333.43 · 245.53 · 372.14 · 327.04 · 228.9 · 220.6 · 170.21 · 159.64 · 120.31 · 117.98 · 89.26 · 130.82 · 151.1 · 207.23 · 218.98 · 115.62 · 111.57 · 139.06 · 107.65 · 125.3 · 125.04 · 142.18 · 222.81 · 198.88 · 74.45 · 79.83 · 136.72 · 201.2 · 209.97 · 104.45 · 77.95 · 91.56 · 92.04 · 112.13 · 120.86 · 138.5 · 188.42 · 150.08 · 118.57 · 150.15 · 149.67 · 151.43 · 174.4 · 176.57 · 172.12 · 95.72 · 73.84 · 83.07 · 132.63 · 158.69 · 125.53 · 138.46 · 98.84 · 53.11 · 63.62 · 105.71 · 70.59 · 127.75 · 141.81 · 107.28 · 83.48 · 60.01 · 42.83 · 99.34 · 86.26 · 82.9 · 31.43 · 21.32 · 40.96 · 72.1 · 58.04 · 84.35 · 55.19 · 56.92 · 68.72 · 39.41 · 60.99 · 61.61 · 69.7 · 68.93 · 90.62 · 90.31 · 82.39 · 78.61 · 54.26 · 55.85 · 37.87 · 45.69 · 26.89 · 21.23 · 27.59 · 35.08 · 35.31 · 32.88 · 28.51 · 31.36 · 40.79 · 51.79 · 59.91 · 33.12 · 29.7 · 36.13 · 57.67 · 53.03 · 38.57 · 60.6 · 58.22 · 58.31 · 69.8 · 26.93 · 29.09 · 35.49 · 52.86 · 37.17 · 58.25 · 54.71 · 65.91 · 33.09 · 40.24 · 27.18 · 41.75 · 55.18 · 49.96 · 61.04 · 34.67 · 32.56 · 26.63 · 56.24 · 58.44 · 36.02 · 51.09 · 54.18 · 44.3 · 43.26 · 43.19 · 68.8 · 56.17 · 47.44 · 81.87 · 98.63 · 101.02 · 106.63 · 67.8 · 55.92 · 94.37 · 74.92 · 55.37 · 61.46 · 55.38 · 43.22 · 39.07 · 24.62 · 23.09 · 21.51 · 41.05 · 45.9 · 50.62 · 68.24 · 42.43 · 35.14 · 39.99 · 46.9 · 64.77 · 58.9 · 54.69 · 64 · 52.36 · 71.08 · 48.69 · 39.87 · 34.35 · 36.84 · 57.6 · 35.96 · 39.7 · 39.47 · 29.02 · 30.91 · 38.37 · 50.01 · 39.8 · 59.52 · 44.86 · 39.8 · 54.01

[77]:
```r
#pm25_data
```

[78]:
```
RF = ts(y, frequency = 365, start = c(2015, 1, 1))
```

[79]:
```
plot(RF,xlab="Time",ylab="PM2.5")
```



[80]:
```
library(tseries)
adf.test(RF) ##p-value = 0.01, stationary
kpss.test(RF) ##  p-value = 0.1, stationary
```

```
Warning message in adf.test(RF):
“p-value smaller than printed p-value”
```

```
        Augmented Dickey-Fuller Test

data:  RF
Dickey-Fuller = -4.8556, Lag order = 12, p-value = 0.01
alternative hypothesis: stationary

        KPSS Test for Level Stationarity

data:  RF
KPSS Level = 0.56828, Truncation lag parameter = 8, p-value = 0.02629
```

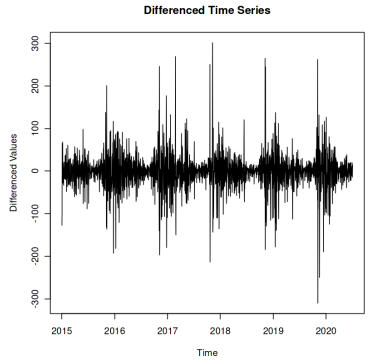
Augmented Dickey-Fuller (ADF) Test: The ADF test is commonly used to detect the presence of a unit root, where the null hypothesis assumes that the series is non-stationary. Here, the ADF test statistic is -4.8556 with a p-value of 0.01, which is less than the typical significance level of 0.05. This result allows us to reject the null hypothesis, suggesting that the time series is stationary. Therefore, according to the ADF test, RF does not have a unit root and is likely stationary.

KPSS Test for Level Stationarity: The KPSS test complements the ADF test by testing for level stationarity, where the null hypothesis assumes that the series is stationary around a level or trend. In this case, the KPSS test statistic is 0.56828, with a p-value of 0.02629, which is below the significance level of 0.05. This leads us to reject the null hypothesis, indicating that the time series may not be stationary in terms of level.

Perform first differencing

> A stationary time series is one where the statistical properties — like mean, variance, and autocorrelation — do not change over time. In other words, the series doesn't trend upward or downward, and the fluctuations around the mean are consistent.

[81]:
```
# Perform first differencing
RF <- diff(RF)

# Plot the differenced time series
plot(RF, main = "Differenced Time Series", ylab = "Differenced Values", xlab = "Time")
```



[82]:
```
library(tseries)
adf.test(RF) ##p-value = 0.01, stationary
kpss.test(RF) ##  p-value = 0.1, stationary
```

```
Warning message in adf.test(RF):
“p-value smaller than printed p-value”
```

```
        Augmented Dickey-Fuller Test

data:  RF
Dickey-Fuller = -17.561, Lag order = 12, p-value = 0.01
alternative hypothesis: stationary
```

```
Warning message in kpss.test(RF):
“p-value greater than printed p-value”
```

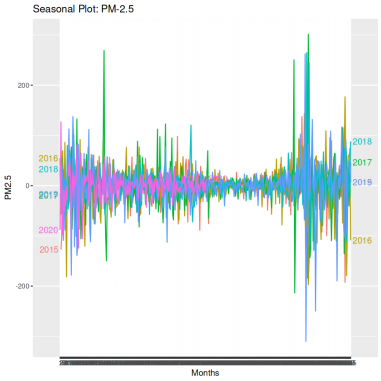
```
        KPSS Test for Level Stationarity

data:  RF
KPSS Level = 0.012556, Truncation lag parameter = 8, p-value = 0.1
```

> Augmented Dickey-Fuller (ADF) Test:

The ADF test checks for the presence of a unit root, with the null hypothesis being that the series is non-stationary. Here, the test statistic is -17.561, which is a highly negative value, indicating strong evidence against the null hypothesis. The printed p-value is shown as 0.01, but the warning message “p-value smaller than printed p-value” suggests that the actual p-value is even lower than 0.01. This strong result leads us to reject the null hypothesis, meaning the series is likely stationary according to the ADF test. KPSS Test for Level Stationarity:

The KPSS test complements the ADF test by checking for level stationarity, where the null hypothesis assumes that the series is stationary. The KPSS test statistic is 0.012556, which is very low, and the printed p-value is 0.1. The warning “p-value greater than printed p-value” implies that the actual p-value is above 0.1, indicating that there is not enough evidence to reject the null hypothesis. Thus, the KPSS test suggests that the series is stationary around a level.

[83]:
```
library(forecast)

# Seasonal Plots
ggseasonplot(RF, year.labels = TRUE, year.labels.left = TRUE) +
  ggtitle("Seasonal Plot: PM-2.5") +
  ylab("PM2.5") +
  xlab("Months")
```



Final Forecasting for the next 5 years

[88]:
```
#install.packages("tidyverse")
#install.packages("lubridate")
#install.packages("prophet")
#install.packages("bsts")
#install.packages("tensorflow")
#install.packages("keras")
```

```
# Load necessary libraries
library(tidyverse)
library(lubridate)
library(forecast)
library(prophet)
library(bsts)
library(xgboost)
library(keras)
library(tensorflow)

# Use stlf() to perform ETS with seasonal decomposition for 5-year (1825 days) forecast
ets_forecast <- stlf(RF, h=1825, s.window="periodic", method="ets")
```

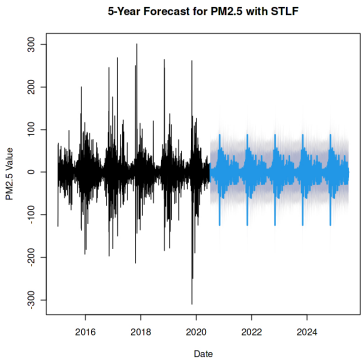
```
Loading required package: BoomSpikeSlab

Loading required package: Boom


Attaching package: ‘Boom’


The following object is masked from ‘package:stats’:

    rWishart


Attaching package: ‘BoomSpikeSlab’


The following object is masked from ‘package:stats’:

    knots


Loading required package: xts
```

```
######################### Warning from 'xts' package #########################
#                                                                            #
# The dplyr lag() function breaks how base R's lag() function is supposed to #
# work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or      #
# source() into this session won't work correctly.                           #
#                                                                            #
# Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
# conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop          #
# dplyr from breaking base R's lag() function.                               #
#                                                                            #
# Code in packages is not affected. It's protected by R's namespace mechanism #
# Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning. #
#                                                                            #
##############################################################################


Attaching package: ‘xts’


The following object is masked from ‘package:leaflet’:

    addLegend


The following objects are masked from ‘package:dplyr’:

    first, last


Attaching package: ‘bsts’


The following object is masked from ‘package:BoomSpikeSlab’:

    SuggestBurn


Attaching package: ‘xgboost’


The following object is masked from ‘package:plotly’:

    slice


The following object is masked from ‘package:dplyr’:

    slice
```

+ Code + Markdown

[87]:

```
# Plot the forecast
plot(ets_forecast, main="5-Year Forecast for PM2.5 with STLF", ylab="PM2.5 Value", xlab="Date")
```



This graph shows a 5-year forecast for PM2.5 values using the STLF (Seasonal and Trend decomposition using LOESS with ETS) model, similar to the previous one but with some distinct characteristics.

Historical Data (in black):

The black line represents historical PM2.5 values from 2015 to 2020. The data appears highly volatile, with frequent spikes and dips, indicating significant fluctuations in PM2.5 levels. The values range widely, occasionally reaching above 200 and dropping below -200, which might suggest either extreme pollution events or data irregularities. Forecasted Values (in blue):

The blue line represents the predicted PM2.5 values from 2020 to 2025. The forecast maintains a repeating seasonal pattern with regular peaks and troughs, capturing the expected cyclical behavior of PM2.5 levels. Prediction Intervals (shaded regions):

The gray shaded areas represent the 80% and 95% prediction intervals. Unlike the previous graph, the intervals here remain relatively stable, suggesting a more consistent level of uncertainty throughout the 5-year forecast. The confidence intervals are tighter compared to the previous graph, possibly indicating the model's higher confidence in its ability to capture seasonal patterns with less variability in the forecast. Long-Term Trend:

The forecasted values are centered around a neutral baseline without a clear upward or downward trend. The model suggests that while seasonal fluctuations will continue, there's no expected long-term increase or decrease in PM2.5 levels. In summary, this forecast displays a stable, seasonally cyclical pattern with consistent prediction intervals. The model appears to account for the seasonal variations but maintains relatively low uncertainty in the forecasted range, implying moderate confidence in the predicted seasonal pattern.

THANK YOU