STAT 443 Project (Final Version) - Group 4

Name

Wasay Siddiqui / Lavanya Upadhyaya / Junseok Yang / Mengjia Zeng

Import packages

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.0      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
```

Dataset Description

No: Row number year: Year of data in this row month: Month of data in this row day: Day of data in this row hour: Hour of data in this row PM2.5: PM2.5 concentration (ug/m^3) PM10: PM10 concentration (ug/m^3) SO2: SO2 concentration (ug/m^3) NO2: NO2 concentration (ug/m^3) CO: CO concentration (ug/m^3) O3: O3 concentration (ug/m^3) TEMP: Temperature (degree Celsius) PRES: Pressure (hPa) DEWP: Dew point temperature (degree Celsius) RAIN: Precipitation (mm) wd: Wind direction WSPM: Wind speed (m/s) station: Name of the air-quality monitoring site

Import dataset

```
df = read_csv("PRSA_Data_Gucheng_20130301-20170228.csv")
```

```
## Rows: 35064 Columns: 18
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): wd, station
## dbl (16): No, year, month, day, hour, PM2.5, PM10, SO2, NO2, CO, O3, TEMP, P...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
head(df)
```

```
## # A tibble: 6 x 18
##      No  year month   day  hour PM2.5  PM10   SO2   NO2    CO    O3  TEMP  PRES
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     1  2013     3     1     0     6    18     5    NA   800    88   0.1 1021.
## 2     2  2013     3     1     1     6    15     5    NA   800    88  -0.3 1022.
## 3     3  2013     3     1     2     5    18    NA    NA   700    52  -0.7 1022.
## 4     4  2013     3     1     3     6    20     6    NA    NA    NA  -1   1023.
## 5     5  2013     3     1     4     5    17     5    NA   600    73  -1.3 1023 
## 6     6  2013     3     1     5     4    11     3    NA   700    87  -1.8 1024.
## # ... with 5 more variables: DEWP <dbl>, RAIN <dbl>, wd <chr>, WSPM <dbl>,
## #   station <chr>
```

```r
# Convert 'wd' into a factor variable for later data manipulation
df$wd = as.factor(df$wd)
```

Dataset Size

```r
dim(df)
```

```
## [1] 35064    18
```

There are total 35064 rows and 18 attributes in this dataset.

Missing Values

```r
# Total number of NA values
sum(is.na(df))
```

```
## [1] 4728
```

```r
# Number of NA values in each attribute
colSums(is.na(df))
```

```
##      No    year   month     day    hour   PM2.5    PM10     SO2     NO2      CO
##       0       0       0       0       0     646     381     507     668    1401
##      O3    TEMP    PRES    DEWP    RAIN      wd    WSPM station
##     729      51      50      51      43     159      42       0
```

```r
# Number of rows with NA values
missing_val = length(unique(which(is.na(df), arr.ind = TRUE)[, 1]))

# Proportion of missing values in the dataset
prop = missing_val / dim(df)[1]
prop
```

```
## [1] 0.07300935
```

Considering that the proportion of missing values in the dataset is about 7.3%, which is pretty low, we could simply drop them all. However, considering our main focus is 'PM2.5', we are going to only drop rows with 'PM2.5' missing value for right now and we may do additional manipulation later.

Double-check of dropping missing values

```r
# Only remove rows with NA value in 'PM2.5' column
df_new = df[!is.na(df$PM2.5), ]

dim(df_new)
```

```
## [1] 34418    18
```

```r
# Re-check the total number of NAs removed
dim(df)[1] - dim(df_new)[1]
```

```
## [1] 646
```

In order to label PM2.5 concentration levels, there should not be any NA in 'PM2.5' to run the function below.

Adding a new label column by the concentration of PM2.5

```r
# Creating a new column, "PM2.5 Type"
df_new$PM2.5_Type = 0

# For loop to label rows by PM2.5
# 3 thresholds - 35/75/105
for (i in 1:dim(df_new)[1]) {
  if (df_new$PM2.5[i] <= 35) {
    df_new$PM2.5_Type[i] = "Low"
  } else if (df_new$PM2.5[i] > 35 & df_new$PM2.5[i] <= 75) {
    df_new$PM2.5_Type[i] = "Medium"
  } else if (df_new$PM2.5[i] > 75 & df_new$PM2.5[i] <= 105) {
    df_new$PM2.5_Type[i] = "High"
  } else if (df_new$PM2.5[i] > 105) {
    df_new$PM2.5_Type[i] = "Dangerous"
  }
}

# Reordering the Average_PM2.5_Type
df_new$PM2.5_Type = factor(df_new$PM2.5_Type, levels = c("Dangerous", "High", "Medium", "Low"))

head(df_new)
```

```
## # A tibble: 6 x 19
##      No  year month   day  hour PM2.5  PM10   SO2   NO2    CO    O3  TEMP  PRES
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     1  2013     3     1     0     6    18     5    NA   800    88   0.1 1021.
## 2     2  2013     3     1     1     6    15     5    NA   800    88  -0.3 1022.
## 3     3  2013     3     1     2     5    18    NA    NA   700    52  -0.7 1022.
## 4     4  2013     3     1     3     6    20     6    NA    NA    NA  -1   1023.
## 5     5  2013     3     1     4     5    17     5    NA   600    73  -1.3 1023
## 6     6  2013     3     1     5     4    11     3    NA   700    87  -1.8 1024.
## # ... with 6 more variables: DEWP <dbl>, RAIN <dbl>, wd <fct>, WSPM <dbl>,
## #   station <chr>, PM2.5_Type <fct>
```

Time-series columns

```r
# Package for converting time-related columns in one column
library("lubridate")
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
df_new$Date_ymd = with(df_new, ymd(paste(year, month, day, sep= ' ')))
df_new$weekday = weekdays(df_new$Date_ymd)

# Convert the variable into factor type & Re-ordering the weekdays
df_new$weekday = factor(df_new$weekday, levels = c("Monday", "Tuesday", "Wednesday", "Thursday", "Frida
```

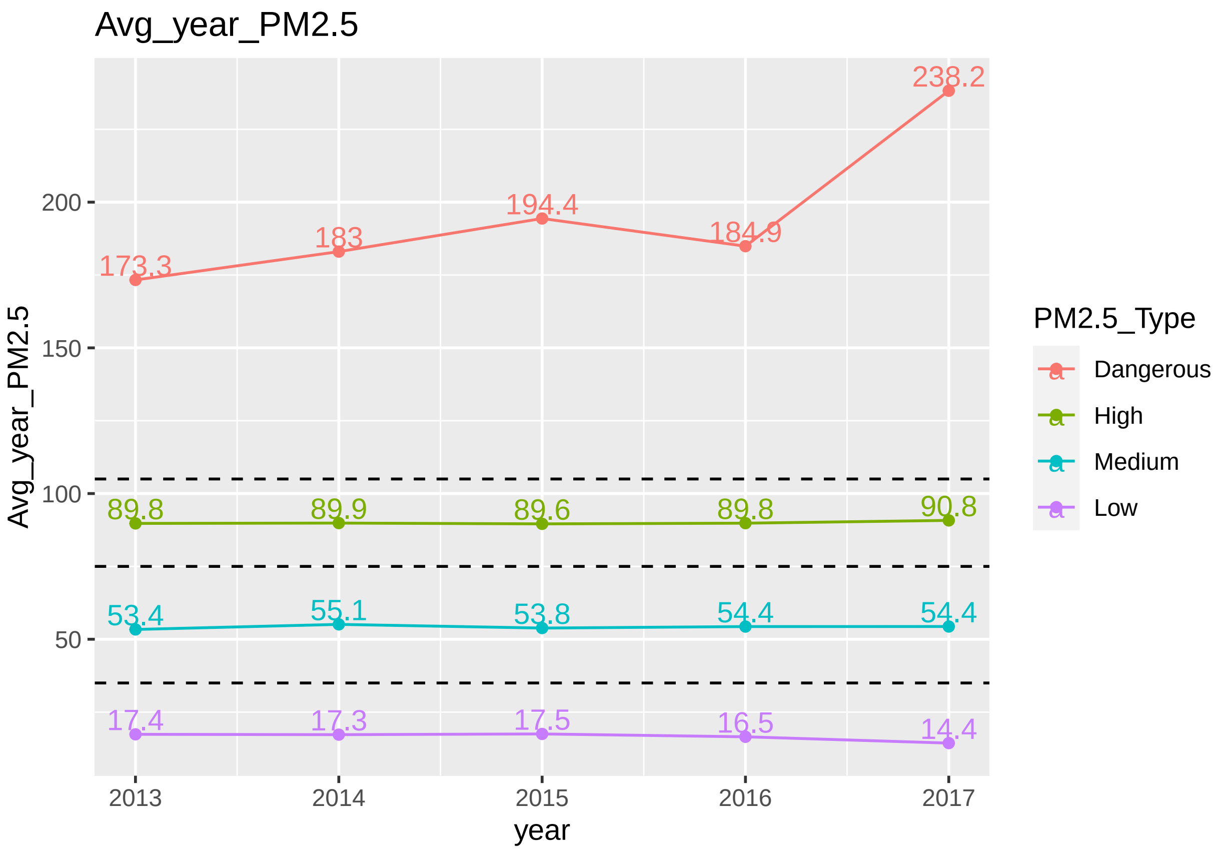
Mode Function - For Wind Direction (wd)

```r
# Create the function.
getmode = function(v) {
   uniqv = unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}

# Test whether the mode function works properly
vec = c("ESE", "ESE", "WS", "NW")
getmode(vec)
```

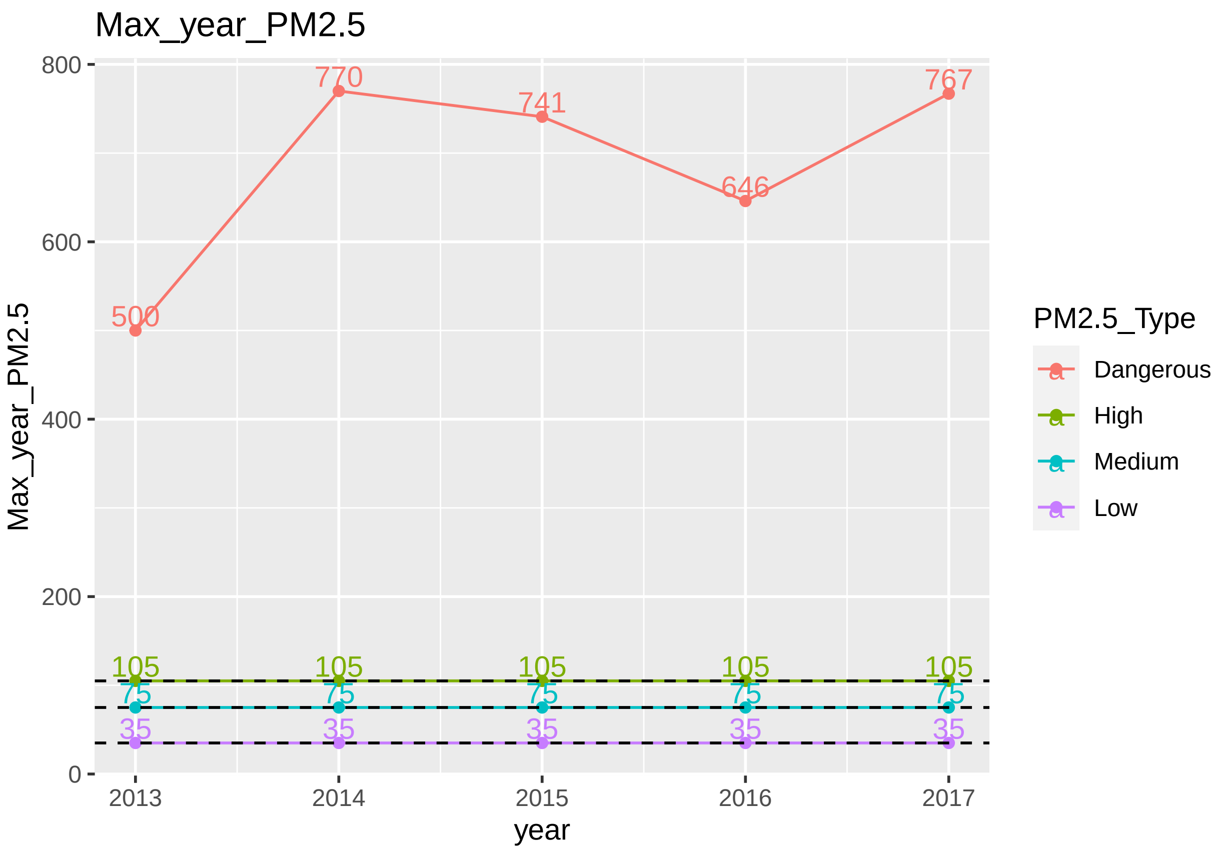
```
## [1] "ESE"
```

Time-series Visualizations (by year)

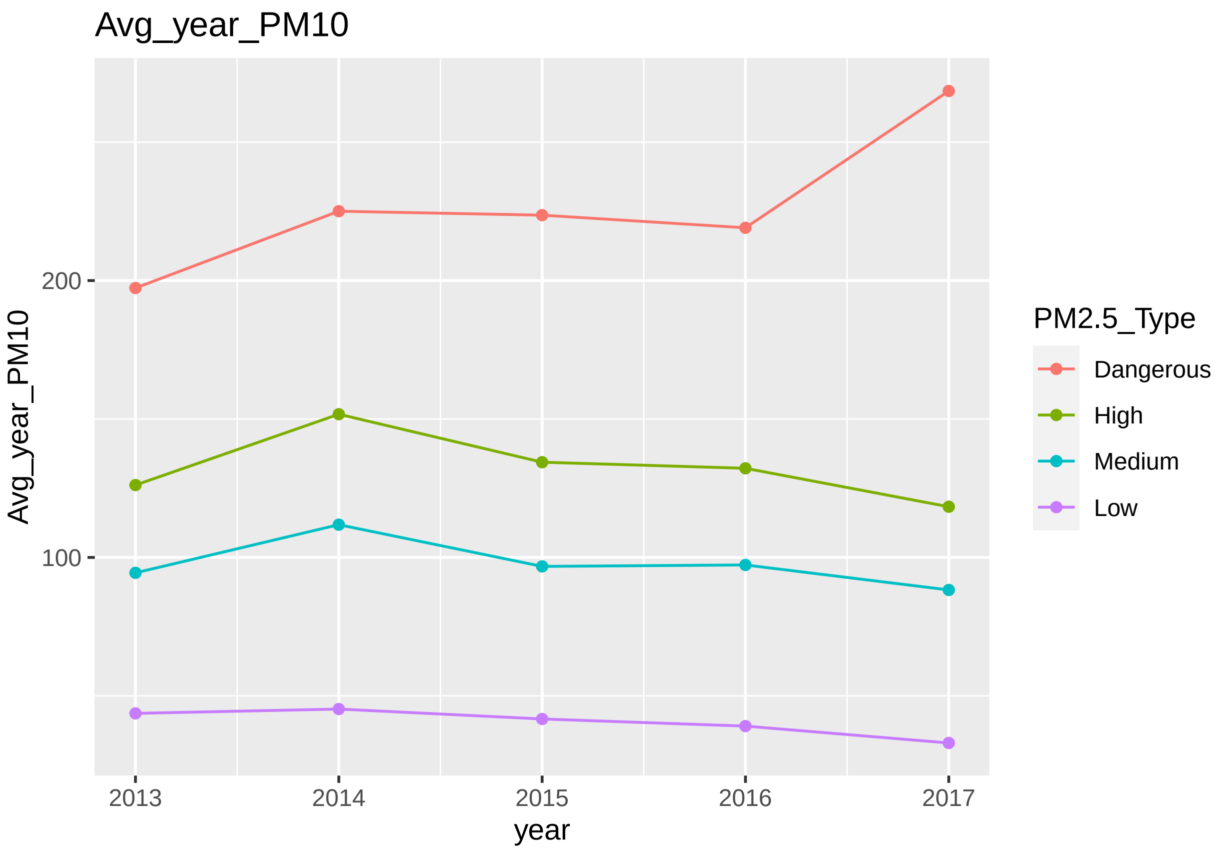
```
PM_year = df_new %>%
  group_by(PM2.5_Type, year) %>%
  summarise(
  Max_year_PM2.5 = max(PM2.5, na.rm=TRUE),
  Avg_year_PM2.5 = mean(PM2.5,na.rm=TRUE),
  Avg_year_PM10 = mean(PM10,na.rm=TRUE),
  Avg_year_SO2 = mean(SO2,na.rm=TRUE),
  Avg_year_NO2 = mean(NO2,na.rm=TRUE),
  Avg_year_CO = mean(CO,na.rm=TRUE),
  Avg_year_O3 = mean(O3,na.rm=TRUE),
  Avg_year_TEMP = mean(TEMP,na.rm=TRUE),
  Avg_year_PRES = mean(PRES,na.rm=TRUE),
  Avg_year_DEWP = mean(DEWP,na.rm=TRUE),
  Total_year_Rain = sum(RAIN,na.rm=TRUE),
  Avg_year_WSPM = mean(WSPM, na.rm=TRUE),
  Freq_year_wd = getmode(wd), .groups = "drop")

# Need to convert into dataframe to run the for loop below
PM_year = as.data.frame(PM_year)

# Option 1 - With values in the plot (Can remove)
ggplot(PM_year, aes(x=year, y=Avg_year_PM2.5, color=PM2.5_Type)) + geom_point(aes(color=PM2.5_Type)) + g
```

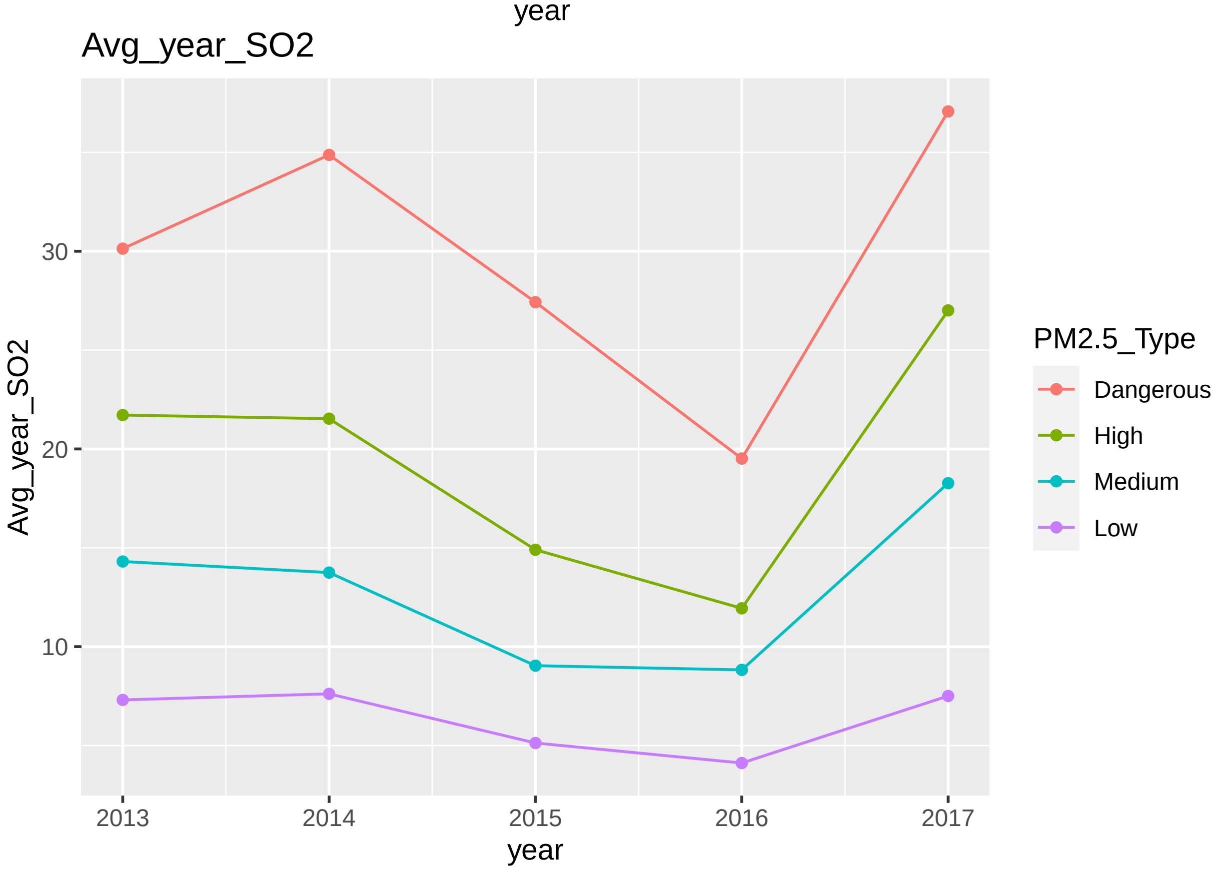


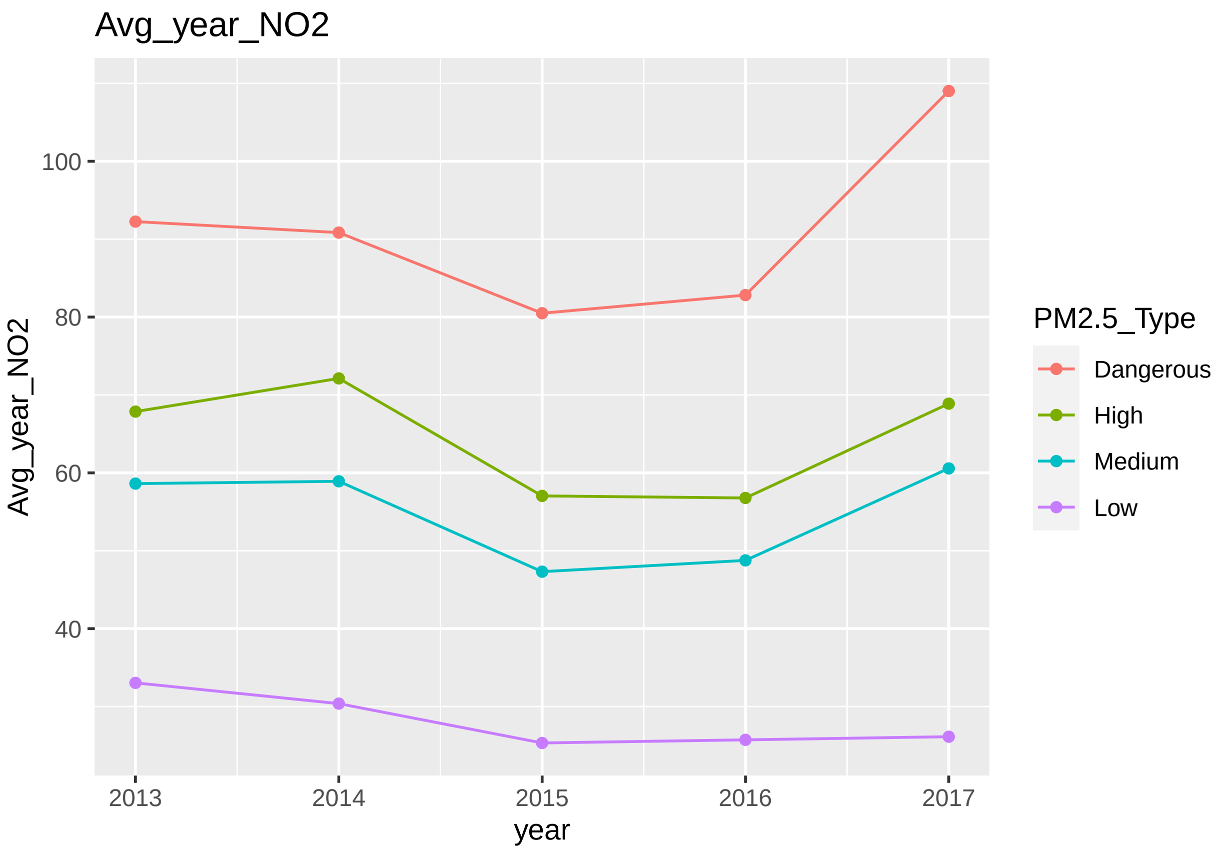

```
# Max_year_PM2.5
ggplot(PM_year, aes(x=year, y=Max_year_PM2.5, color=PM2.5_Type)) + geom_point(aes(color=PM2.5_Type)) + g
```

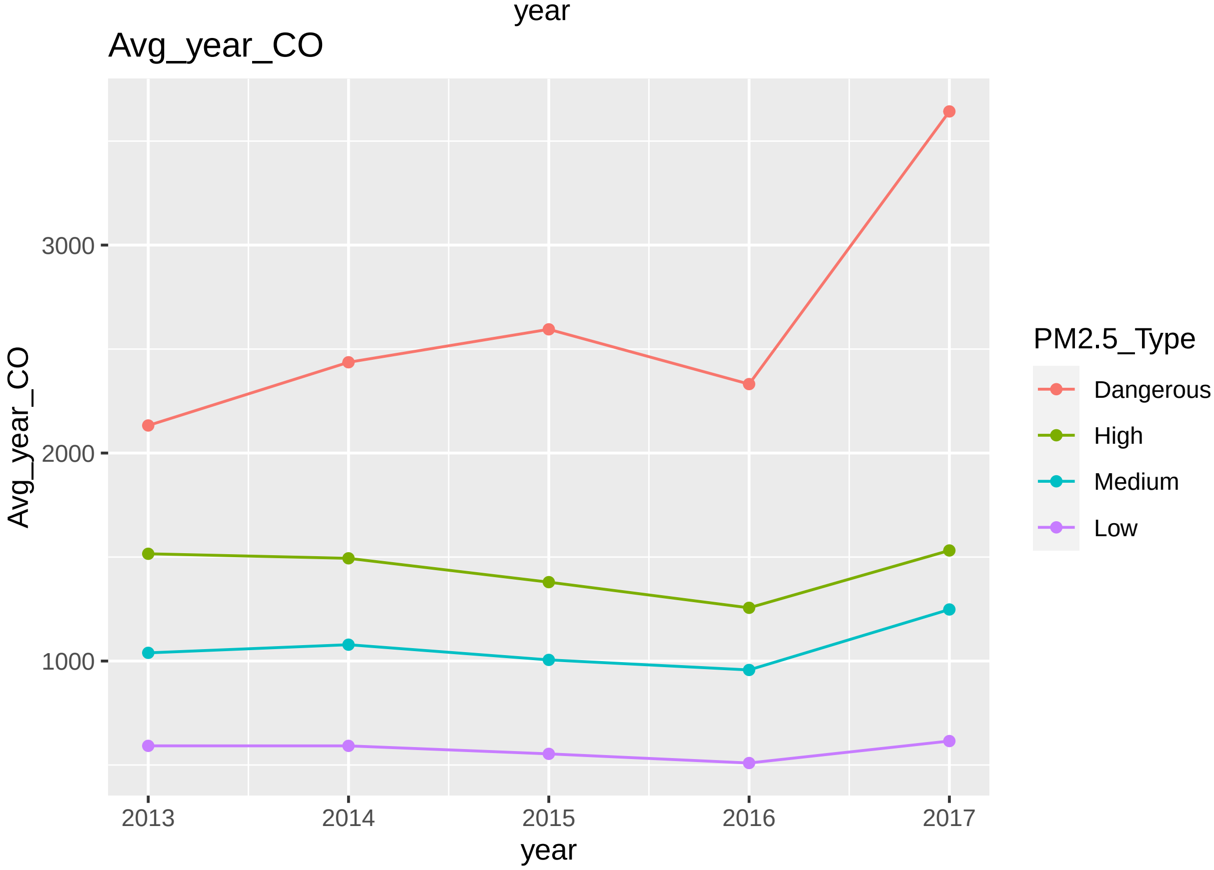


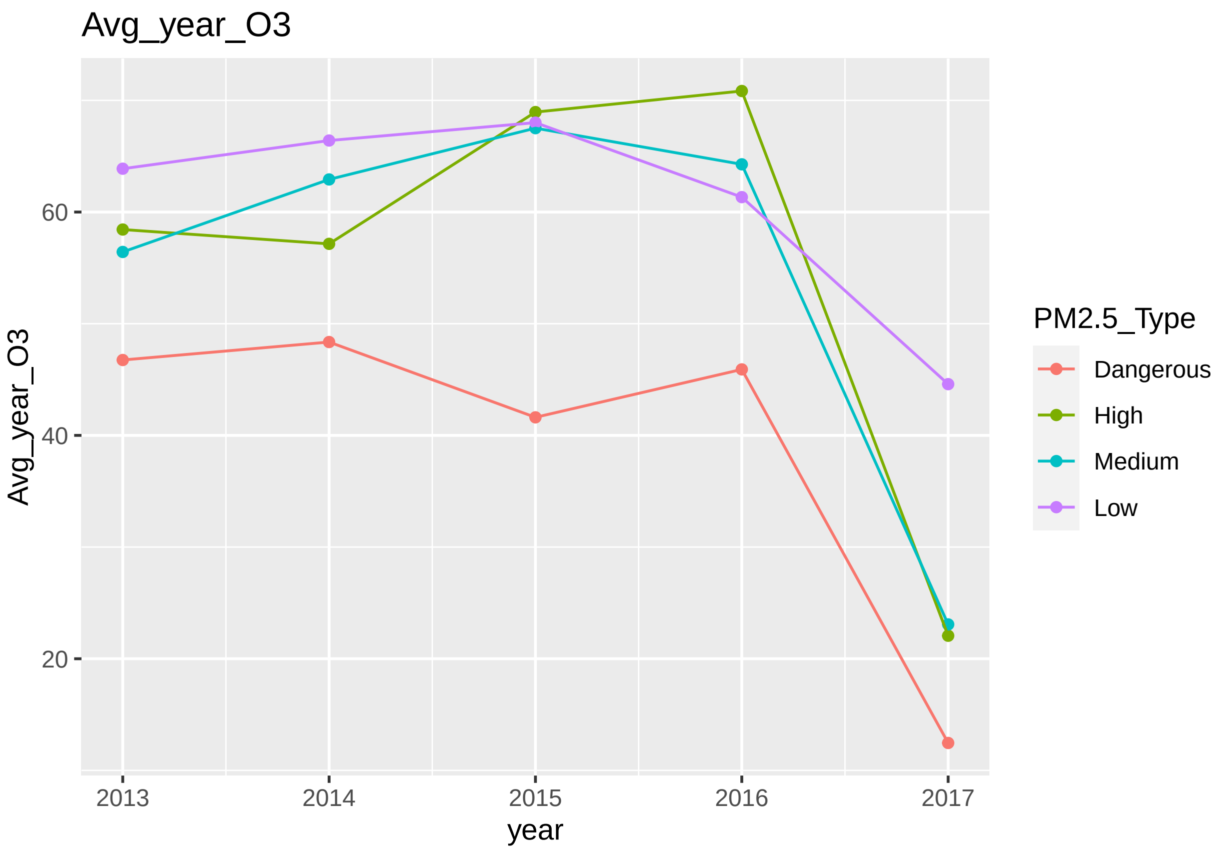

```
# Multiple plots with for loop
for (i in c(5:14)) {
  print(ggplot(PM_year, aes(x=year, y=PM_year[, i], color=PM2.5_Type)) + geom_point(aes(color=PM2.5_Type
}
```


Avg_year_PM10
200
100
Avg_year_PM10
2013
2014
2015
2016
2017
year
PM2.5_Type
Dangerous
High
Medium
Low

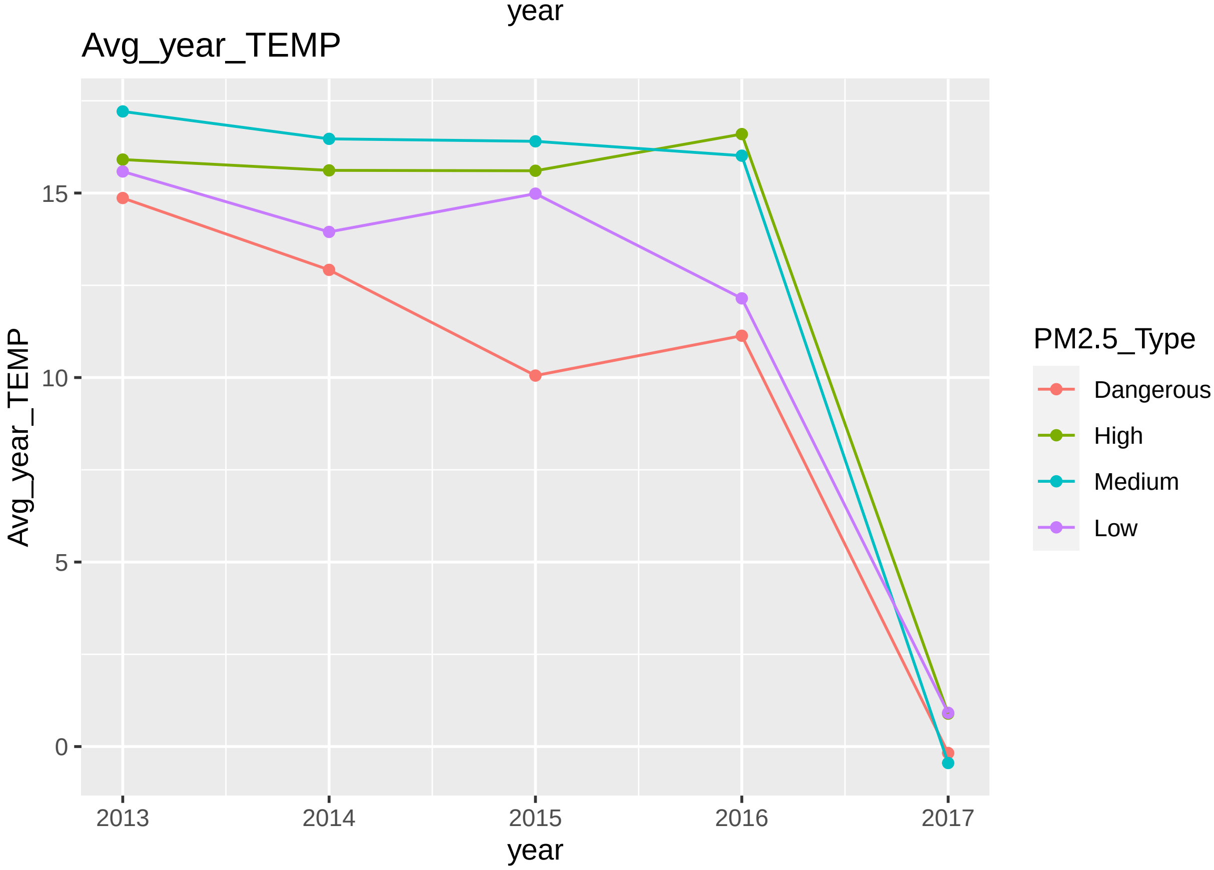

Avg_year_SO2
30
20
10
Avg_year_SO2
2013
2014
2015
2016
2017
year
PM2.5_Type
Dangerous
High
Medium
Low


Avg_year_NO2
100
80
60
40
Avg_year_NO2
2013
2014
2015
2016
2017
year
PM2.5_Type
Dangerous
High
Medium
Low

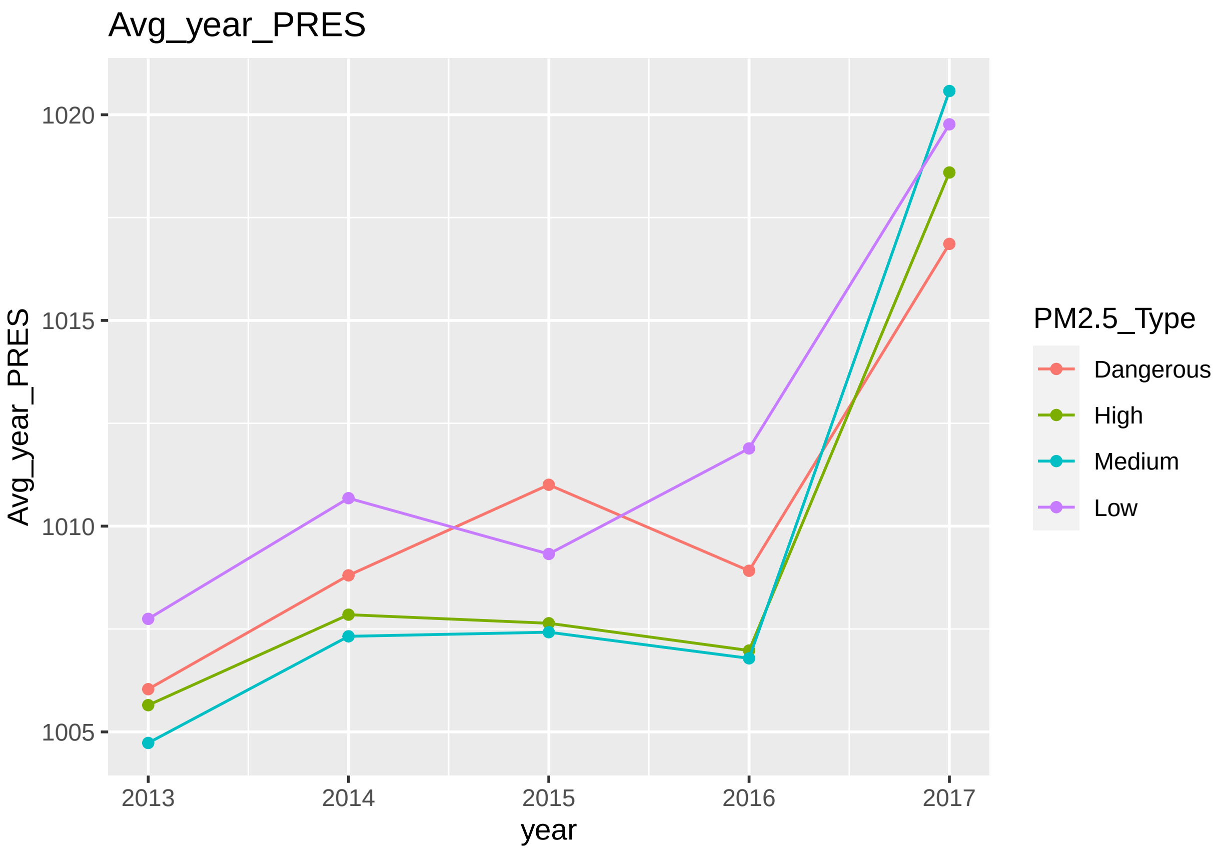

Avg_year_CO
3000
2000
1000
Avg_year_CO
2013
2014
2015
2016
2017
year
PM2.5_Type
Dangerous
High
Medium
Low

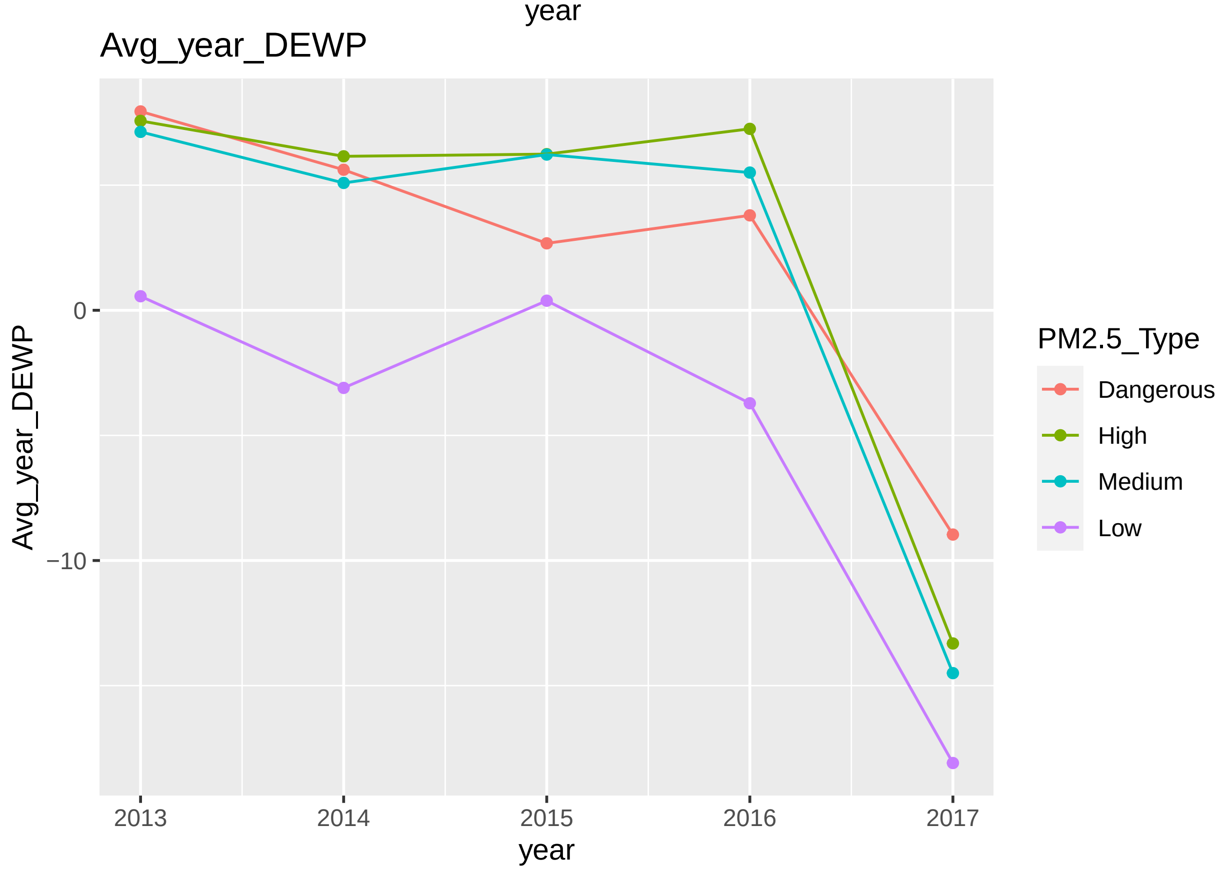


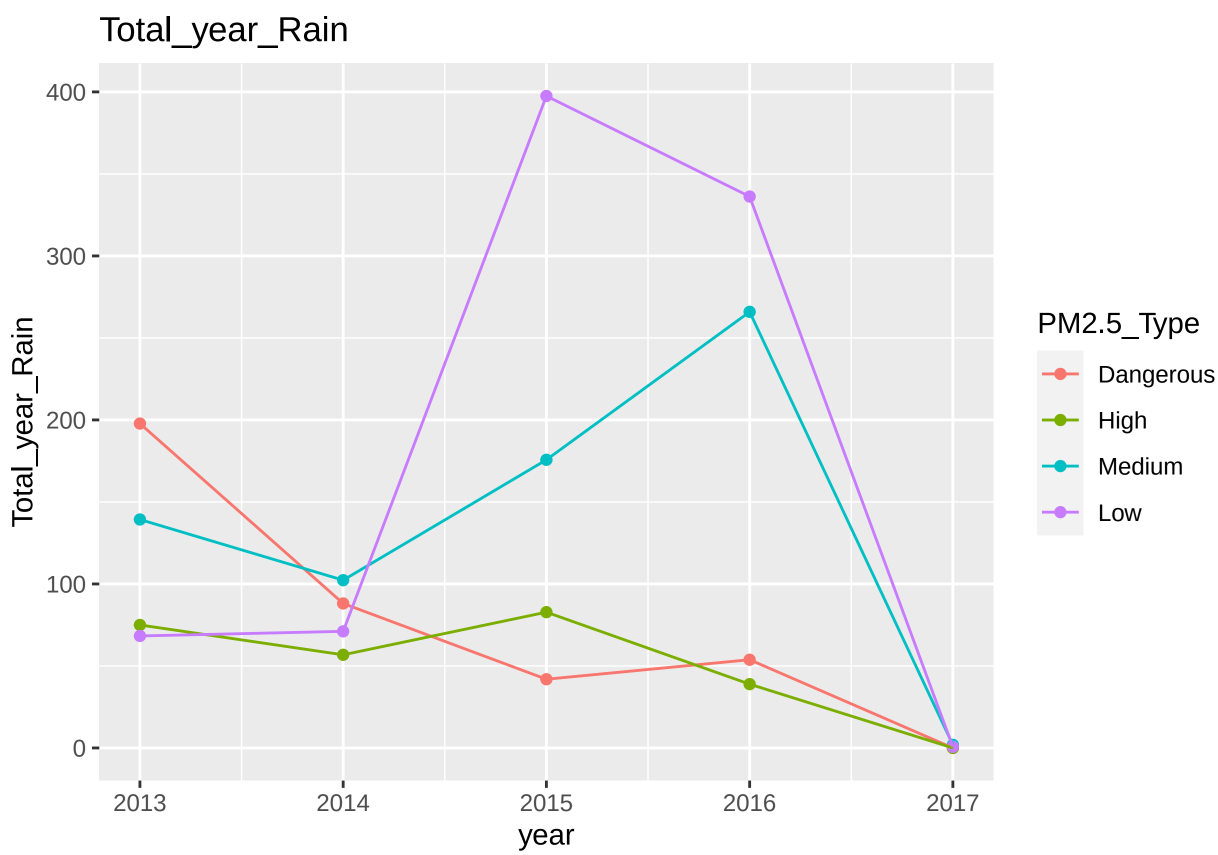
Avg_year_O3
Avg_year_O3
60
40
20
2013
2014
2015
2016
2017
year
PM2.5_Type
Dangerous
High
Medium
Low

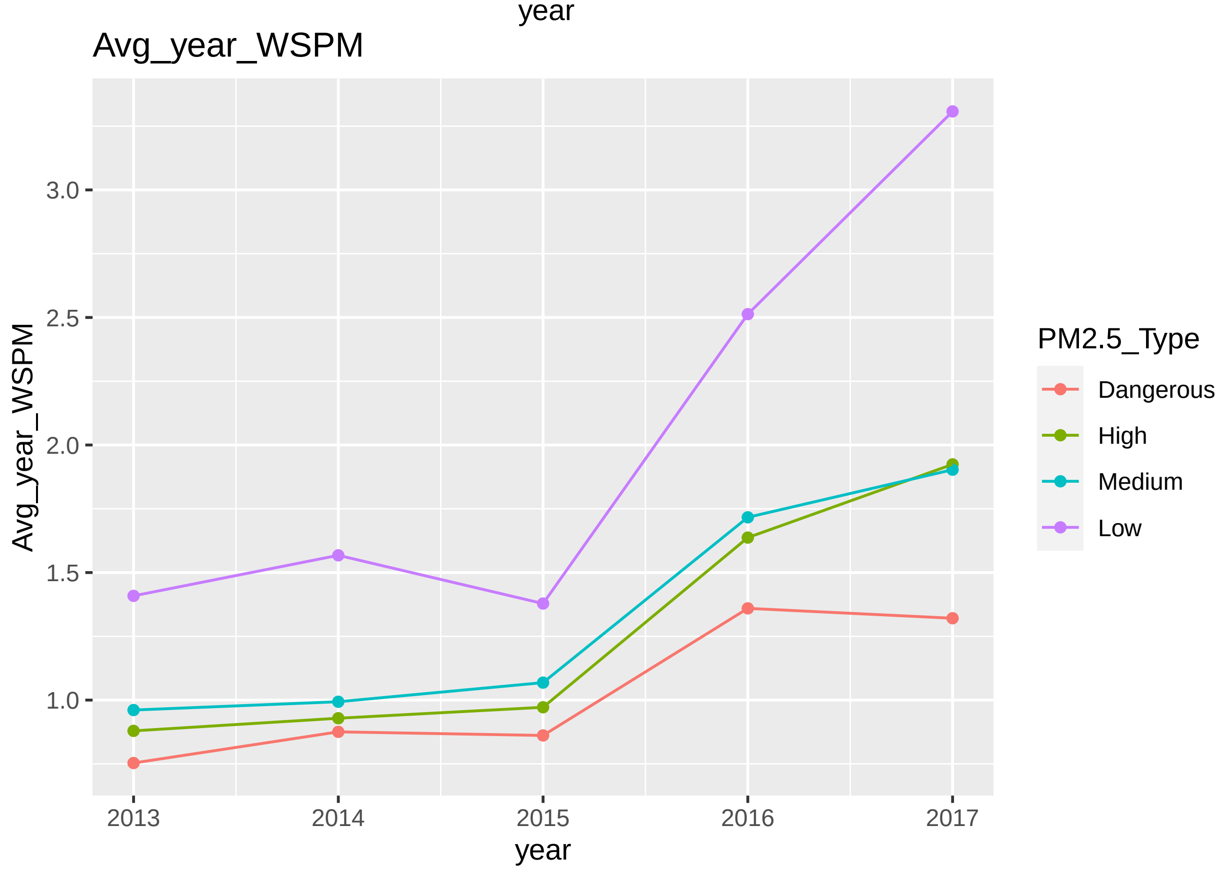


Avg_year_TEMP
Avg_year_TEMP
15
10
5
0
2013
2014
2015
2016
2017
year
PM2.5_Type
Dangerous
High
Medium
Low

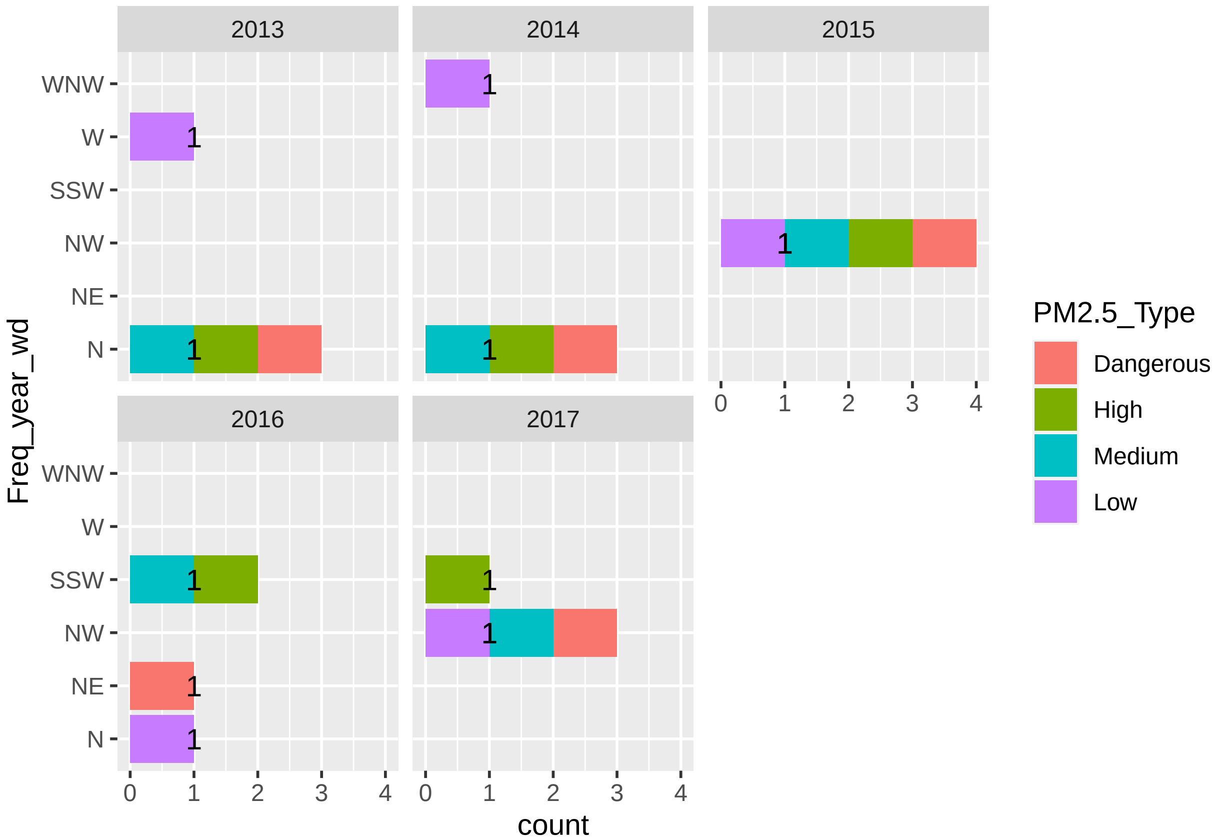

Avg_year_PRES
1020
1015
1010
1005
Avg_year_PRES
2013
2014
2015
2016
2017
year
PM2.5_Type
Dangerous
High
Medium
Low


Avg_year_DEWP
0
−10
Avg_year_DEWP
2013
2014
2015
2016
2017
year
PM2.5_Type
Dangerous
High
Medium
Low


Total_year_Rain
Total_year_Rain
0
100
200
300
400
2013
2014
2015
2016
2017
year
PM2.5_Type
Dangerous
High
Medium
Low


Avg_year_WSPM
Avg_year_WSPM
1.0
1.5
2.0
2.5
3.0
2013
2014
2015
2016
2017
year
PM2.5_Type
Dangerous
High
Medium
Low

```r
# Plot for 'Freq_year_wd'
ggplot(PM_year, aes(y = Freq_year_wd, fill =PM2.5_Type)) + geom_bar() + facet_wrap(~year) + geom_text(ae
```

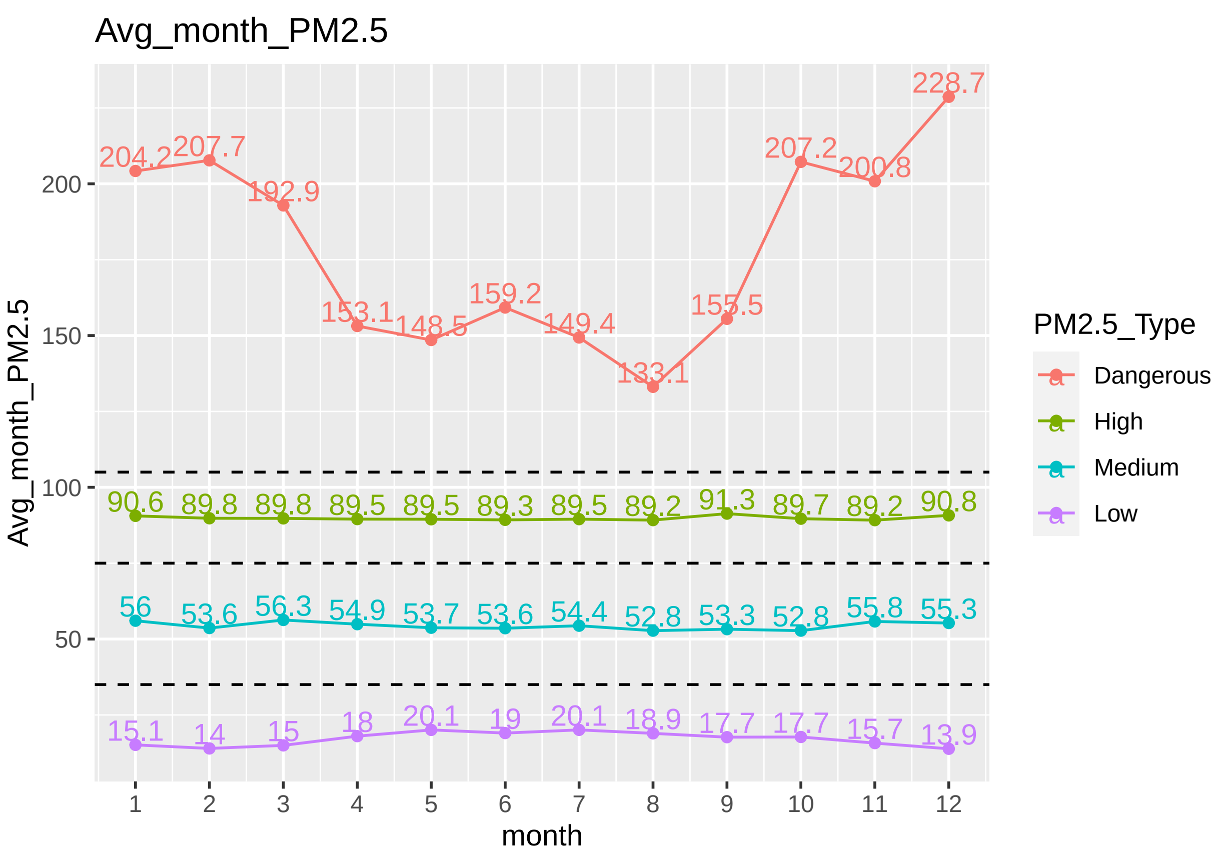


Above is the visualizations of different variables' trends by year. Although the highest maximum PM2.5 concentration has been detected in 2014, the average PM2.5 concentration was highest in 2017. Considering that the dataset contains from March 1, 2013 to February 28, 2017, meaning that 2017 has only January and February, we could make a naive assumption that the winter would more likely to have higher PM2.5 concentration. Also, there are some air pollutants that follow similar trends to PM2.5 such as PM10 and CO whereas WSPM (wind speed) seems to show the opposite PM2.5 level order.

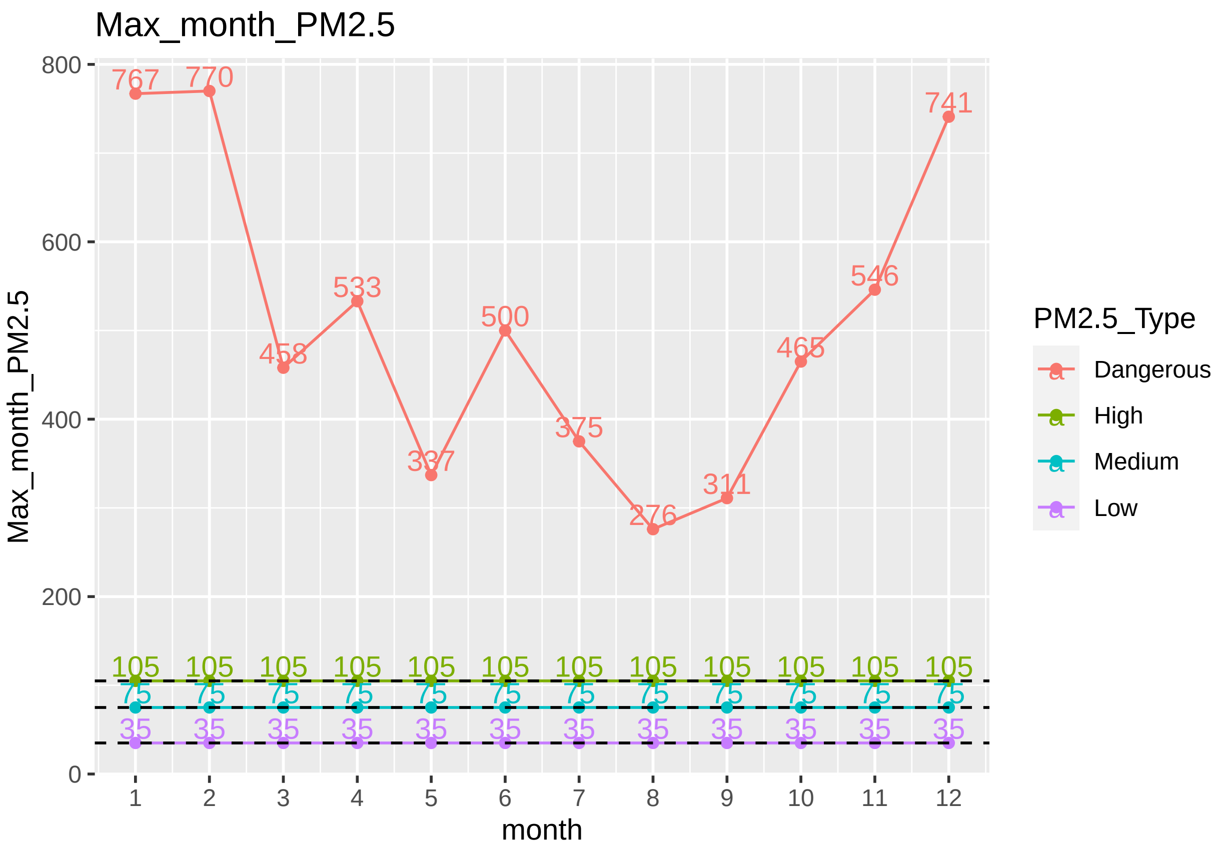
Time-series Visualizations (by month)

```r
PM_month = df_new %>%
  group_by(PM2.5_Type, month) %>%
  summarise(
  Max_month_PM2.5 = max(PM2.5, na.rm=TRUE),
  Avg_month_PM2.5 = mean(PM2.5,na.rm=TRUE),
  Avg_month_PM10 = mean(PM10,na.rm=TRUE),
  Avg_month_SO2 = mean(SO2,na.rm=TRUE),
  Avg_month_NO2 = mean(NO2,na.rm=TRUE),
  Avg_month_CO = mean(CO,na.rm=TRUE),
  Avg_month_O3 = mean(O3,na.rm=TRUE),
  Avg_month_TEMP = mean(TEMP,na.rm=TRUE),
  Avg_month_PRES = mean(PRES,na.rm=TRUE),
  Avg_month_DEWP = mean(DEWP,na.rm=TRUE),
```

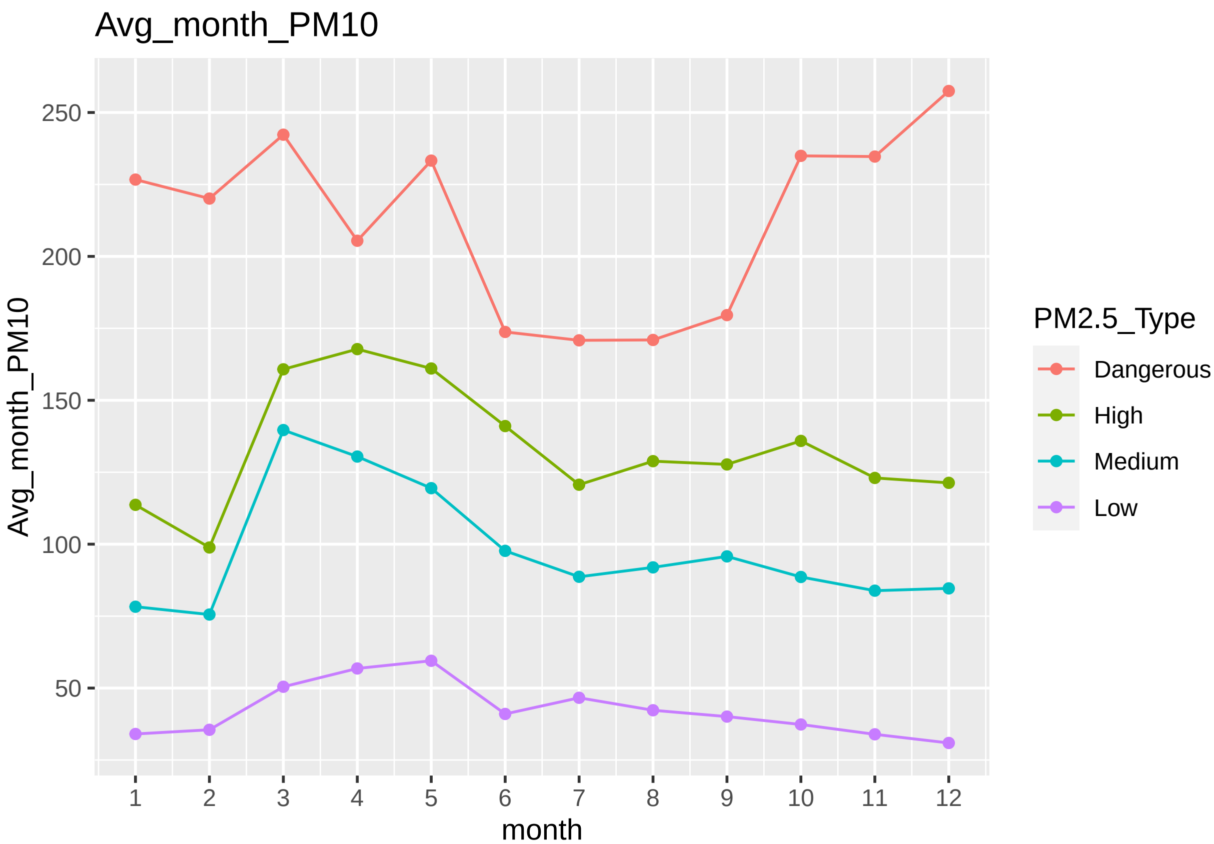
```
  Total_month_Rain = sum(RAIN,na.rm=TRUE),
  Avg_month_WSPM = mean(WSPM, na.rm=TRUE),
  Freq_month_wd = getmode(wd), .groups = "drop")

# Need to convert into dataframe to run the for loop below
PM_month = as.data.frame(PM_month)

# Option 1 - With values in the plot Avg_month_PM2.5
ggplot(PM_month, aes(x=month, y=Avg_month_PM2.5, color=PM2.5_Type)) + geom_point(aes(color=PM2.5_Type))
```

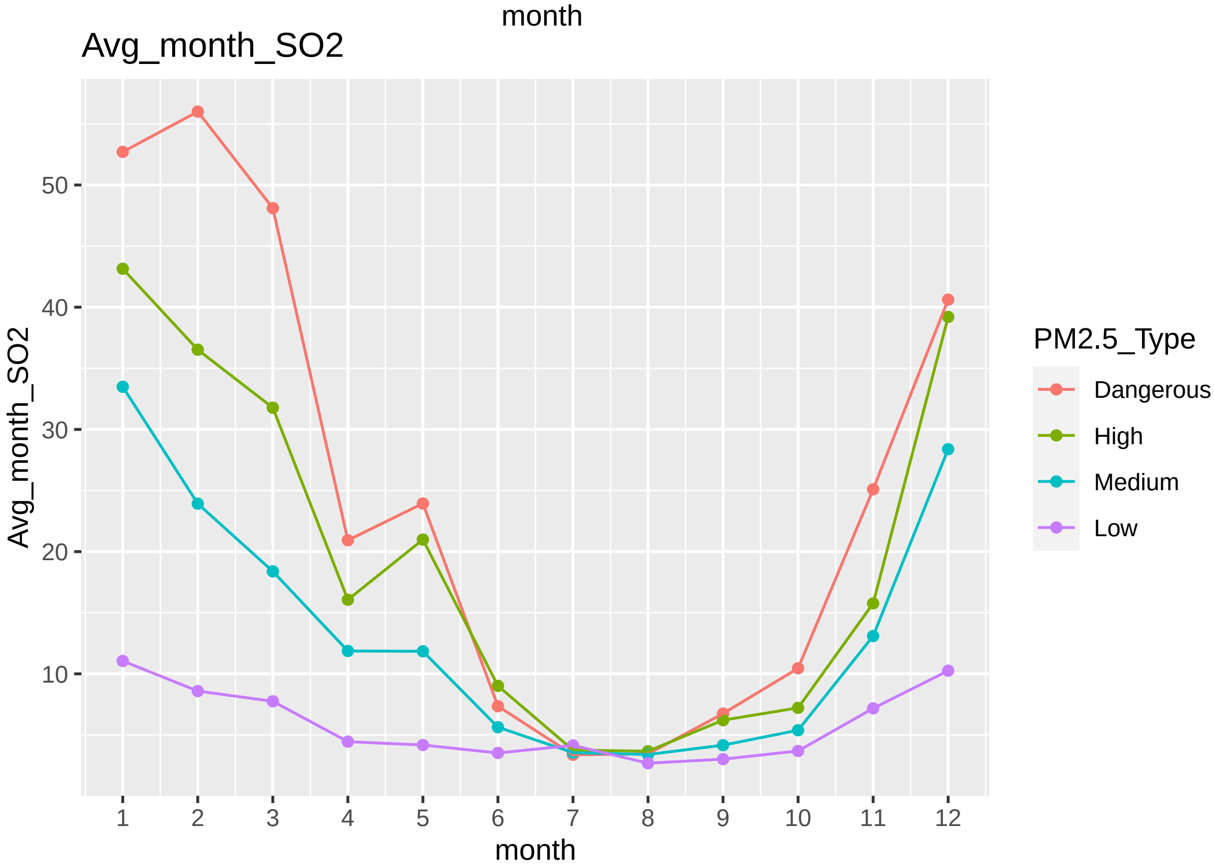


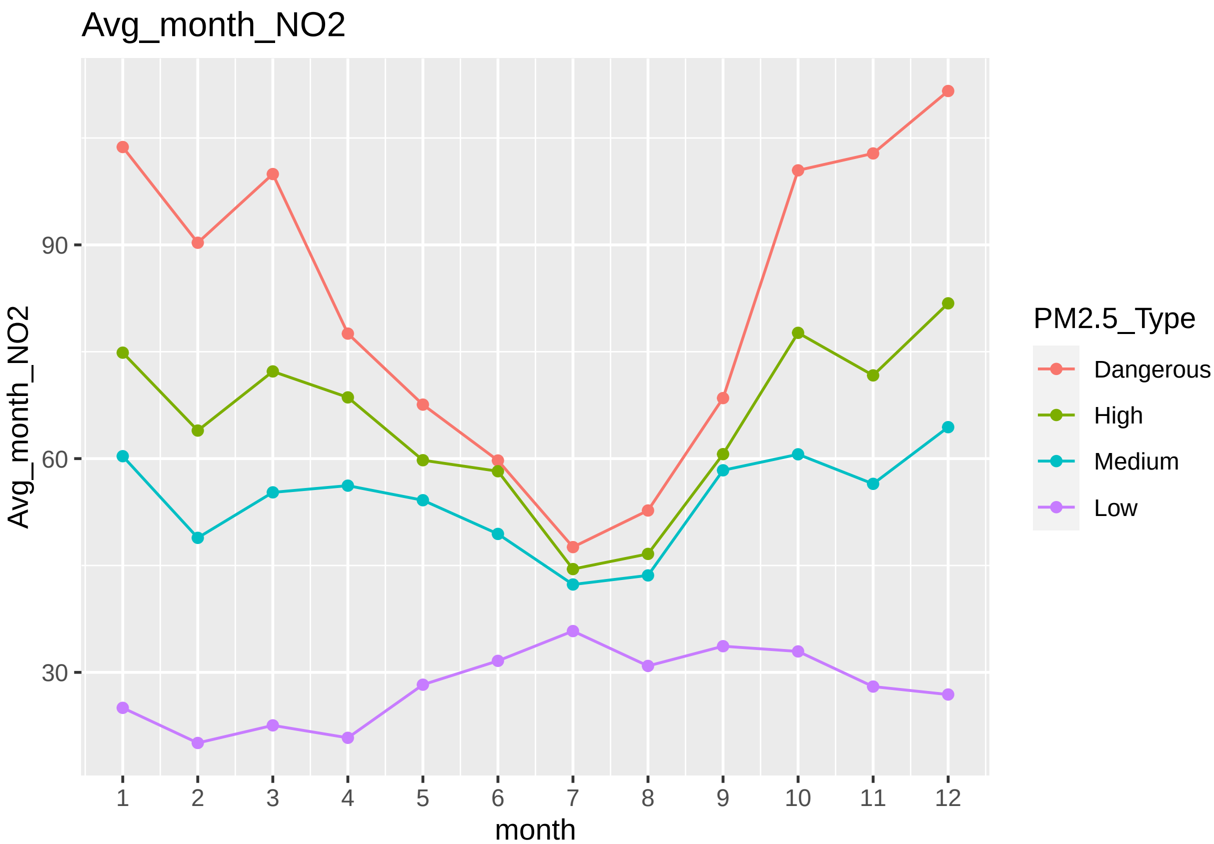

```
# Max_month_PM2.5
ggplot(PM_month, aes(x=month, y=Max_month_PM2.5, color=PM2.5_Type)) + geom_point(aes(color=PM2.5_Type))
```

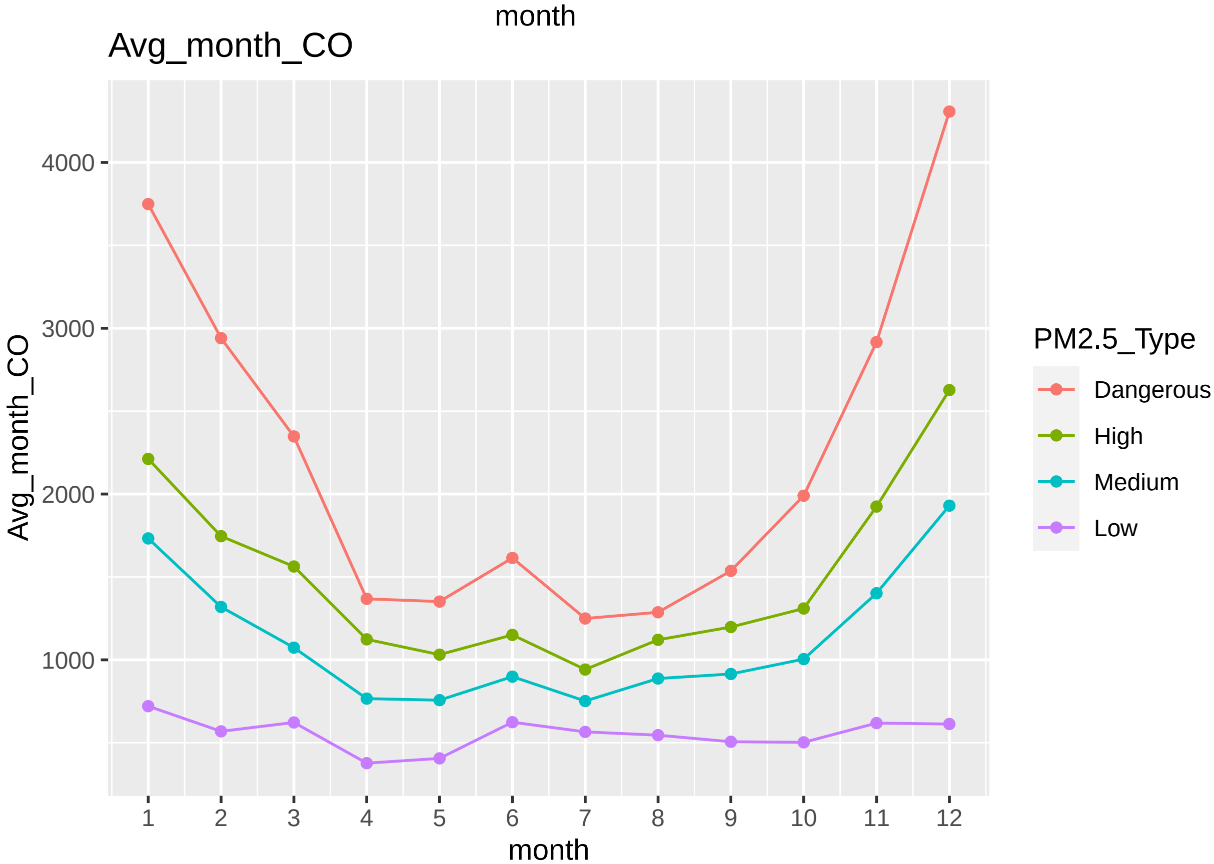


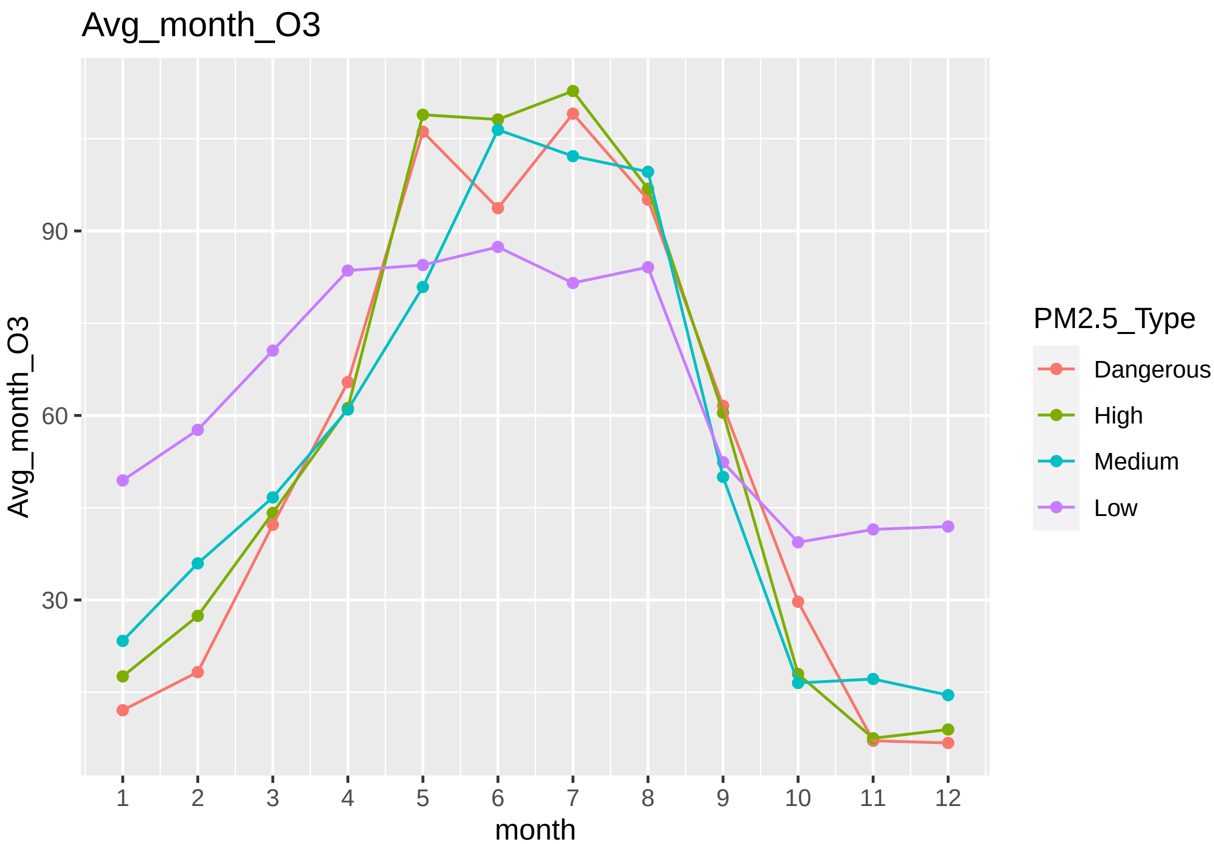

```
# Multiple plots with for loop
for (i in c(5:14)) {
  print(ggplot(PM_month, aes(x=month, y=PM_month[, i], color=PM2.5_Type)) + geom_point(aes(color=PM2.5_T
}
```

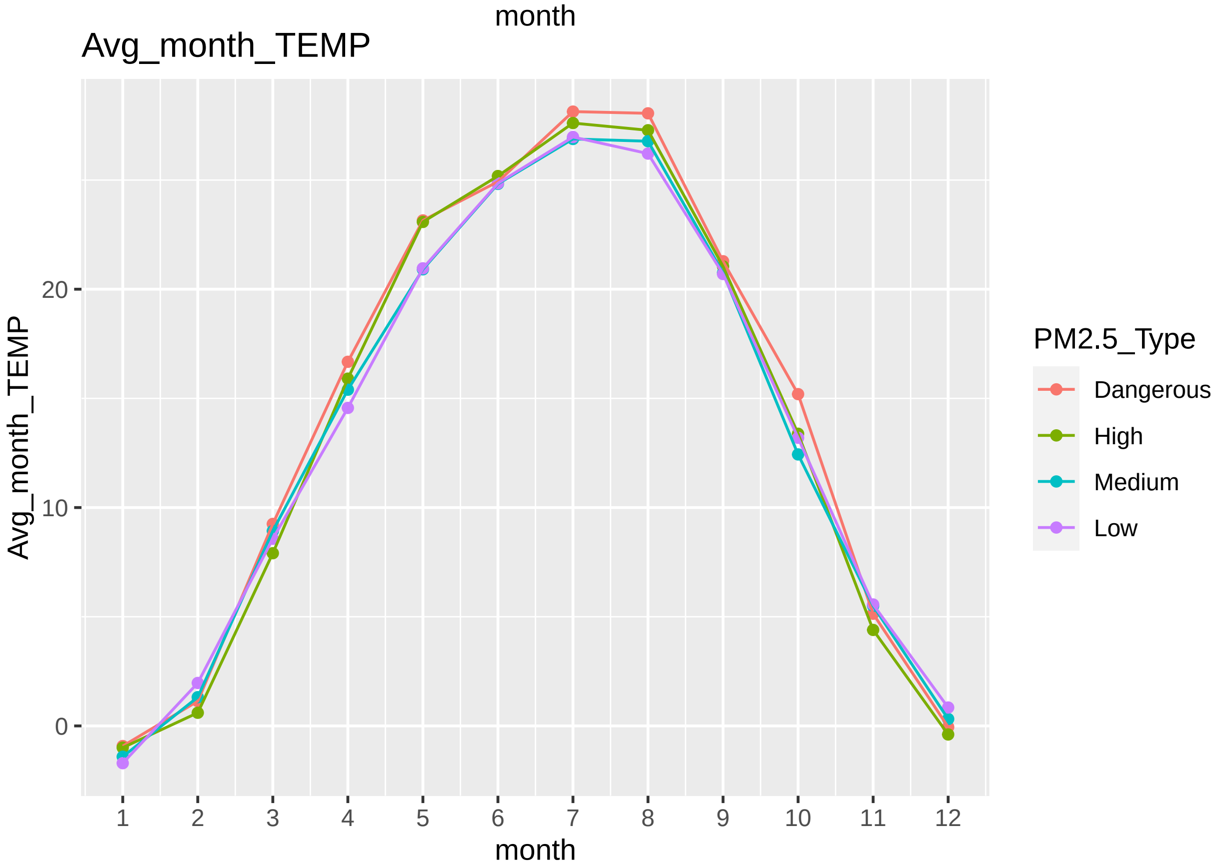

Avg_month_PM10
Avg_month_PM10
month
PM2.5_Type
Dangerous
High
Medium
Low

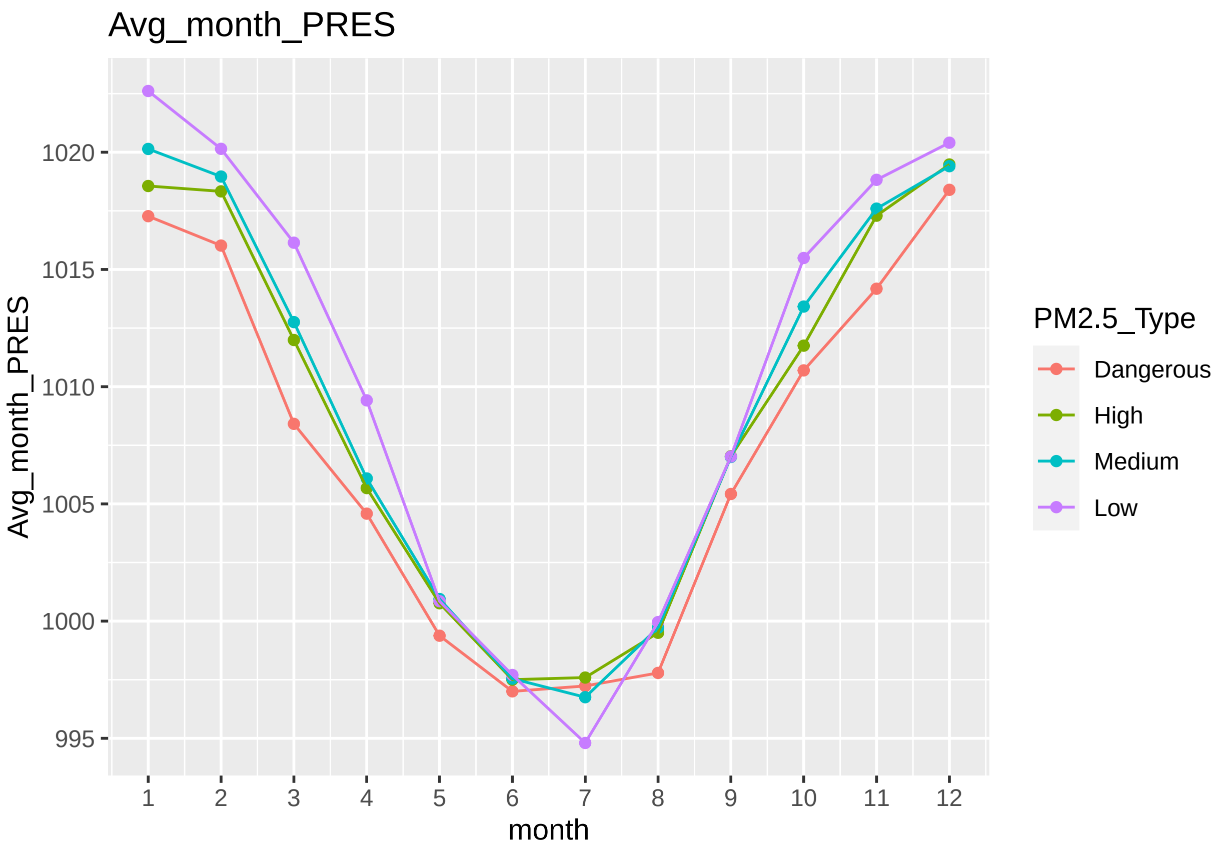

Avg_month_SO2
Avg_month_SO2
month
PM2.5_Type
Dangerous
High
Medium
Low

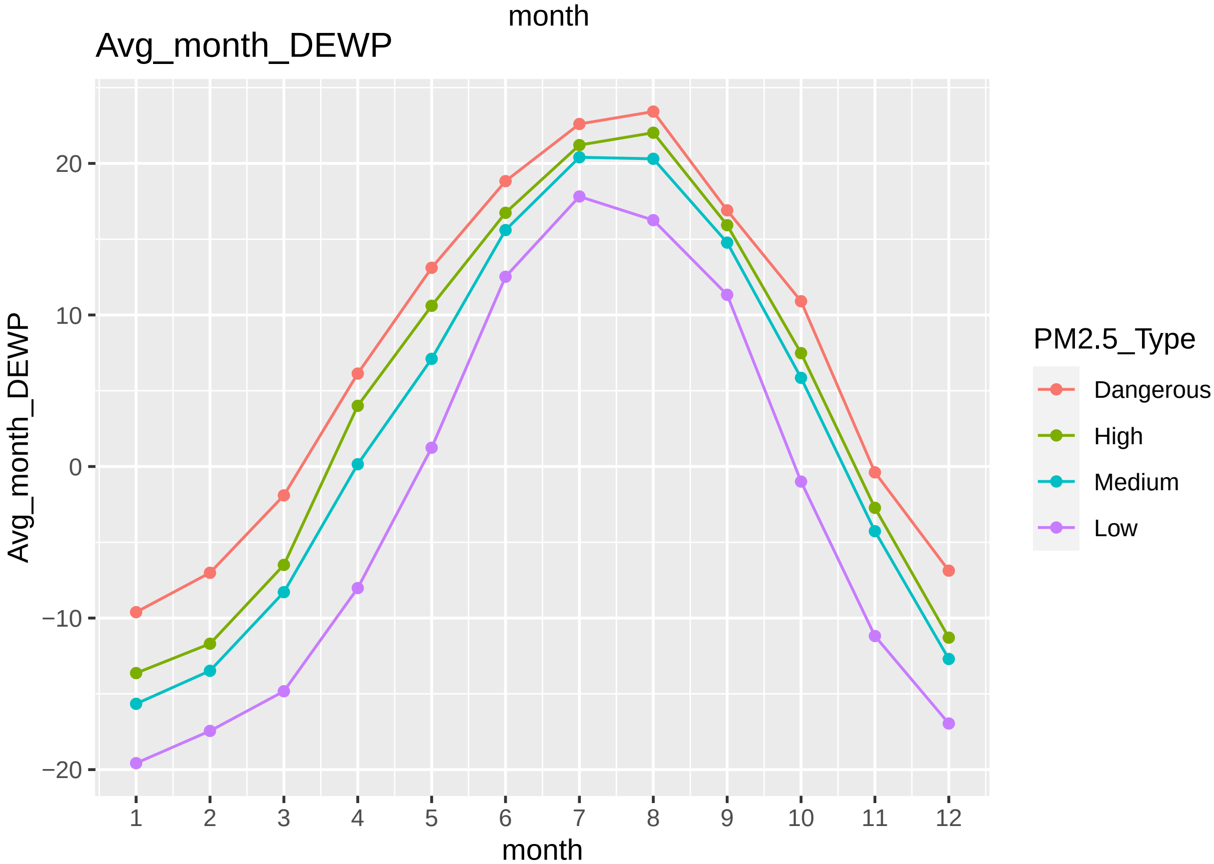

Avg_month_NO2
90
60
30
Avg_month_NO2
1
2
3
4
5
6
7
8
9
10
11
12
month
PM2.5_Type
Dangerous
High
Medium
Low

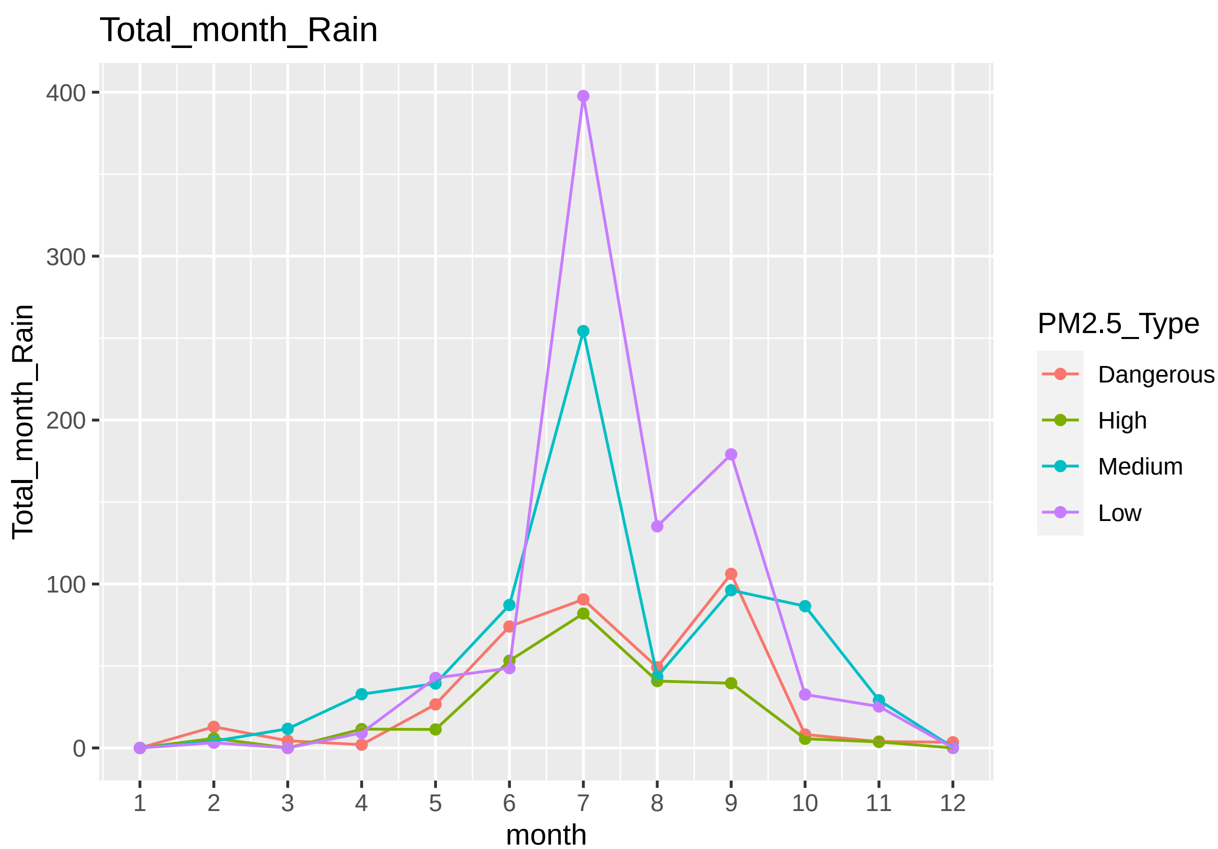

Avg_month_CO
4000
3000
2000
1000
Avg_month_CO
1
2
3
4
5
6
7
8
9
10
11
12
month
PM2.5_Type
Dangerous
High
Medium
Low

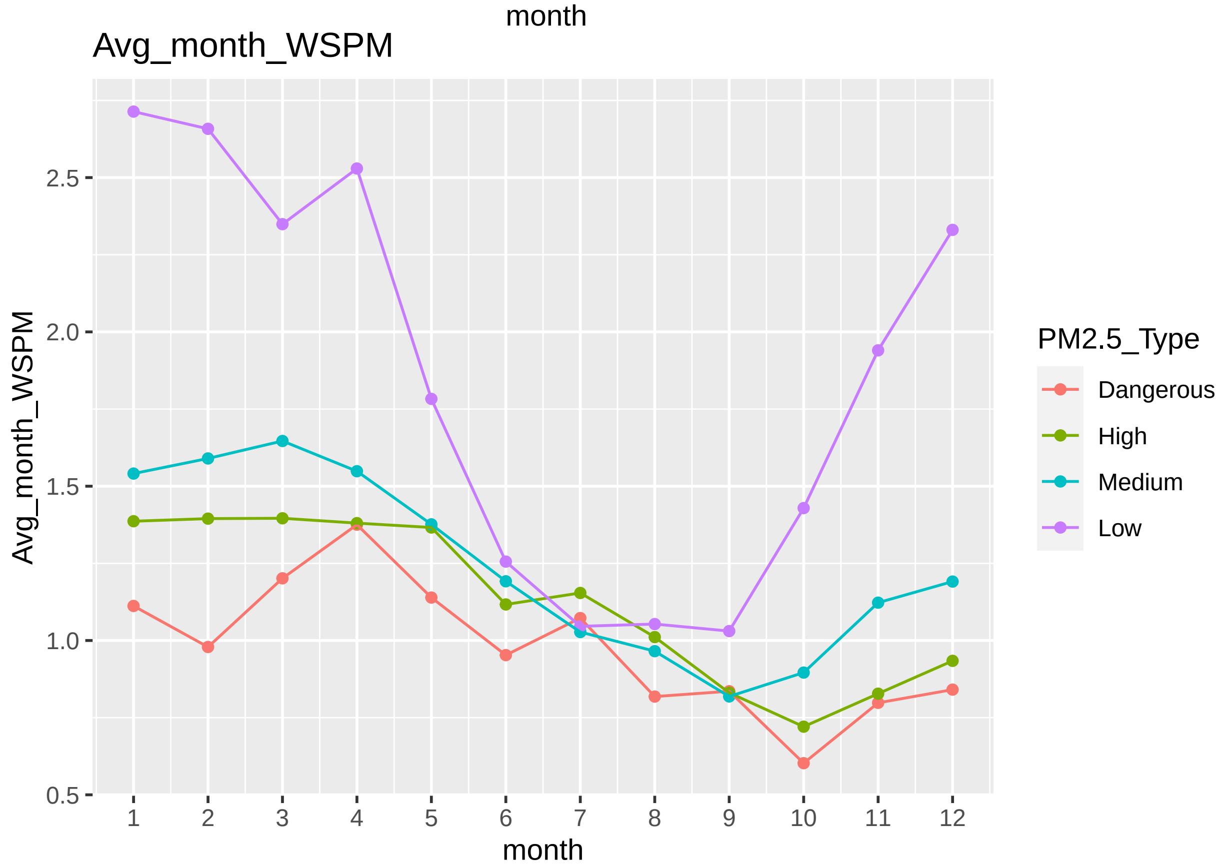

Avg_month_O3
Avg_month_O3
90
60
30
1
2
3
4
5
6
7
8
9
10
11
12
month
PM2.5_Type
Dangerous
High
Medium
Low


Avg_month_TEMP
Avg_month_TEMP
20
10
0
1
2
3
4
5
6
7
8
9
10
11
12
month
PM2.5_Type
Dangerous
High
Medium
Low

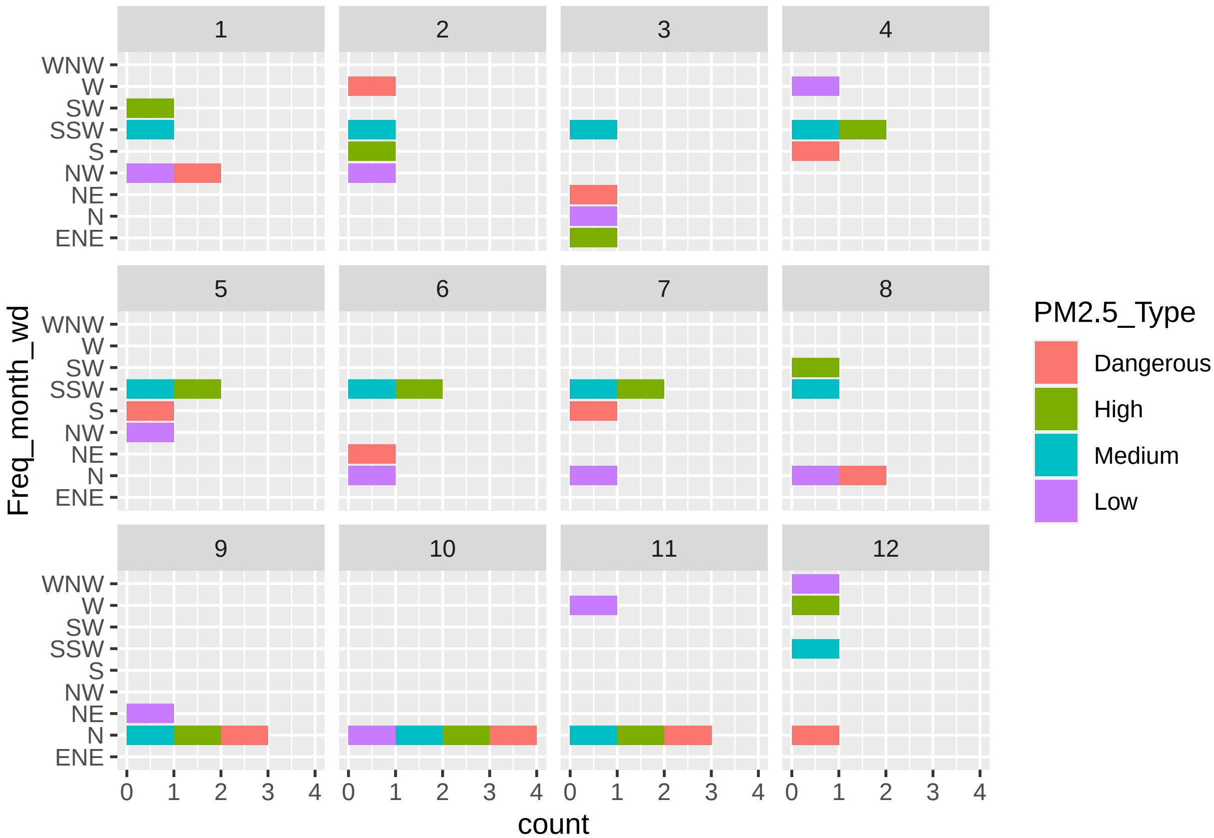

Avg_month_PRES
Avg_month_PRES
1020
1015
1010
1005
1000
995
1
2
3
4
5
6
7
8
9
10
11
12
month
PM2.5_Type
Dangerous
High
Medium
Low


Avg_month_DEWP
Avg_month_DEWP
20
10
0
−10
−20
1
2
3
4
5
6
7
8
9
10
11
12
month
PM2.5_Type
Dangerous
High
Medium
Low


Total_month_Rain
400
300
200
100
0
Total_month_Rain
1
2
3
4
5
6
7
8
9
10
11
12
month
PM2.5_Type
Dangerous
High
Medium
Low


Avg_month_WSPM
2.5
2.0
1.5
1.0
0.5
Avg_month_WSPM
1
2
3
4
5
6
7
8
9
10
11
12
month
PM2.5_Type
Dangerous
High
Medium
Low

```r
# Plot for 'Freq_month_wd'
ggplot(PM_month, aes(y = Freq_month_wd, fill =PM2.5_Type)) + geom_bar() + facet_wrap(~month)
```

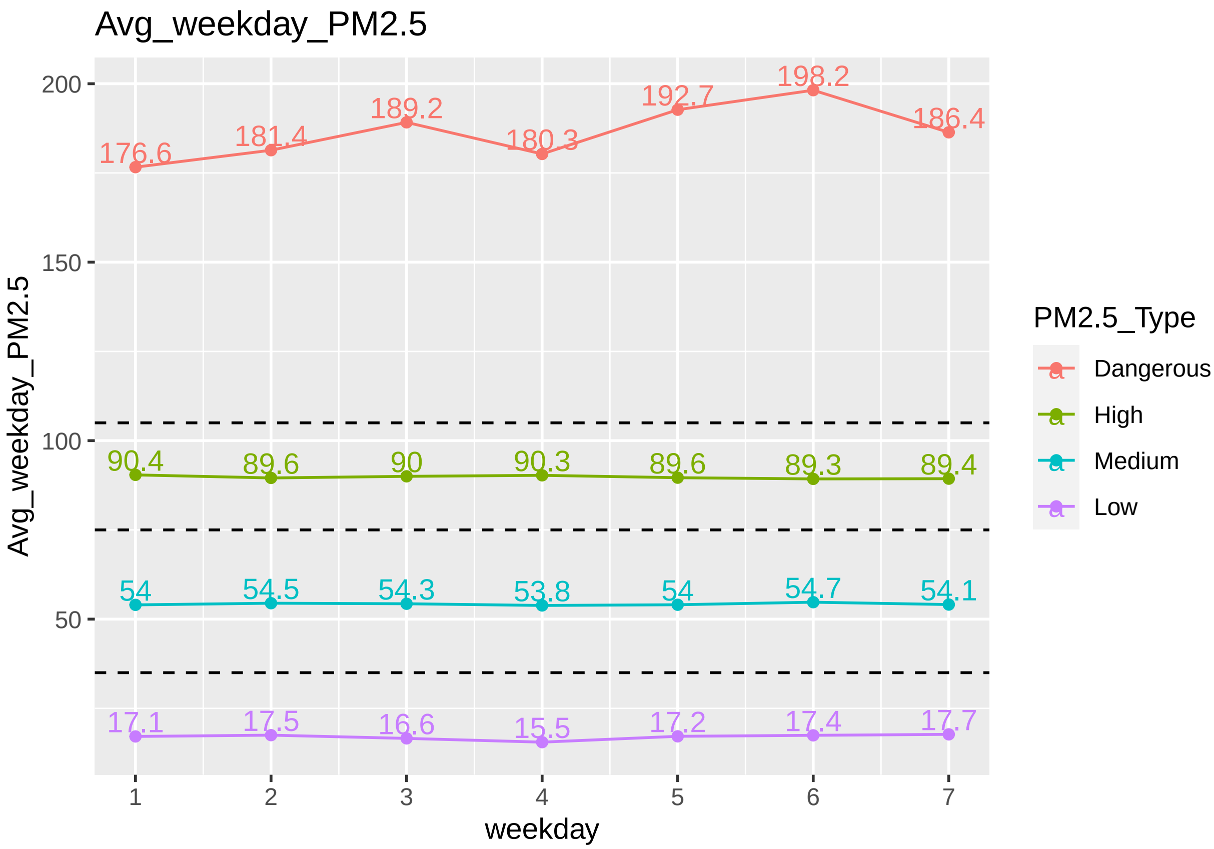


Recalling our naive assumption made from the 'year' visualizations above, both the maximum and average PM2.5 concentration are higher in the winter months (December / January / February) and lower in the summer months (July / August). Our intuition was somewhat correct. Similar to the 'year', air pollutants like SO2 and NO2 have pretty identical ups and downs while O3 and DEWP (Dew point) have the opposite spikes.

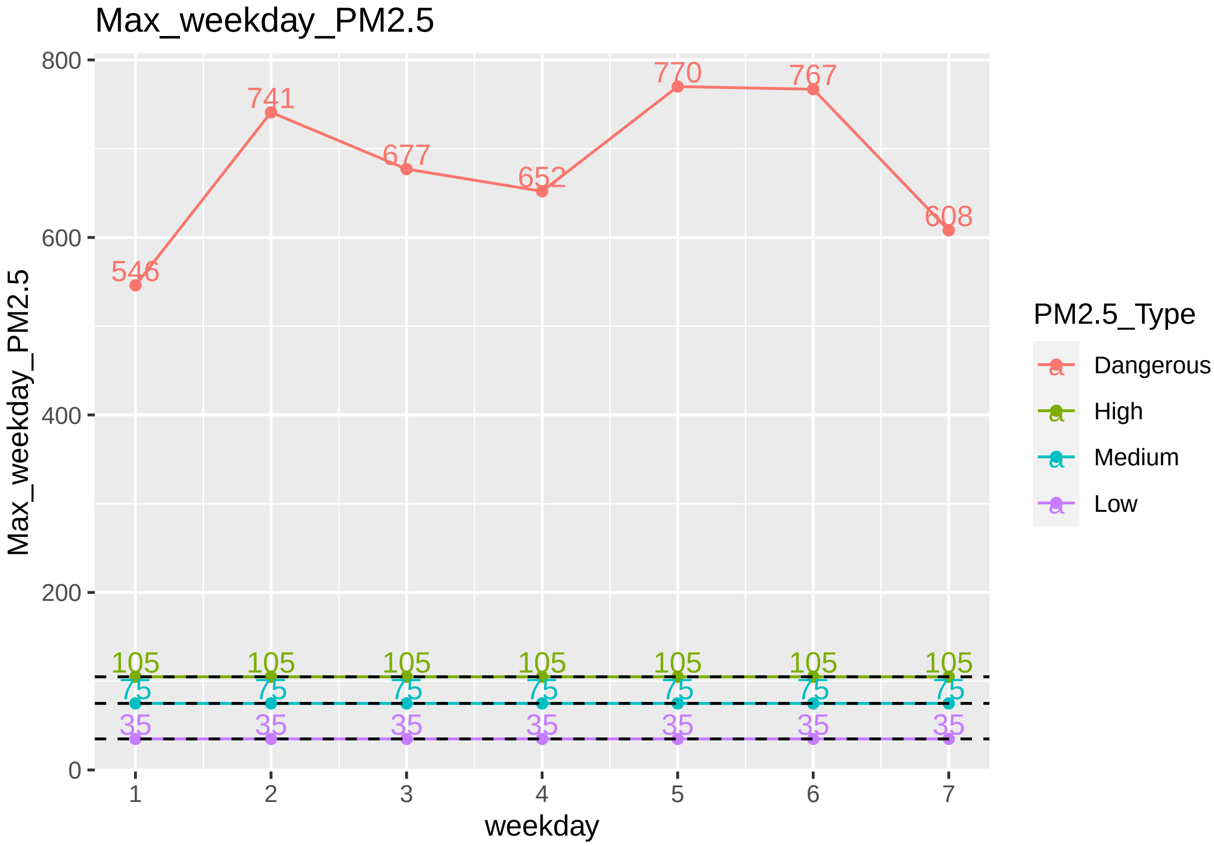
Time-series Visualizations (by day)

```r
PM_weekday = df_new %>%
  group_by(PM2.5_Type, weekday) %>%
  summarise(
  Max_weekday_PM2.5 = max(PM2.5, na.rm=TRUE),
  Avg_weekday_PM2.5 = mean(PM2.5,na.rm=TRUE),
  Avg_weekday_PM10 = mean(PM10,na.rm=TRUE),
  Avg_weekday_SO2 = mean(SO2,na.rm=TRUE),
  Avg_weekday_NO2 = mean(NO2,na.rm=TRUE),
  Avg_weekday_CO = mean(CO,na.rm=TRUE),
  Avg_weekday_O3 = mean(O3,na.rm=TRUE),
  Avg_weekday_TEMP = mean(TEMP,na.rm=TRUE),
  Avg_weekday_PRES = mean(PRES,na.rm=TRUE),
  Avg_weekday_DEWP = mean(DEWP,na.rm=TRUE),
  Total_weekday_Rain = sum(RAIN,na.rm=TRUE),
```

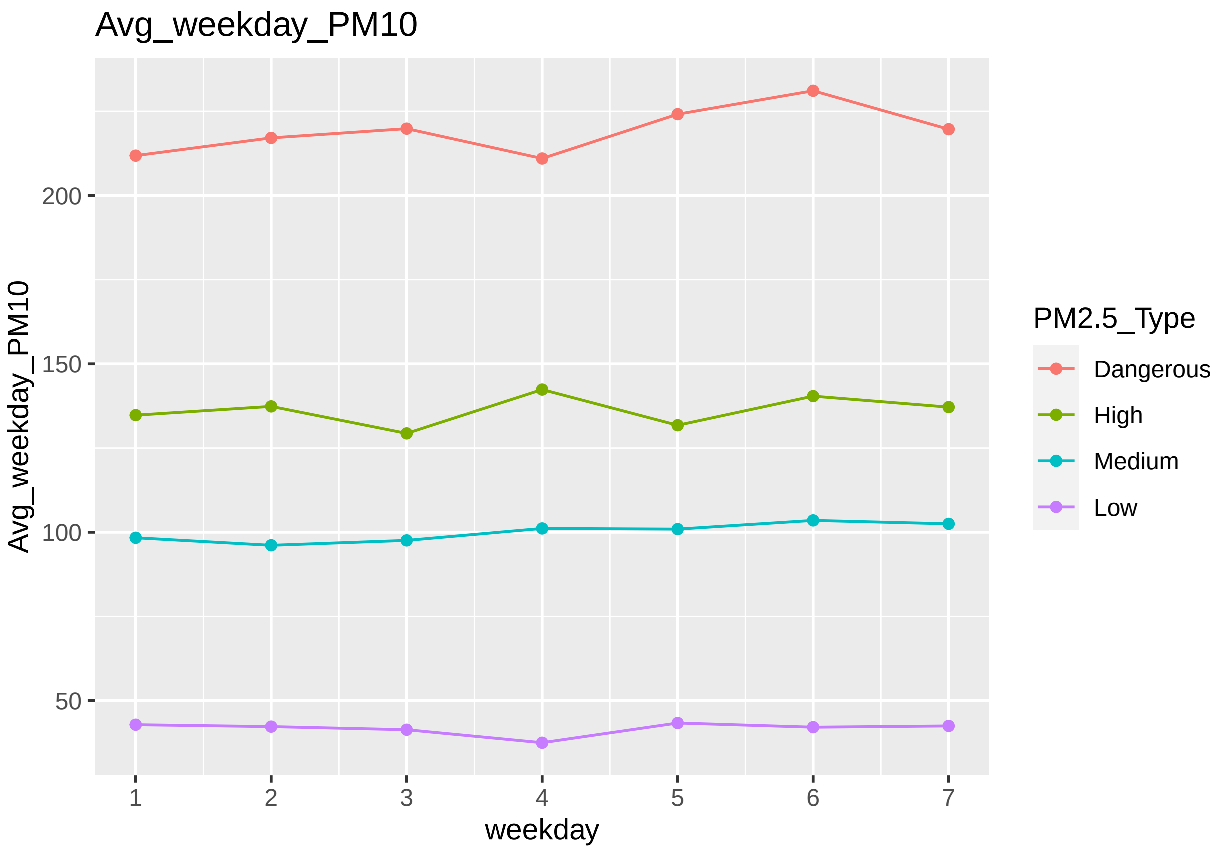
```
  Avg_weekday_WSPM = mean(WSPM, na.rm=TRUE),
  Freq_weekday_wd = getmode(wd), .groups = "drop")

PM_weekday$weekday = as.numeric(PM_weekday$weekday)

# Need to convert into dataframe to run the for loop below
PM_weekday = as.data.frame(PM_weekday)

# Option 1 - With values in the plot (Can remove)
ggplot(PM_weekday, aes(x=weekday, y=Avg_weekday_PM2.5, color=PM2.5_Type,)) + geom_point(aes(color=PM2.5_
```

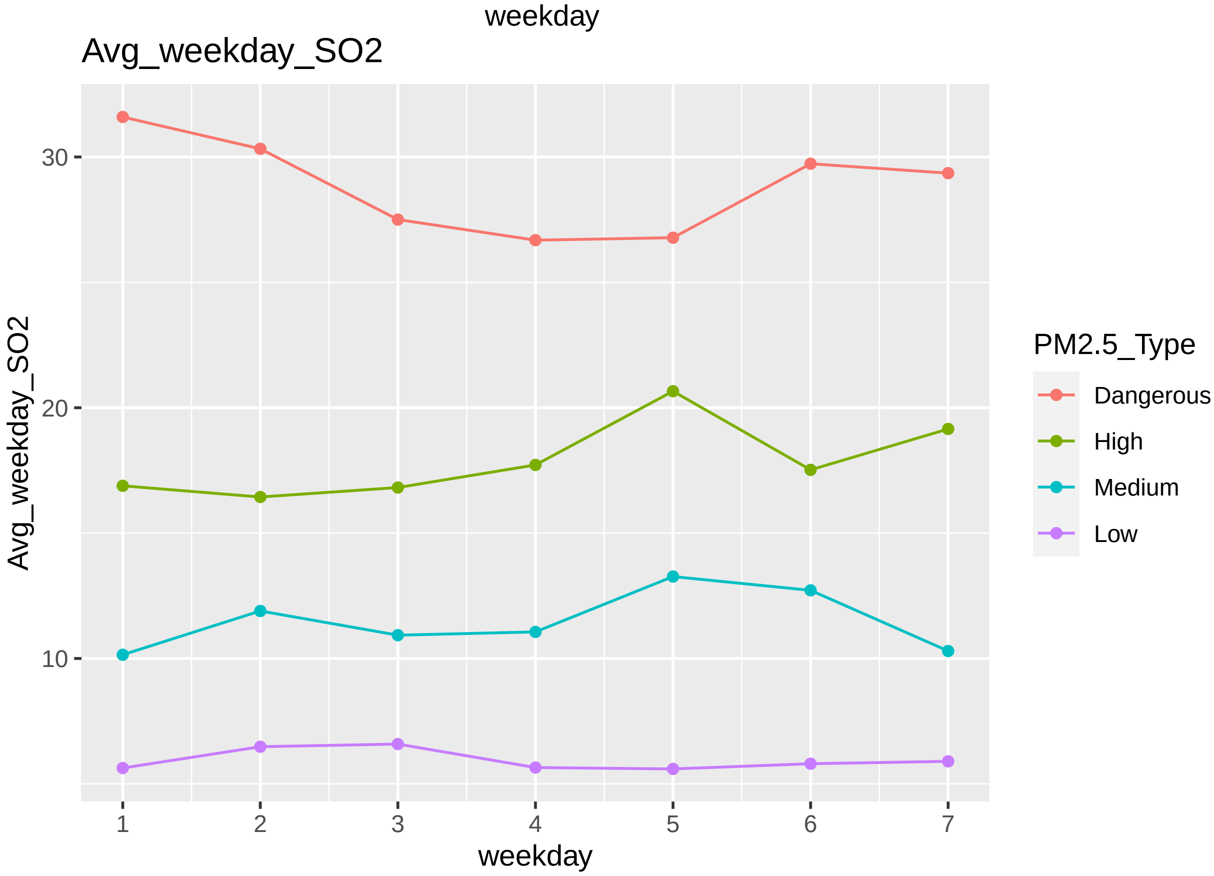


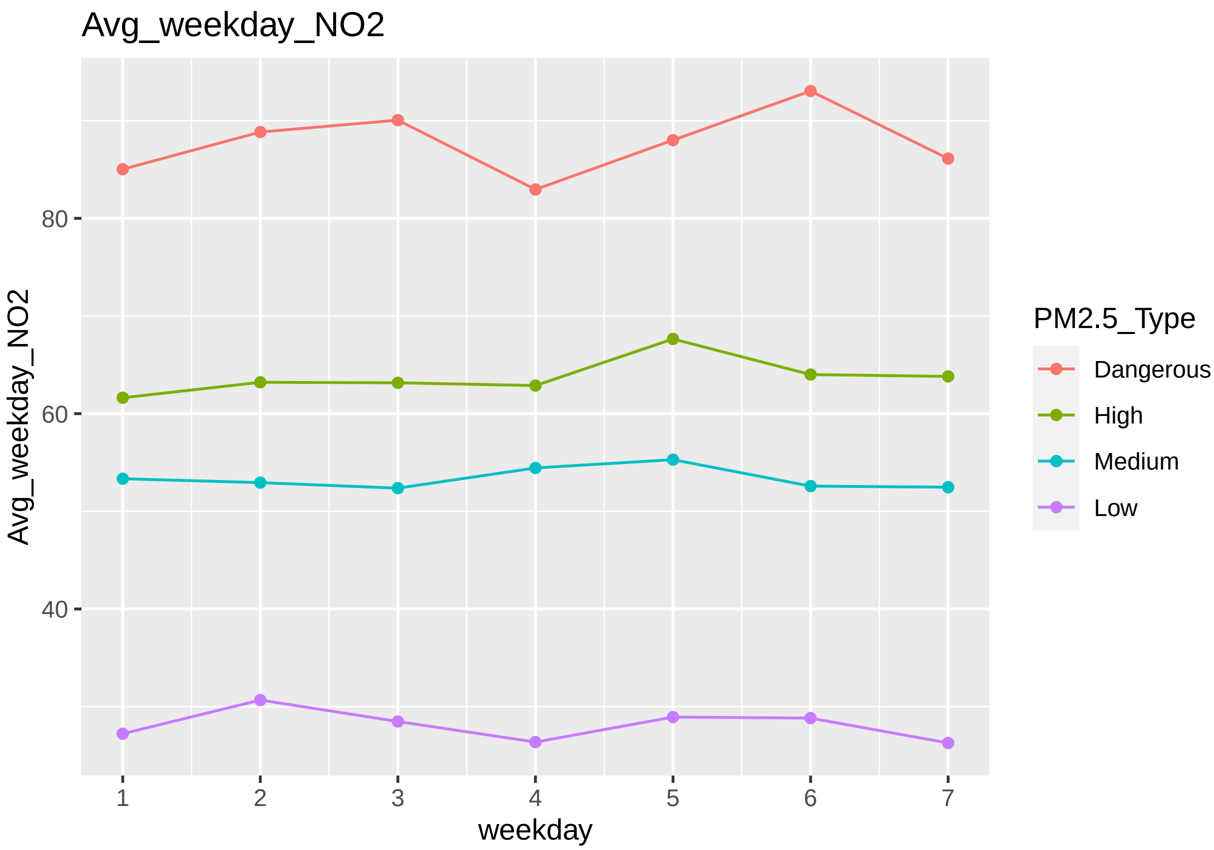

```
# Max_weekday_PM2.5
ggplot(PM_weekday, aes(x=weekday, y=Max_weekday_PM2.5, color=PM2.5_Type)) + geom_point(aes(color=PM2.5_
```

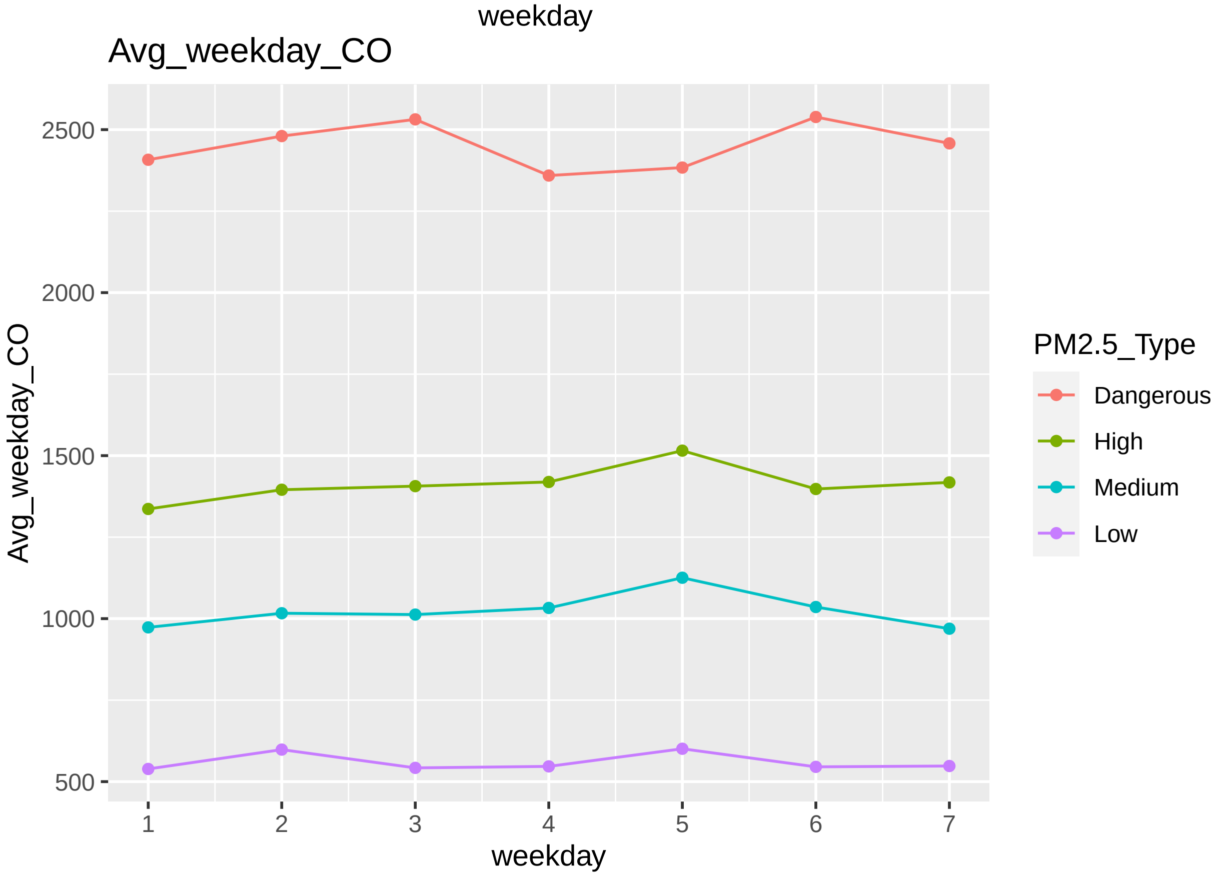


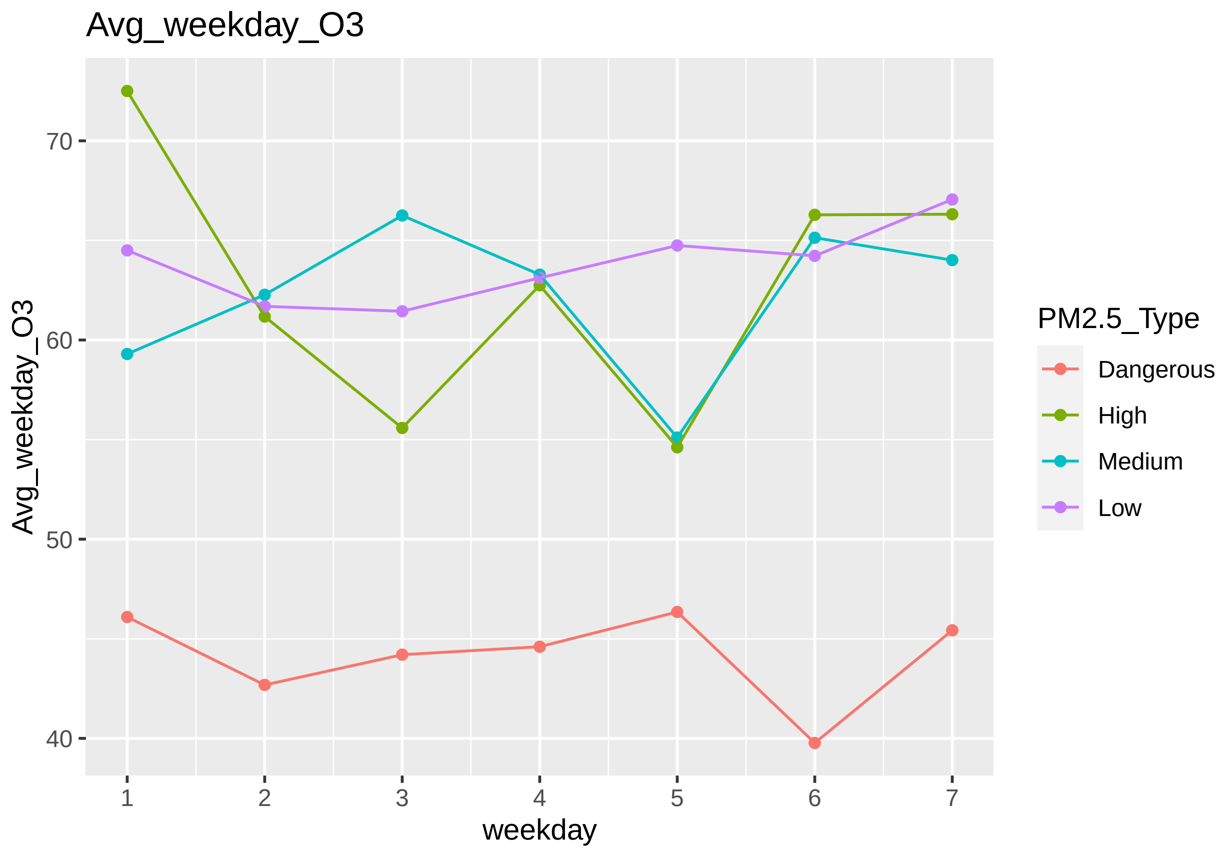

```r
# Multiple plots with for loop
for (i in c(5:14)) {
  print(ggplot(PM_weekday, aes(x=weekday, y=PM_weekday[, i], color=PM2.5_Type)) + geom_point(aes(color=I
}
```

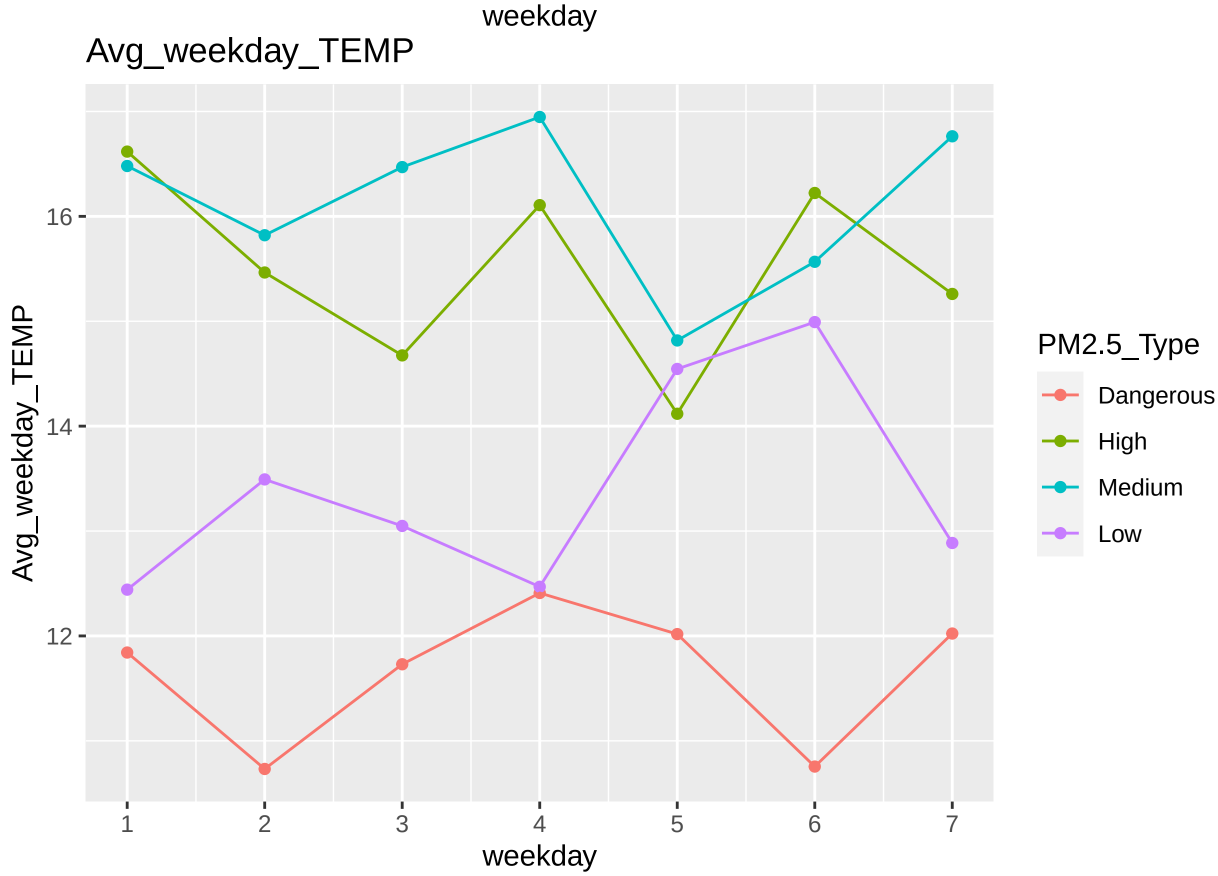

Avg_weekday_PM10
Avg_weekday_PM10
200
150
100
50
1
2
3
4
5
6
7
weekday
PM2.5_Type
Dangerous
High
Medium
Low

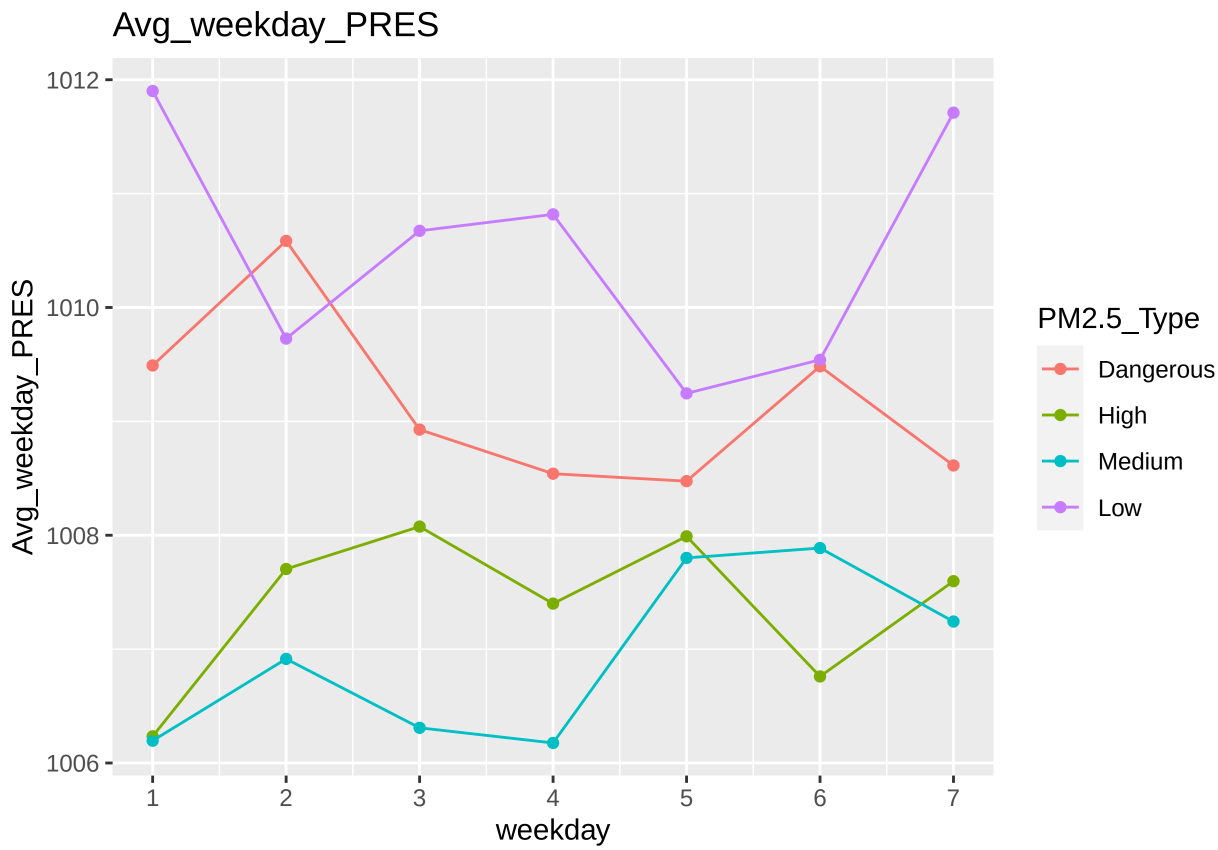

Avg_weekday_SO2
Avg_weekday_SO2
30
20
10
1
2
3
4
5
6
7
weekday
PM2.5_Type
Dangerous
High
Medium
Low

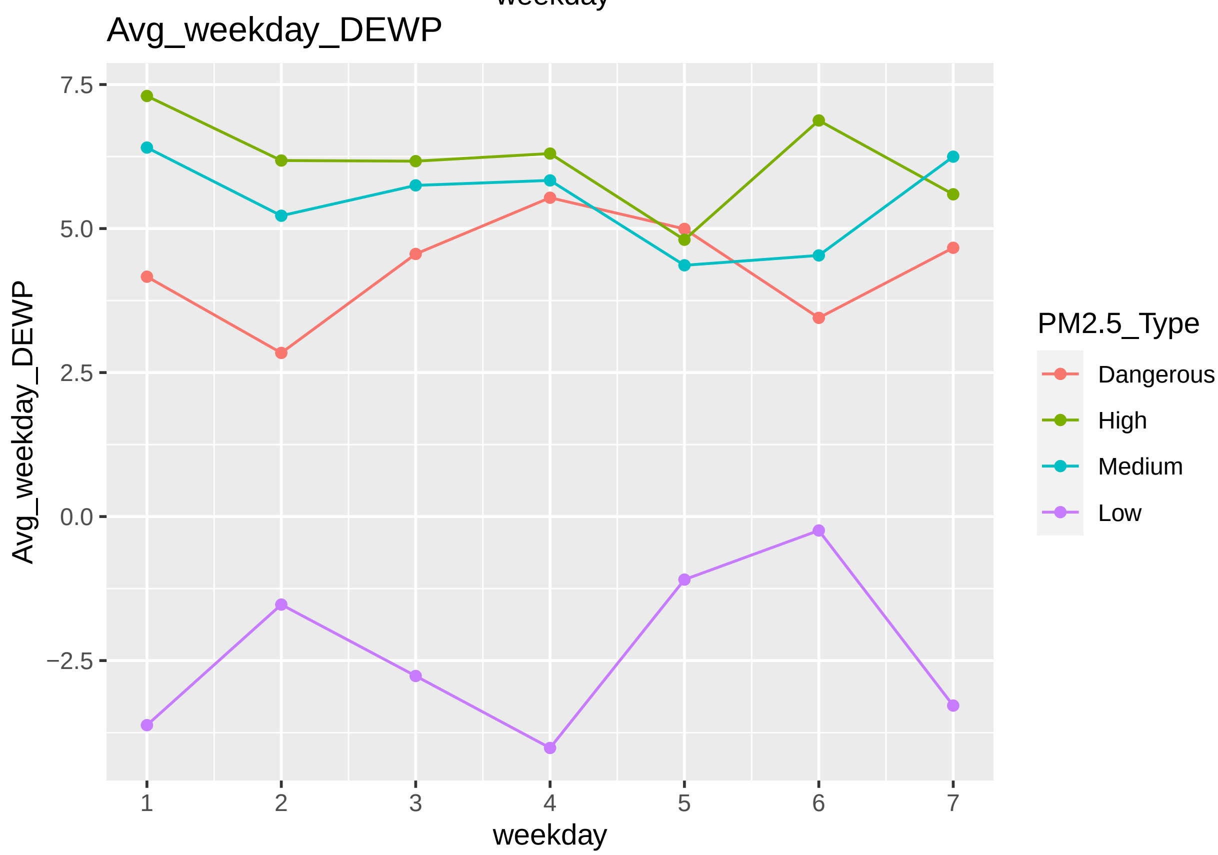

Avg_weekday_NO2
Avg_weekday_NO2
weekday
80
60
40
1
2
3
4
5
6
7
PM2.5_Type
Dangerous
High
Medium
Low

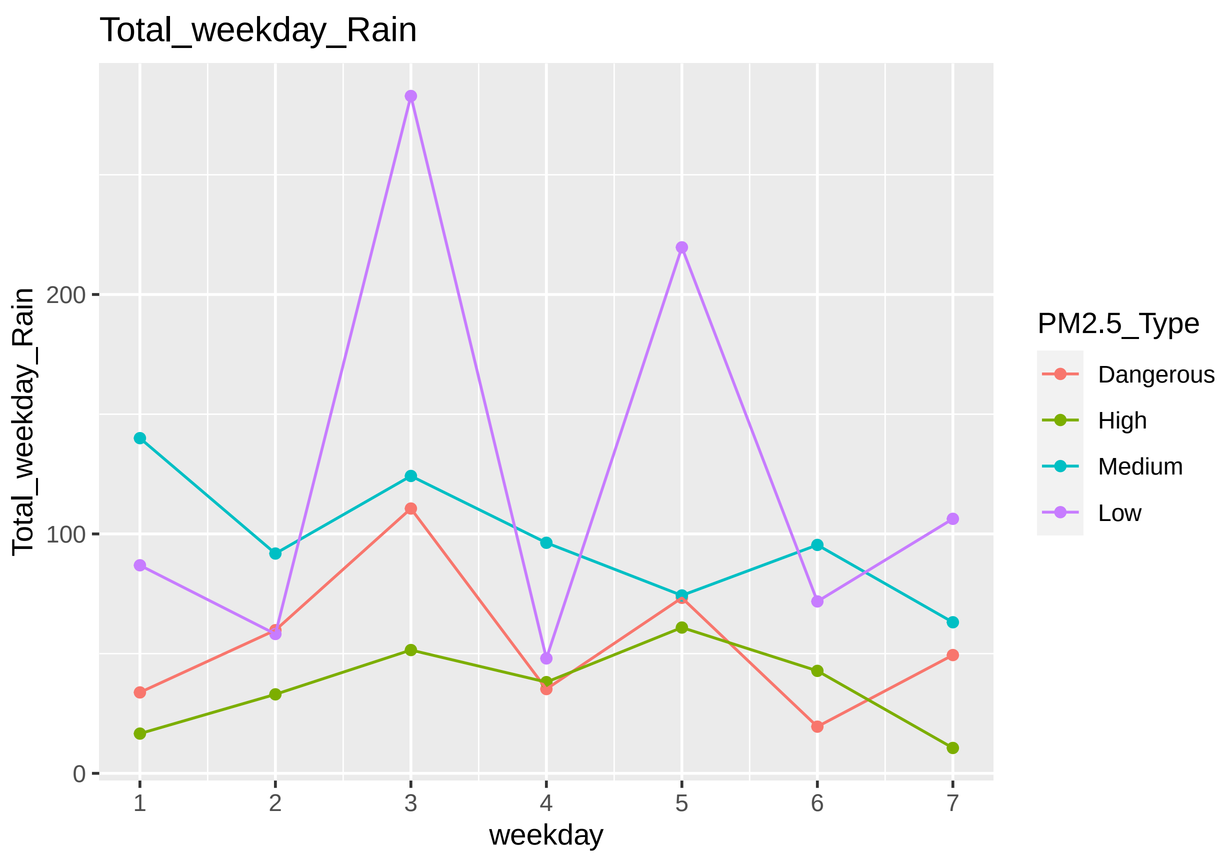

Avg_weekday_CO
Avg_weekday_CO
weekday
2500
2000
1500
1000
500
1
2
3
4
5
6
7
PM2.5_Type
Dangerous
High
Medium
Low

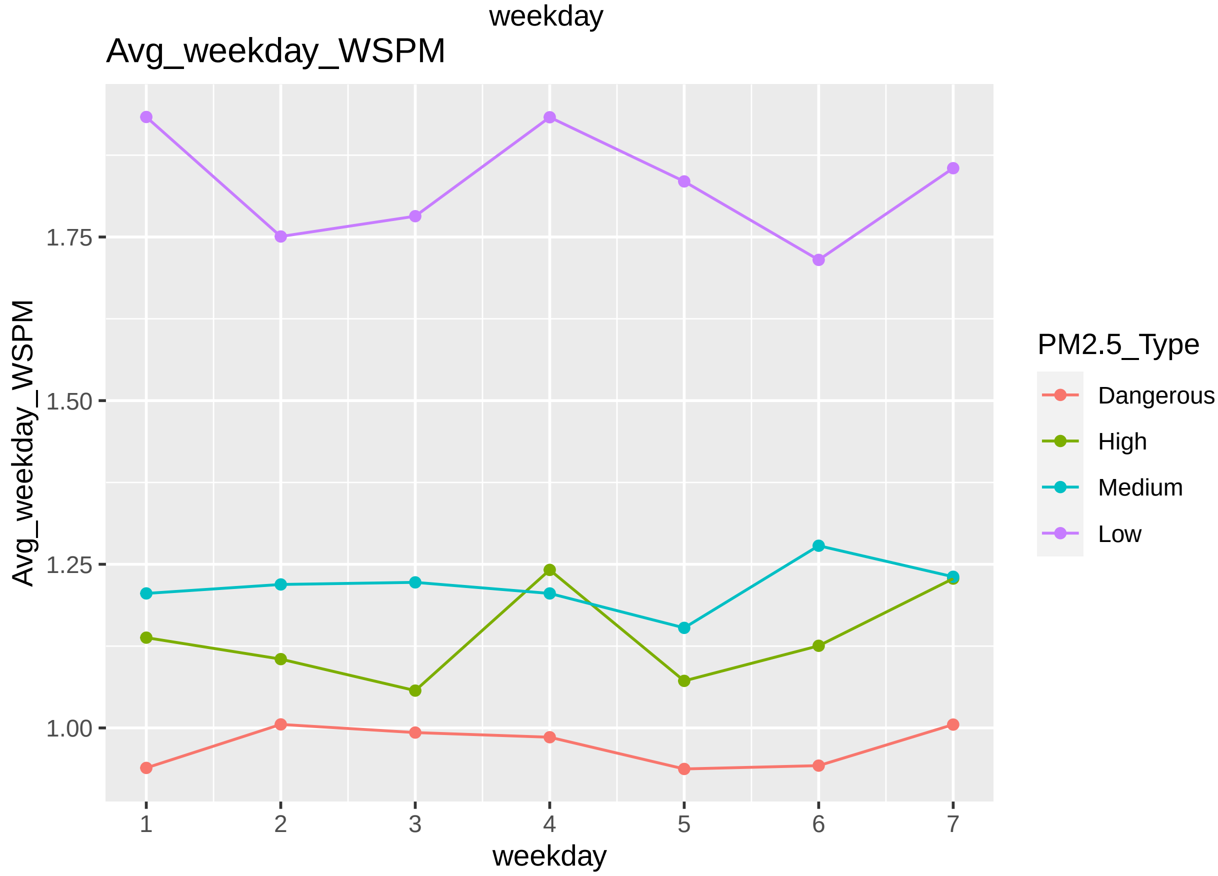

Avg_weekday_O3
Avg_weekday_O3
70
60
50
40
1
2
3
4
5
6
7
weekday
PM2.5_Type
Dangerous
High
Medium
Low


Avg_weekday_TEMP
Avg_weekday_TEMP
16
14
12
1
2
3
4
5
6
7
weekday
PM2.5_Type
Dangerous
High
Medium
Low

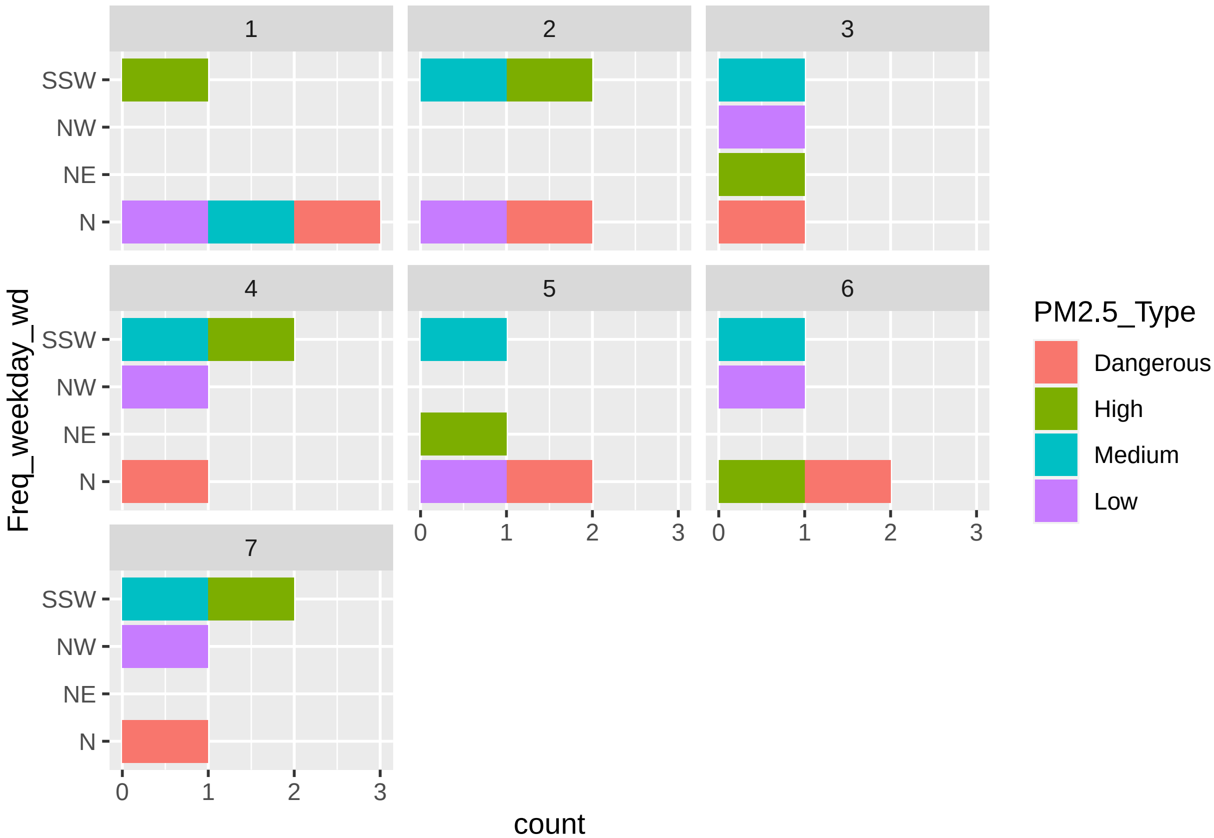

Avg_weekday_PRES
1012
1010
1008
1006
Avg_weekday_PRES
1
2
3
4
5
6
7
weekday
PM2.5_Type
Dangerous
High
Medium
Low


Avg_weekday_DEWP
7.5
5.0
2.5
0.0
−2.5
Avg_weekday_DEWP
1
2
3
4
5
6
7
weekday
PM2.5_Type
Dangerous
High
Medium
Low


Total_weekday_Rain
Total_weekday_Rain
200
100
0
1
2
3
4
5
6
7
weekday
PM2.5_Type
Dangerous
High
Medium
Low


Avg_weekday_WSPM
Avg_weekday_WSPM
1.75
1.50
1.25
1.00
1
2
3
4
5
6
7
weekday
PM2.5_Type
Dangerous
High
Medium
Low

```
# Plot for 'Freq_weekday_wd'
ggplot(PM_weekday, aes(y = Freq_weekday_wd, fill =PM2.5_Type)) + geom_bar() + facet_wrap(~weekday)
```



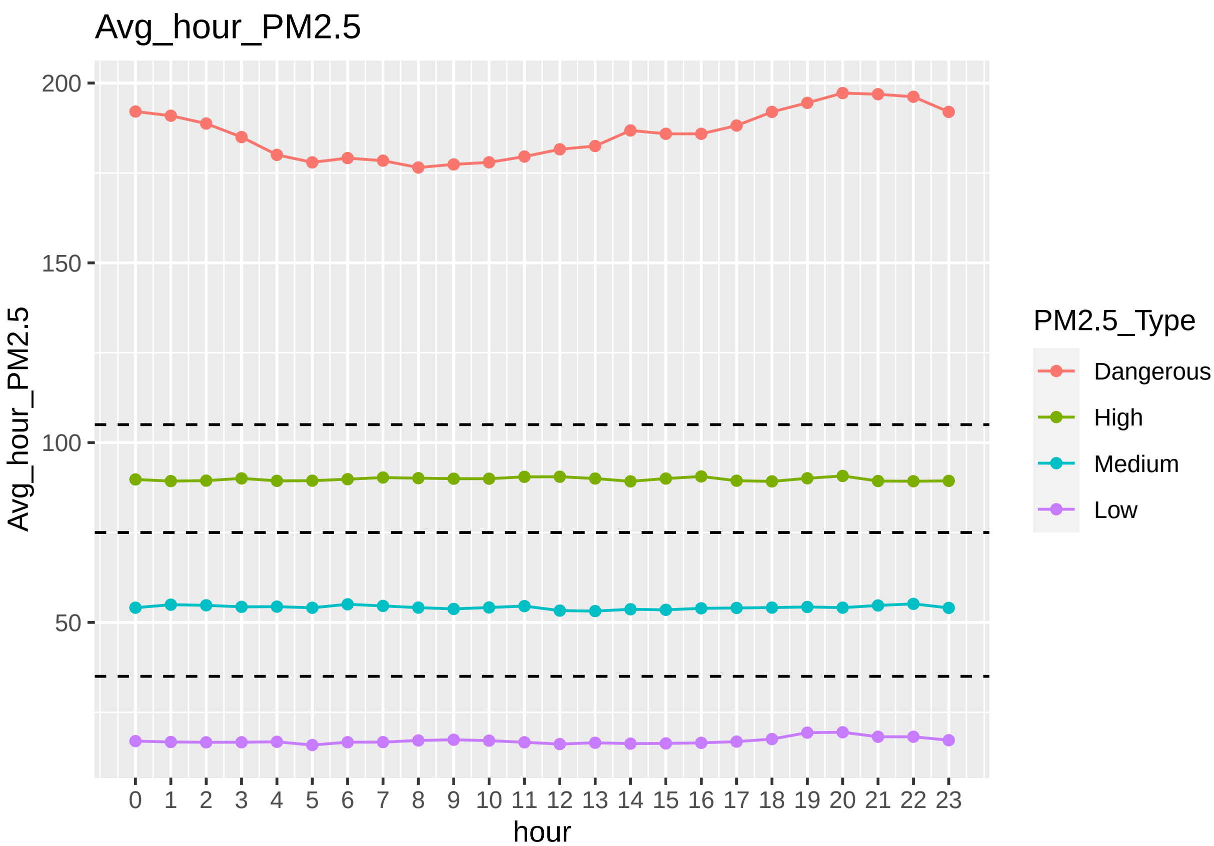
(1 = Monday, 2 = Tuesday, 3 = Wednesday, 4 = Thursday, 5 = Friday, 6 = Saturday, 7 = Sunday)

From the weekday visualizations above, both the maximum and average PM2.5 concentration start to increase from Thursday through Saturday and decrease from Sunday. We may set up an assumption or interpret this trend like this: People tend to do more outdoor activities on Friday and Saturday (Rush hours + Night outdoor activities (ex. Friends / Colleagues from work) on Friday, More Outdoor activities (ex. Family) on Saturday), therefore PM2.5 concentration is higher on these days compared to other weekdays. Based on our result, we may recommend people try not to do outdoor activities or stay outside too long on these days.

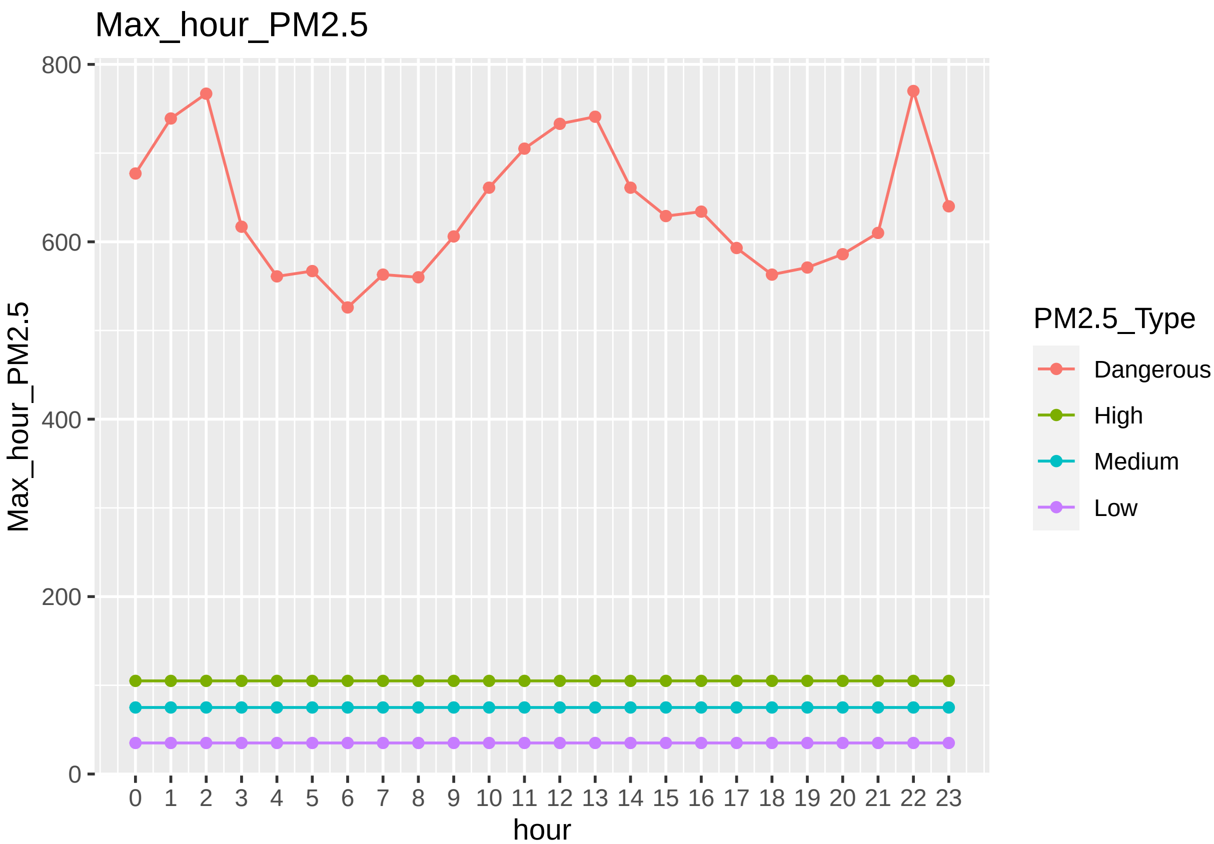
Time-series Visualizations (by hour)

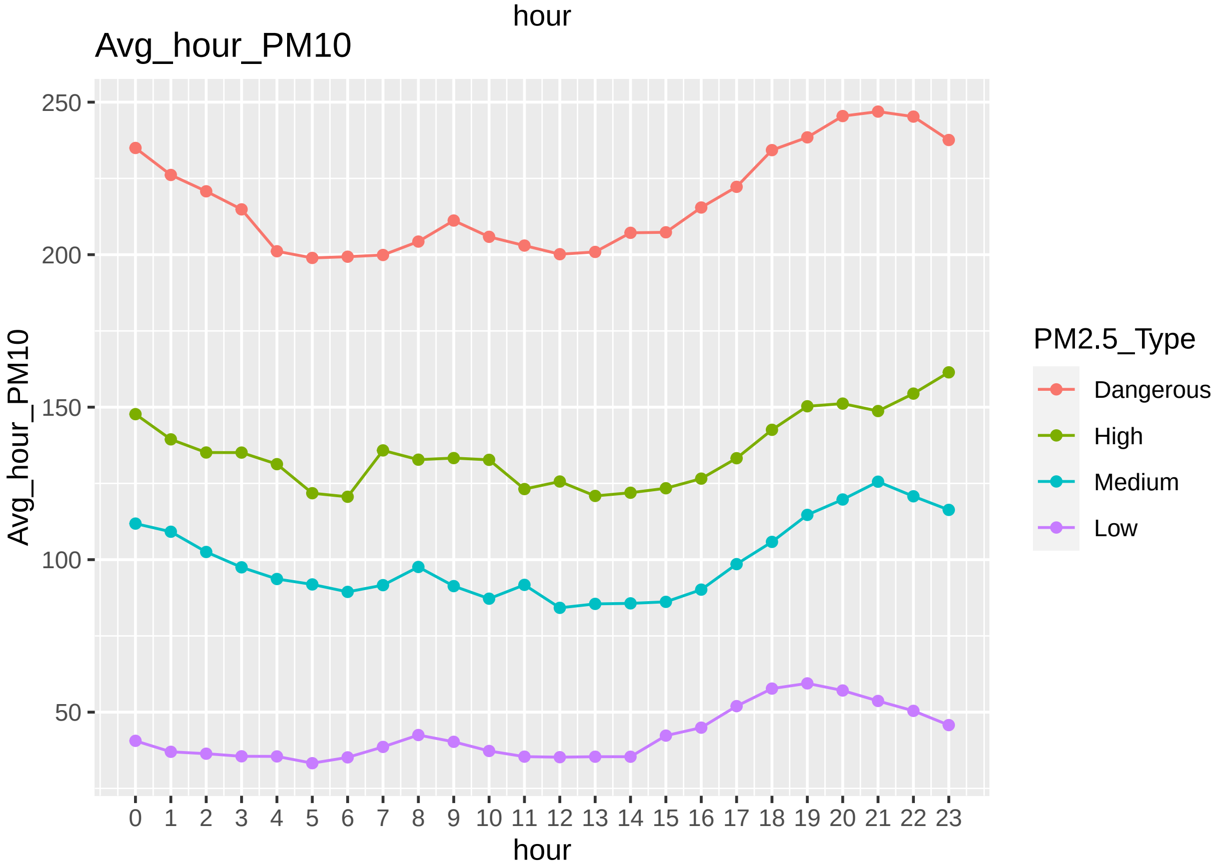
```
PM_hour = df_new %>%
  group_by(PM2.5_Type, hour) %>%
  summarise(
  Max_hour_PM2.5 = max(PM2.5, na.rm=TRUE),
  Avg_hour_PM2.5 = mean(PM2.5,na.rm=TRUE),
  Avg_hour_PM10 = mean(PM10,na.rm=TRUE),
  Avg_hour_SO2 = mean(SO2,na.rm=TRUE),
  Avg_hour_NO2 = mean(NO2,na.rm=TRUE),
  Avg_hour_CO = mean(CO,na.rm=TRUE),
  Avg_hour_O3 = mean(O3,na.rm=TRUE),
  Avg_hour_TEMP = mean(TEMP,na.rm=TRUE),
```

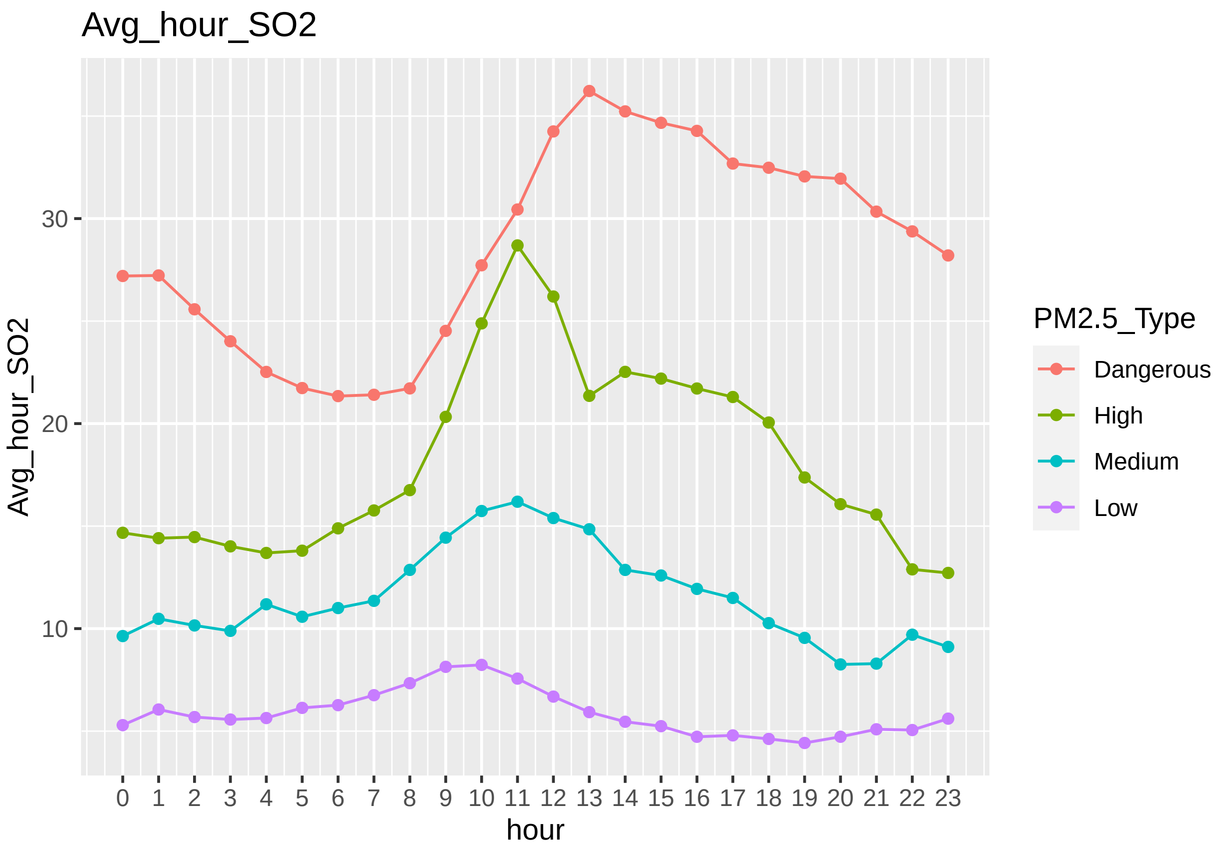
```
  Avg_hour_PRES = mean(PRES,na.rm=TRUE),
  Avg_hour_DEWP = mean(DEWP,na.rm=TRUE),
  Total_hour_Rain = sum(RAIN,na.rm=TRUE),
  Avg_hour_WSPM = mean(WSPM, na.rm=TRUE),
  Freq_hour_wd = getmode(wd), .groups = "drop")

# Need to convert into dataframe to run the for loop below
PM_hour = as.data.frame(PM_hour)

# Option 1 - With values in the plot (Can remove)
ggplot(PM_hour, aes(x=hour, y=Avg_hour_PM2.5, color=PM2.5_Type)) + geom_point(aes(color=PM2.5_Type)) + g
```

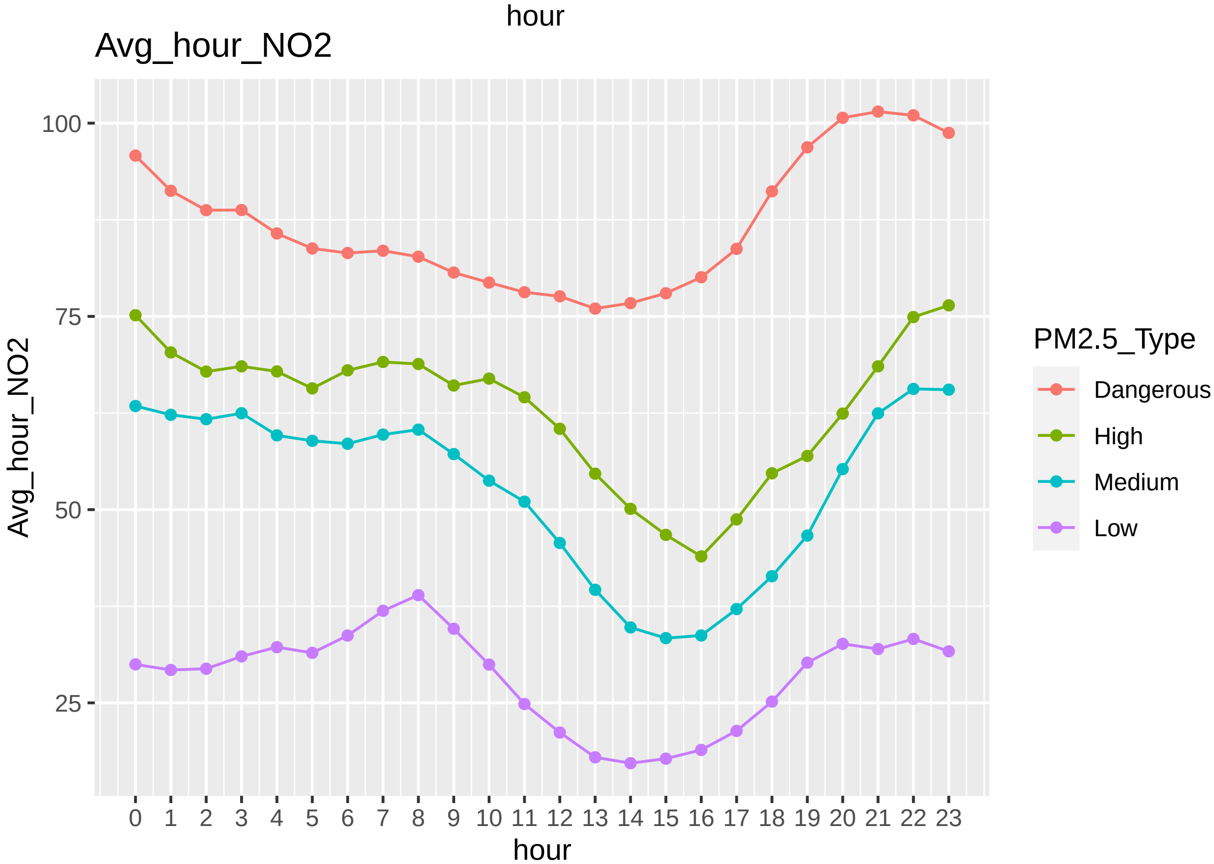


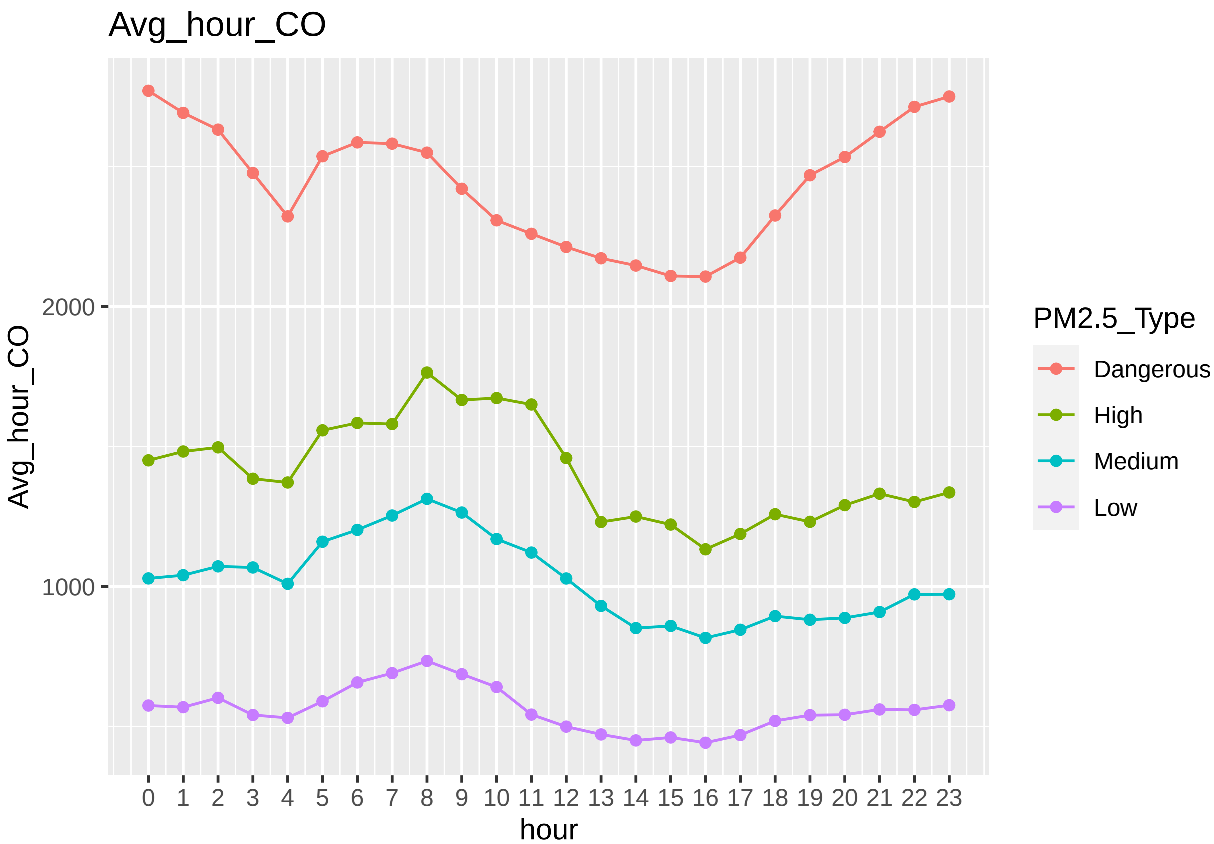

```
# Multiple plots with for loop
for (i in c(3, 5:14)) {
  print(ggplot(PM_hour, aes(x=hour, y=PM_hour[, i], color=PM2.5_Type)) + geom_point(aes(color=PM2.5_Typ
}
```

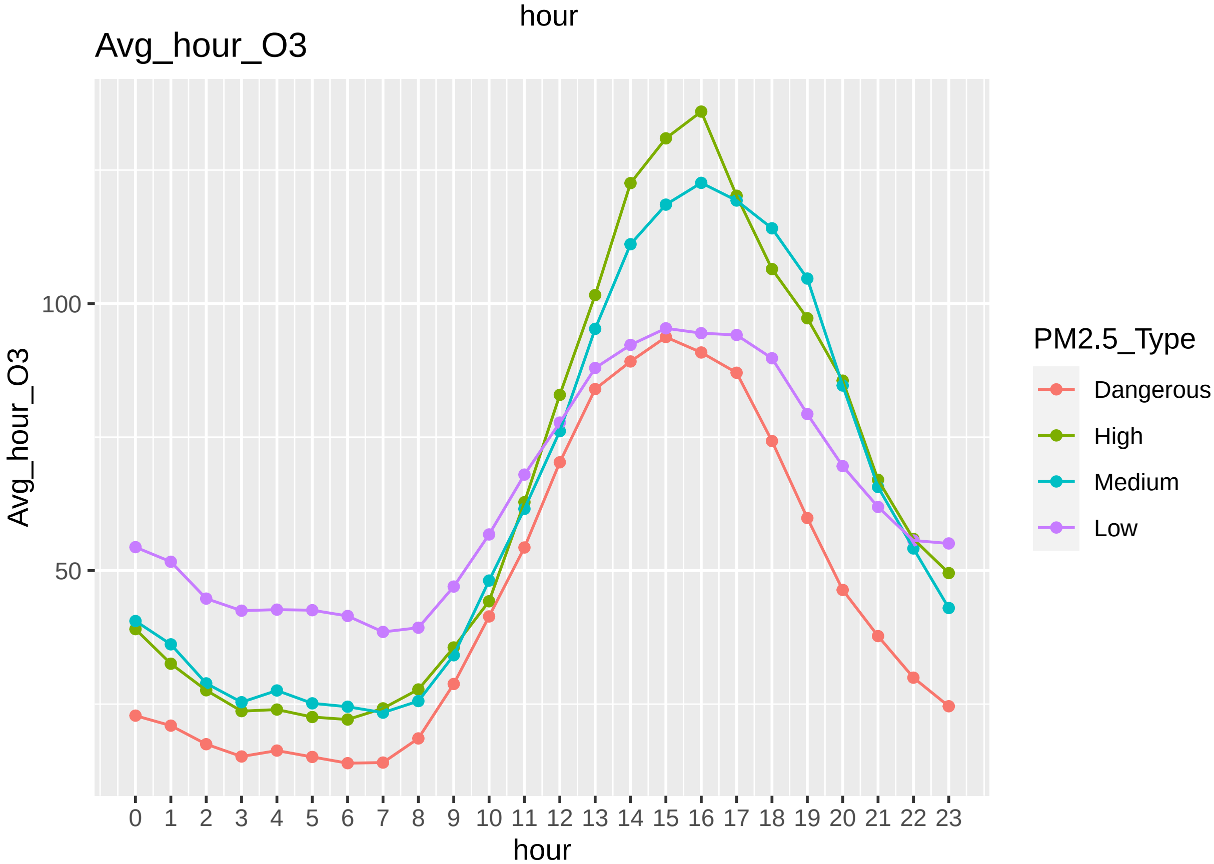

Max_hour_PM2.5
Max_hour_PM2.5
800
600
400
200
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
hour
PM2.5_Type
Dangerous
High
Medium
Low

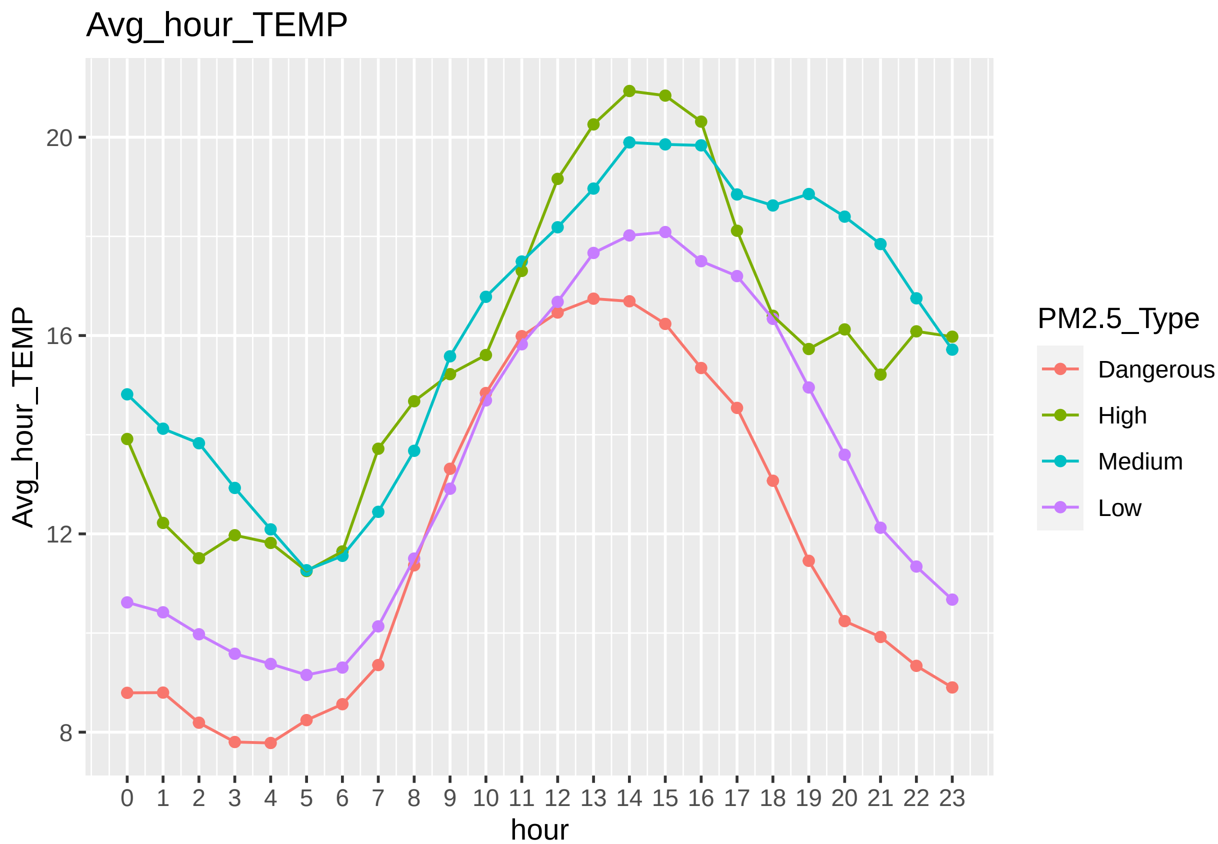

Avg_hour_PM10
Avg_hour_PM10
250
200
150
100
50
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
hour
PM2.5_Type
Dangerous
High
Medium
Low

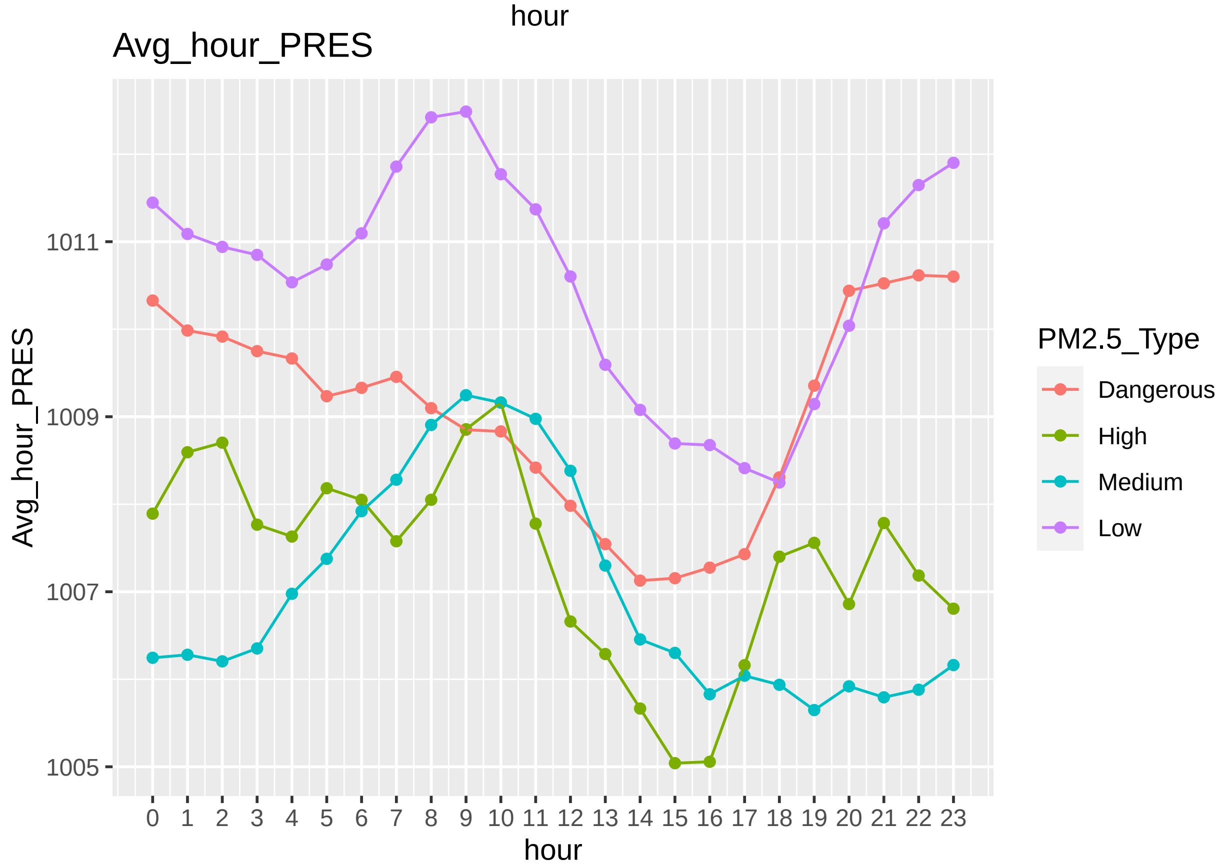

Avg_hour_SO2
Avg_hour_SO2
30
20
10
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
hour
PM2.5_Type
Dangerous
High
Medium
Low

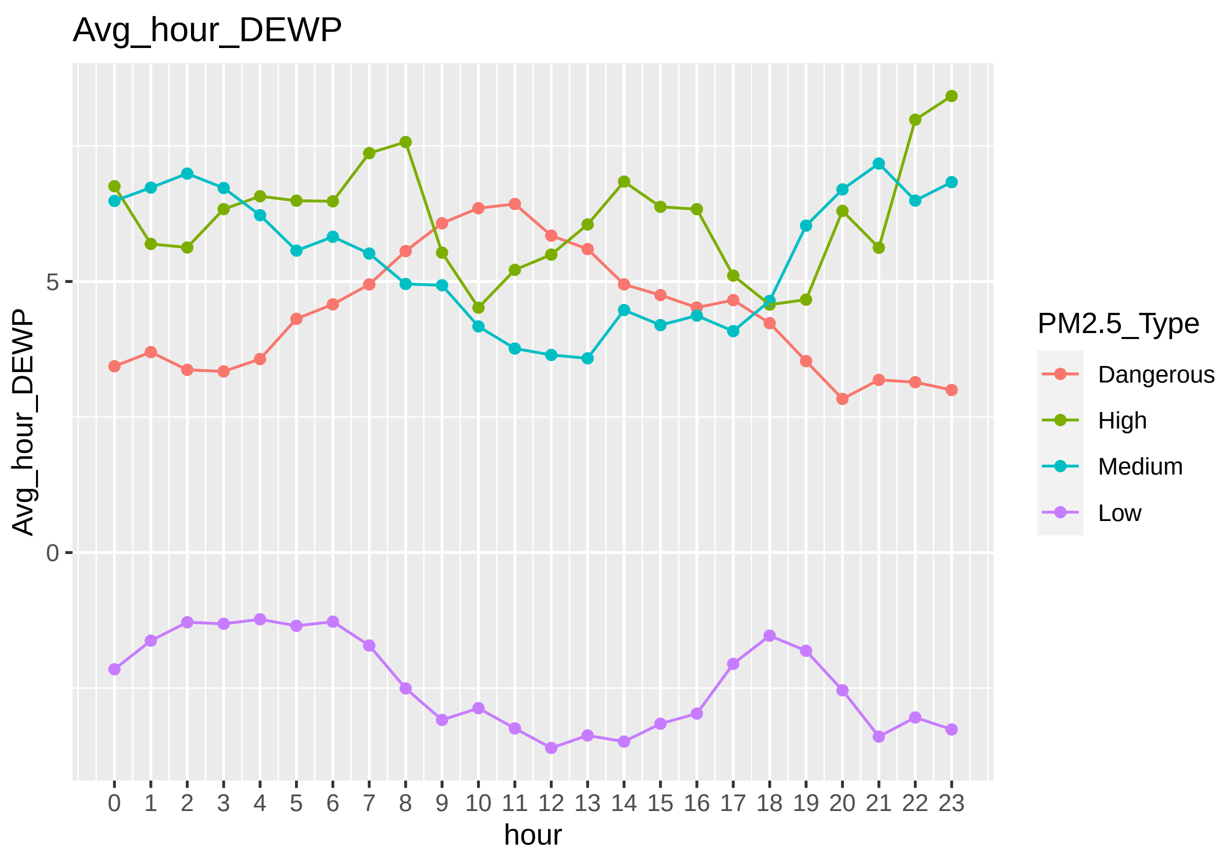

Avg_hour_NO2
Avg_hour_NO2
100
75
50
25
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
hour
PM2.5_Type
Dangerous
High
Medium
Low

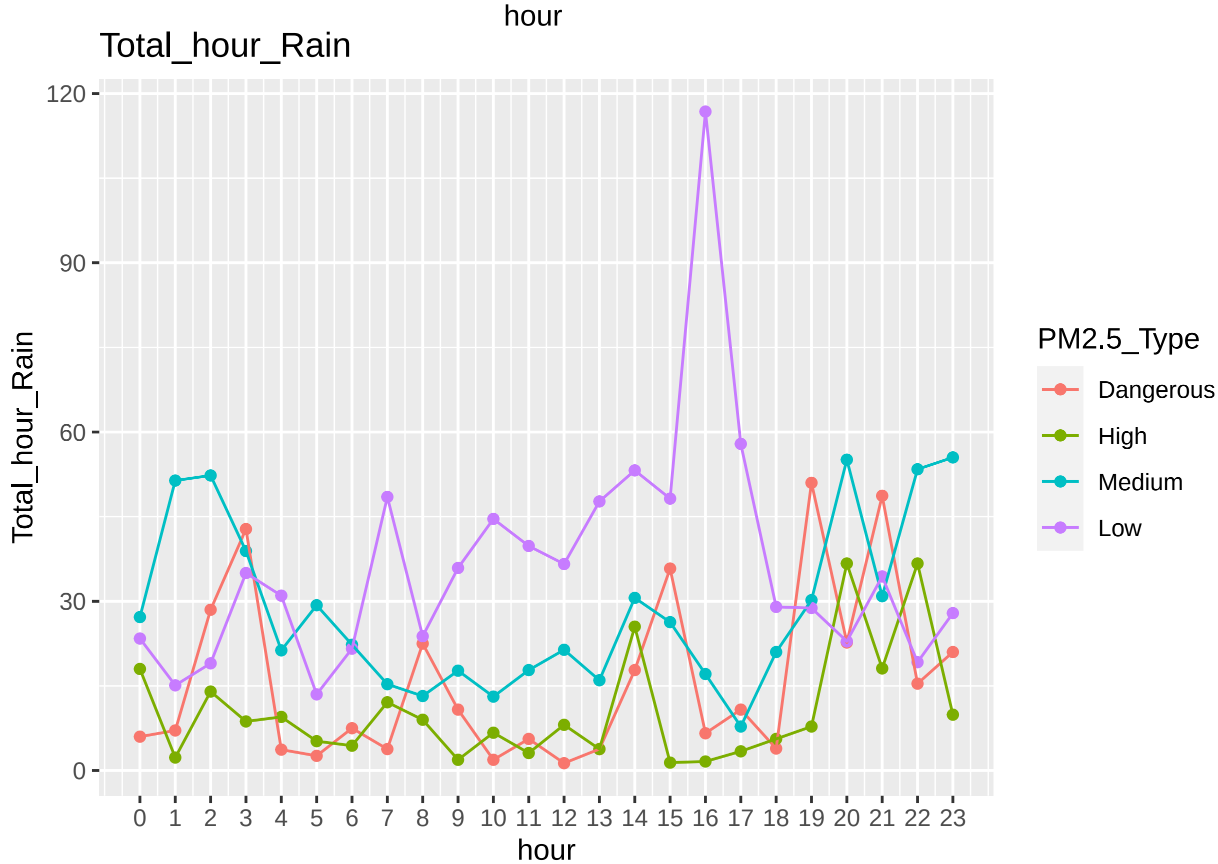

Avg_hour_CO
Avg_hour_CO
2000
1000
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
hour
PM2.5_Type
Dangerous
High
Medium
Low

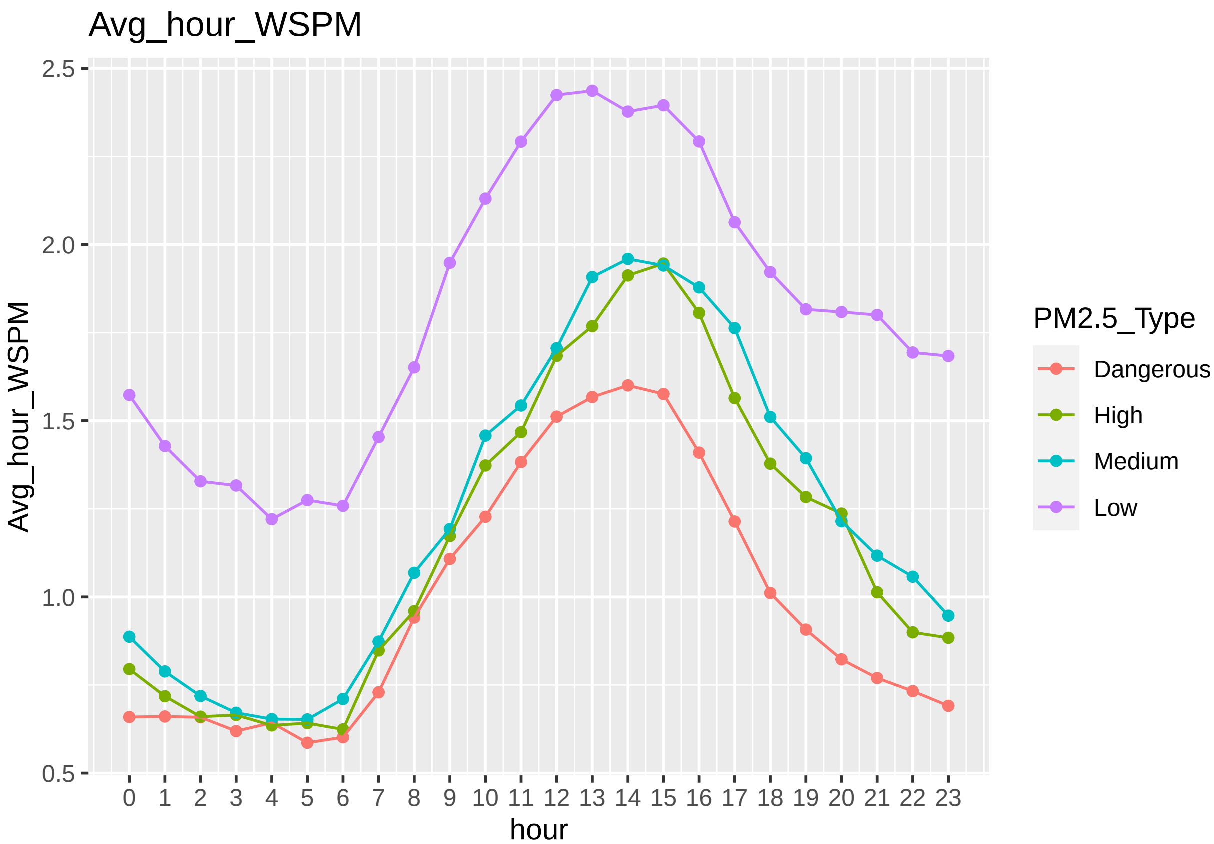

Avg_hour_O3
Avg_hour_O3
100
50
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
hour
PM2.5_Type
Dangerous
High
Medium
Low


Avg_hour_TEMP
Avg_hour_TEMP
20
16
12
8
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
hour
PM2.5_Type
Dangerous
High
Medium
Low

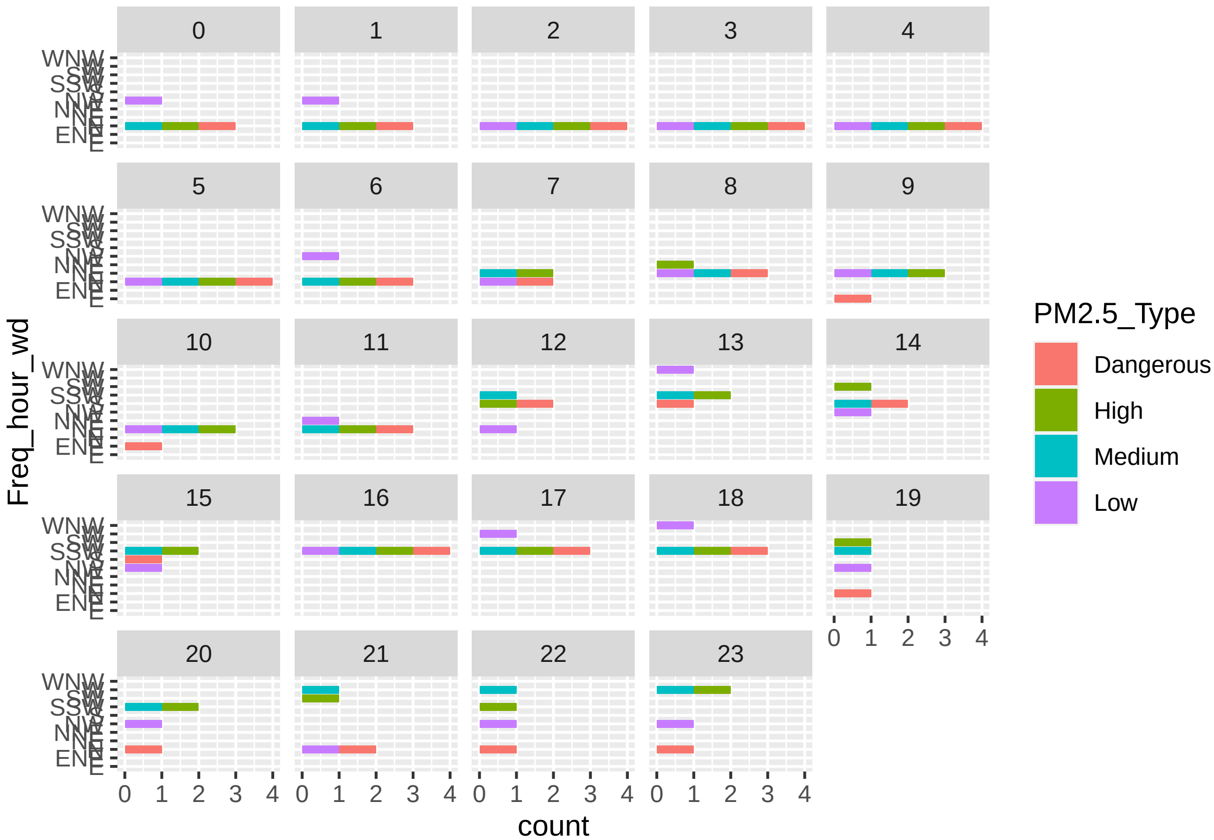

Avg_hour_PRES
Avg_hour_PRES
1011
1009
1007
1005
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
hour
PM2.5_Type
Dangerous
High
Medium
Low


Avg_hour_DEWP
Avg_hour_DEWP
5
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
hour
PM2.5_Type
Dangerous
High
Medium
Low


Total_hour_Rain
Total_hour_Rain
120
90
60
30
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
hour
PM2.5_Type
Dangerous
High
Medium
Low




```
# Plot for 'Freq_hour_wd'
ggplot(PM_hour, aes(y = Freq_hour_wd, fill =PM2.5_Type)) + geom_bar() + facet_wrap(~hour)
```



Although the average PM2.5 plot shows relatively mild spikes compared to the maximum PM2.5, both of them are higher in the nighttime (8pm-2am) and around noon (12-2pm), and lower in the early morning and afternoon. One major factor that could affect PM2.5 would be rush hours, and this result is somewhat interesting considering that most rush hours are either in the early morning (6-9am) or early evening (6-8pm). The PM2.5 concentration is actually lower in these time, which could indicate that it takes some time for PM2.5 level to increase. Based on our results, we may suggest people not to go outside during these time periods.

To sum up, we were able to see some significant trends of PM2.5 concentration based on different time-series. There were noticeable ups in the winter months and downs in the summer months. Also, Friday and Saturday were more likely to have higher PM2.5 concentration compared to other weekdays. Lastly, we would advise people try not to do any outdoor activity during the nighttime and around noon considering these time periods had relatively higher spikes. Comparing with the PM2.5 concentration, there seemed to have some pollutants that followed similar trend such as PM10, NO2, and CO whereas TEMP (temperature) and WSPM (wind speed) showed the opposite trend.

Linear Regression

Since we want to predict average PM2.5 level of a specific day based on its yesterday's various features values, we would like to create a new column called 'Max_Tmrw_PM2.5'. (ex. We would like to predict March 14th's PM2.5 level with March 13th's information)

New dataframe (for Regression)

```
PM_Daily = df_new %>%
  group_by(Date_ymd) %>%
  summarise(
  Max_Day_PM2.5 = max(PM2.5,na.rm=TRUE),
  Avg_Day_PM10 = mean(PM10,na.rm=TRUE),
  Avg_Day_SO2 = mean(SO2,na.rm=TRUE),
  Avg_Day_NO2 = mean(NO2,na.rm=TRUE),
  Avg_Day_CO = mean(CO,na.rm=TRUE),
  Avg_Day_O3 = mean(O3,na.rm=TRUE),
  Avg_Day_TEMP = mean(TEMP,na.rm=TRUE),
  Avg_Day_PRES = mean(PRES,na.rm=TRUE),
  Avg_Day_DEWP = mean(DEWP,na.rm=TRUE),
  Total_Day_Rain = sum(RAIN,na.rm=TRUE),
  Avg_Day_WSPM = mean(WSPM, na.rm=TRUE),
  Freq_Day_wd = getmode(wd))

head(PM_Daily)
```

```
## # A tibble: 6 x 13
##   Date_ymd   Max_Day_P~1 Avg_D~2 Avg_D~3 Avg_D~4 Avg_D~5 Avg_D~6 Avg_D~7 Avg_D~8
##   <date>           <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 2013-03-01          16    16.9    7.39    14.2    870.    77.1   1.69    1025.
## 2 2013-03-02          98    51.4   37.3     37.6   1479.    39.6   0.821   1025.
## 3 2013-03-03         150   120.    47.5     63.3   2350.    33.7   6.56    1013.
## 4 2013-03-04          83    51.3   18.7     32.7   1171.    67.2   9.80    1016.
## 5 2013-03-05         228   173.    75.1     73.5   1382.    84.4   6.75    1009.
## 6 2013-03-06         292   256.   114.     163.    2287.    17.5   7.35    1006.
## # ... with 4 more variables: Avg_Day_DEWP <dbl>, Total_Day_Rain <dbl>,
## #   Avg_Day_WSPM <dbl>, Freq_Day_wd <fct>, and abbreviated variable names
## #   1: Max_Day_PM2.5, 2: Avg_Day_PM10, 3: Avg_Day_SO2, 4: Avg_Day_NO2,
## #   5: Avg_Day_CO, 6: Avg_Day_O3, 7: Avg_Day_TEMP, 8: Avg_Day_PRES
```

Dataset Modification for Regression

```
# Creating a new column
PM_Daily$Max_Tmrw_PM2.5 = 0

for (i in 1:dim(PM_Daily)[1]) {
  if (i < dim(PM_Daily)[1]) {
    # Move Max_Day_PM2.5 above
    PM_Daily$Max_Tmrw_PM2.5[i] = PM_Daily$Max_Day_PM2.5[i+1]
  } else {
    # Last row does not have its tomorrow's average PM2.5 level value
    PM_Daily$Max_Tmrw_PM2.5[i] = "X"
  }
}

# Check the last 5 rows
tail(PM_Daily)
```

```
## # A tibble: 6 x 14
```

```
##   Date_ymd   Max_Day_P~1 Avg_D~2 Avg_D~3 Avg_D~4 Avg_D~5 Avg_D~6 Avg_D~7 Avg_D~8
##   <date>           <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 2017-02-23          50    39.8    6.65    25.6    650     48.5    2.38   1021.
## 2 2017-02-24          46    52.7    9.71    65.5    900     36.9    5.35   1017.
## 3 2017-02-25          19    24.7    4.46    12.9    433.    54.5    8.21   1015.
## 4 2017-02-26          60    71.3   10.5     48.3    792.    27.2    6.54   1017.
## 5 2017-02-27          99   114.    16.2     69.8   1425     19.6    7.22   1014.
## 6 2017-02-28          34    34      5.71    28.8    529.    41.8    9.75   1011.
## # ... with 5 more variables: Avg_Day_DEWP <dbl>, Total_Day_Rain <dbl>,
## #   Avg_Day_WSPM <dbl>, Freq_Day_wd <fct>, Max_Tmrw_PM2.5 <chr>, and
## #   abbreviated variable names 1: Max_Day_PM2.5, 2: Avg_Day_PM10,
## #   3: Avg_Day_SO2, 4: Avg_Day_NO2, 5: Avg_Day_CO, 6: Avg_Day_O3,
## #   7: Avg_Day_TEMP, 8: Avg_Day_PRES
```

Additional Data Manipulation & Subsetting

```r
# Dataset without the last row
df_reg = PM_Daily[1:dim(PM_Daily)[1]-1, ]

# Dataset excluding some unnecessary columns for the regression
df_reg = df_reg[, c(-1)]

# Convert 'Max_Tmrw_PM2.5' into double type
df_reg$Max_Tmrw_PM2.5 = as.double(df_reg$Max_Tmrw_PM2.5)

tail(df_reg)
```

```
## # A tibble: 6 x 13
##   Max_Day_PM2.5 Avg_Da~1 Avg_D~2 Avg_D~3 Avg_D~4 Avg_D~5 Avg_D~6 Avg_D~7 Avg_D~8
##           <dbl>    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1           156     79.3    5.94    29.5    959.    36.2  -0.959   1016.   -7.83
## 2            50     39.8    6.65    25.6    650     48.5   2.38    1021.  -13.3
## 3            46     52.7    9.71    65.5    900     36.9   5.35    1017.  -12.5
## 4            19     24.7    4.46    12.9    433.    54.5   8.21    1015.  -12.4
## 5            60     71.3   10.5     48.3    792.    27.2   6.54    1017.   -8.8
## 6            99    114.    16.2     69.8   1425     19.6   7.22    1014.   -7.86
## # ... with 4 more variables: Total_Day_Rain <dbl>, Avg_Day_WSPM <dbl>,
## #   Freq_Day_wd <fct>, Max_Tmrw_PM2.5 <dbl>, and abbreviated variable names
## #   1: Avg_Day_PM10, 2: Avg_Day_SO2, 3: Avg_Day_NO2, 4: Avg_Day_CO,
## #   5: Avg_Day_O3, 6: Avg_Day_TEMP, 7: Avg_Day_PRES, 8: Avg_Day_DEWP
```

Missing Values in df_reg

```r
# Total number of rows with at least one NA
missing_val_reg = length(unique(which(is.na(df_reg), arr.ind = TRUE)[, 1]))
missing_val_reg
```

```
## [1] 43
```

```r
# Check the number of missing values in each column
colSums(is.na(df_reg))
```

```
##  Max_Day_PM2.5   Avg_Day_PM10    Avg_Day_SO2    Avg_Day_NO2     Avg_Day_CO
##              0              0              3              3             24
##     Avg_Day_O3   Avg_Day_TEMP   Avg_Day_PRES   Avg_Day_DEWP Total_Day_Rain
##             10              0              0              0              0
##   Avg_Day_WSPM    Freq_Day_wd Max_Tmrw_PM2.5
##              0              7              0
```

```r
# Proportion of missing value rows
dim(df_reg)[1]
```

```
## [1] 1459
```

```r
prop_reg = missing_val_reg / dim(df_reg)[1]
prop_reg
```

```
## [1] 0.02947224
```

When checking the number of rows with missing values in 'df_reg' dataset, we can see that there are total 43 rows with at least one NA. Considering that the proportion of rows with missing values is only about 2.9%, which is relatively small, we would simply drop these rows and do the regression.

Drop missing values

```r
df_reg = drop_na(df_reg)
dim(df_reg)[1]
```

```
## [1] 1416
```

Split the dataset into Train & Test

```r
# Split the dataset into train and test (80% training / 20% test)
set.seed(443)
train_index = sample(nrow(df_reg), size = trunc(0.8 * nrow(df_reg)))
reg_trn = df_reg[train_index, ]
reg_tst = df_reg[-train_index, ]

head(reg_trn)
```

```
## # A tibble: 6 x 13
##   Max_Day_PM2.5 Avg_Da~1 Avg_D~2 Avg_D~3 Avg_D~4 Avg_D~5 Avg_D~6 Avg_D~7 Avg_D~8
##           <dbl>    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1            30     40.5    14.2    34.6   2383.    67.6    22.5   1001.    8.78
## 2            39     28.5     2      22.8    929.    72.2    27.5    998.   20.7
```

```
## 3           157    158.    42.3     89.6   1287.    34.5    5.18   1013.   -6.93
## 4            62     38.1    2.92    37.0    733.    17.8   15.5    1011.   10.3
## 5           314    272.    14.0    126.    2375     55.2   16.2    1009.    7.28
## 6            64     65.7    3.12    43.0    758.    65.4   22.6    1008.   14.5
## # ... with 4 more variables: Total_Day_Rain <dbl>, Avg_Day_WSPM <dbl>,
## #   Freq_Day_wd <fct>, Max_Tmrw_PM2.5 <dbl>, and abbreviated variable names
## #   1: Avg_Day_PM10, 2: Avg_Day_SO2, 3: Avg_Day_NO2, 4: Avg_Day_CO,
## #   5: Avg_Day_O3, 6: Avg_Day_TEMP, 7: Avg_Day_PRES, 8: Avg_Day_DEWP
```

Initial Linear Regression Model

```
# Initial model with the training dataset
init_mod = lm(Max_Tmrw_PM2.5 ~ ., data = reg_trn)
summary(init_mod)
```

```
##
## Call:
## lm(formula = Max_Tmrw_PM2.5 ~ ., data = reg_trn)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -483.88  -39.82   -5.42   34.94  560.43
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -9.788e+02  5.128e+02  -1.909 0.056556 .
## Max_Day_PM2.5   3.771e-01  4.168e-02   9.047  < 2e-16 ***
## Avg_Day_PM10    9.884e-02  7.039e-02   1.404 0.160573
## Avg_Day_SO2    -7.621e-01  2.051e-01  -3.716 0.000212 ***
## Avg_Day_NO2     9.333e-01  1.983e-01   4.706 2.85e-06 ***
## Avg_Day_CO      3.332e-04  4.824e-03   0.069 0.944946
## Avg_Day_O3      6.312e-01  1.085e-01   5.815 7.91e-09 ***
## Avg_Day_TEMP   -3.235e+00  8.525e-01  -3.795 0.000156 ***
## Avg_Day_PRES    1.070e+00  5.001e-01   2.140 0.032581 *
## Avg_Day_DEWP   -4.748e-03  5.399e-01  -0.009 0.992984
## Total_Day_Rain -1.168e+00  3.936e-01  -2.969 0.003053 **
## Avg_Day_WSPM   -2.664e+01  4.383e+00  -6.079 1.66e-09 ***
## Freq_Day_wdENE -1.428e+01  2.032e+01  -0.703 0.482383
## Freq_Day_wdESE  4.310e+00  2.269e+01   0.190 0.849381
## Freq_Day_wdN   -1.839e+01  1.555e+01  -1.183 0.237143
## Freq_Day_wdNE  -3.066e+01  1.610e+01  -1.904 0.057165 .
## Freq_Day_wdNNE -3.209e+01  1.679e+01  -1.911 0.056323 .
## Freq_Day_wdNNW  8.897e+00  2.033e+01   0.438 0.661784
## Freq_Day_wdNW  -1.804e+01  1.630e+01  -1.107 0.268599
## Freq_Day_wdS   -1.009e+01  1.743e+01  -0.579 0.562806
## Freq_Day_wdSE   9.571e+00  2.488e+01   0.385 0.700553
## Freq_Day_wdSSE -3.034e+01  2.061e+01  -1.472 0.141233
## Freq_Day_wdSSW  6.665e+00  1.636e+01   0.407 0.683817
## Freq_Day_wdSW   7.357e-01  1.791e+01   0.041 0.967234
## Freq_Day_wdW   -2.439e+01  1.670e+01  -1.461 0.144350
## Freq_Day_wdWNW -1.807e+01  1.734e+01  -1.042 0.297551
## Freq_Day_wdWSW  2.230e+00  1.934e+01   0.115 0.908235
```

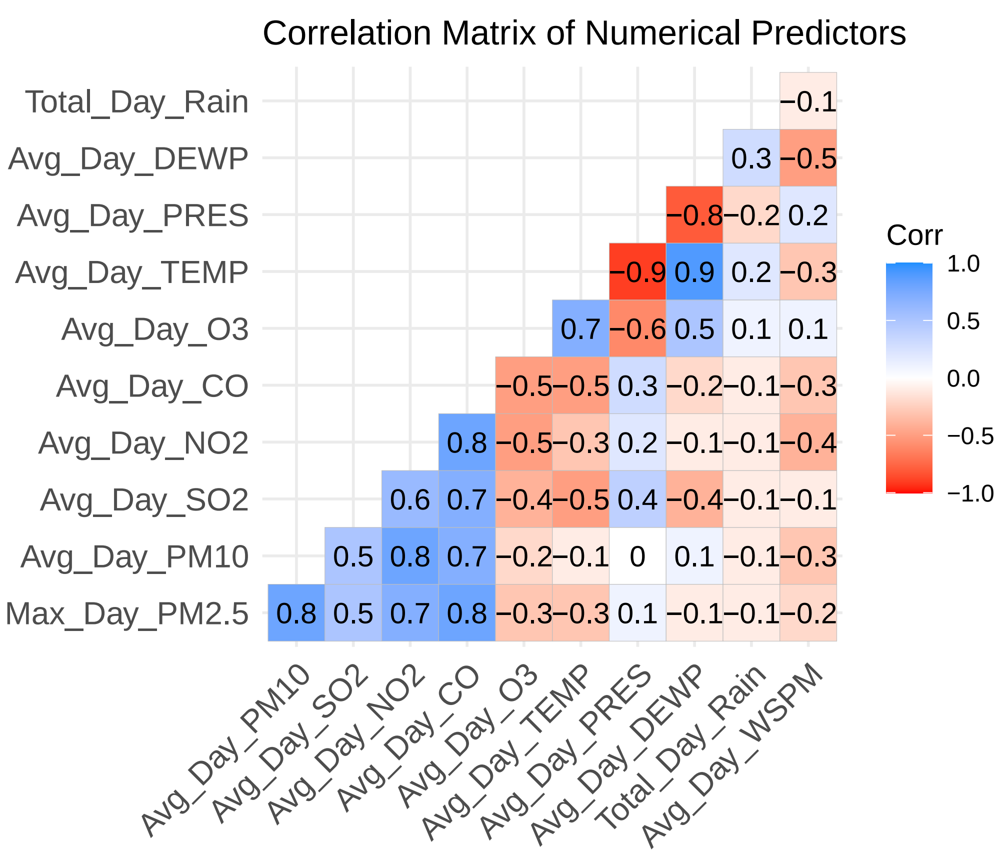
```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 75.29 on 1105 degrees of freedom
## Multiple R-squared:  0.5124, Adjusted R-squared:  0.501
## F-statistic: 44.67 on 26 and 1105 DF,  p-value: < 2.2e-16
```

Based on the summary table, we can see that some explanatory variables such as 'Avg_PM10', 'Avg_Day_CO', 'Avg_Day_DEWP', and most indicator variables of 'Freq_Day_wd' have relatively high p-values. This could be a clue that these attributes might not be statistically significant and eventually affect the model's performance.

Diagonostics - Initial Model

1. Multicollinearity

```r
library(ggcorrplot)

# Correlation Matrix with only numerical predictors
corr = round(cor(subset(reg_trn, select = -c(Max_Tmrw_PM2.5, Freq_Day_wd))), 1)

ggcorrplot(corr,lab = TRUE, type = "lower", colors = c("red", "white", "dodgerblue")) + labs(title="Cor
```



Above is the correlation heatmap of numerical explanatory variables. We can find out that there seems to have strong positive linear relationships between ('Max_Day_PM2.5'-'Avg_Day_PM10') and ('Avg_Day_TEMP'-'Avg_Day_DEWP'). Conversely, 'Avg_Day_TEMP' and 'Avg_Day_PRES' seem to have strong negative linear relationship. We may consider removing some of these variables for better prediction.

2. Predictor Redundancy - Variance Inflation Factor (VIF)

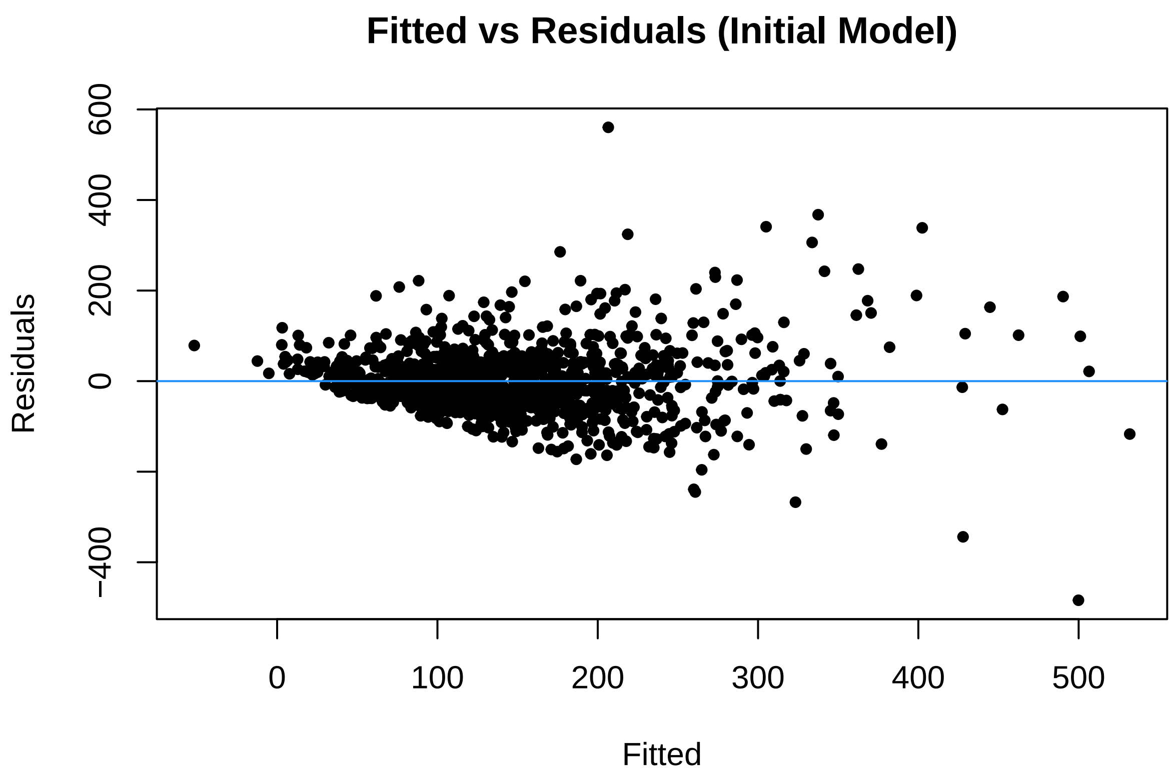
```
library(faraway)
vif(init_mod)
```

```
##  Max_Day_PM2.5   Avg_Day_PM10    Avg_Day_SO2    Avg_Day_NO2     Avg_Day_CO
##       4.047543       5.594299       2.734729       6.241134       5.210580
##     Avg_Day_O3   Avg_Day_TEMP   Avg_Day_PRES   Avg_Day_DEWP Total_Day_Rain
##       3.484382      16.490545       4.806309      10.587860       1.184944
##   Avg_Day_WSPM Freq_Day_wdENE Freq_Day_wdESE   Freq_Day_wdN  Freq_Day_wdNE
##       2.280476       2.057509       1.696508       7.479786       5.157031
## Freq_Day_wdNNE Freq_Day_wdNNW  Freq_Day_wdNW   Freq_Day_wdS  Freq_Day_wdSE
##       4.080110       2.129975       4.992313       3.330882       1.509782
## Freq_Day_wdSSE Freq_Day_wdSSW  Freq_Day_wdSW   Freq_Day_wdW Freq_Day_wdWNW
##       1.974211       4.878177       2.908535       4.073836       3.620716
## Freq_Day_wdWSW
##       2.300371
```

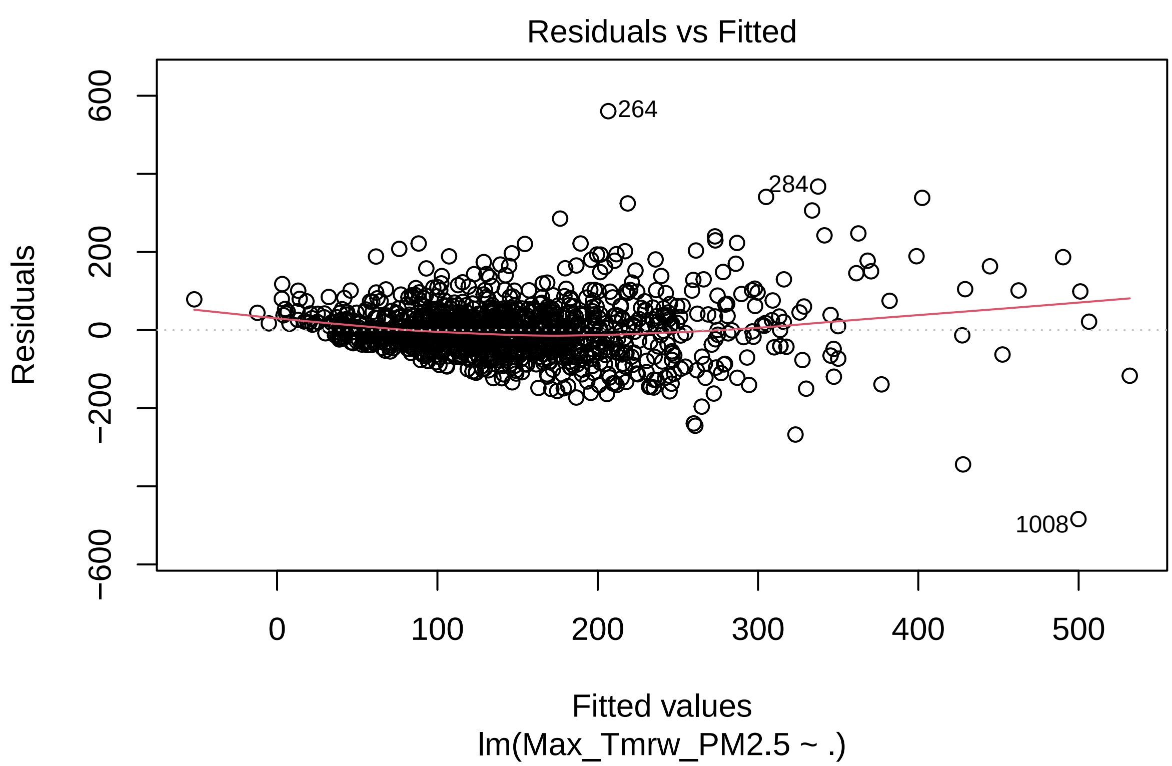
The VIF score indicates the redundancy of a variable, and the most common VIF thresholds are either VIF>5 or VIF>10. We can see that there are some variables with relatively high VIF score such as 'Avg_Day_TEMP' (16.5) and 'Avg_Day_DEWP' (10.6). We may consider removing these predictors later to fix the redundancy problem.

Constant Variance Assumption

```
# Fitted vs Residuals (Graphical)
plot(fitted(init_mod), resid(init_mod), pch = 20, xlab = "Fitted", ylab = "Residuals", main = "Fitted vs
abline(h = 0, col = "dodgerblue")
```




```r
plot(init_mod, 1)
```




```r
# Breusch-Pagn Test (Statistical)
library(lmtest)
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
bptest(init_mod)
```

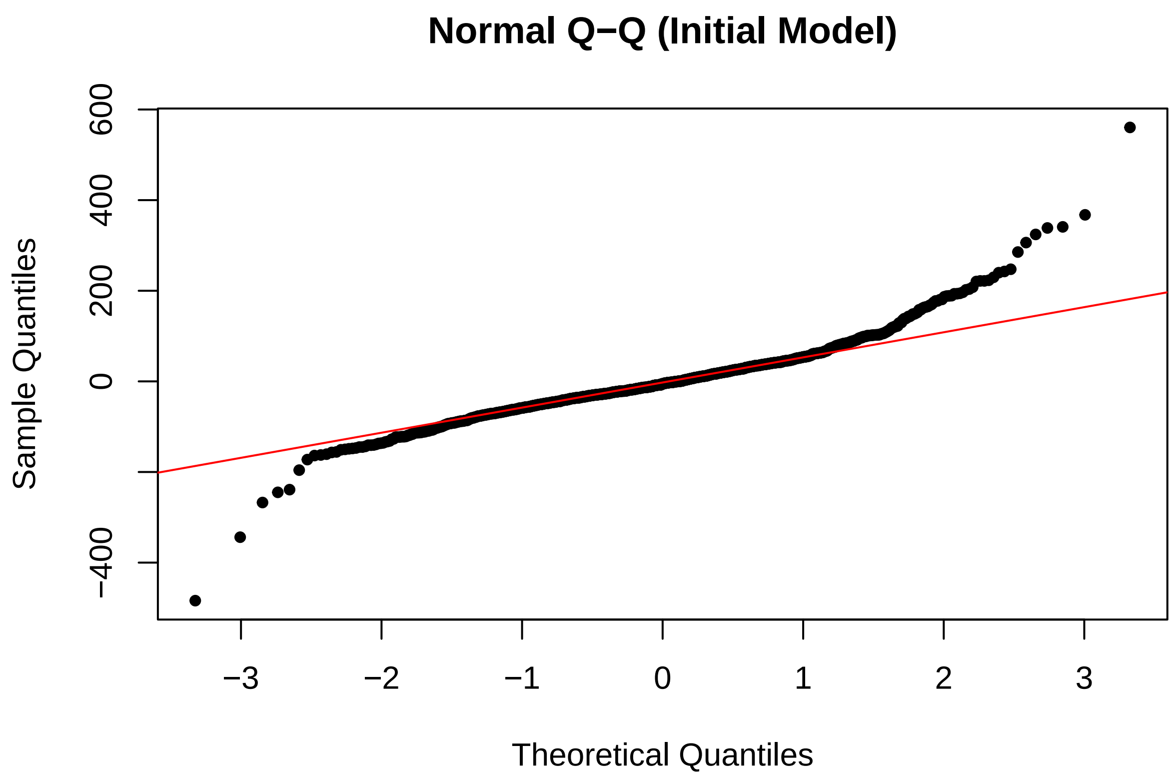
```
##
##  studentized Breusch-Pagan test
##
## data:  init_mod
## BP = 228.19, df = 26, p-value < 2.2e-16
```

H_0 : The variance of the model is constant. (Homoscedasticity) H_A : The variance is not constant. (Heteroscedasticity)

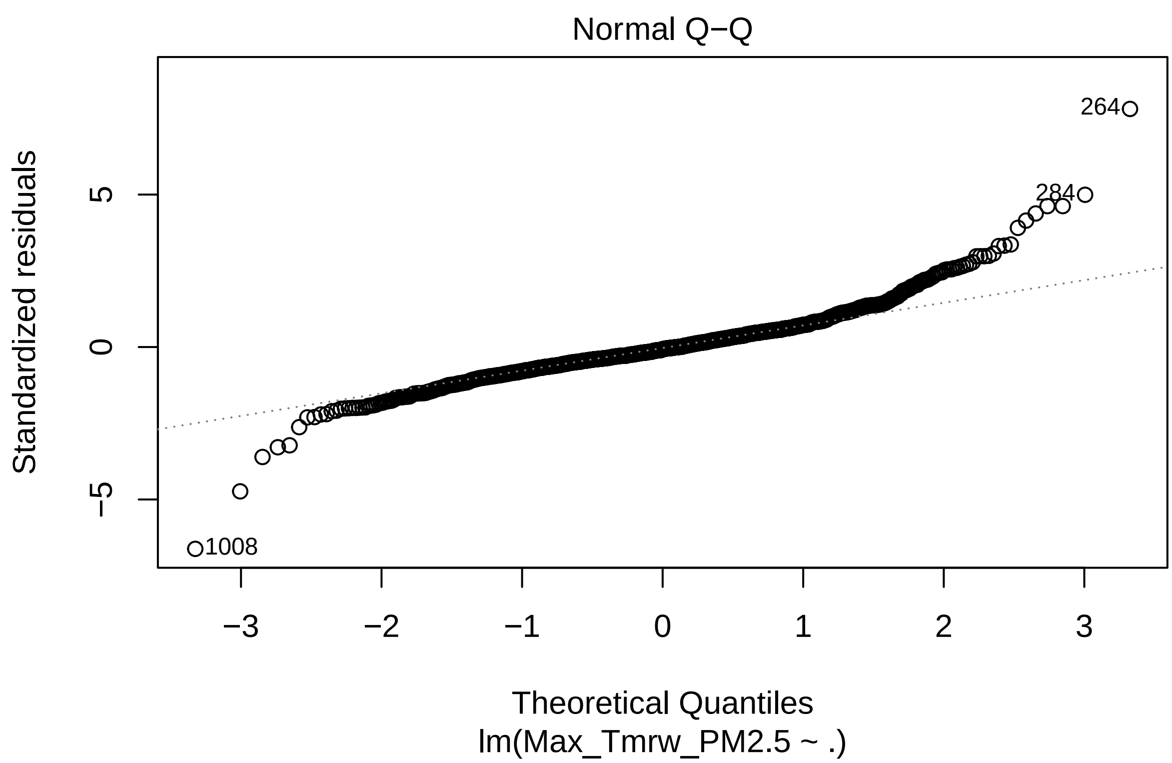
We can figure out that the constant variance condition is not met from both the graphical and statistical test. From the fitted vs residuals plot, the spread of points gets larger as we move from left to right, forming a cone shape. Also, the scale of y-axis (residuals) is pretty large (-400 ~ 600). Since the p-value of Breusch-Pagan test is significantly small, we would reject the null hypothesis, which is the variance is constant, and conclude that the homoscedasticity is violated. Thus, we might consider variable transformations such as log or box-cox transformation to fix the problem.

Normality Assumption

```
# Q-Q plot (Graphical)
qqnorm(resid(init_mod), pch = 20, main="Normal Q-Q (Initial Model)")
qqline(resid(init_mod), col = "red")
```




```
plot(init_mod, 2)
```




```
# Shapiro-Wilk / Kolmogorov-Smirnov - Statistical Test
# Shapiro-Wilk (Good for n < 50)
```

```
shapiro.test(resid(init_mod))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(init_mod)
## W = 0.92347, p-value < 2.2e-16
```

```
# Kolmogorov-Smirnov (Good for n > 50)
ks.test(resid(init_mod), y = pnorm)
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  resid(init_mod)
## D = 0.51501, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

H_0 : The residuals of the model follow a normal distribution. H_A : The residuals do not follow a normal distribution.

Similar to the constant variance assumption, the normality assumption seems to be violated as well. We can find some observations that are not on the line in the Q-Q plot. Plus, the Kolomogorov-Smirnov test p-value is significantly low, we would reject the null hypothesis, which is the residuals follow a normal distribution, and conclude that the normality condition is not satisfied. In this case, variable transformation could also fix the problem.

Influential - Leverage

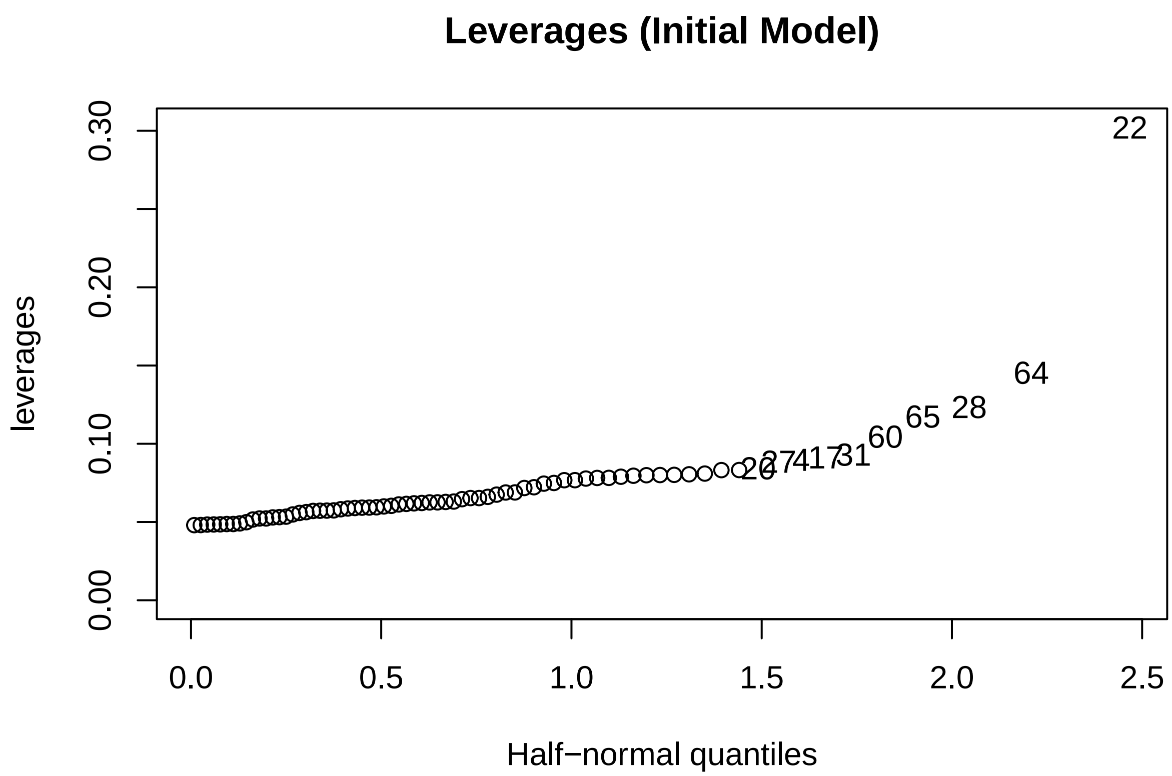
```
n = dim(reg_trn)[1]
p = 27 # number of pred = 26 + 1 intercept
lev = influence(init_mod)$hat
high_lev = lev[lev > (2*p/n)]
length(high_lev)
```

```
## [1] 73
```

```
max(high_lev)
```

```
## [1] 0.3022001
```

```
# Graphic
halfnorm(high_lev, 10, ylab = "leverages", main="Leverages (Initial Model)")
```



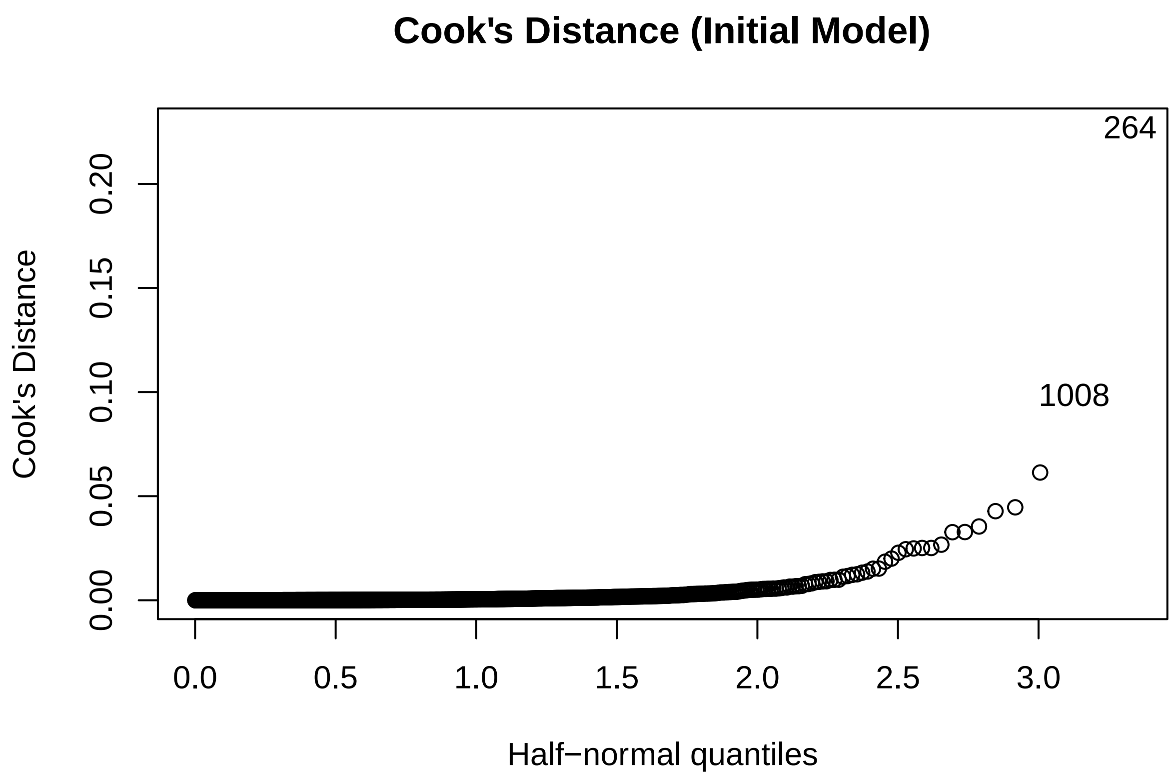
As we can see from the leverage plot above, almost all points follow a line and have relatively low leverages. However, observation 22 is relatively located higher than the others and does not follow the main trend. Therefore, it could be considered as a 'bad' high leverage point and thus possibly removed.

Influential - Cook's distance

```
cook = cooks.distance(init_mod)
max(cook)
```

```
## [1] 0.2271908
```

```
# Graphic
halfnorm(cook, 2, labs = as.character(1:length(cook)), ylab = "Cook's Distance", main="Cook's Distance
```



The common rule of thumb for cook's distance is $D_i \geq 1$, and since no point is higher than 1, we could assume that there is no highly influential point in the dataset.

Variable selection with AIC

```
library(stats)

# AIC - Recommend slightly larger model than BIC
# Tip: trace=0 (or False) as an argument to suppress lengthy output
step(init_mod, direction = "both", trace = 0)
```

```
##
## Call:
## lm(formula = Max_Tmrw_PM2.5 ~ Max_Day_PM2.5 + Avg_Day_PM10 +
##     Avg_Day_SO2 + Avg_Day_NO2 + Avg_Day_O3 + Avg_Day_TEMP + Avg_Day_PRES +
##     Total_Day_Rain + Avg_Day_WSPM + Freq_Day_wd, data = reg_trn)
##
## Coefficients:
##    (Intercept)   Max_Day_PM2.5    Avg_Day_PM10     Avg_Day_SO2     Avg_Day_NO2
##     -976.68484         0.37785         0.09965        -0.75936         0.93709
##     Avg_Day_O3    Avg_Day_TEMP    Avg_Day_PRES  Total_Day_Rain    Avg_Day_WSPM
##        0.63112        -3.24931         1.06831        -1.16698       -26.66396
## Freq_Day_wdENE  Freq_Day_wdESE    Freq_Day_wdN   Freq_Day_wdNE  Freq_Day_wdNNE
##      -14.18446         4.36296       -18.37164       -30.60593       -32.01298
## Freq_Day_wdNNW   Freq_Day_wdNW    Freq_Day_wdS   Freq_Day_wdSE  Freq_Day_wdSSE
##        8.91145       -18.00371       -10.03966         9.49494       -30.32439
## Freq_Day_wdSSW   Freq_Day_wdSW    Freq_Day_wdW  Freq_Day_wdWNW  Freq_Day_wdWSW
##        6.68548         0.71867       -24.32698       -17.96763         2.27966
```

Variable selection with BIC

```r
# BIC - Favor simpler model as it penalizes more parameters whenever log(n) > 2
n = length(resid(init_mod))
step(init_mod, direction = "both", k=log(n), trace = 0)
```

```
##
## Call:
## lm(formula = Max_Tmrw_PM2.5 ~ Max_Day_PM2.5 + Avg_Day_SO2 + Avg_Day_NO2 +
##     Avg_Day_O3 + Avg_Day_TEMP + Total_Day_Rain + Avg_Day_WSPM,
##     data = reg_trn)
##
## Coefficients:
##    (Intercept)   Max_Day_PM2.5     Avg_Day_SO2     Avg_Day_NO2      Avg_Day_O3
##        88.2088          0.4067         -0.8485          1.1519          0.7183
##   Avg_Day_TEMP  Total_Day_Rain    Avg_Day_WSPM
##        -4.0967         -1.4976        -25.1992
```

Reduced model with AIC

```r
aic_mod = lm(Max_Tmrw_PM2.5 ~ Max_Day_PM2.5 + Avg_Day_PM10 + Avg_Day_SO2 +
    Avg_Day_NO2 + Avg_Day_O3 + Avg_Day_TEMP + Avg_Day_PRES +
    Total_Day_Rain + Avg_Day_WSPM + Freq_Day_wd,
    data = reg_trn)

summary(aic_mod)
```

```
##
## Call:
## lm(formula = Max_Tmrw_PM2.5 ~ Max_Day_PM2.5 + Avg_Day_PM10 +
##     Avg_Day_SO2 + Avg_Day_NO2 + Avg_Day_O3 + Avg_Day_TEMP + Avg_Day_PRES +
##     Total_Day_Rain + Avg_Day_WSPM + Freq_Day_wd, data = reg_trn)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -483.46  -39.77   -5.39   34.91  560.24
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -976.68484  509.51184  -1.917 0.055507 .
## Max_Day_PM2.5     0.37785    0.03926   9.624  < 2e-16 ***
## Avg_Day_PM10      0.09965    0.06915   1.441 0.149874
## Avg_Day_SO2      -0.75936    0.19587  -3.877 0.000112 ***
## Avg_Day_NO2       0.93709    0.19050   4.919 1.00e-06 ***
## Avg_Day_O3        0.63112    0.10791   5.849 6.51e-09 ***
## Avg_Day_TEMP     -3.24931    0.62000  -5.241 1.91e-07 ***
## Avg_Day_PRES      1.06831    0.49659   2.151 0.031671 *
## Total_Day_Rain   -1.16698    0.38174  -3.057 0.002290 **
## Avg_Day_WSPM    -26.66396    3.98383  -6.693 3.47e-11 ***
## Freq_Day_wdENE  -14.18446   20.24917  -0.700 0.483765
```

```
## Freq_Day_wdESE    4.36296   22.63021   0.193 0.847156
## Freq_Day_wdN    -18.37164   15.53133  -1.183 0.237112
## Freq_Day_wdNE   -30.60593   16.06892  -1.905 0.057082 .
## Freq_Day_wdNNE  -32.01298   16.73899  -1.912 0.056073 .
## Freq_Day_wdNNW    8.91145   20.31171   0.439 0.660940
## Freq_Day_wdNW   -18.00371   16.27719  -1.106 0.268936
## Freq_Day_wdS    -10.03966   17.39783  -0.577 0.564014
## Freq_Day_wdSE     9.49494   24.83187   0.382 0.702261
## Freq_Day_wdSSE  -30.32439   20.56931  -1.474 0.140698
## Freq_Day_wdSSW    6.68548   16.33736   0.409 0.682462
## Freq_Day_wdSW     0.71867   17.88616   0.040 0.967957
## Freq_Day_wdW    -24.32698   16.60684  -1.465 0.143238
## Freq_Day_wdWNW  -17.96763   17.16137  -1.047 0.295337
## Freq_Day_wdWSW    2.27966   19.29838   0.118 0.905988
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 75.22 on 1107 degrees of freedom
## Multiple R-squared:  0.5124, Adjusted R-squared:  0.5019
## F-statistic: 48.48 on 24 and 1107 DF,  p-value: < 2.2e-16
```

Reduced Model with BIC

```
bic_mod = lm(Max_Tmrw_PM2.5 ~ Max_Day_PM2.5 + Avg_Day_SO2 + Avg_Day_NO2 +
    Avg_Day_O3 + Avg_Day_TEMP + Total_Day_Rain + Avg_Day_WSPM, data = reg_trn)
summary(bic_mod)
```

```
##
## Call:
## lm(formula = Max_Tmrw_PM2.5 ~ Max_Day_PM2.5 + Avg_Day_SO2 + Avg_Day_NO2 +
##     Avg_Day_O3 + Avg_Day_TEMP + Total_Day_Rain + Avg_Day_WSPM,
##     data = reg_trn)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -473.68  -39.81   -6.01   33.91  582.45
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     88.20882   12.14468   7.263 7.05e-13 ***
## Max_Day_PM2.5    0.40666    0.02976  13.666  < 2e-16 ***
## Avg_Day_SO2     -0.84849    0.19188  -4.422 1.07e-05 ***
## Avg_Day_NO2      1.15191    0.15467   7.447 1.89e-13 ***
## Avg_Day_O3       0.71827    0.10433   6.885 9.60e-12 ***
## Avg_Day_TEMP    -4.09672    0.41851  -9.789  < 2e-16 ***
## Total_Day_Rain  -1.49764    0.37711  -3.971 7.60e-05 ***
## Avg_Day_WSPM   -25.19923    3.81335  -6.608 6.00e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 76 on 1124 degrees of freedom
## Multiple R-squared:  0.4946, Adjusted R-squared:  0.4915
```

```
## F-statistic: 157.2 on 7 and 1124 DF,  p-value: < 2.2e-16
```

The BIC model favors a simpler model compared to the AIC model, and we could use the AIC model if we want to include 'Freq_Day_wd'. However, since most of the 'Freq_Day_wd' indicator variables have relatively high p-values, indicating not statistically significant, they might not add meaningful predictive power to the model. Therefore, we would prefer the BIC model.

Final selection - BIC Model

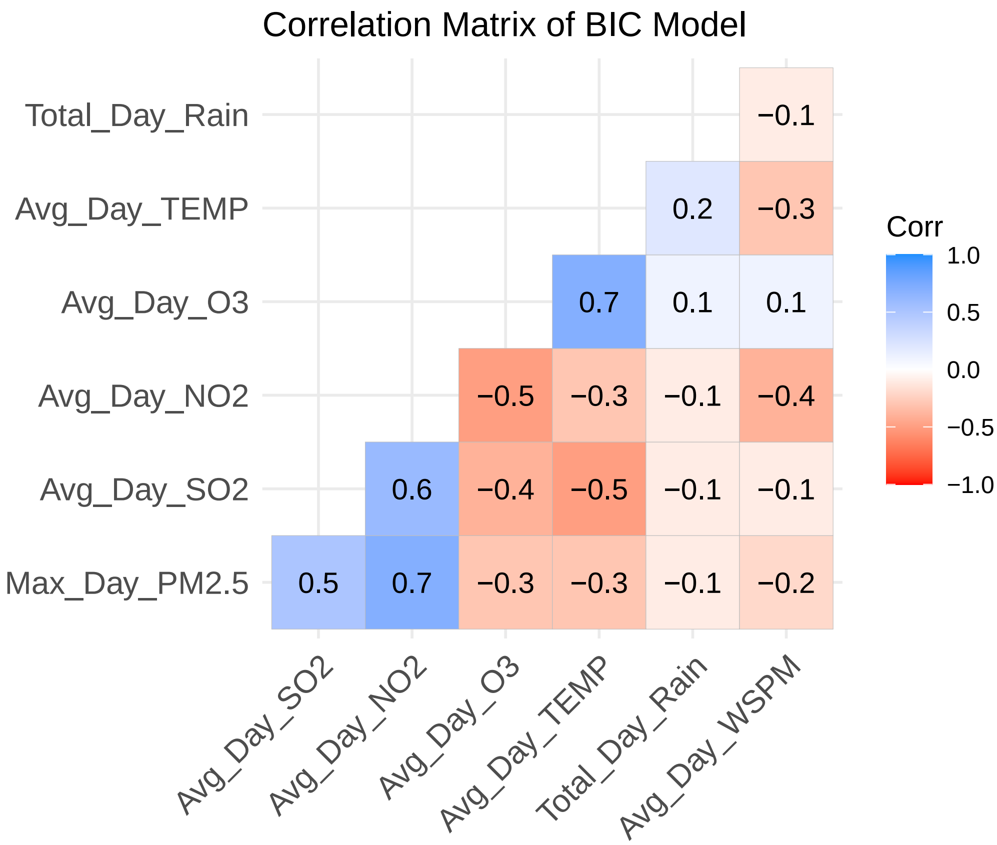
```r
bic_mod = lm(Max_Tmrw_PM2.5 ~ Max_Day_PM2.5 + Avg_Day_SO2 + Avg_Day_NO2 +
    Avg_Day_O3 + Avg_Day_TEMP + Total_Day_Rain + Avg_Day_WSPM, data = reg_trn)

summary(bic_mod)
```

```
##
## Call:
## lm(formula = Max_Tmrw_PM2.5 ~ Max_Day_PM2.5 + Avg_Day_SO2 + Avg_Day_NO2 +
##     Avg_Day_O3 + Avg_Day_TEMP + Total_Day_Rain + Avg_Day_WSPM,
##     data = reg_trn)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -473.68  -39.81   -6.01   33.91  582.45
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    88.20882   12.14468   7.263 7.05e-13 ***
## Max_Day_PM2.5   0.40666    0.02976  13.666  < 2e-16 ***
## Avg_Day_SO2    -0.84849    0.19188  -4.422 1.07e-05 ***
## Avg_Day_NO2     1.15191    0.15467   7.447 1.89e-13 ***
## Avg_Day_O3      0.71827    0.10433   6.885 9.60e-12 ***
## Avg_Day_TEMP   -4.09672    0.41851  -9.789  < 2e-16 ***
## Total_Day_Rain -1.49764    0.37711  -3.971 7.60e-05 ***
## Avg_Day_WSPM  -25.19923    3.81335  -6.608 6.00e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 76 on 1124 degrees of freedom
## Multiple R-squared:  0.4946, Adjusted R-squared:  0.4915
## F-statistic: 157.2 on 7 and 1124 DF,  p-value: < 2.2e-16
```

All explanatory variables have significantly small p-values, indicating statistical significance.

Diagnostic with BIC model

```r
# Correlation Matrix with only numerical predictors
corr_bic = round(cor(subset(reg_trn, select = -c(Max_Tmrw_PM2.5, Avg_Day_PM10, Avg_Day_CO, Avg_Day_DEWP

ggcorrplot(corr_bic,lab = TRUE, type = "lower", colors = c("red", "white", "dodgerblue")) + labs(title=
```



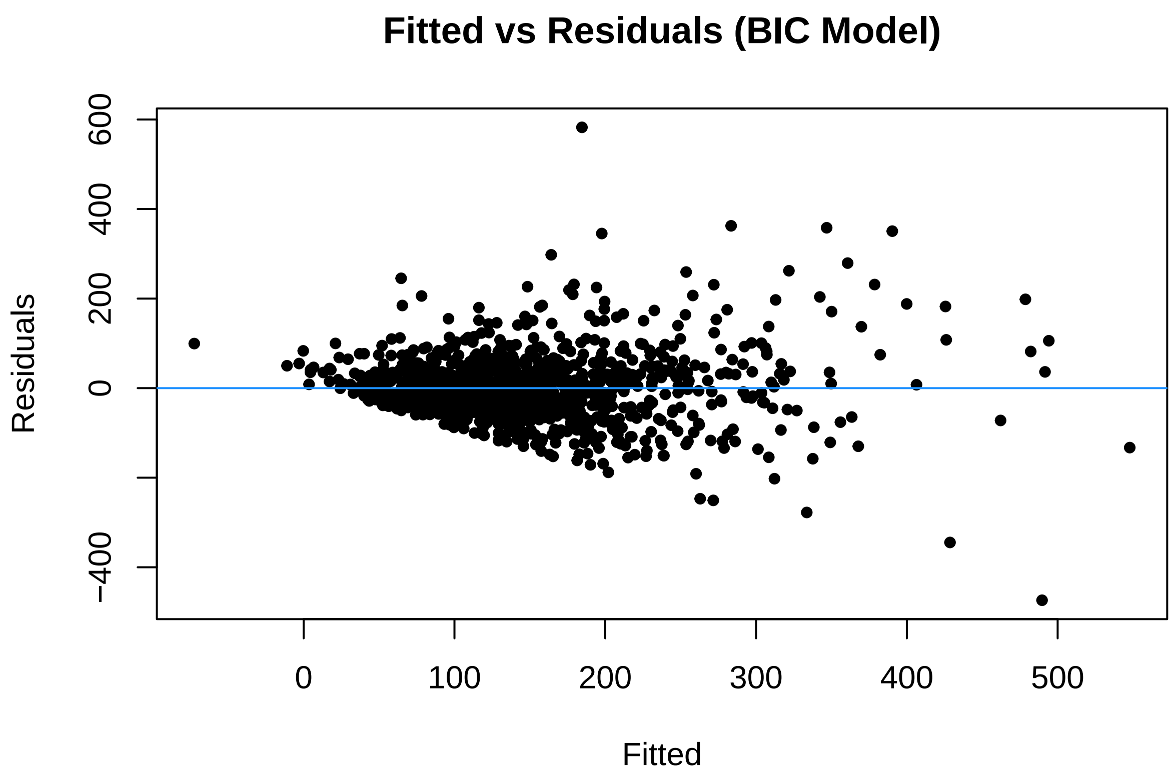

```r
# VIF of BIC
vif(bic_mod)
```

```
##   Max_Day_PM2.5     Avg_Day_SO2     Avg_Day_NO2      Avg_Day_O3    Avg_Day_TEMP
##        2.024976        2.349487        3.724884        3.159032        3.900002
## Total_Day_Rain    Avg_Day_WSPM
##        1.067629        1.694318
```

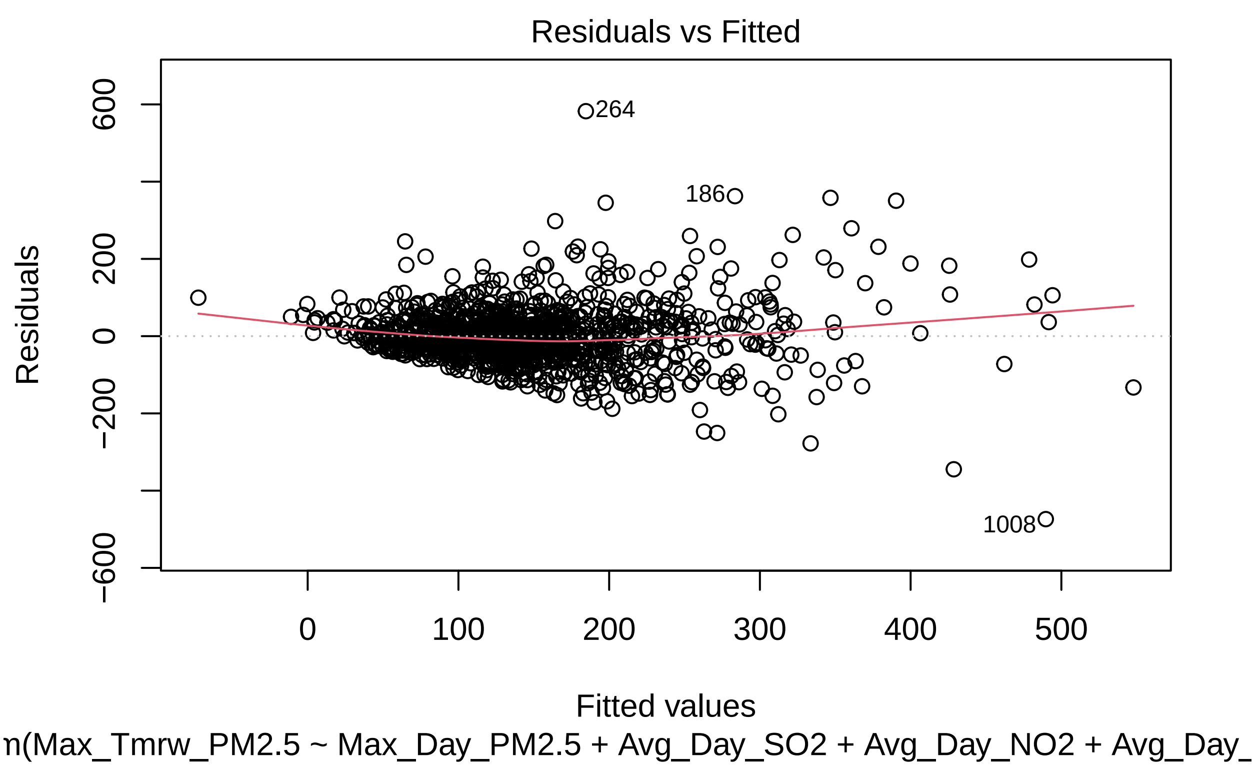
For the correlation plot, although there are some predictors with relatively high coefficient (Max_Day_PM2.5 - Avg_Day_NO2, 0.7), most predictors have indeed lower correlation coefficients to each other than the initial model's correlation coefficients. When comparing the VIF summary with the initial model ('init_mod'), we can see that all predictors have small VIF values.

Constant Variance Assumption - BIC Model

```r
# Fitted vs Residuals (Graphical)
plot(fitted(bic_mod), resid(bic_mod), pch = 20, xlab = "Fitted", ylab = "Residuals", main = "Fitted vs R
abline(h = 0, col = "dodgerblue")
```




```r
plot(bic_mod, 1)
```




```r
# Breusch-Pagn Test (Statistical)
bptest(bic_mod)
```

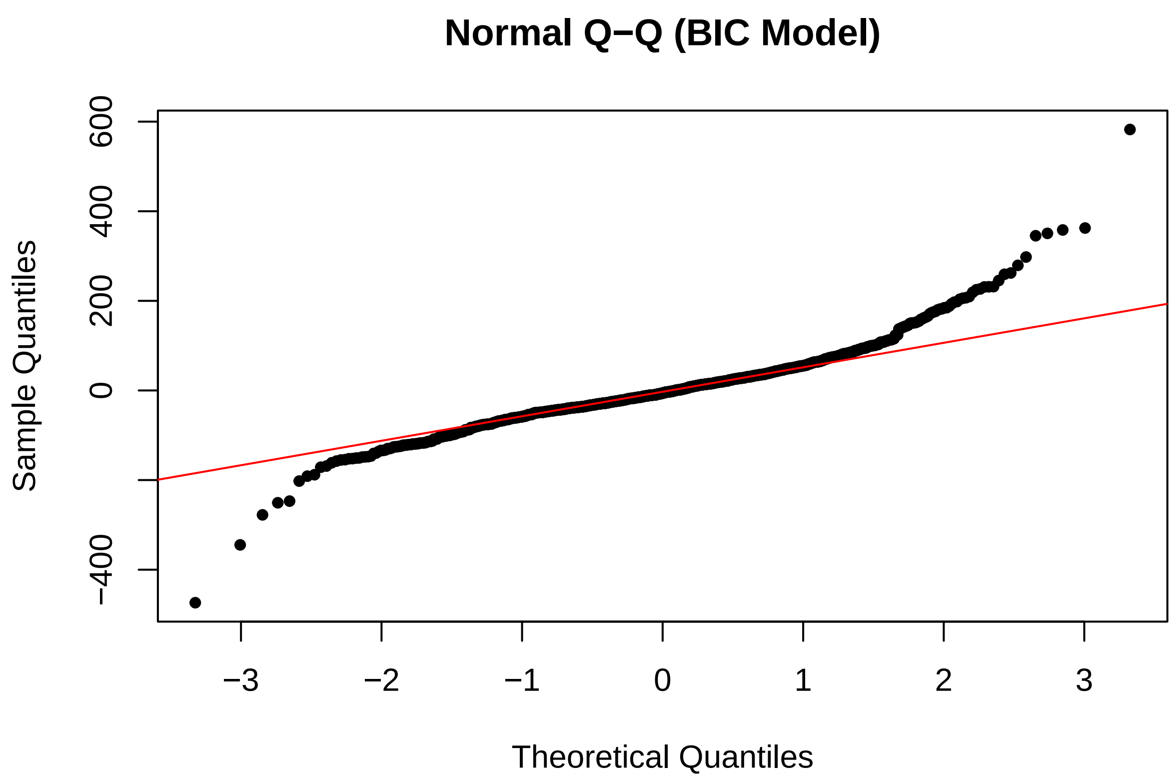
```
##
##  studentized Breusch-Pagan test
##
## data:  bic_mod
## BP = 180.74, df = 7, p-value < 2.2e-16
```

H_0 : The variance of the model is constant. (Homoscedasticity) H_A : The variance is not constant. (Heteroscedasticity)

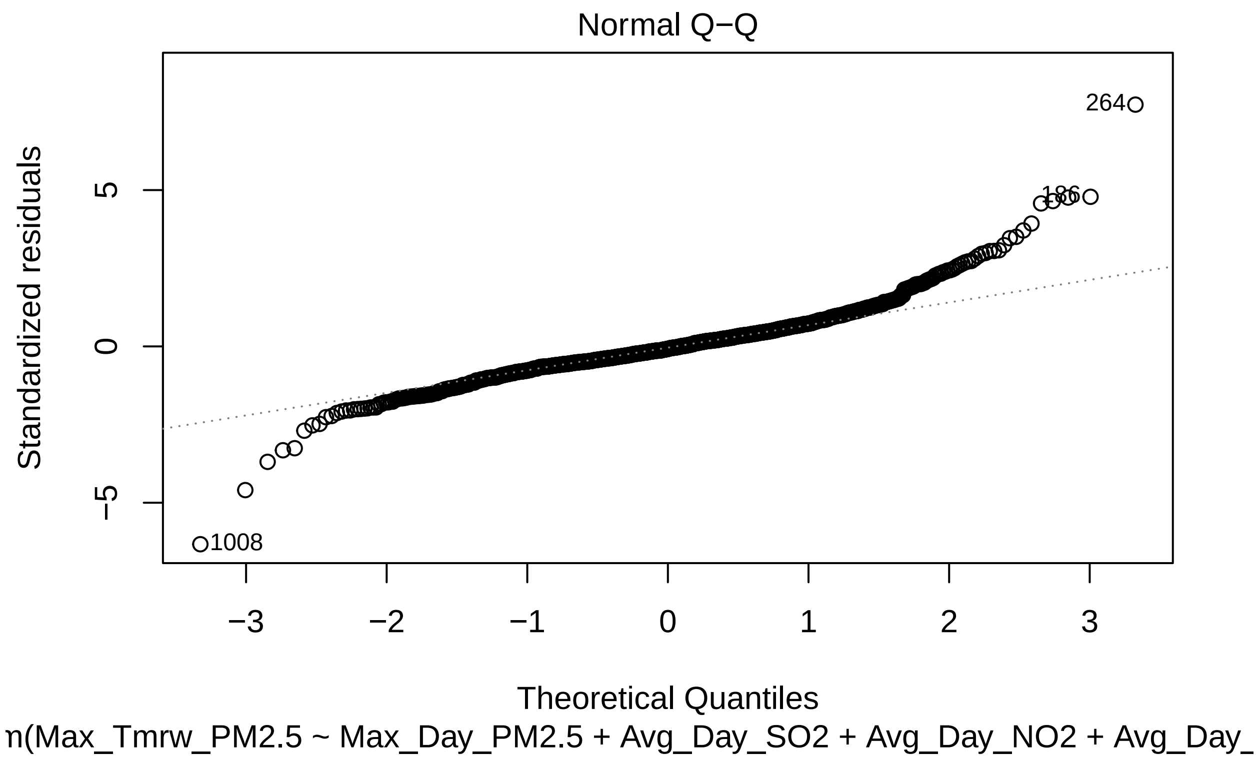
Similar to the initial model, the constant variance assumption is still not met. Therefore, we may consider transforming the response variable to fix the problem.

Normality Assumption - BIC Model

```
# Q-Q plot (Graphical)
qqnorm(resid(bic_mod), pch = 20, main = "Normal Q-Q (BIC Model)")
qqline(resid(bic_mod), col = "red")
```




```
plot(bic_mod, 2)
```




```r
# Shapiro-Wilk / Kolmogorov-Smirnov - Statistical Test
# Shapiro-Wilk (Good for n < 50)
shapiro.test(resid(bic_mod))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(bic_mod)
## W = 0.92259, p-value < 2.2e-16
```

```r
# Kolmogorov-Smirnov (Good for n > 50)
ks.test(resid(bic_mod), y = pnorm)
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  resid(bic_mod)
## D = 0.51778, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

H_0 : The residuals of the model follow a normal distribution. H_A : The residuals do not follow a normal distribution.

Same as the initial model, the normality assumption is not met as well for the BIC model, and response variable tranformation may be the solution to satisfy the condition.

Influential - Leverage (BIC Model)

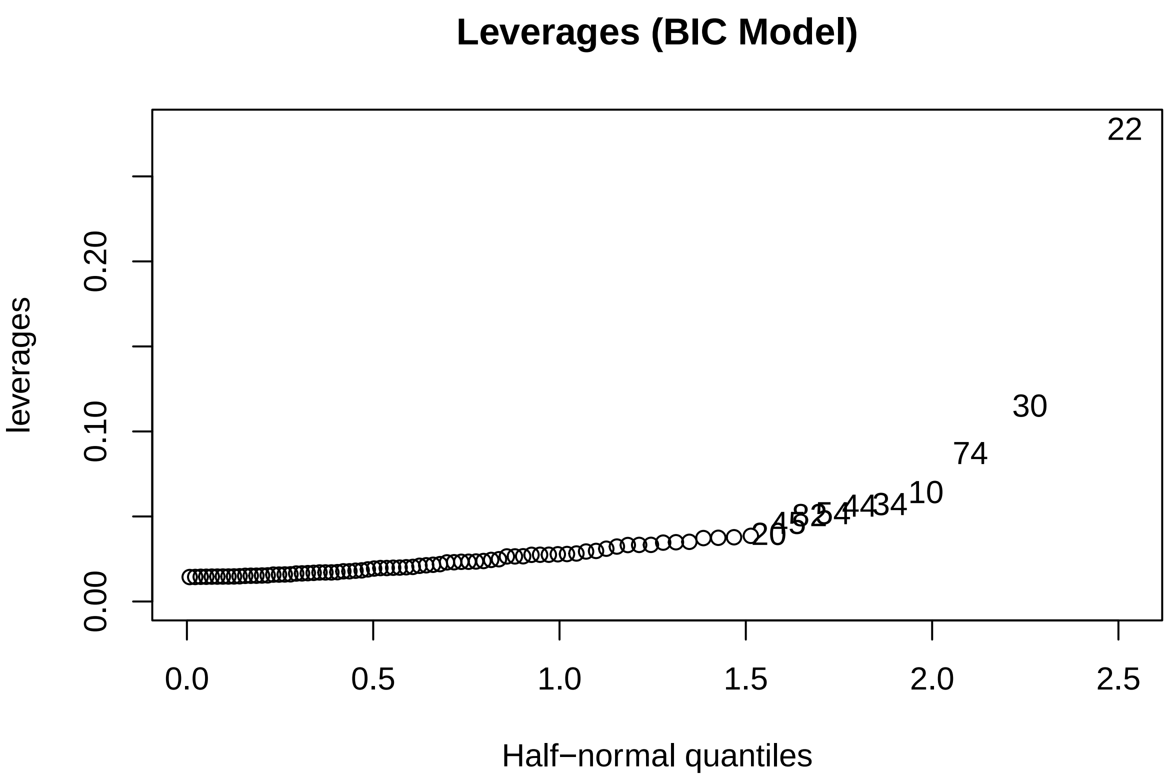
```
n = dim(reg_trn)[1]
p = 8 # number of pred = 7 + 1 intercept
lev = influence(bic_mod)$hat
high_lev = lev[lev > (2*p/n)]
length(high_lev)
```

```
## [1] 84
```

```
max(high_lev)
```

```
## [1] 0.2780974
```

```
# Graphic
halfnorm(high_lev, 10, ylab = "leverages", main="Leverages (BIC Model)")
```



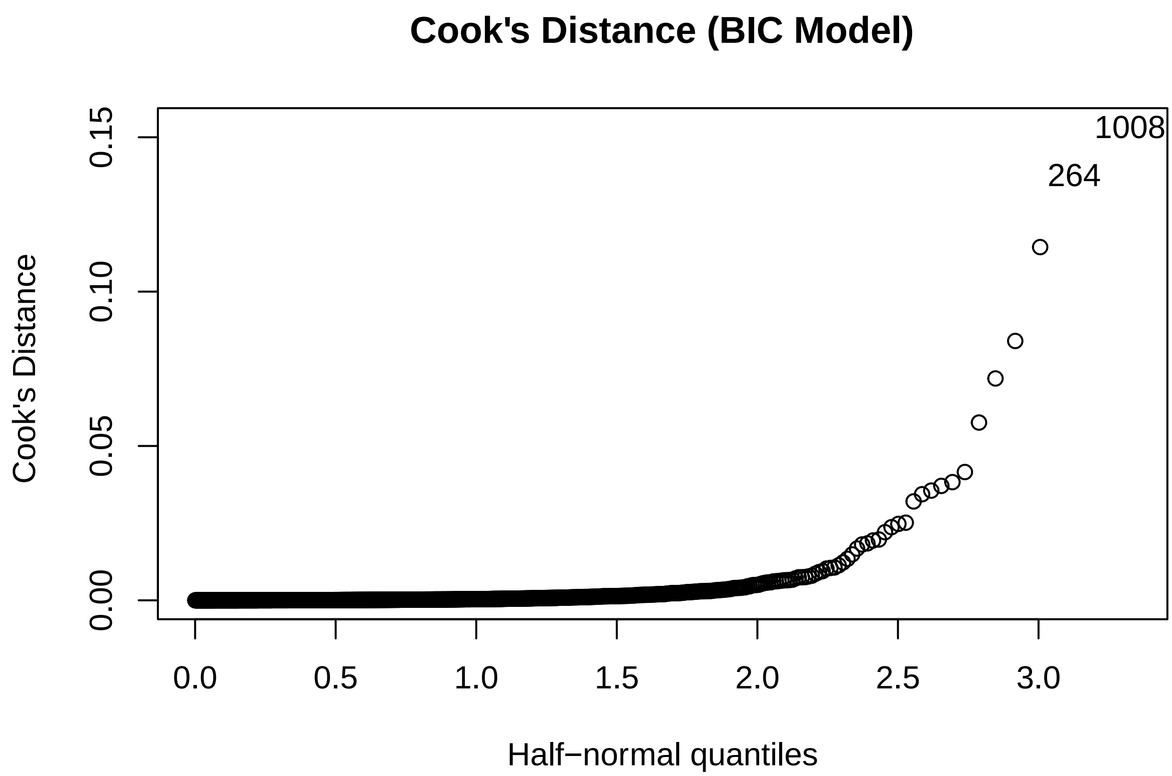
The leverage plot shows similar trend compared to the initial model. Most observations are on the line, but observation 22 again has relatively high leverages, and this point could be considered as 'bad' high leverage point and may have to remove.

Influential - Cook's Distance (BIC Model)

```
cook = cooks.distance(bic_mod)
max(cook)
```

```
## [1] 0.1532926
```

```
# Graphic
halfnorm(cook, 2, labs = as.character(1:length(cook)), ylab = "Cook's Distance ", main = "Cook's Distanc
```



Similar to the initial model, the highest Cook's distance is about 0.15, which is pretty small, we may conclude that there is no highly influential observation from the reduced model.

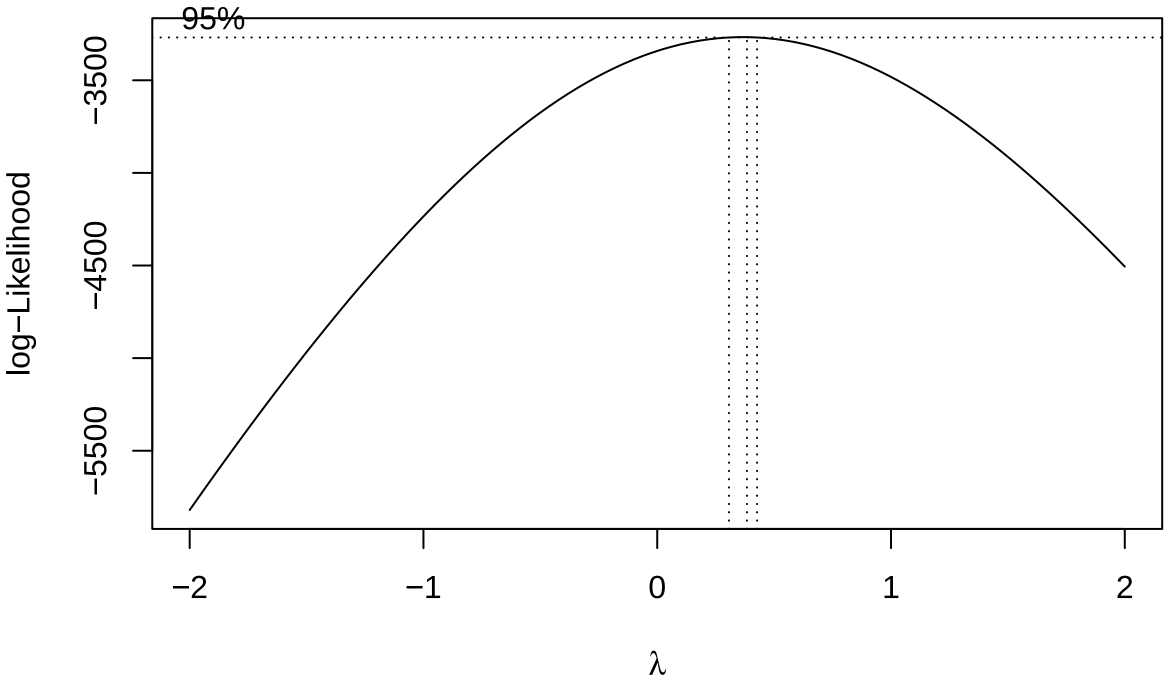
Since the constant variance and normality assumptions are still violated, we may consider transforming the response variable.

Variable Transformation - Box-Cox (BIC Model)

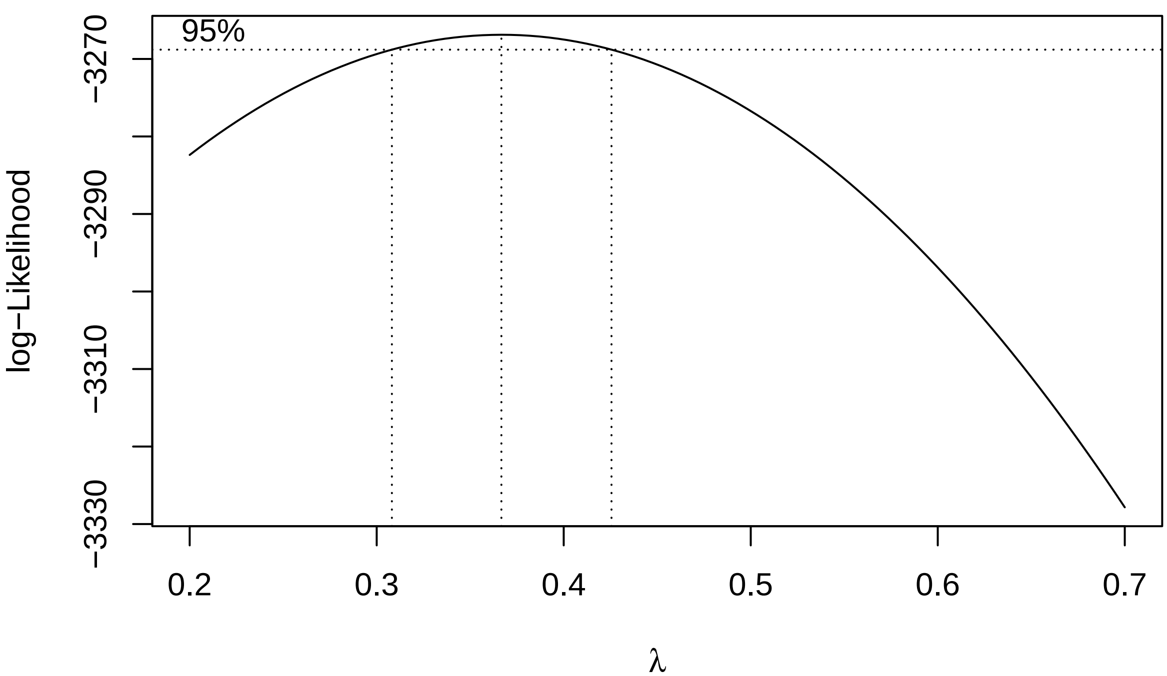
```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
boxcox(bic_mod, plotit = TRUE)
```




```r
boxcox(bic_mod, lambda = seq(0.2, 0.7, by = 0.05))
```



When observing the box-cox plot above, it seems like λ value around 0.4 has the largest negative log-likelihood value. For a simple calculation, it would be a good idea to choose $\lambda = 0.4$

Transformed BIC Model

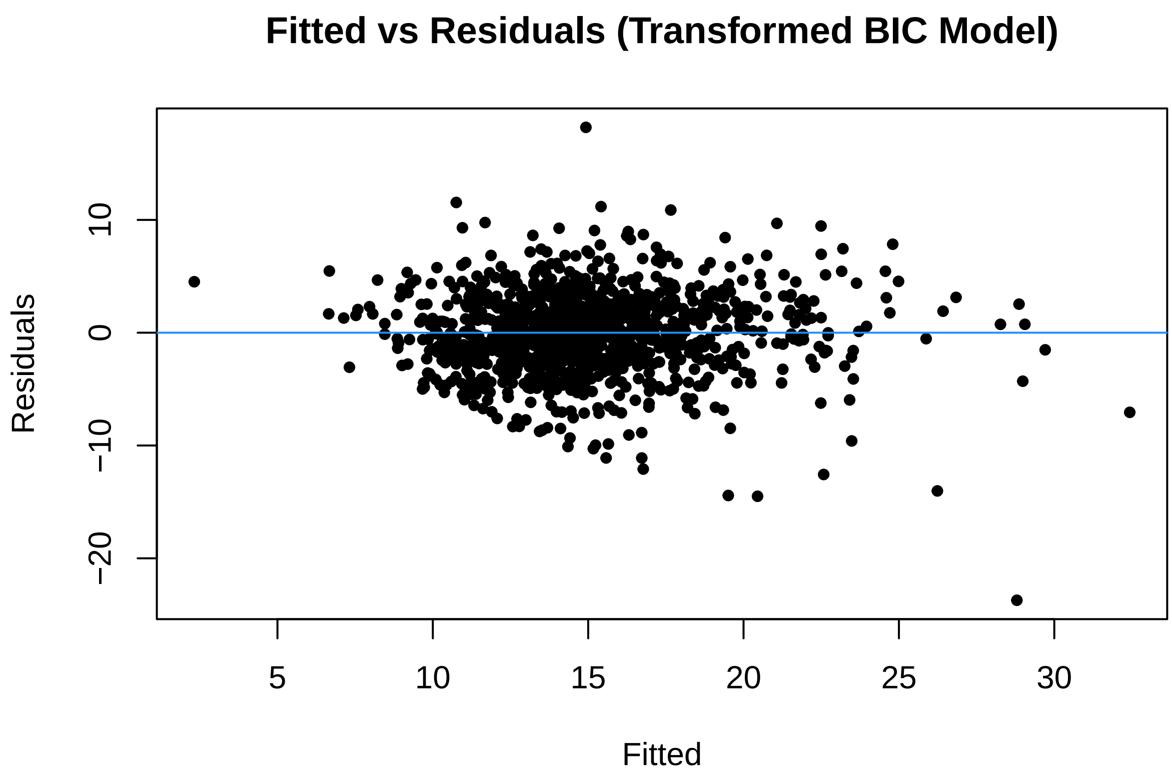
```
bic_mod_trns = lm((((Max_Tmrw_PM2.5^0.4) - 1)/0.4) ~ Max_Day_PM2.5 + Avg_Day_SO2 + Avg_Day_NO2 + Avg_Da

summary(bic_mod_trns)
```

```
##
## Call:
## lm(formula = (((Max_Tmrw_PM2.5^0.4) - 1)/0.4) ~ Max_Day_PM2.5 +
##     Avg_Day_SO2 + Avg_Day_NO2 + Avg_Day_O3 + Avg_Day_TEMP + Total_Day_Rain +
##     Avg_Day_WSPM, data = reg_trn)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.7170  -2.0531   0.0264   2.2299  18.2060
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    12.314490   0.575321  21.405  < 2e-16 ***
## Max_Day_PM2.5   0.014175   0.001410  10.056  < 2e-16 ***
## Avg_Day_SO2    -0.039806   0.009090  -4.379 1.30e-05 ***
## Avg_Day_NO2     0.064954   0.007327   8.865  < 2e-16 ***
## Avg_Day_O3      0.041950   0.004942   8.488  < 2e-16 ***
## Avg_Day_TEMP   -0.202482   0.019826 -10.213  < 2e-16 ***
## Total_Day_Rain -0.093033   0.017865  -5.208 2.27e-07 ***
## Avg_Day_WSPM   -1.436010   0.180647  -7.949 4.54e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.601 on 1124 degrees of freedom
## Multiple R-squared:  0.4658, Adjusted R-squared:  0.4625
## F-statistic:   140 on 7 and 1124 DF,  p-value: < 2.2e-16
```

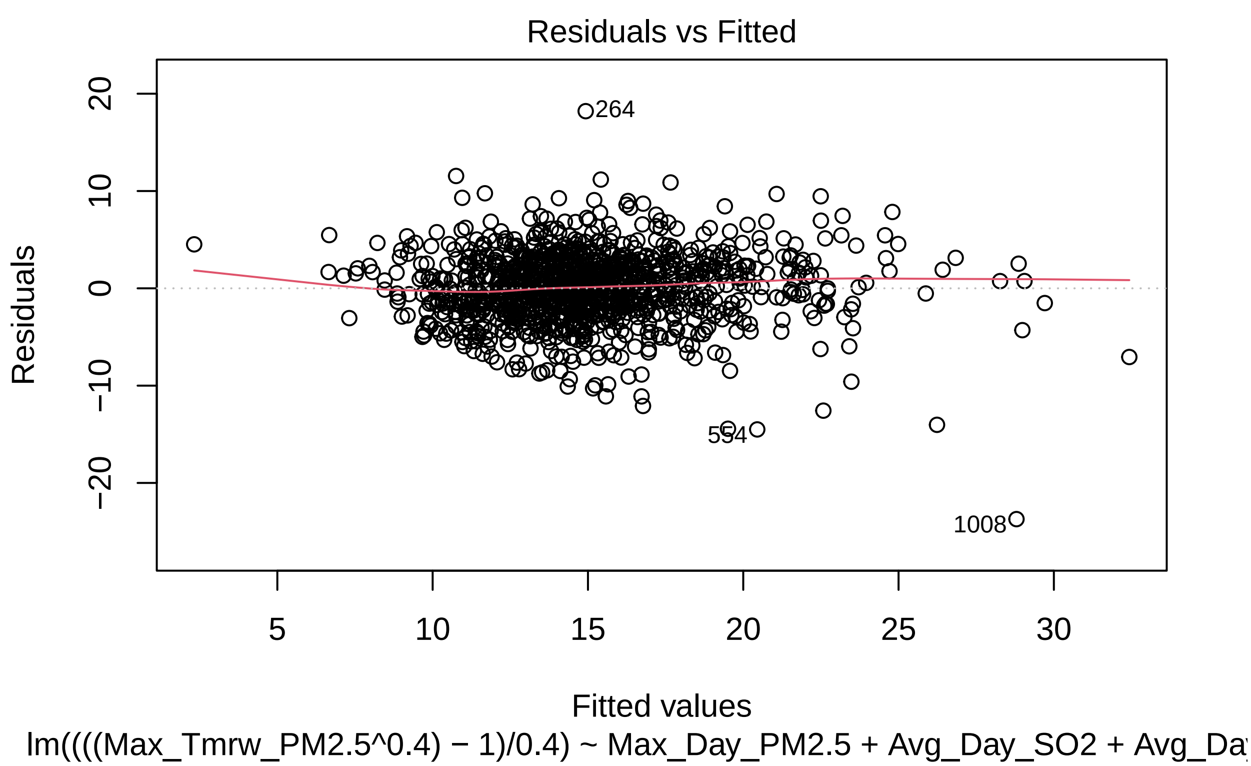
After transforming the response variable, the p-values for numerical predictors decreased significantly compared to the original reduced model (bic_mod).

Constant Variance Assumption - New Transformed BIC Model

```
# Fitted vs Residuals (Graphical)
plot(fitted(bic_mod_trns), resid(bic_mod_trns), pch = 20, xlab = "Fitted", ylab = "Residuals", main = "
abline(h = 0, col = "dodgerblue")
```




```
plot(bic_mod_trns, 1)
```



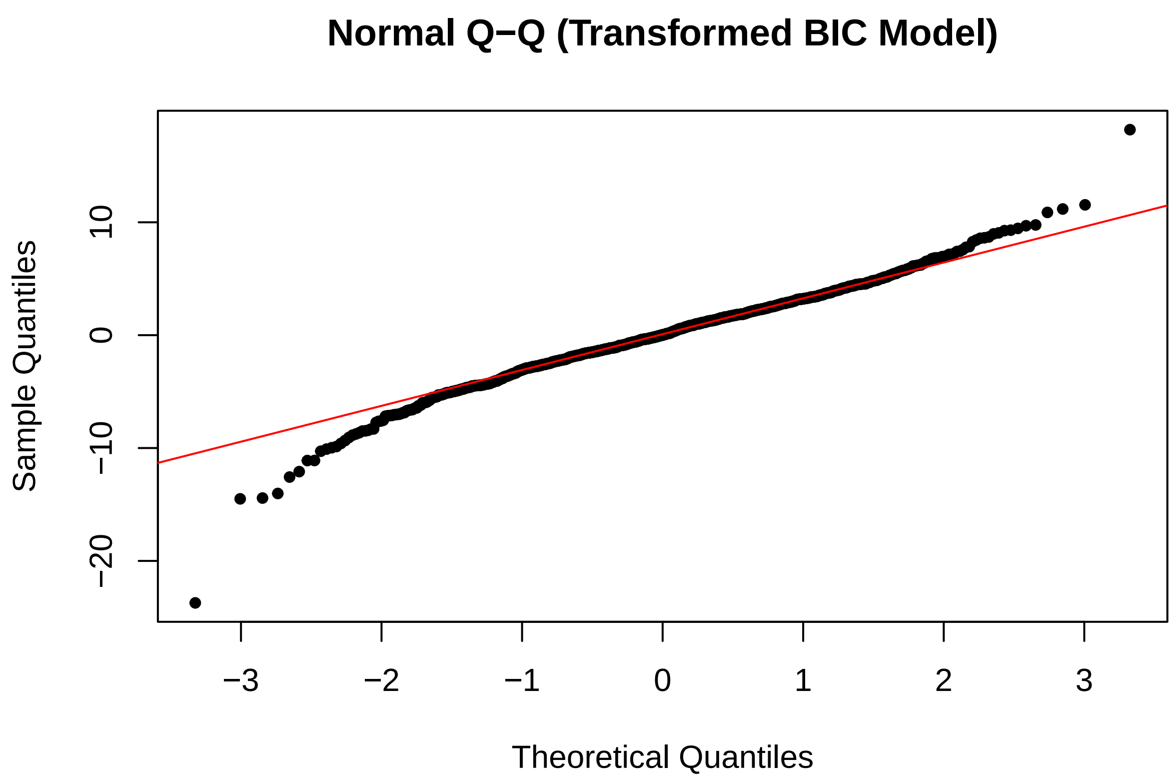

```
# Breusch-Pagn Test (Statistical)
bptest(bic_mod_trns)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  bic_mod_trns
## BP = 103.55, df = 7, p-value < 2.2e-16
```

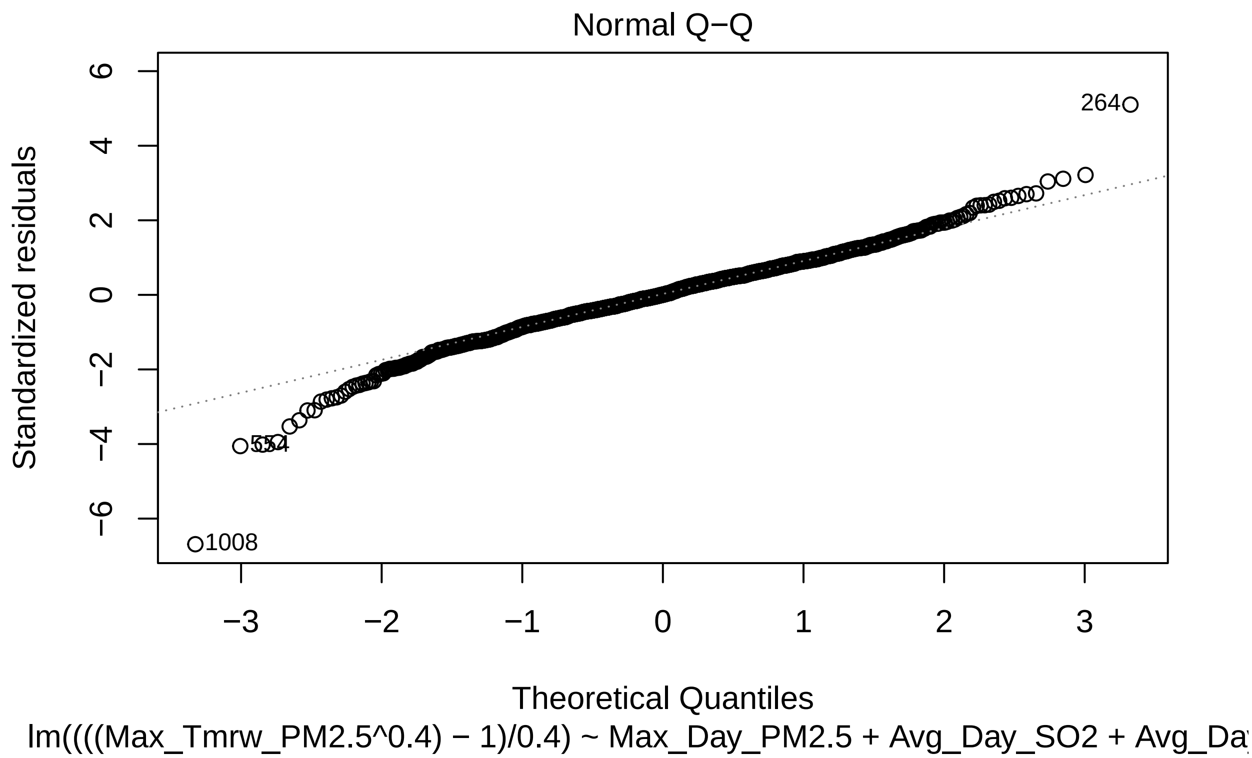
It is surprising that the fitted vs residuals plot seems to display fairly equivalent spread from left to right. Also, the scale of y-axis (-20 ~ 20) is significantly smaller than the original reduced model (bic_mod). However, the statistical test value is still not met.

Normality Assumption - New Transformed BIC Model

```r
# Q-Q plot (Graphical)
qqnorm(resid(bic_mod_trns), pch = 20, main = "Normal Q-Q (Transformed BIC Model)")
qqline(resid(bic_mod_trns), col = "red")
```




```r
plot(bic_mod_trns, 2)
```




```r
# Shapiro-Wilk / Kolmogorov-Smirnov - Statistical Test
# Shapiro-Wilk (Good for n < 50)
shapiro.test(resid(bic_mod_trns))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(bic_mod_trns)
## W = 0.97581, p-value = 8.117e-13
```

```r
# Kolmogorov-Smirnov (Good for n > 50)
ks.test(resid(bic_mod_trns), y = pnorm)
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  resid(bic_mod_trns)
## D = 0.27525, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

Same for the normality assumption, almost all observations seem to be on the line compared to the original reduced model (bic_mod)'s Q-Q plot. However, the statistical test values still indicates the violation of the condition.

Influential - Leverage (Transformed BIC Model)

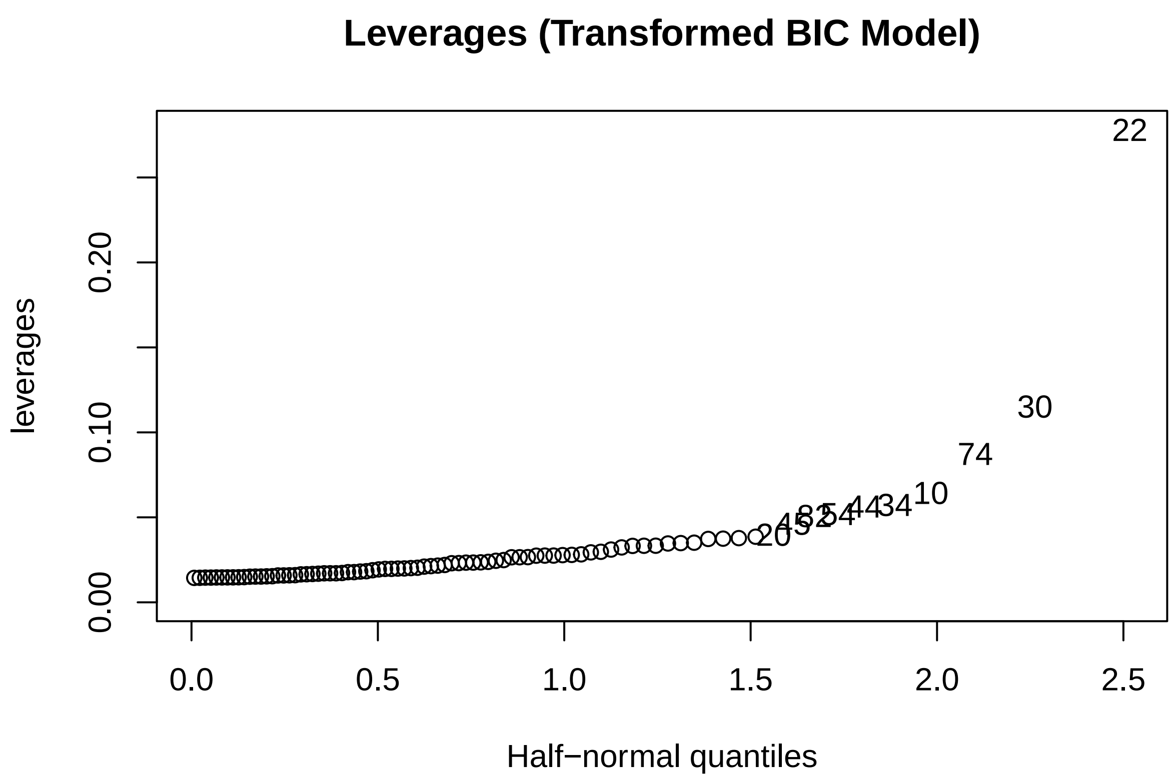
```
n = dim(reg_trn)[1]
p = 8 # number of pred = 7 + 1 intercept
lev = influence(bic_mod_trns)$hat
high_lev = lev[lev > (2*p/n)]
length(high_lev)
```

```
## [1] 84
```

```
max(high_lev)
```

```
## [1] 0.2780974
```

```
# Graphic
halfnorm(high_lev, 10, ylab = "leverages", main= "Leverages (Transformed BIC Model)")
```

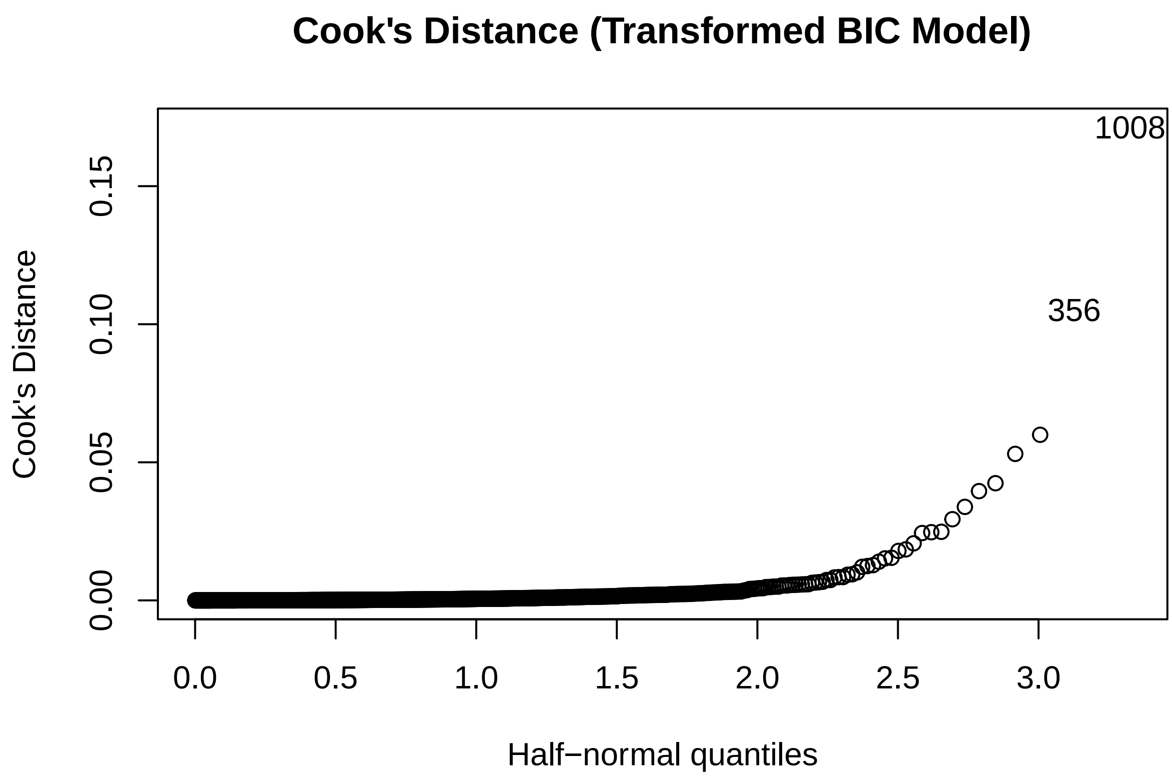


Influential - Cook's Distance (Transformed BIC Model)

```
cook = cooks.distance(bic_mod_trns)
max(cook)
```

```
## [1] 0.1712471
```

```r
# Graphic
halfnorm(cook, 2, labs = as.character(1:length(cook)), ylab = "Cook's Distance", main="Cook's Distance
```



Both the leverage and Cook's distance plots of transformed model (bic_mod_trns) are almost identical to the original reduced model (bic_mod), indicating there might be no highly 'bad' influential point.

To sum up, the transformation of the response variable was somewhat effective in terms of fixing some assumptions, and therefore we would choose it as our final model.

Final Model - Transformed BIC Model

```r
# Same as bic_mod_trns
trans_mod = lm((((Max_Tmrw_PM2.5^0.4) - 1)/0.4) ~ Max_Day_PM2.5 + Avg_Day_SO2 + Avg_Day_NO2 + Avg_Day_O3

summary(trans_mod)
```

```
##
## Call:
## lm(formula = (((Max_Tmrw_PM2.5^0.4) - 1)/0.4) ~ Max_Day_PM2.5 +
##     Avg_Day_SO2 + Avg_Day_NO2 + Avg_Day_O3 + Avg_Day_TEMP + Total_Day_Rain +
##     Avg_Day_WSPM, data = reg_trn)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.7170  -2.0531   0.0264   2.2299  18.2060
##
## Coefficients:
```

```
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    12.314490   0.575321  21.405  < 2e-16 ***
## Max_Day_PM2.5   0.014175   0.001410  10.056  < 2e-16 ***
## Avg_Day_SO2    -0.039806   0.009090  -4.379 1.30e-05 ***
## Avg_Day_NO2     0.064954   0.007327   8.865  < 2e-16 ***
## Avg_Day_O3      0.041950   0.004942   8.488  < 2e-16 ***
## Avg_Day_TEMP   -0.202482   0.019826 -10.213  < 2e-16 ***
## Total_Day_Rain -0.093033   0.017865  -5.208 2.27e-07 ***
## Avg_Day_WSPM   -1.436010   0.180647  -7.949 4.54e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.601 on 1124 degrees of freedom
## Multiple R-squared:  0.4658, Adjusted R-squared:  0.4625
## F-statistic:   140 on 7 and 1124 DF,  p-value: < 2.2e-16
```

Model Performance

Calculating Adjusted R squared

Problem with R^2

It cannot get worse when adding "bad" predictors/terms to a model.

Why does it go up?

Even if fitting a predictor that has no correlation to the response (or no additional correlation beyond the predictors already present), we will expect some correlation to be explained "by random chance"

Adjusted R squared

Adjusts the SSE and SST terms by dividing by their degrees of freedom

The degrees of freedom for the SSE term is related to how many parameters are used to fit the model. The more parameters, the more opportunities for some "random chance correlation" to get explained.

$R_a^2 = 1 - \frac{SSE/(n-p)}{SST/(n-1)} = 1 - (\frac{n-1}{n-p})(1 - R^2)$

Interpretation

After adjusting for correlation due to random chance, approximately X% of the variance in [Response] is explained by this model.

Evaluation Function - RMSE / R squared / Adjusted R squared

```
eval_results <- function(true, predicted, df, mod) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)

  # Additional variables for Adjusted R squared
  n = dim(df)[1]
  p = sum(coef(mod) != 0) - 1 # -1 is for the intercept

  RMSE = sqrt(SSE/nrow(df))
  R_square <- 1 - SSE / SST
```

```r
  Adj_R_sq <- 1 - ((n-1)/(n-p))*(SSE/SST)

  # Model performance metrics
data.frame(
  RMSE = RMSE,
  Rsquare = R_square,
  Adjusted_Rsquare = Adj_R_sq
)
}
```

Model Performance - Transformed Model

```r
ypred = predict(trans_mod, newdata = reg_tst, type = "response")

# Reconverting
yprednew = (ypred*0.4 + 1)^2.5

# Need to omit missing values to run the eval_results function
yprednew = na.omit(yprednew)
yprednew[1:5]
```

```
##        1        2        3        4        5
## 122.2152 262.2627 208.6926 416.0826 231.2529
```

Function to categorize predicted Max_Tmrw_PM2.5 concentrations' levels

```r
category_PM2.5 = function(x){
  newx = numeric()
  for (i in 1:length(x)){
    if (x[i] <= 35) {
      newx[i] = "Low"
      } else if (x[i] > 35 & x[i] <= 75) {
        newx[i] = "Medium"
        } else if (x[i] > 75 & x[i] <= 105) {
          newx[i] = "High"
          } else if (x[i] > 105) {
           newx[i] = "Dangerous"
  }
  }

  # Reordering the Average_PM2.5_Type
  newx = factor(newx, levels = c("Dangerous", "High", "Medium", "Low"))
  return(newx)
}
```

Check the predicted levels' accuracy to the test data

```r
# Comparing predicted PM2.5 levels with the test labels (Actual/Observation)
YPM = category_PM2.5(yprednew)
testPM = category_PM2.5(reg_tst$Max_Tmrw_PM2.5)

# Remove the row that was dropped from yprenew
testPM = testPM[-c(243)]

# Number of observations that are correctly predicted by the Transformed model
sum(YPM == testPM)
```

```
## [1] 171
```

```r
length(testPM)
```

```
## [1] 283
```

```r
# Test Accuracy
trans_acc = sum(YPM == testPM) / length(testPM)
trans_acc
```

```
## [1] 0.6042403
```

Performance of the Transformed BIC Model

```r
trans_perf = eval_results(reg_tst$Max_Tmrw_PM2.5, yprednew, reg_tst, trans_mod)
```

```
## Warning in predicted - true: longer object length is not a multiple of shorter
## object length
```

```r
trans_perf
```

```
##      RMSE  Rsquare Adjusted_Rsquare
## 1 94.7177 0.284619        0.2691233
```

As we can see from the evaluation above, both the R^2 and adjusted R^2 are relatively low, indicating only 28.5% of the variability is explained by the transformed model.

Prediction vs Actual Table (Transformed BIC Model)

```r
table(Pred.trans = YPM, Actual.Obs = testPM)
```

```
##             Actual.Obs
## Pred.trans  Dangerous High Medium Low
##   Dangerous       140   19     16  10
##   High             19   16     20   6
##   Medium            8    9     15   5
##   Low               0    0      0   0
```

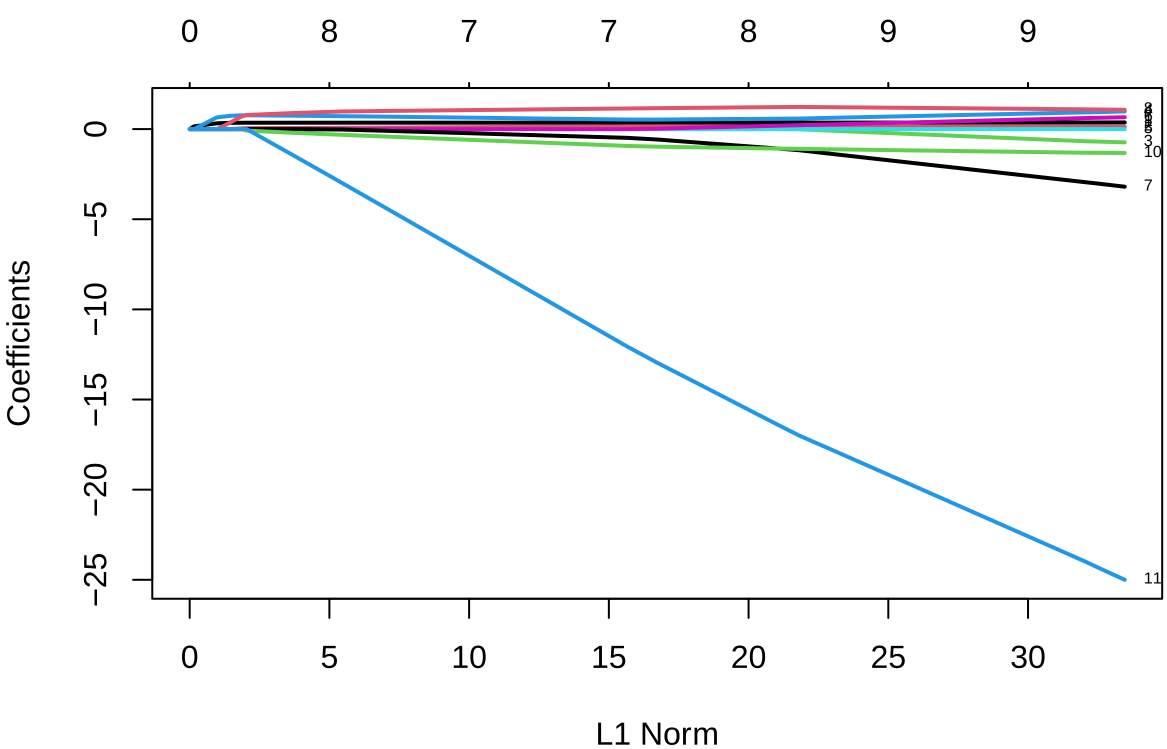
Lasso Regression

```
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-4
```

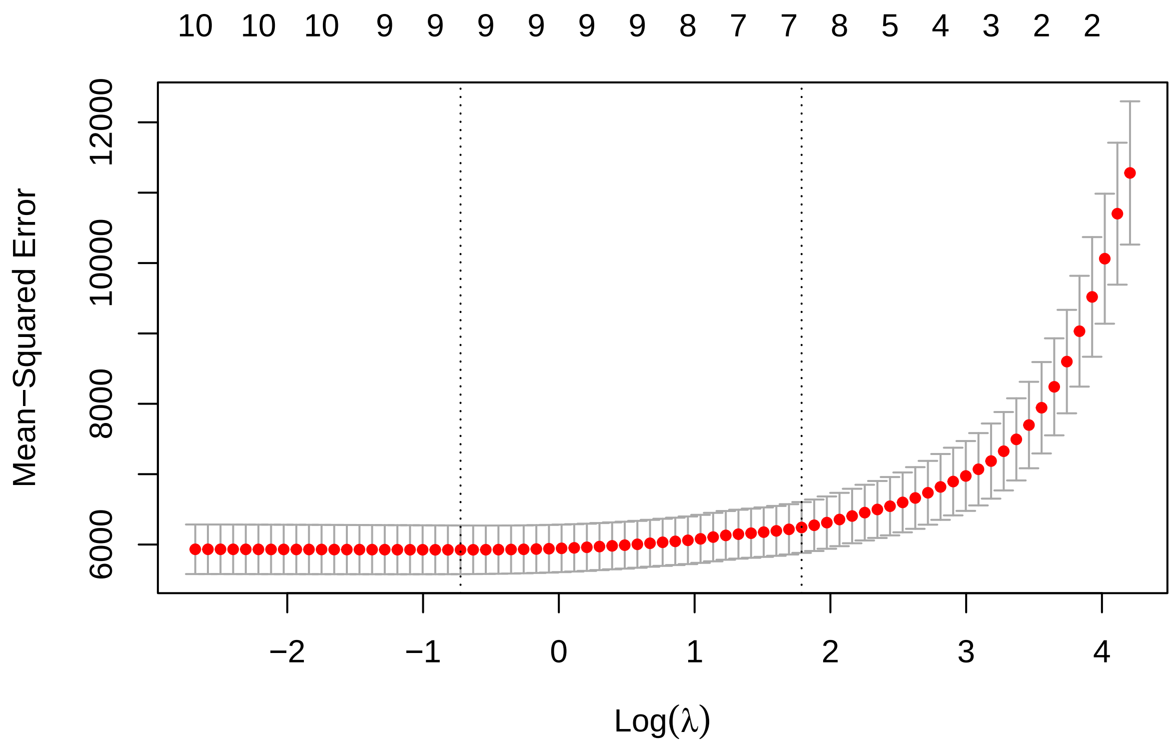
```
# Converting dataframes into matrix to fit glmnet function
x = as.matrix(reg_trn[,-c(12, 13)])
y = as.matrix(reg_trn[,13])
```

```
modlasso = glmnet(x, y)
plot(modlasso, label=TRUE, lwd=2)
```



Fit LASSO Model 1

```r
# Default -> alpha = 1 (LASSO)
cvfit = cv.glmnet(x, y)
plot(cvfit)
```




```r
# Converting the test data into matrix
newtest = reg_tst[,-c(12, 13)]
newX = as.matrix(newtest)
```

Fit LASSO Model 2

```r
# Fit LASSO model with the minimum lambda
lambda = cvfit$lambda.min
bestlasso = glmnet(x, y, lambda = lambda)
coef(bestlasso)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                            s0
## (Intercept)    -1042.4962144
## Max_Day_PM2.5      0.3641061
## Avg_Day_PM10       0.1297935
## Avg_Day_SO2       -0.6011327
## Avg_Day_NO2        0.8876815
## Avg_Day_CO         .
## Avg_Day_O3         0.5703099
## Avg_Day_TEMP      -2.7589021
## Avg_Day_PRES       1.1100001
```

```
## Avg_Day_DEWP        .
## Total_Day_Rain    -1.2873017
## Avg_Day_WSPM     -23.2653461
```

```r
Ytest.pred.las = predict(bestlasso, s = lambda, newx = newX)

# MSE
mean((Ytest.pred.las - as.numeric(reg_tst$Max_Tmrw_PM2.5))^2)
```

```
## [1] 7962.442
```

```r
# Comparing predicted PM2.5 levels of LASSO Model with the test labels (Actual/Observation)
YPM.las = category_PM2.5(Ytest.pred.las)
testPM = category_PM2.5(reg_tst$Max_Tmrw_PM2.5)

# Number of observations that are correctly predicted by the LASSO model
sum(YPM.las == testPM)
```

```
## [1] 178
```

```r
# Test Accuracy
las_acc = sum(YPM.las == testPM) / length(testPM)
las_acc
```

```
## [1] 0.6267606
```

Prediction vs Actual Table of LASSO Model

```r
table(Pred.las = YPM.las, Actual.Obs = testPM)
```

```
##              Actual.Obs
## Pred.las     Dangerous High Medium Low
##   Dangerous        151   23     24  12
##   High               9   10     10   4
##   Medium             6   10     17   5
##   Low                1    2      0   0
```

Performance of LASSO Model

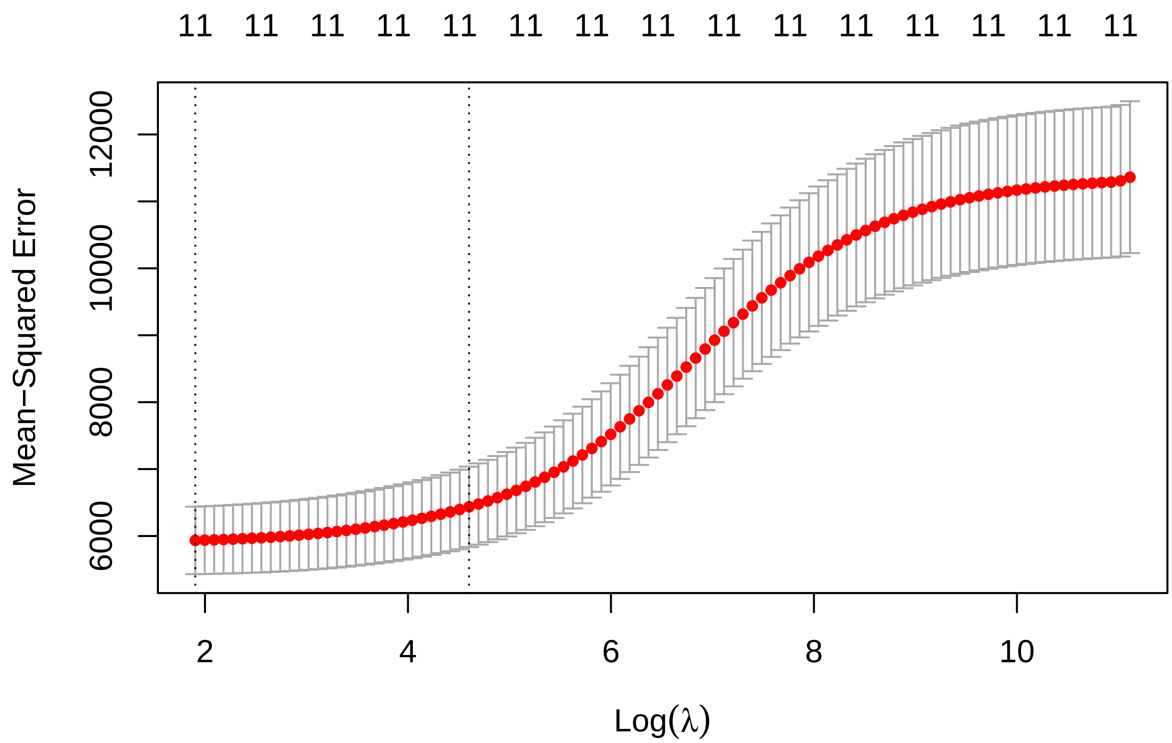
```r
las_perf = eval_results(reg_tst$Max_Tmrw_PM2.5, Ytest.pred.las, reg_tst, bestlasso)
las_perf
```

```
##       RMSE   Rsquare Adjusted_Rsquare
## 1 89.23252 0.3650764        0.3466059
```

Compared to the transformed model, the overall performance scores are better.

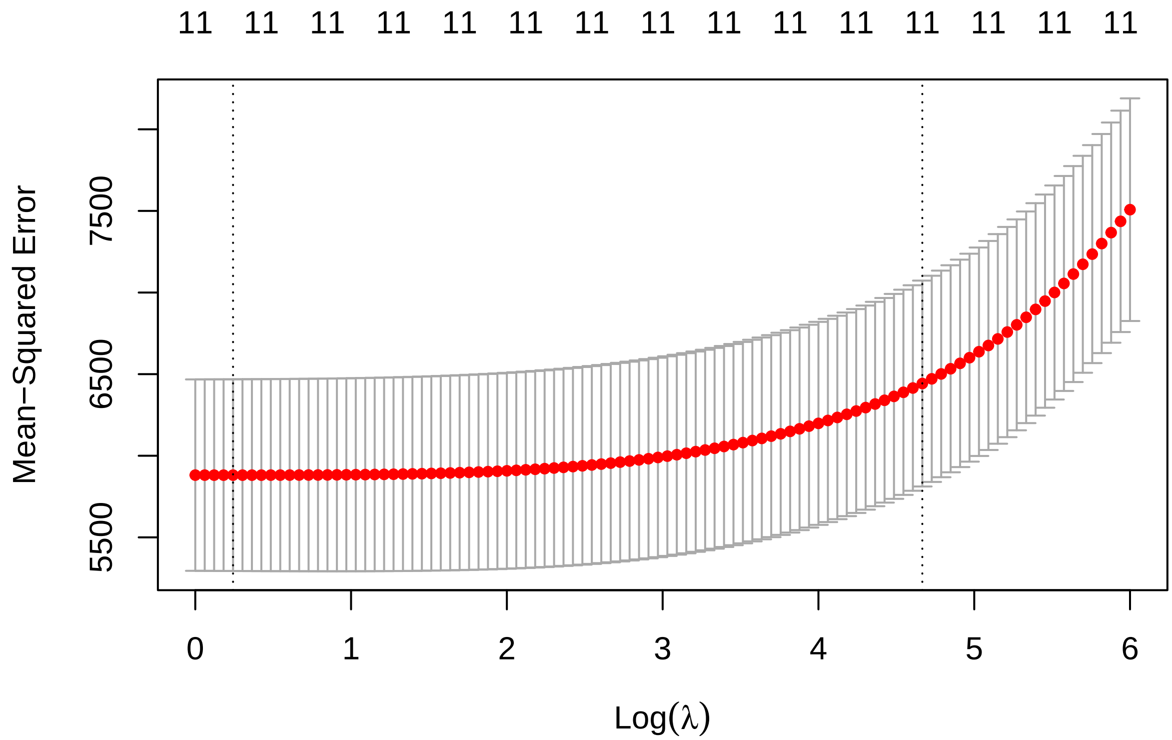
Ridge Regression

```
cvfit1 = cv.glmnet(x, y, alpha = 0)
plot(cvfit1)
```



Fit Ridge Model 1

```
lam.seq = exp(seq(0, 6, length=100))
cvfit1 = cv.glmnet(x, y, alpha = 0, lambda=lam.seq)
plot(cvfit1)
```



Fit Ridge Model 2

```
# Fit Ridge model with the minimum lambda
lambda1 = cvfit1$lambda.min
bestridge = glmnet(x, y, alpha = 0, lambda = lambda1)
coef(bestridge)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                            s0
## (Intercept)    -1.105782e+03
## Max_Day_PM2.5   3.597283e-01
## Avg_Day_PM10    1.452975e-01
## Avg_Day_SO2    -6.854923e-01
## Avg_Day_NO2     9.184284e-01
## Avg_Day_CO     -5.341606e-04
## Avg_Day_O3      6.065010e-01
## Avg_Day_TEMP   -2.798371e+00
## Avg_Day_PRES    1.172401e+00
## Avg_Day_DEWP   -1.073800e-01
## Total_Day_Rain -1.302772e+00
## Avg_Day_WSPM   -2.461248e+01
```

```
Ytest.pred.rid = predict(bestridge, s = lambda1, newx = newX)
mean((Ytest.pred.rid - as.numeric(reg_tst$Max_Tmrw_PM2.5))^2)
```

```
## [1] 7998.947
```

```r
# Comparing predicted PM2.5 levels of Ridge Model with the test labels (Actual/Observation)
YPM.rid = category_PM2.5(Ytest.pred.rid)
testPM = category_PM2.5(reg_tst$Max_Tmrw_PM2.5)

# Number of observations that are correctly predicted by the Ridge model
sum(YPM.rid == testPM)
```

```
## [1] 177
```

```r
# Test Accuracy
rid_acc = sum(YPM.rid == testPM) / length(testPM)
rid_acc
```

```
## [1] 0.6232394
```

Prediction vs Actual Table of Ridge Model

```r
table(Pred.Ridge = YPM.rid, Actual.Obs = testPM)
```

```
##              Actual.Obs
## Pred.Ridge   Dangerous High Medium Low
##   Dangerous        151   24     24  12
##   High               9    9     10   4
##   Medium             6   10     17   5
##   Low                1    2      0   0
```

Performance of Ridge Model

```r
rid_perf = eval_results(reg_tst$Max_Tmrw_PM2.5, Ytest.pred.rid, reg_tst, bestridge)
rid_perf
```

```
##       RMSE   Rsquare Adjusted_Rsquare
## 1 89.43683 0.3621656        0.3388017
```

Compared to the LASSO model, both the R^2 and adjusted R^2 is slightly lower.

Model Performance on Test Data Dataframe

```r
mod_perf_df = data.frame(
  Model = c("Transformed", "LASSO", "Ridge"),
  RMSE = c(trans_perf[1,1], las_perf[1,1], rid_perf[1,1]),
  Rsquare = c(trans_perf[1,2], las_perf[1,2], rid_perf[1,2]),
  Adjusted_Rsquare = c(trans_perf[1,3], las_perf[1,3], rid_perf[1,3]),
  Correct_Prediction = c(sum(YPM == testPM), sum(YPM.las == testPM), sum(YPM.rid == testPM)),
  Test_Prediction_Accuracy = c(trans_acc, las_acc, rid_acc)
)
```

```
## Warning in `==.default`(YPM, testPM): longer object length is not a multiple of
## shorter object length

## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length
```

```
mod_perf_df
```

```
##         Model     RMSE   Rsquare Adjusted_Rsquare Correct_Prediction
## 1 Transformed 94.71770 0.2846190        0.2691233                165
## 2       LASSO 89.23252 0.3650764        0.3466059                178
## 3       Ridge 89.43683 0.3621656        0.3388017                177
##   Test_Prediction_Accuracy
## 1                0.6042403
## 2                0.6267606
## 3                0.6232394
```

When comparing three different models' performances, we would prefer either the LASSO or Ridge more over the transformed model.

Multinomial

```
# Creating a new column
reg_trn$PM_type = category_PM2.5(reg_trn$Max_Tmrw_PM2.5)
reg_tst$PM_type = category_PM2.5(reg_tst$Max_Tmrw_PM2.5)
```

```
library(nnet)
multimodel = multinom(PM_type ~.-Max_Tmrw_PM2.5, data = reg_trn)
```

```
## # weights:  112 (81 variable)
## initial  value 1569.285217
## iter  10 value 1061.848121
## iter  20 value 1012.073561
## iter  30 value 987.605170
## iter  40 value 937.835172
## iter  50 value 928.874834
## iter  60 value 928.016084
## iter  70 value 927.420378
## iter  80 value 925.781668
## iter  90 value 925.192869
## iter 100 value 924.587218
## final  value 924.587218
## stopped after 100 iterations
```

```
multimodel1 = multinom(PM_type ~.-Max_Tmrw_PM2.5-Freq_Day_wd, data = reg_trn)
```

```
## # weights:  52 (36 variable)
## initial  value 1569.285217
## iter  10 value 1061.862169
```

```
## iter  20 value 1012.394278
## iter  30 value 998.282750
## iter  40 value 967.386828
## iter  50 value 963.833865
## iter  60 value 963.069529
## final  value 962.998584
## converged
```

```r
multimodel2 = multinom(PM_type ~ Max_Day_PM2.5 + Avg_Day_NO2 + Avg_Day_CO + Avg_Day_O3 + Avg_Day_PRES +
```

```
## # weights:  32 (21 variable)
## initial  value 1569.285217
## iter  10 value 1089.315997
## iter  20 value 1054.282322
## iter  30 value 1012.759046
## iter  40 value 1000.463629
## iter  50 value 1000.393890
## iter  60 value 1000.386926
## iter  70 value 1000.383899
## iter  70 value 1000.383890
## iter  80 value 1000.382777
## iter  80 value 1000.382774
## final  value 1000.382698
## converged
```

```r
multimodel3 = multinom(PM_type ~ Max_Day_PM2.5 + Avg_Day_SO2 + Avg_Day_NO2 + Avg_Day_CO + Avg_Day_O3 + A
```

```
## # weights:  44 (30 variable)
## initial  value 1569.285217
## iter  10 value 1079.464725
## iter  20 value 1035.799780
## iter  30 value 979.677255
## iter  40 value 976.069628
## iter  50 value 974.133650
## iter  60 value 973.923108
## final  value 973.918365
## converged
```

```r
AIC(multimodel)
```

```
## [1] 2011.174
```

```r
AIC(multimodel1)
```

```
## [1] 1997.997
```

```r
AIC(multimodel2)
```

```
## [1] 2042.765
```

```r
AIC(multimodel3)
```

```
## [1] 2007.837
```

```r
newpred = predict(multimodel, newdata = reg_tst, "class")
tab = table(newpred, test_pred = reg_tst$PM_type)
round((sum(diag(tab))/sum(tab))*100,2)
```

```
## [1] 63.73
```

```r
tab
```

```
##              test_pred
## newpred       Dangerous High Medium Low
##    Dangerous        151   28     22  13
##    High               2    2      2   0
##    Medium            12   12     26   6
##    Low                2    3      1   2
```

```r
newpred = predict(multimodel1, newdata = reg_tst, "class")
tab = table(Predicted = newpred, Actual_Obs = reg_tst$PM_type)
round((sum(diag(tab))/sum(tab))*100,2)
```

```
## [1] 64.44
```

```r
tab
```

```
##              Actual_Obs
## Predicted     Dangerous High Medium Low
##    Dangerous        156   29     24  13
##    High               0    0      0   0
##    Medium            10   13     26   7
##    Low                1    3      1   1
```

```r
summary(multimodel1)
```

```
## Call:
## multinom(formula = PM_type ~ . - Max_Tmrw_PM2.5 - Freq_Day_wd,
##     data = reg_trn)
##
## Coefficients:
##          (Intercept) Max_Day_PM2.5 Avg_Day_PM10 Avg_Day_SO2 Avg_Day_NO2
## High        1.150122   -0.004144494 -0.001659262  0.00203975 -0.01050711
## Medium     26.970564   -0.003458433 -0.014738241  0.02791218 -0.03728169
## Low        87.281097   -0.004589069 -0.009658746  0.03739640 -0.08711299
##            Avg_Day_CO  Avg_Day_O3 Avg_Day_TEMP Avg_Day_PRES Avg_Day_DEWP
## High   -0.0001797619 -0.01153043   0.05033082  -0.00176550  0.009264475
## Medium  0.0003977829 -0.03219292   0.20410930  -0.02713351 -0.043494719
## Low     0.0013231865 -0.04789430   0.09725895  -0.08521980 -0.012075609
##         Total_Day_Rain Avg_Day_WSPM
```

```
## High       0.06815594    0.4702726
## Medium     0.07614522    0.7143663
## Low        0.08617513    0.9238511
##
## Std. Errors:
##         (Intercept) Max_Day_PM2.5 Avg_Day_PM10 Avg_Day_SO2 Avg_Day_NO2
## High   0.0003289219    0.002052032  0.003010258  0.01276235 0.008694958
## Medium 0.0002844465    0.002127886  0.003821768  0.01497212 0.010620166
## Low    0.0002121215    0.002919650  0.005002081  0.01697479 0.015670691
##          Avg_Day_CO  Avg_Day_O3 Avg_Day_TEMP Avg_Day_PRES Avg_Day_DEWP
## High   0.0002942240 0.004455561   0.03122111 0.0006034205   0.02192370
## Medium 0.0003204394 0.004910829   0.03119875 0.0006554160   0.02127853
## Low    0.0003340430 0.008018019   0.04143565 0.0008421461   0.02852060
##        Total_Day_Rain Avg_Day_WSPM
## High       0.02206835    0.1797942
## Medium     0.02184770    0.1804258
## Low        0.02381778    0.2047281
##
## Residual Deviance: 1925.997
## AIC: 1997.997
```

```r
newpred = predict(multimodel2, newdata = reg_tst, "class")
tab = table(newpred, test_pred = reg_tst$PM_type)
round((sum(diag(tab))/sum(tab))*100,2)
```

```
## [1] 63.03
```

```r
tab
```

```
##              test_pred
## newpred      Dangerous High Medium Low
##   Dangerous        157   28     28  15
##   High               0    0      0   0
##   Medium             9   14     21   5
##   Low                1    3      2   1
```

```r
summary(multimodel2)
```

```
## Call:
## multinom(formula = PM_type ~ Max_Day_PM2.5 + Avg_Day_NO2 + Avg_Day_CO +
##     Avg_Day_O3 + Avg_Day_PRES + Avg_Day_WSPM, data = reg_trn)
##
## Coefficients:
##        (Intercept) Max_Day_PM2.5 Avg_Day_NO2    Avg_Day_CO  Avg_Day_O3
## High      49.11184  -0.004041688 -0.01508397 -2.976594e-04 -0.00897315
## Medium   115.11948  -0.007758898 -0.04918239 -6.295213e-06 -0.02458342
## Low      123.43095  -0.007346721 -0.08930191  1.311441e-03 -0.04114388
##        Avg_Day_PRES Avg_Day_WSPM
## High    -0.04803419    0.1741805
## Medium  -0.11094305    0.2453506
## Low     -0.11949726    0.5929032
##
```

```
## Std. Errors:
##         (Intercept) Max_Day_PM2.5 Avg_Day_NO2   Avg_Day_CO  Avg_Day_O3
## High   0.0004074701   0.001684090 0.006649306 0.0002383980 0.002690489
## Medium 0.0003916942   0.001941669 0.007556114 0.0002748507 0.003014867
## Low    0.0004133895   0.002470311 0.012197297 0.0002760135 0.005433935
##        Avg_Day_PRES Avg_Day_WSPM
## High   0.0004737341    0.1398905
## Medium 0.0005005980    0.1329302
## Low    0.0007118339    0.1563103
##
## Residual Deviance: 2000.765
## AIC: 2042.765
```

```
newpred = predict(multimodel3, newdata = reg_tst, "class")
tab = table(newpred, test_pred = reg_tst$PM_type)
round((sum(diag(tab))/sum(tab))*100,2)
```

```
## [1] 63.38
```

```
tab
```

```
##            test_pred
## newpred     Dangerous High Medium Low
##   Dangerous       156   29     26  14
##   High              0    0      1   0
##   Medium            9   14     23   6
##   Low               2    2      1   1
```

We have explored different combinations of predictors based on the exploration of linear regression models as mentioned before, like the model selected by BIC or the full model. The best model turns out to be the multinomial model without wind direction as explored before with the accuracy to be 0.6444.

KNN

```
creatednorm = function(x){
  (x - min(x))/(max(x) - min(x))
}
```

```
df_norm = as.data.frame(lapply(df_reg[, -c(12, 13)], creatednorm))
```

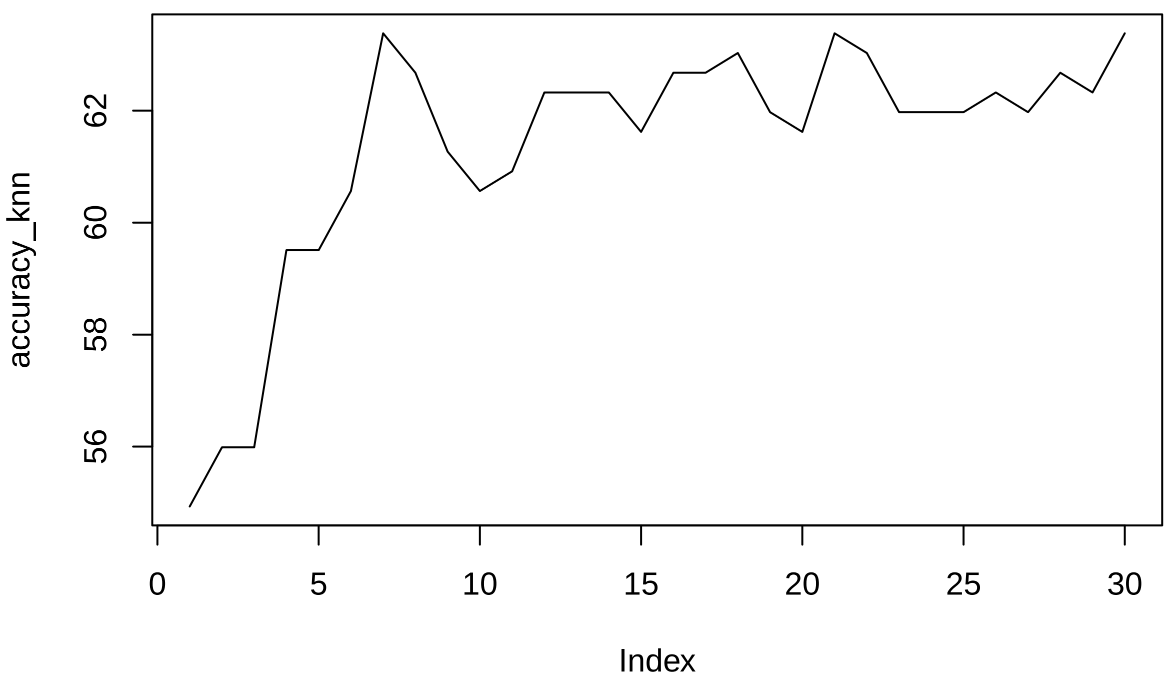
```
reg_trn1 = df_norm[train_index,]
reg_tst1 = df_norm[-train_index,]
```

```
PM_type = category_PM2.5(df_reg$Max_Tmrw_PM2.5)
reg_target_category = PM_type[train_index]
reg_test_category = PM_type[-train_index]
```

```
library(class)
set.seed(1)
accuracy_knn = numeric()
for (i in 1:30) {
  pr = knn(reg_trn1, reg_tst1, cl = reg_target_category, k = i)
  tab = table(pr, reg_test_category)
  accuracy = function(x){
    sum(diag(x)/(sum(rowSums(x)))) * 100
  }
  accuracy_knn[i]=accuracy(tab)
}
which.max(accuracy_knn)
```

```
## [1] 21
```

```
plot(accuracy_knn, type = "l")
```




```
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'lattice'

## The following object is masked from 'package:faraway':
##
##     melanoma

##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
set.seed(1)
pr = knn(reg_trn1, reg_tst1, cl = reg_target_category, k = 7)
tab = table(x = pr, y = reg_test_category)
confusionMatrix(table(pr ,reg_test_category))
```

```
## Confusion Matrix and Statistics
##
##            reg_test_category
## pr          Dangerous High Medium Low
##   Dangerous       146   22     25  12
##   High              6   11      6   3
##   Medium           13    6     17   3
##   Low               2    6      3   3
##
## Overall Statistics
##
##                Accuracy : 0.6232
##                  95% CI : (0.5641, 0.6798)
##     No Information Rate : 0.588
##     P-Value [Acc > NIR] : 0.125779
##
##                   Kappa : 0.2928
##
##  Mcnemar's Test P-Value : 0.001778
##
## Statistics by Class:
##
##                      Class: Dangerous Class: High Class: Medium Class: Low
## Sensitivity                    0.8743     0.24444       0.33333    0.14286
## Specificity                    0.4957     0.93724       0.90558    0.95817
## Pos Pred Value                 0.7122     0.42308       0.43590    0.21429
## Neg Pred Value                 0.7342     0.86822       0.86122    0.93333
## Prevalence                     0.5880     0.15845       0.17958    0.07394
## Detection Rate                 0.5141     0.03873       0.05986    0.01056
## Detection Prevalence           0.7218     0.09155       0.13732    0.04930
## Balanced Accuracy              0.6850     0.59084       0.61946    0.55052
```

We have explored another machine learning techniques, KNN, in this situation. For the exploration from 1 to 30 different neighbors, the best model with the highest accuracy should be the model with 21 nearest neighbors. As from the accuracy table, there may be a lot of cases in the category of dangerous, while few cases in the category of low, which may be a issue in choosing maximum PM2.5 as the daily representative.

Prediction vs Actual Table Interpretation

From the Prediction vs Actual table above, we would like to focus on the numbers of 'underpredicted', specifically 'Dangerous' or 'High' to be predicted as 'Medium' or 'Low' (Left bottom corner box, 4 numbers). One assumption we made wass that people may not likely to feel big difference between 'Dangerous' and 'High' compared to ('Dangerous'-'Medium') or ('High'-'Medium') although this might not be true for all people in the district. It would be great if we could have as least number of observations that are predicted incorrectly as possible, but considering our ultimate goal is to alarm 'Dangerous' (and 'High' as well) to people and prevent them from any outdoor activity to protect their health, predicting observations as ‘Dangerous’ or ‘High’ that are actually ‘Medium’ or ‘Low’ may be acceptable in this case but not vice versa.