IE6600 - Modeling_Fung

Yuqiao Feng

Inputing Bejing Air Quality Dataset

Data pre-processing

```
# Input data by using read_csv and make easy data formating----
mydata<-read.csv("C:\\Users\\kings\\Downloads\\df.csv")
#mydata
#summarize the data
summary(mydata)
```

```
##       year          month             day              hour
##  Min.   :2013   Min.   : 1.000   Min.   : 1.00   Min.   : 0.0
##  1st Qu.:2014   1st Qu.: 4.000   1st Qu.: 8.00   1st Qu.: 6.0
##  Median :2015   Median : 7.000   Median :16.00   Median :12.0
##  Mean   :2015   Mean   : 6.524   Mean   :15.72   Mean   :11.5
##  3rd Qu.:2016   3rd Qu.:10.000   3rd Qu.:23.00   3rd Qu.:17.0
##  Max.   :2017   Max.   :12.000   Max.   :31.00   Max.   :23.0
##      PM2.5             PM10            SO2                NO2
##  Min.   :  2.00   Min.   :  2.0   Min.   :  0.2856   Min.   :  1.026
##  1st Qu.: 21.00   1st Qu.: 36.0   1st Qu.:  3.0000   1st Qu.: 24.000
##  Median : 55.00   Median : 82.0   Median :  7.0000   Median : 43.000
##  Mean   : 79.19   Mean   :104.2   Mean   : 15.6534   Mean   : 50.373
##  3rd Qu.:109.00   3rd Qu.:144.0   3rd Qu.: 19.0000   3rd Qu.: 70.000
##  Max.   :999.00   Max.   :999.0   Max.   :500.0000   Max.   :290.000
##        CO              O3                TEMP             PRES
##  Min.   :  100   Min.   :   0.2142   Min.   :-19.90   Min.   : 982.4
##  1st Qu.:  500   1st Qu.:  12.0000   1st Qu.:  3.20   1st Qu.:1002.2
##  Median :  900   Median :  45.0000   Median : 14.50   Median :1010.4
##  Mean   : 1213   Mean   :  57.1151   Mean   : 13.56   Mean   :1010.7
##  3rd Qu.: 1500   3rd Qu.:  80.0000   3rd Qu.: 23.30   3rd Qu.:1019.0
##  Max.   :10000   Max.   :1071.0000   Max.   : 41.60   Max.   :1042.8
##       DEWP             RAIN              wd                WSPM
##  Min.   :-36.000   Min.   : 0.0000   Length:418946      Min.   : 0.000
##  1st Qu.: -8.900   1st Qu.: 0.0000   Class :character   1st Qu.: 0.900
##  Median :  3.100   Median : 0.0000   Mode  :character   Median : 1.400
##  Mean   :  2.493   Mean   : 0.0646                      Mean   : 1.735
##  3rd Qu.: 15.100   3rd Qu.: 0.0000                      3rd Qu.: 2.200
##  Max.   : 29.100   Max.   :72.5000                      Max.   :13.200
##    station
##  Length:418946
##  Class :character
##  Mode  :character
```

```
##
##
##
```

```
#convert orginial data to standard date format
mydata$Date<-as.Date(with(mydata,paste(year,month,day,sep="-")),"%Y-%m-%d")
#mydata
```

Calculating Mean of PM 2.5 and PM10

```
#mean of PM2.5 by year
df1 <- mydata[, c("PM2.5", "year")][order(mydata$year),] %>%
  group_by(year) %>%
  summarise(mean = mean(PM2.5)) %>%
  arrange(desc(year))
df1
```

```
## # A tibble: 5 x 2
##    year  mean
##   <int> <dbl>
## 1  2017  91.6
## 2  2016  71.4
## 3  2015  79.0
## 4  2014  84.7
## 5  2013  79.7
```

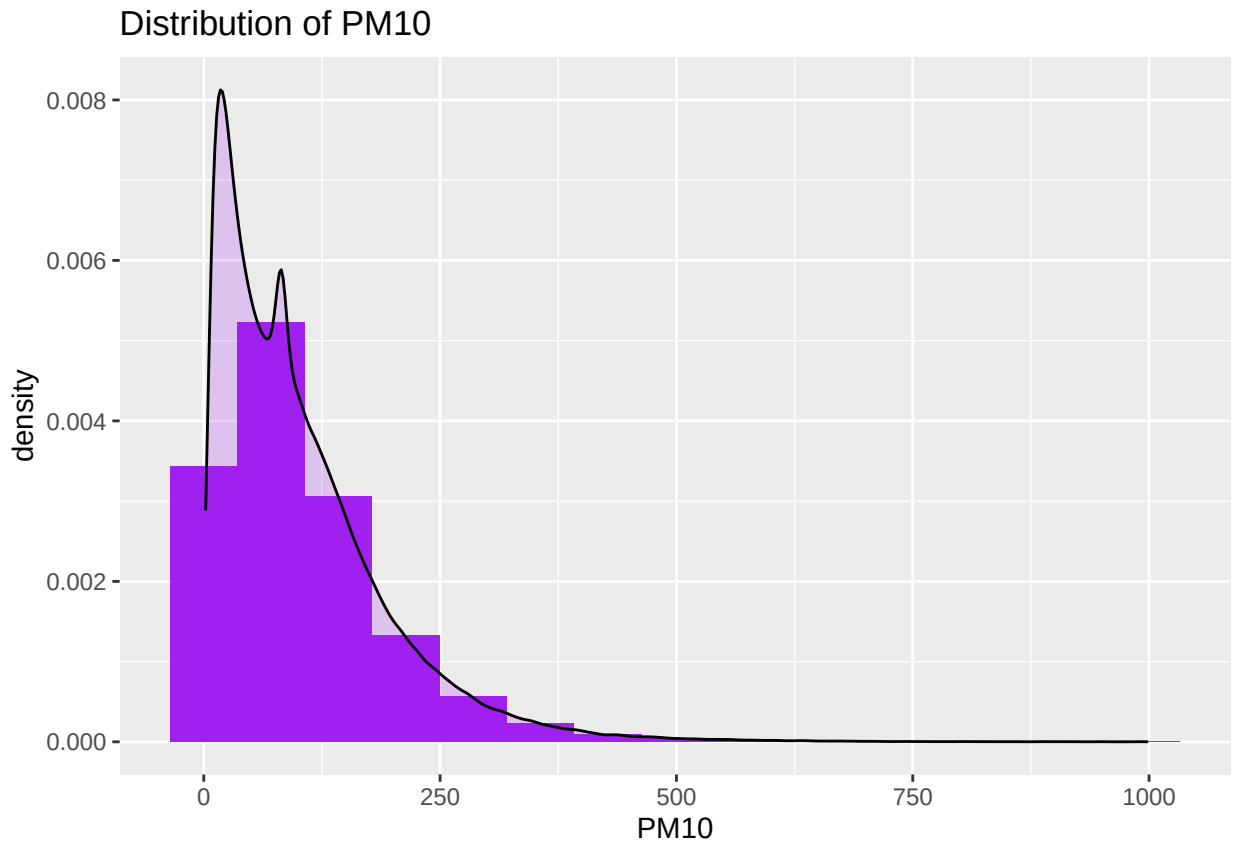
```
#mean of PM10 by year
df2 <- mydata[, c("PM10", "year")][order(mydata$year),] %>%
  group_by(year) %>%
  summarise(mean = mean(PM10)) %>%
  arrange(desc(year))

df2
```

```
## # A tibble: 5 x 2
##    year  mean
##   <int> <dbl>
## 1  2017 110.
## 2  2016  94.2
## 3  2015 103.
## 4  2014 115.
## 5  2013 103.
```

Plot graph by distribution of PM 2.5 and PM10

```
#distribution of PM10
ggplot(mydata, aes(PM10)) +
  geom_histogram(bins = 15, aes(y = ..density..), fill = "purple") +
  geom_density(alpha = 0.2, fill = "purple") +
  ggtitle("Distribution of PM10")
```

```
## Warning: The dot-dot notation (‘..density..‘) was deprecated in ggplot2 3.4.0.
## i Please use ‘after_stat(density)‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

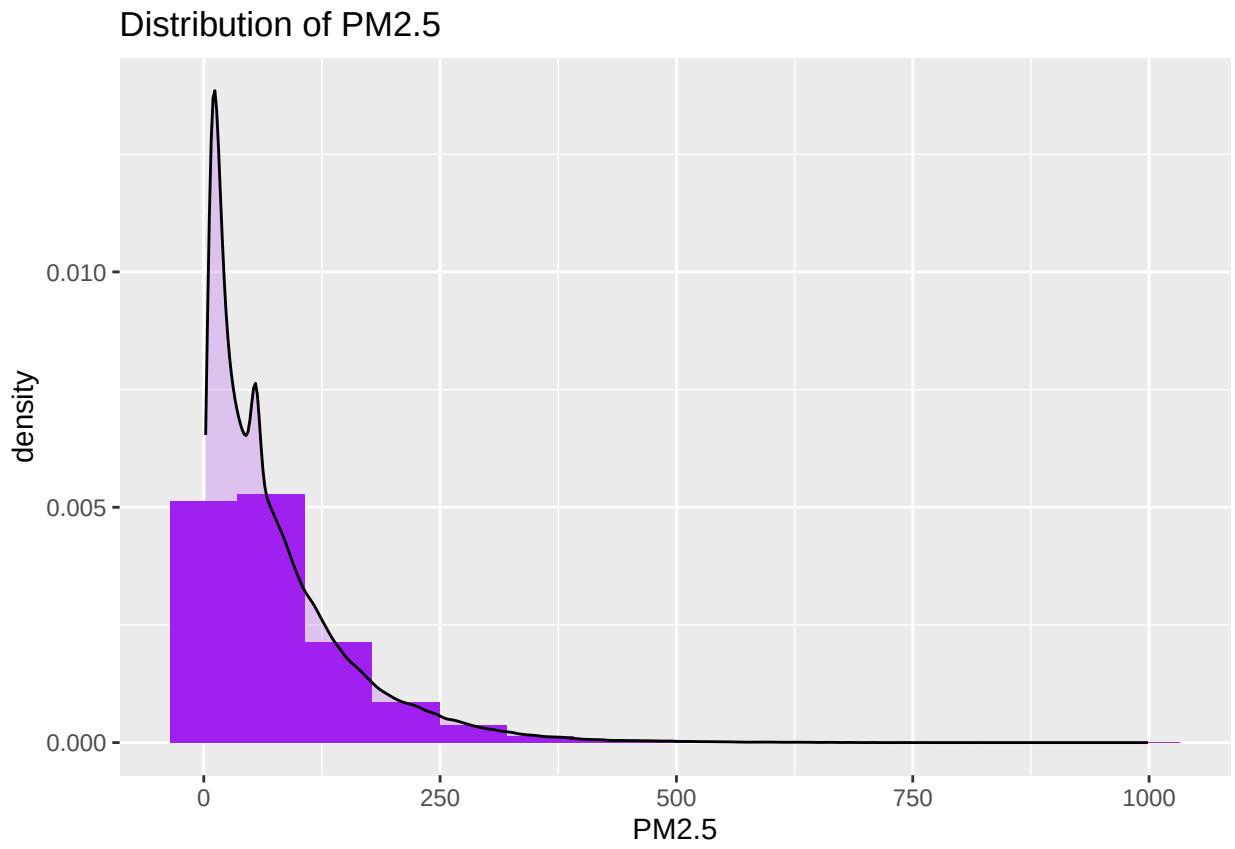



```
theme(axis.title = element_text(), axis.title.x = element_text())
```

```
## List of 2
##  $ axis.title  :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi FALSE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x:List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
```

```
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi FALSE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  - attr(*, "class")= chr [1:2] "theme" "gg"
##  - attr(*, "complete")= logi FALSE
##  - attr(*, "validate")= logi TRUE
```

```
#distribution of PM2.5
ggplot(mydata, aes(PM2.5)) +
  geom_histogram(bins = 15, aes(y = ..density..), fill = "purple") +
  geom_density(alpha = 0.2, fill = "purple") +
  ggtitle("Distribution of PM2.5")
```




```
theme(axis.title = element_text(), axis.title.x = element_text())
```

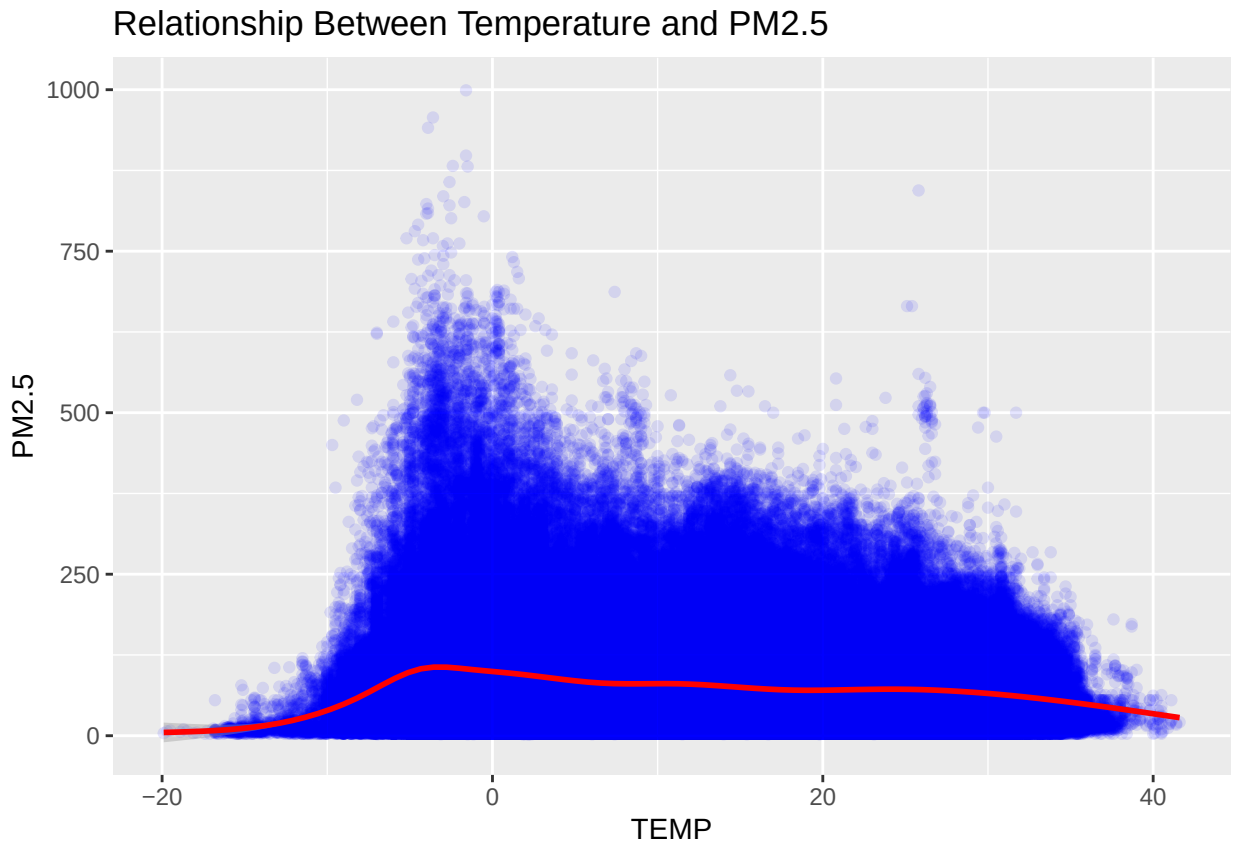
```
## List of 2
##  $ axis.title  :List of 11
##   ..$ family       : NULL
```

```
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi FALSE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x:List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi FALSE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  - attr(*, "class")= chr [1:2] "theme" "gg"
##  - attr(*, "complete")= logi FALSE
##  - attr(*, "validate")= logi TRUE
```

Plot graph by relationship of PM 2.5 and other variables

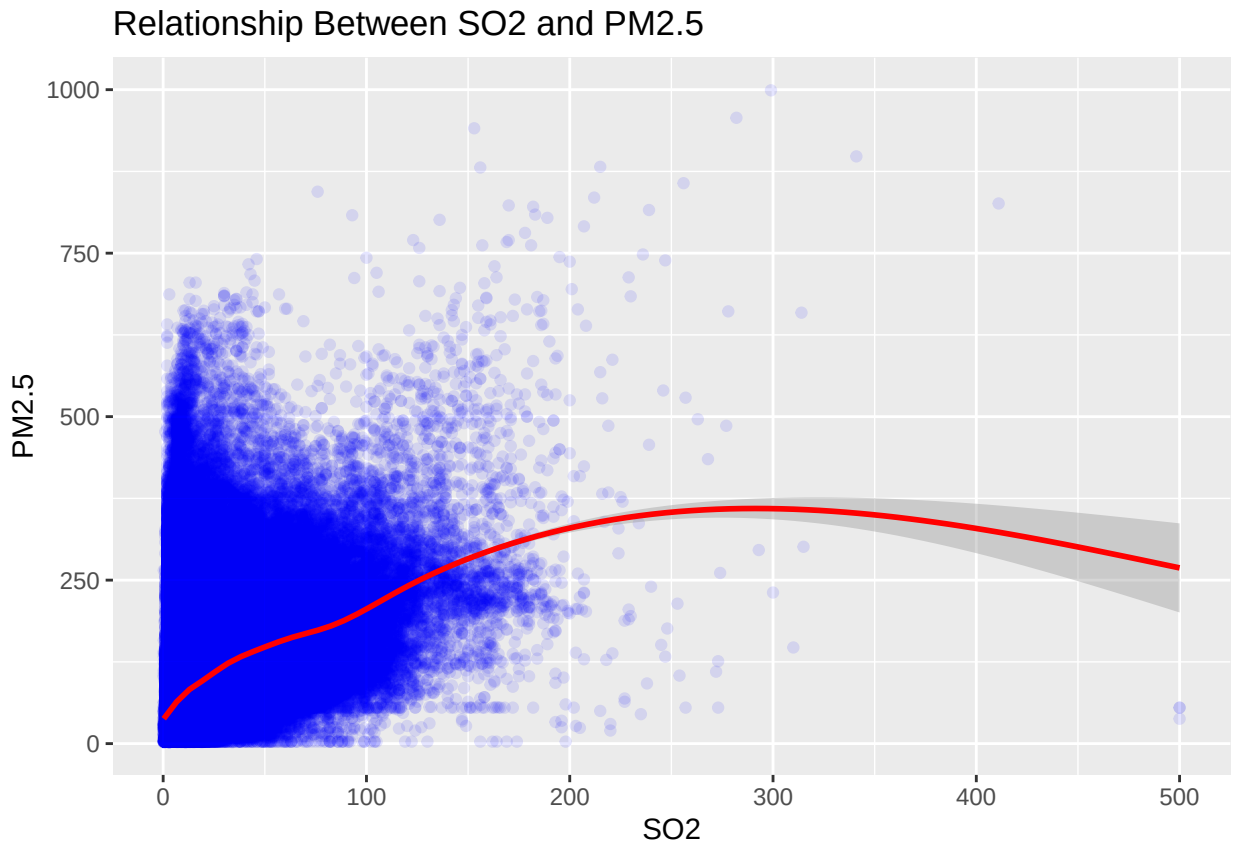
```
#plot the relationship between Temperature and PM2.5
ggplot(mydata, aes(x = TEMP, y = PM2.5)) +
  geom_point(alpha = 0.1, color = "blue") +
  geom_smooth(color = "red") +
  ggtitle("Relationship Between Temperature and PM2.5")
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



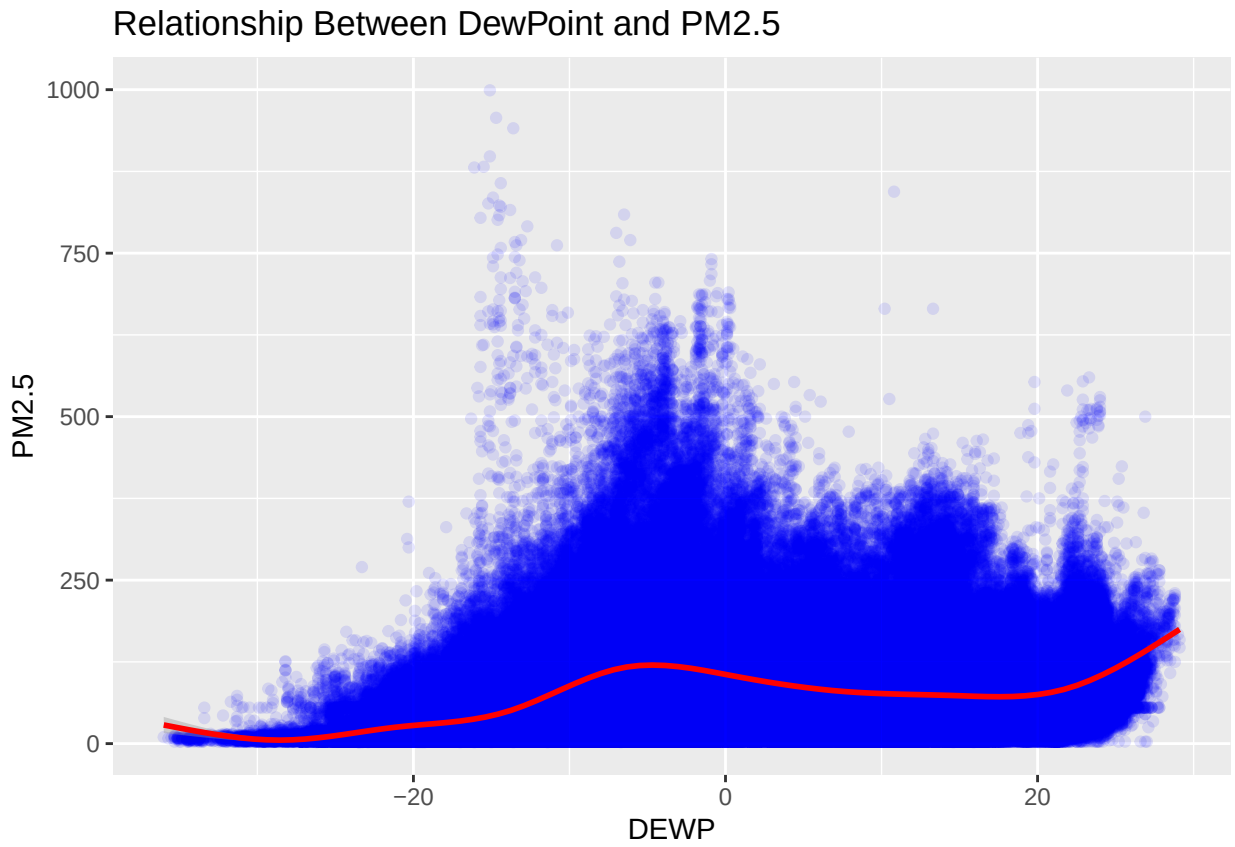

```
#plot the relationship between SO2 and PM2.5
ggplot(mydata, aes(x = SO2, y = PM2.5)) +
  geom_point(alpha = 0.1, color = "blue") +
  geom_smooth(color = "red") +
  ggtitle("Relationship Between SO2 and PM2.5")
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



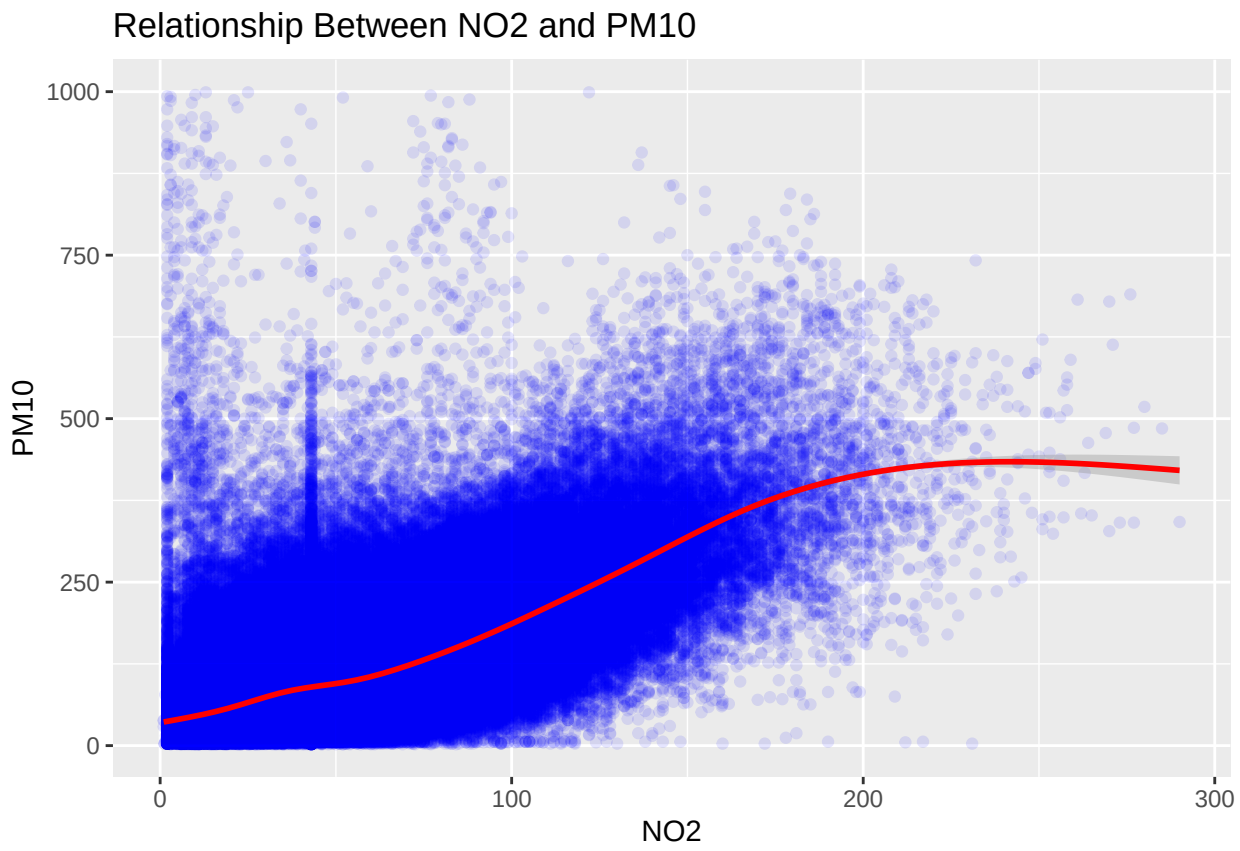

```
#plot the relationship between DewPoint and PM2.5
ggplot(mydata, aes(x = DEWP, y = PM2.5)) +
  geom_point(alpha = 0.1, color = "blue") +
  geom_smooth(color = "red") +
  ggtitle("Relationship Between DewPoint and PM2.5")
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

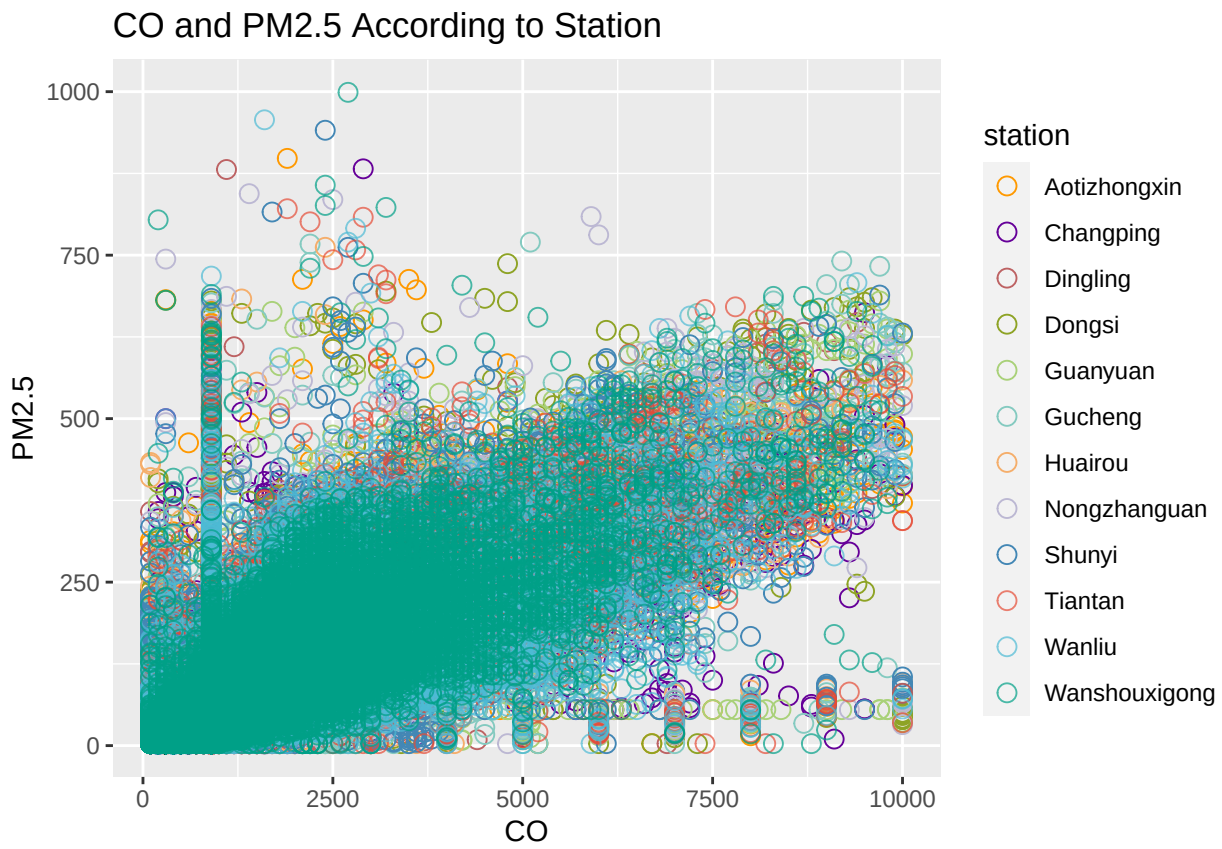



```
#plot the relationship between NO2 and PM10
ggplot(mydata, aes(x = NO2, y = PM10)) +
  geom_point(alpha = 0.1, color = "blue") +
  geom_smooth(color = "red") +
  ggtitle("Relationship Between NO2 and PM10")
```

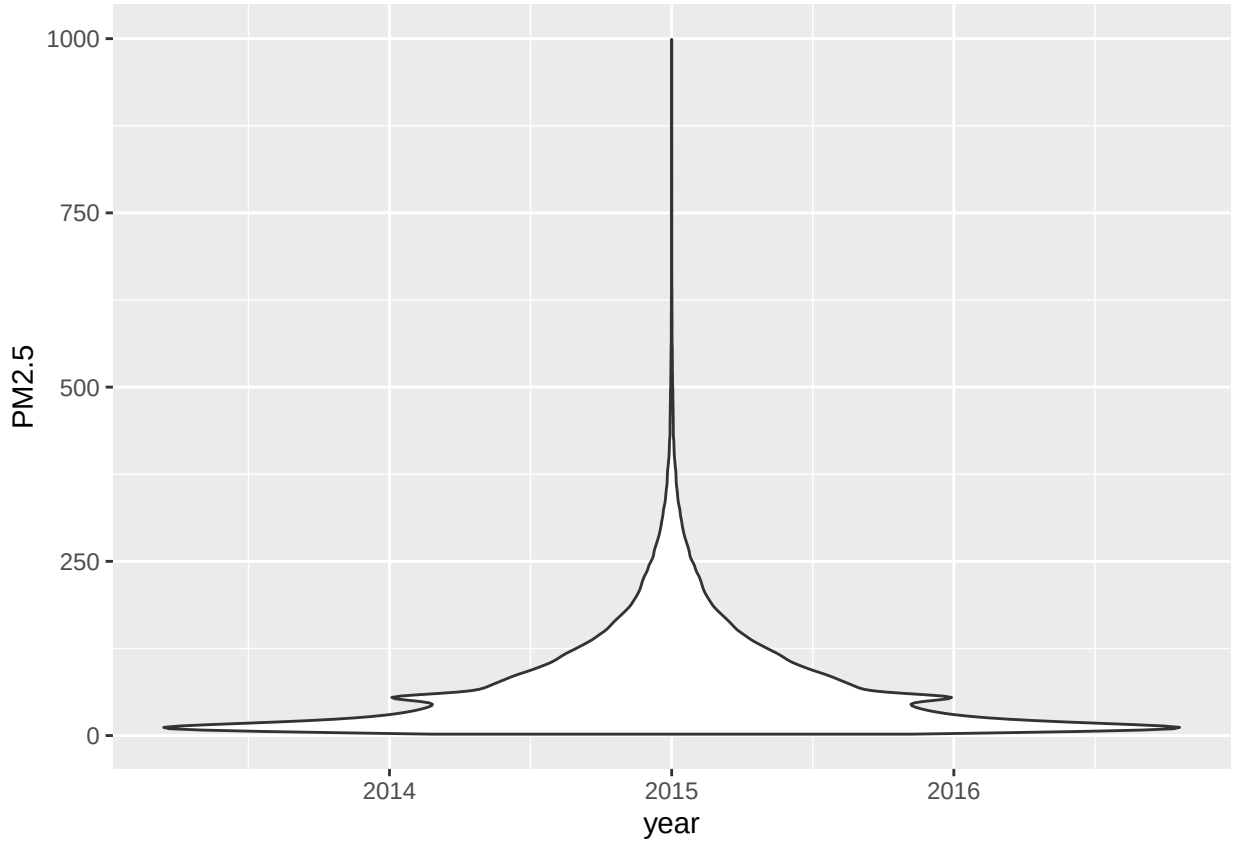
```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```




```
#plot the relationship between CO and PM2.5 according to station
ggplot(mydata, aes(x = CO, y = PM2.5, color = station)) +
  geom_point(size = 3, shape = 21) +
  scale_color_manual(values = c("#ff9900", "#660099","#BD6263","#8EA325","#A9D179","#84CAC0","#F5AE6B",
  ggtitle("CO and PM2.5 According to Station")
```




```
#plot violin graph to visualize PM 2.5
ggplot(mydata, aes(x = year, y = PM2.5)) +
  geom_violin()
```



Linear Regression:set wind direction as a factor

```
#Regression of wind direction
mydata$wd <- as.factor(mydata$wd)


m1 <- lm(mydata$PM2.5 ~mydata$year+ mydata$PM10 + mydata$SO2 + mydata$CO + mydata$O3 +mydata$NO2+mydata$
summary(m1)
```

```
##
## Call:
## lm(formula = mydata$PM2.5 ~ mydata$year + mydata$PM10 + mydata$SO2 +
##     mydata$CO + mydata$O3 + mydata$NO2 + mydata$wd)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -602.41  -13.11    0.49   12.51  894.94
##
## Coefficients:
##                 Estimate  Std. Error t value             Pr(>|t|)
## (Intercept)  323.0466663  90.8839074   3.554             0.000379 ***
## mydata$year   -0.1658310   0.0451090  -3.676             0.000237 ***
## mydata$PM10    0.5627373   0.0008683 648.091 < 0.0000000000000002 ***
## mydata$SO2     0.0277592   0.0029614   9.374 < 0.0000000000000002 ***
```

```
## mydata$CO      0.0209087   0.0000730 286.421 < 0.0000000000000002 ***
## mydata$O3      0.0532714   0.0011711  45.489 < 0.0000000000000002 ***
## mydata$NO2     0.1147336   0.0025187  45.553 < 0.0000000000000002 ***
## mydata$wdENE  -1.5494715   0.2646332  -5.855        0.00000000477 ***
## mydata$wdESE   1.5318550   0.2887373   5.305        0.00000011251 ***
## mydata$wdN    -3.0191627   0.2721007 -11.096 < 0.0000000000000002 ***
## mydata$wdNE   -1.4130597   0.2516705  -5.615        0.00000001970 ***
## mydata$wdNNE  -2.2885976   0.2778975  -8.235 < 0.0000000000000002 ***
## mydata$wdNNW  -4.7529146   0.2876131 -16.525 < 0.0000000000000002 ***
## mydata$wdNW   -6.6724568   0.2705992 -24.658 < 0.0000000000000002 ***
## mydata$wdS    -1.4592792   0.3108265  -4.695        0.00000266906 ***
## mydata$wdSE   -0.2885909   0.3056148  -0.944             0.345019
## mydata$wdSSE  -0.7959560   0.3191559  -2.494             0.012634 *
## mydata$wdSSW  -3.0193421   0.2983384 -10.121 < 0.0000000000000002 ***
## mydata$wdSW   -5.2934062   0.2790201 -18.971 < 0.0000000000000002 ***
## mydata$wdW    -6.7710134   0.3177255 -21.311 < 0.0000000000000002 ***
## mydata$wdWNW  -7.8804945   0.2900055 -27.174 < 0.0000000000000002 ***
## mydata$wdWSW  -5.3495092   0.3058519 -17.491 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 33.32 on 418924 degrees of freedom
## Multiple R-squared:  0.8266, Adjusted R-squared:  0.8266
## F-statistic: 9.508e+04 on 21 and 418924 DF,  p-value: < 0.00000000000000022
```

```
#m1:#PM2.5 = 323.04 + -0.16(year) + 0.56(PM10) + 0.02(SO2)+0.02(co)+0.05(O3)+0.11(NO2)-1.54(wdENE)-3.01
#-0.28(WdSE)-0.79(WdSSE)-3.01(wdSSW)-5.29(wdSW)-6.77(wdW)-7.88(wdWNW)-5.34(wdWSW)

m2 <- lm(mydata$PM10 ~ mydata$year+ mydata$PM2.5 + mydata$SO2 + mydata$CO + mydata$O3 +mydata$NO2+mydata
summary(m2)
```

```
##
## Call:
## lm(formula = mydata$PM10 ~ mydata$year + mydata$PM2.5 + mydata$SO2 +
##     mydata$CO + mydata$O3 + mydata$NO2 + mydata$wd)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -875.47  -19.72   -8.75    9.22  938.23
##
## Coefficients:
##                 Estimate  Std. Error t value             Pr(>|t|)
## (Intercept)  375.2548557 114.2750409   3.284             0.001024 **
## mydata$year   -0.1860509   0.0567190  -3.280             0.001037 **
## mydata$PM2.5   0.8896774   0.0013728 648.091 < 0.0000000000000002 ***
## mydata$SO2     0.0963912   0.0037210  25.905 < 0.0000000000000002 ***
## mydata$CO     -0.0012710   0.0001004 -12.666 < 0.0000000000000002 ***
## mydata$O3      0.1320017   0.0014619  90.292 < 0.0000000000000002 ***
## mydata$NO2     0.4634968   0.0030929 149.856 < 0.0000000000000002 ***
## mydata$wdENE  -2.1047449   0.3327398  -6.325  0.00000000025267023 ***
## mydata$wdESE   1.2469200   0.3630570   3.435             0.000594 ***
## mydata$wdN     1.8930801   0.3421693   5.533  0.00000003157373756 ***
## mydata$wdNE   -2.4926198   0.3164317  -7.877  0.00000000000000335 ***
## mydata$wdNNE  -1.3701541   0.3494422  -3.921  0.00008820528114206 ***
```

```
## mydata$wdNNW   5.0790467   0.3616691  14.043 < 0.0000000000000002 ***
## mydata$wdNW    8.5654268   0.3402332  25.175 < 0.0000000000000002 ***
## mydata$wdS     2.2972934   0.3908184   5.878  0.00000000415162155 ***
## mydata$wdSE    3.6315169   0.3842307   9.451 < 0.0000000000000002 ***
## mydata$wdSSE   3.7571700   0.4012583   9.363 < 0.0000000000000002 ***
## mydata$wdSSW   3.4116649   0.3751308   9.095 < 0.0000000000000002 ***
## mydata$wdSW    4.3152711   0.3509191  12.297 < 0.0000000000000002 ***
## mydata$wdW     4.4390231   0.3996564  11.107 < 0.0000000000000002 ***
## mydata$wdWNW   9.6738959   0.3646595  26.529 < 0.0000000000000002 ***
## mydata$wdWSW   2.5842081   0.3846890   6.718  0.00000000001849070 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 41.9 on 418924 degrees of freedom
## Multiple R-squared:  0.7884, Adjusted R-squared:  0.7884
## F-statistic: 7.433e+04 on 21 and 418924 DF,  p-value: < 0.00000000000000022
```

```
#m2:#PM10 = 375.25 + -0.186(year) + 0.889(PM2.5) + 0.096(SO2)-0.001(co)+0.132(O3)+0.463(NO2)-2.104(wdEN
#-0.28(WdSE)-0.79(WdSSE)-3.01(wdSSW)-5.29(wdSW)-6.77(wdW)-7.88(wdWNW)-5.34(wdWSW)
```

Linear Regression:set station as a factor

```
#Linear regression of station
#In order to fit this regression model and tell R that the variable "program" is a categorical variable

mydata$station <- as.factor(mydata$station)

#fit linear regression model
fit <- lm(PM2.5 ~ PM10 + SO2+NO2+CO+O3+station, data = mydata)
summary(fit)
```

```
##
## Call:
## lm(formula = PM2.5 ~ PM10 + SO2 + NO2 + CO + O3 + station, data = mydata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -601.37  -13.23    0.26   12.54  895.36
##
## Coefficients:
##                          Estimate   Std. Error t value             Pr(>|t|)
## (Intercept)          -19.36164253   0.22870213 -84.659 < 0.0000000000000002 ***
## PM10                   0.55757854   0.00086983 641.023 < 0.0000000000000002 ***
## SO2                    0.02263090   0.00289788   7.809  0.00000000000000576 ***
## NO2                    0.17725825   0.00261932  67.673 < 0.0000000000000002 ***
## CO                     0.02078761   0.00007222 287.852 < 0.0000000000000002 ***
## O3                     0.06581994   0.00110067  59.800 < 0.0000000000000002 ***
## stationChangping       1.75232492   0.25311203   6.923  0.00000000000442435 ***
## stationDingling        9.89969138   0.25798217  38.374 < 0.0000000000000002 ***
## stationDongsi          3.19562259   0.25174020  12.694 < 0.0000000000000002 ***
## stationGuanyuan        0.91201733   0.25123799   3.630             0.000283 ***
```

```
## stationGucheng        -4.55285130   0.25177386 -18.083 < 0.0000000000000002 ***
## stationHuairou         6.43547601   0.25724136  25.017 < 0.0000000000000002 ***
## stationNongzhanguan    1.35567414   0.25131501   5.394  0.00000006881978917 ***
## stationShunyi          7.34119512   0.25424860  28.874 < 0.0000000000000002 ***
## stationTiantan         1.82663843   0.25177366   7.255  0.00000000000040211 ***
## stationWanliu         -1.04904552   0.25156166  -4.170  0.00003044834656828 ***
## stationWanshouxigong  -0.54635720   0.25160079  -2.172             0.029892 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 33.22 on 418929 degrees of freedom
## Multiple R-squared:  0.8276, Adjusted R-squared:  0.8276
## F-statistic: 1.257e+05 on 16 and 418929 DF,  p-value: < 0.00000000000000022
```

```
#From the values in the Estimate column, we can write the fitted regression model:

#PM2.5 = -19.36 + .55(PM10) + 0.02(SO2) + 0.17(NO2)+0.02(co)+0.06(O3)+1.75(Changping)+9.89(Dingling)+3.
```

Ridge Model Regression

```
#Ridge regression is a method we can use to fit a regression model when multicollinearity is present in
#To perform ridge regression, we'll use functions from the glmnet package. This package requires the re

library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.2.3
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

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

```
## Loaded glmnet 4.1-7
```

```
y <- mydata$PM2.5
x <- data.matrix(mydata[, c('PM10','SO2', 'NO2', 'CO', 'O3')])
#fit ridge model

#we'll use the glmnet() function to fit the ridge regression model and specify alpha=0.

# ridge regression requires the data to be standardized such that each predictor variable has a mean of


model <- glmnet(x, y, alpha = 0)

#view the summary of model
```

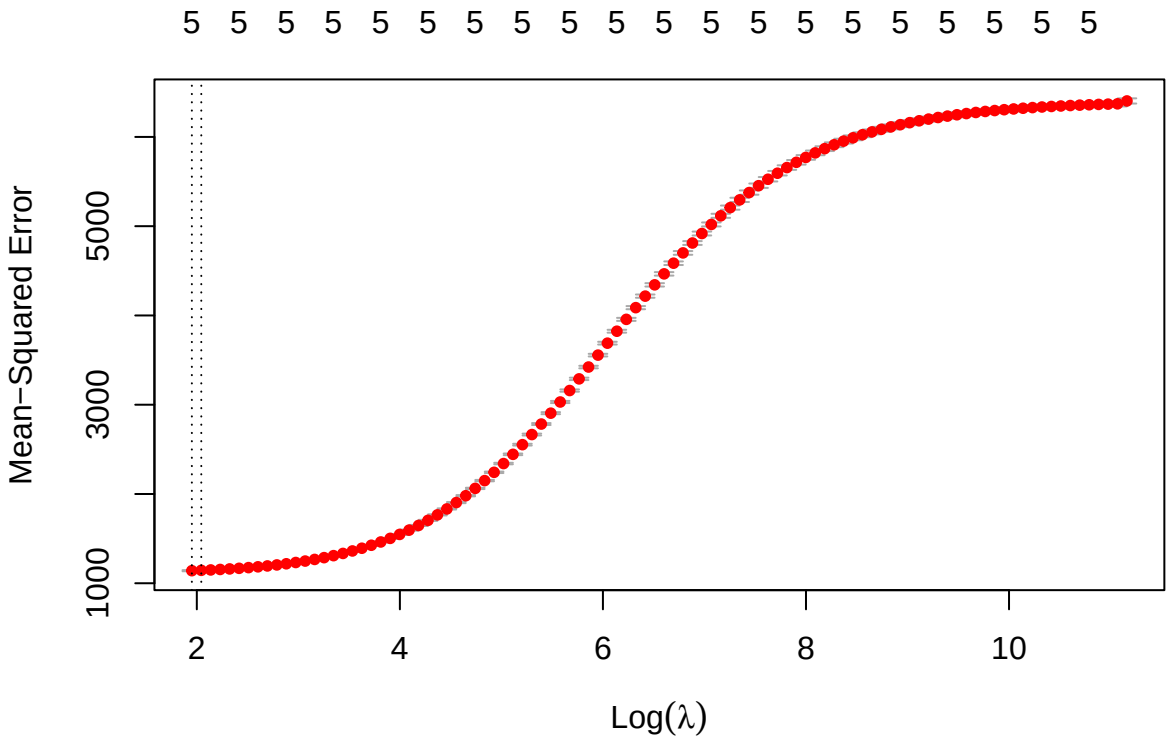
```
summary(model)
```

```
##           Length Class     Mode
## a0        100    -none-    numeric
## beta      500    dgCMatrix S4
## df        100    -none-    numeric
## dim         2    -none-    numeric
## lambda    100    -none-    numeric
## dev.ratio 100    -none-    numeric
## nulldev     1    -none-    numeric
## npasses     1    -none-    numeric
## jerr        1    -none-    numeric
## offset      1    -none-    logical
## call        4    -none-    call
## nobs        1    -none-    numeric
```

```
#Next, we'll identify the lambda value that produces the lowest test mean squared error (MSE) by using

#perform k-fold cross-validation to find optimal lambda value


cv_model <- cv.glmnet(x, y, alpha = 0)

#find optimal lambda value that minimizes test MSE
best_lambda <- cv_model$lambda.min
best_lambda
```

```
## [1] 7.036282
```

```
#produce plot of test MSE by lambda value
plot(cv_model)
```




```
#The lambda value that minimizes the test MSE turns out to be 7.0362.


best_model <- glmnet(x, y, alpha = 0, lambda = best_lambda)
coef(best_model)
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept) -13.72545524
## PM10          0.49314785
## SO2           0.09795717
## NO2           0.22288125
## CO            0.02053946
## O3            0.06771158
```

```
#fitted model:PM2.5=-13.725+0.49PM10+0.09so2+0.22NO2+0.02CO+0.06703




#Lastly, we can analyze the final model produced by the optimal lambda value.

#We can use the following code to obtain the coefficient estimates for this model:
```

```
y_predicted <- predict(model, s = best_lambda, newx = x)

#find SST and SSE
sst <- sum((y - mean(y))^2)
sse <- sum((y_predicted - y)^2)

#find R-Squared
rsq <- 1 - sse/sst

rsq
```

```
## [1] 0.8218718
```

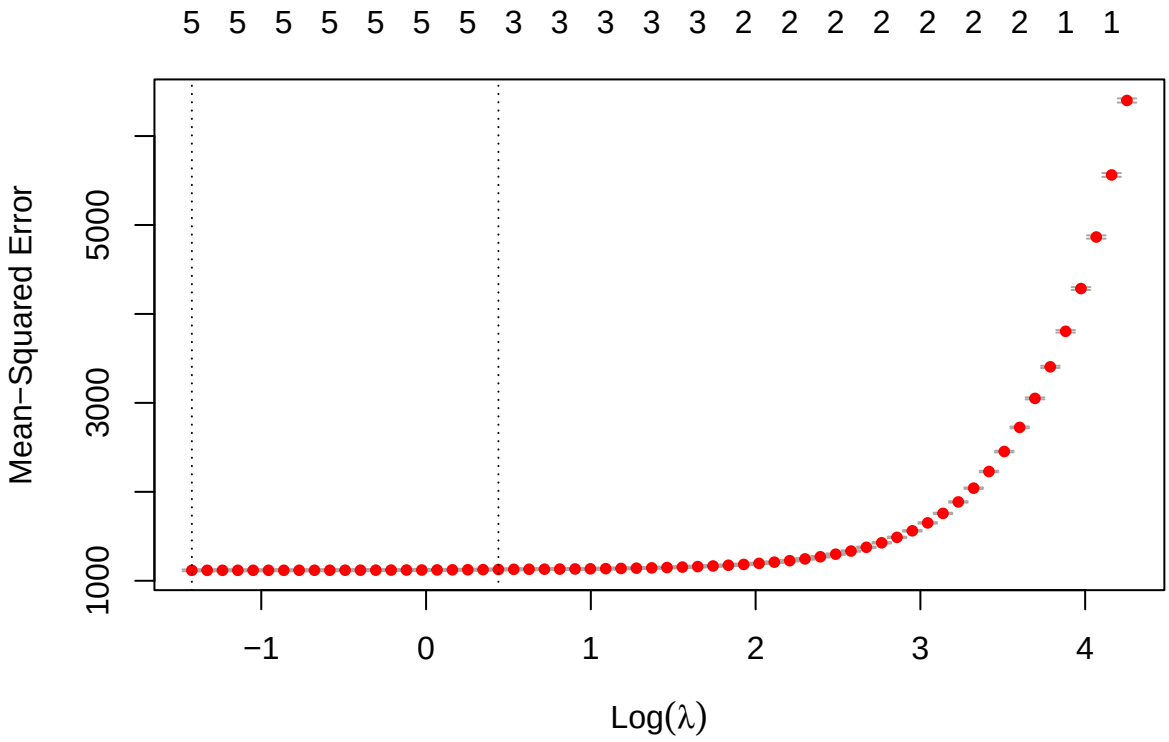
```
#The R-squared turns out to be 0.8218. That is, the best model was able to explain 82.1% of the variati
```

Lasso regression

```
#Lasso regression

#To perform lasso regression, we'll use functions from the glmnet package. This package requires the re
#define response variable
y <- mydata$PM2.5

#define matrix of predictor variables
x <- data.matrix(mydata[, c('PM10','SO2', 'NO2', 'CO', 'O3')])


library(glmnet)

#perform k-fold cross-validation to find optimal lambda value
cv_model <- cv.glmnet(x, y, alpha = 1)

#find optimal lambda value that minimizes test MSE
best_lambda <- cv_model$lambda.min
best_lambda
```

```
## [1] 0.2413775
```

```
#produce plot of test MSE by lambda value
plot(cv_model)
```




```
#The lambda value that minimizes the test MSE turns out to be 0.21.


#Lastly, we can analyze the final model produced by the optimal lambda value.

#We can use the following code to obtain the coefficient estimates for this model:
#find coefficients of best model
best_model <- glmnet(x, y, alpha = 1, lambda = best_lambda)
coef(best_model)
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept) -13.89451463
## PM10          0.56332051
## SO2           0.02401005
## NO2           0.11445905
## CO            0.02099390
## O3            0.04890430
```

```
#                       s0
#(Intercept) -13.89451463
#PM10          0.56332051
#SO2           0.02401005
#NO2           0.11445905
#CO            0.02099390
```

```
#O3            0.04890430

#model PM2.5=-13.894+0.56PM10+0.02SO2+0.11NO2+0.02CO+0.04O3

#use fitted best model to make predictions
y_predicted <- predict(best_model, s = best_lambda, newx = x)
#y_predicted
#find SST and SSE
sst <- sum((y - mean(y))^2)
sse <- sum((y_predicted - y)^2)

#find R-Squared
rsq <- 1 - sse/sst
rsq
```

```
## [1] 0.8255041
```

```
#The R-squared turns out to be 0.825. That is, the best model was able to explain 82.5% of the variatio
```