Appendix B

Group1: Sean Torres and Anusia Edward

```r
#Importing Dataset
air <- read.csv("~/Desktop/Shunyi.csv")
```

Data Pre-Processing

```r
sum(is.na(air$PM2.5))
```

```
## [1] 913
```

```r
# won't let me split because of the NAs in the outcome var
# filling outcome var
air.o <- kNN(air, variable = c("PM2.5"))
air.o <- subset(air.o, select = year:WSPM)
x <- subset(air.o, select = -PM2.5)
y <- subset(air.o, select = PM2.5)
# splitting dataset to ensure no further data leakage
set.seed(1)
trainset <- createDataPartition(air.o$PM2.5, p = 0.8, list = FALSE)
x.train <- x[trainset, ]
y.train <- y[trainset, ]
x.test <- x[-trainset, ]
y.test <- y[-trainset, ]
y.train1 <- as.data.frame(y.train)
y.test1 <- as.data.frame(y.test)
#imputing missing values using KNN
sum(is.na(x.train))
```

```
## [1] 5782
```

```r
x.train <- kNN(x.train)
x.train <- subset(x.train, select = year:WSPM)
sum(is.na(x.test))
```

```
## [1] 1345
```

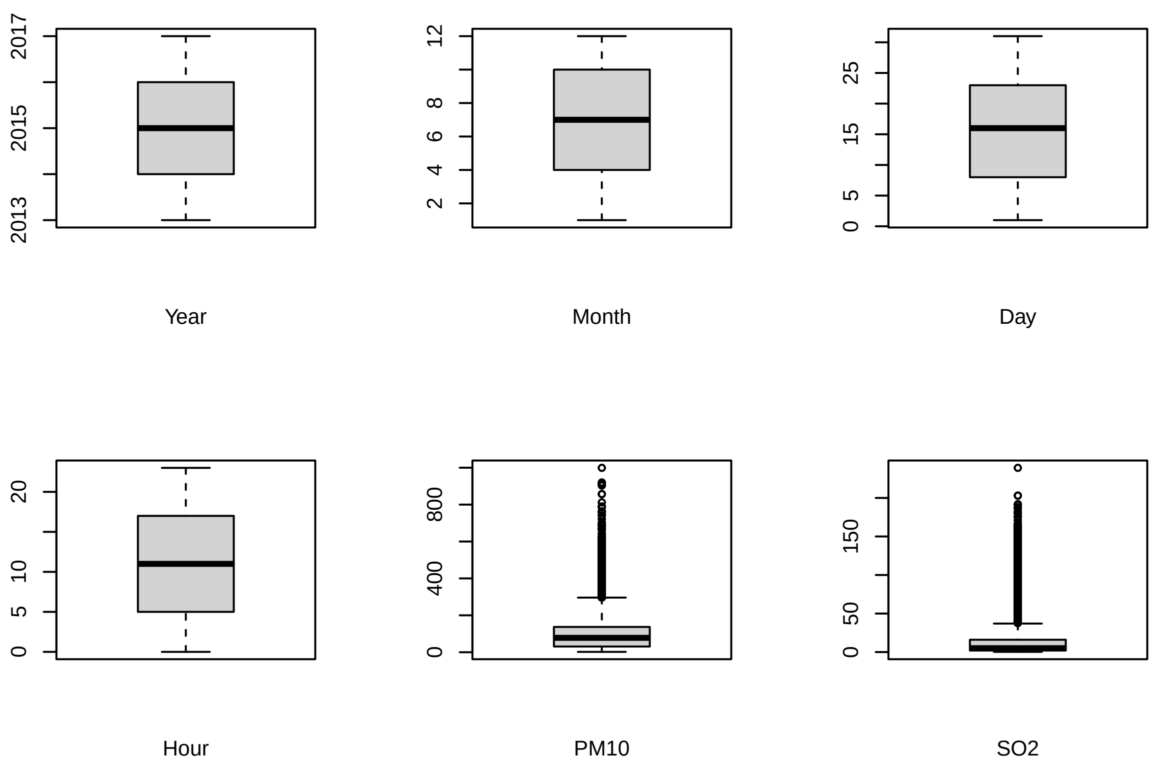
```r
x.test <- kNN(x.test)
x.test <- subset(x.test, select = year:WSPM)
sum(is.na(y.train))
```

```
## [1] 0
```

```
sum(is.na(y.test))
```

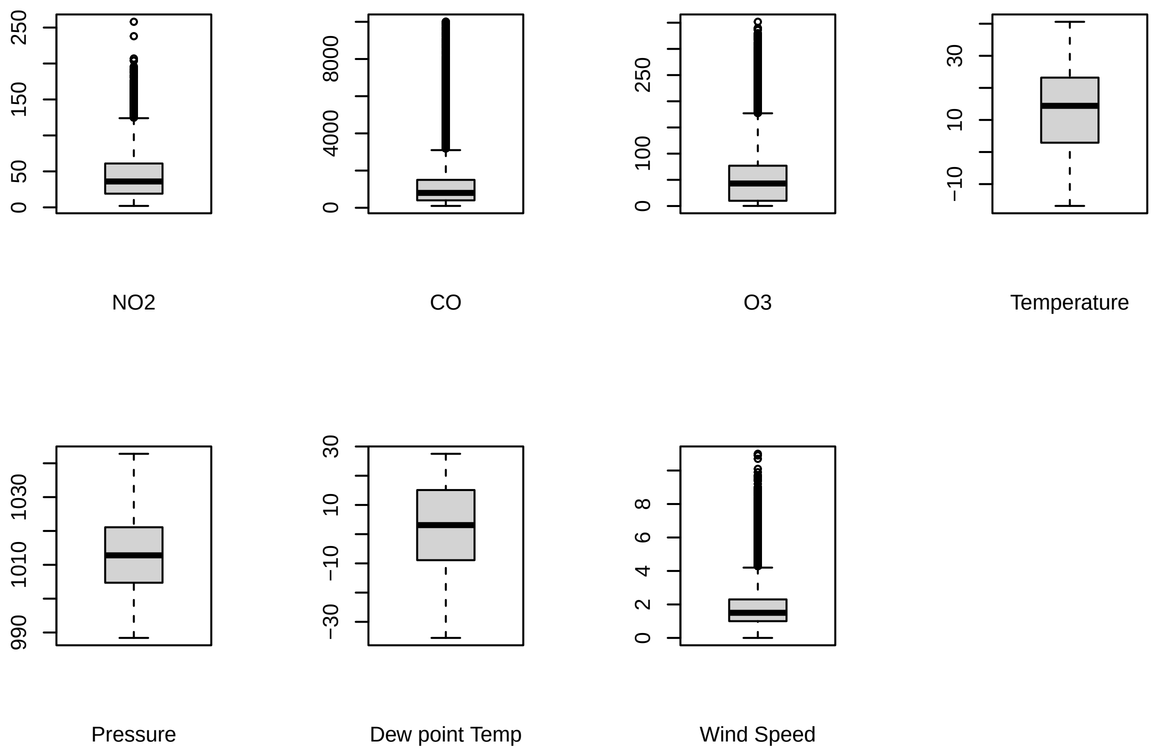
```
## [1] 0
```

```
# near Zero Variance removal
nZV.x <- nearZeroVar(x.train)
x.train <- x.train[, -nZV.x]
x.test <- x.test[, -nZV.x]
# visualizing outliers
par(mfrow = c(2,3))
boxplot(x.train$year,xlab = "Year")
boxplot(x.train$month, xlab = "Month")
boxplot(x.train$day, xlab = "Day")
boxplot(x.train$hour, xlab = "Hour")
boxplot(x.train$PM10, xlab = "PM10")
boxplot(x.train$SO2, xlab = "SO2")
```

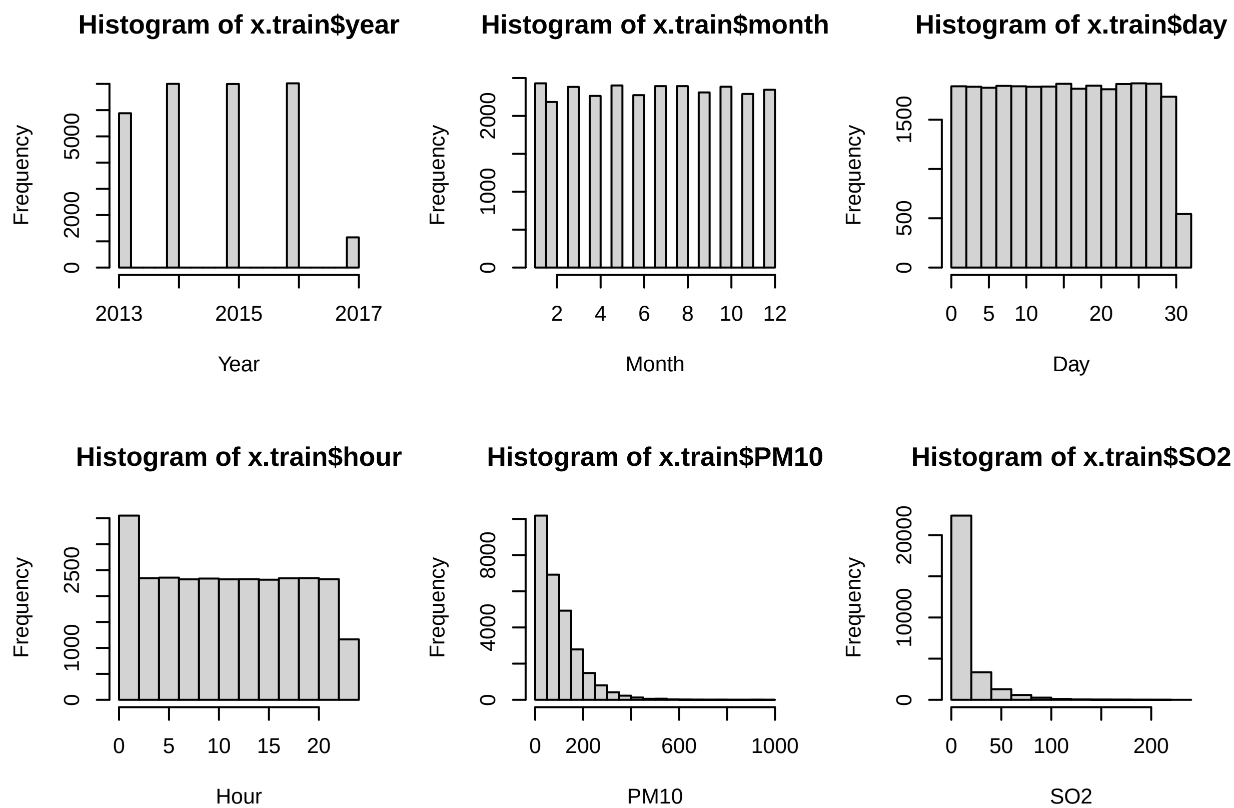



```
par(mfrow = c(2,4))
boxplot(x.train$NO2, xlab = "NO2")
boxplot(x.train$CO, xlab = "CO")
boxplot(x.train$O3, xlab = "O3")
boxplot(x.train$TEMP, xlab = "Temperature")
boxplot(x.train$PRES, xlab = "Pressure")
boxplot(x.train$DEWP, xlab = "Dew point Temp")
```

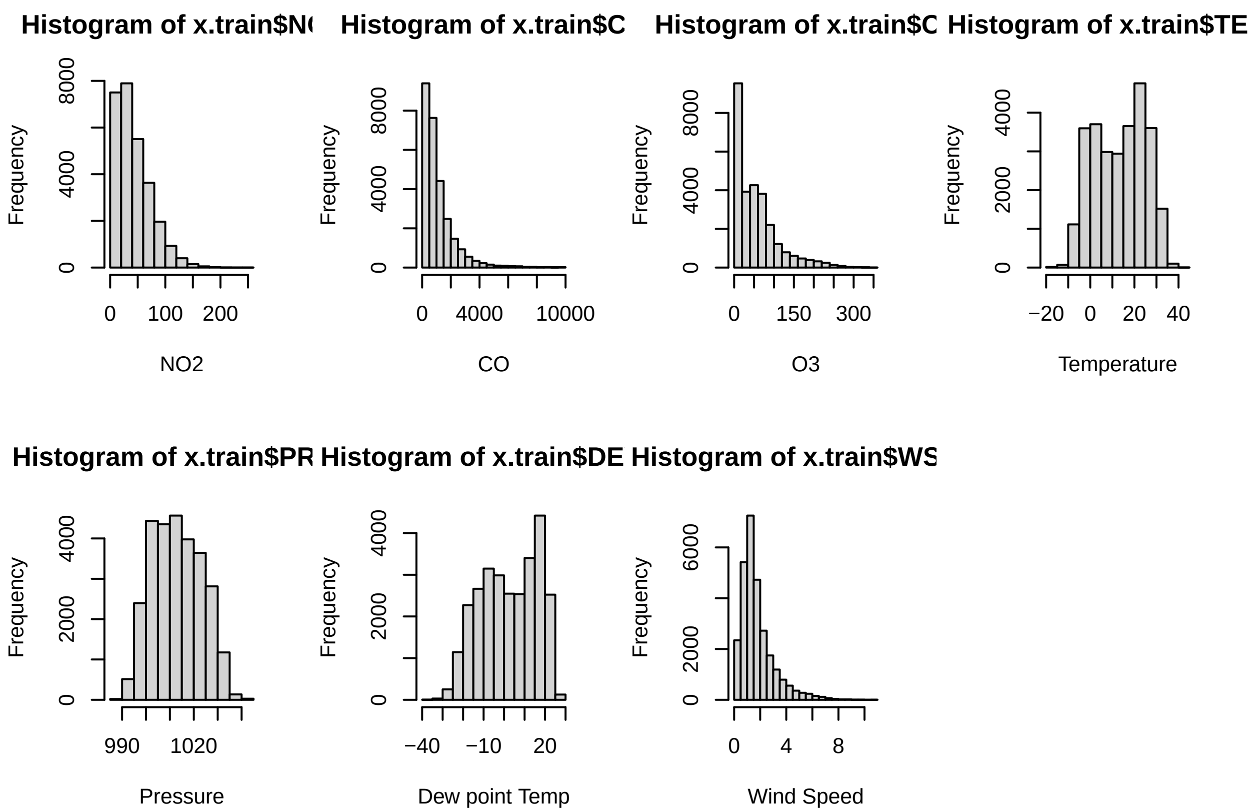
```
boxplot(x.train$WSPM, xlab = "Wind Speed")
# visualizing distributions
par(mfrow = c(2,3))
```



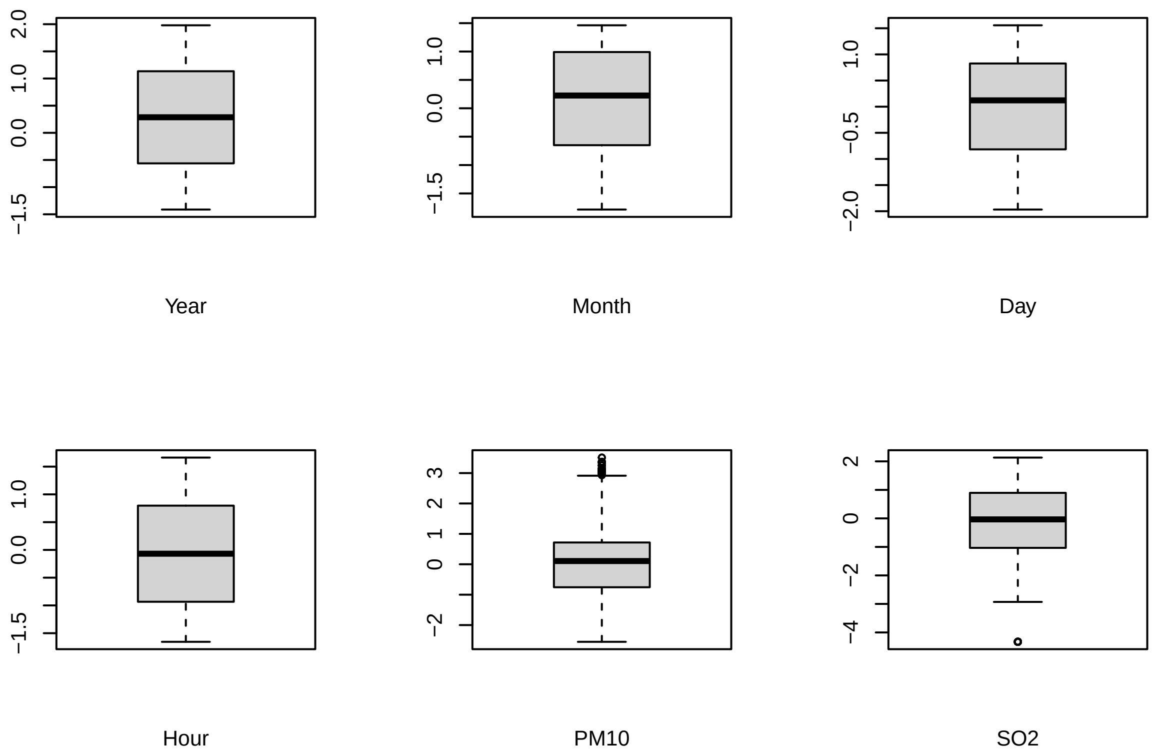

```
hist(x.train$year,xlab = "Year")
hist(x.train$month, xlab = "Month")
hist(x.train$day, xlab = "Day")
hist(x.train$hour, xlab = "Hour")
hist(x.train$PM10, xlab = "PM10")
hist(x.train$SO2, xlab = "SO2")
```



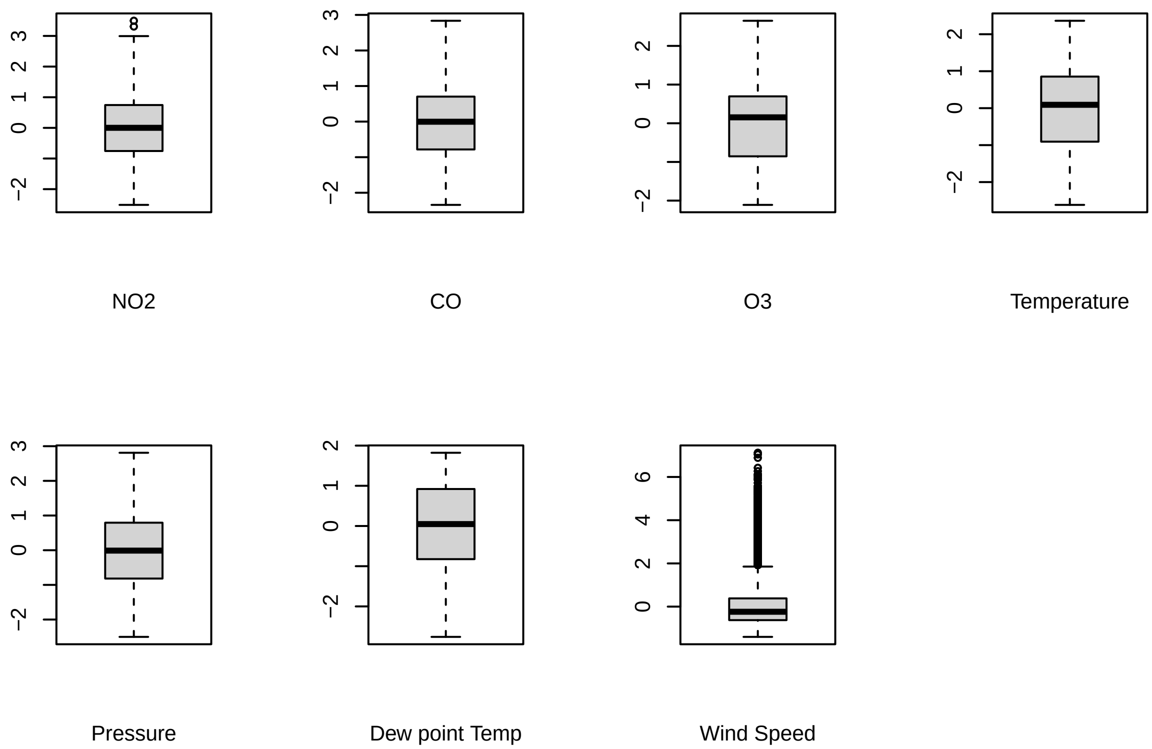

```r
par(mfrow = c(2,4))
hist(x.train$NO2, xlab = "NO2")
hist(x.train$CO, xlab = "CO")
hist(x.train$O3, xlab = "O3")
hist(x.train$TEMP, xlab = "Temperature")
hist(x.train$PRES, xlab = "Pressure")
hist(x.train$DEWP, xlab = "Dew point Temp")
hist(x.train$WSPM, xlab = "Wind Speed")
# box-cox, center, scaling
trans <- preProcess(x.train,
                       method = c("BoxCox", "center", "scale"))
x.trainp <- predict(trans, x.train)
trans1 <- preProcess(y.train1,
                       method = c("BoxCox", "center", "scale"))
y.trainp <- predict(trans1, y.train1)
trans2 <- preProcess(x.test,
                       method = c("BoxCox", "center", "scale"))
x.testp <- predict(trans2, x.test)
trans3 <- preProcess(y.test1,
                       method = c("BoxCox", "center", "scale"))
y.testp <- predict(trans3, y.test1)
# visualizing outliers after transformations
par(mfrow = c(2,3))
```



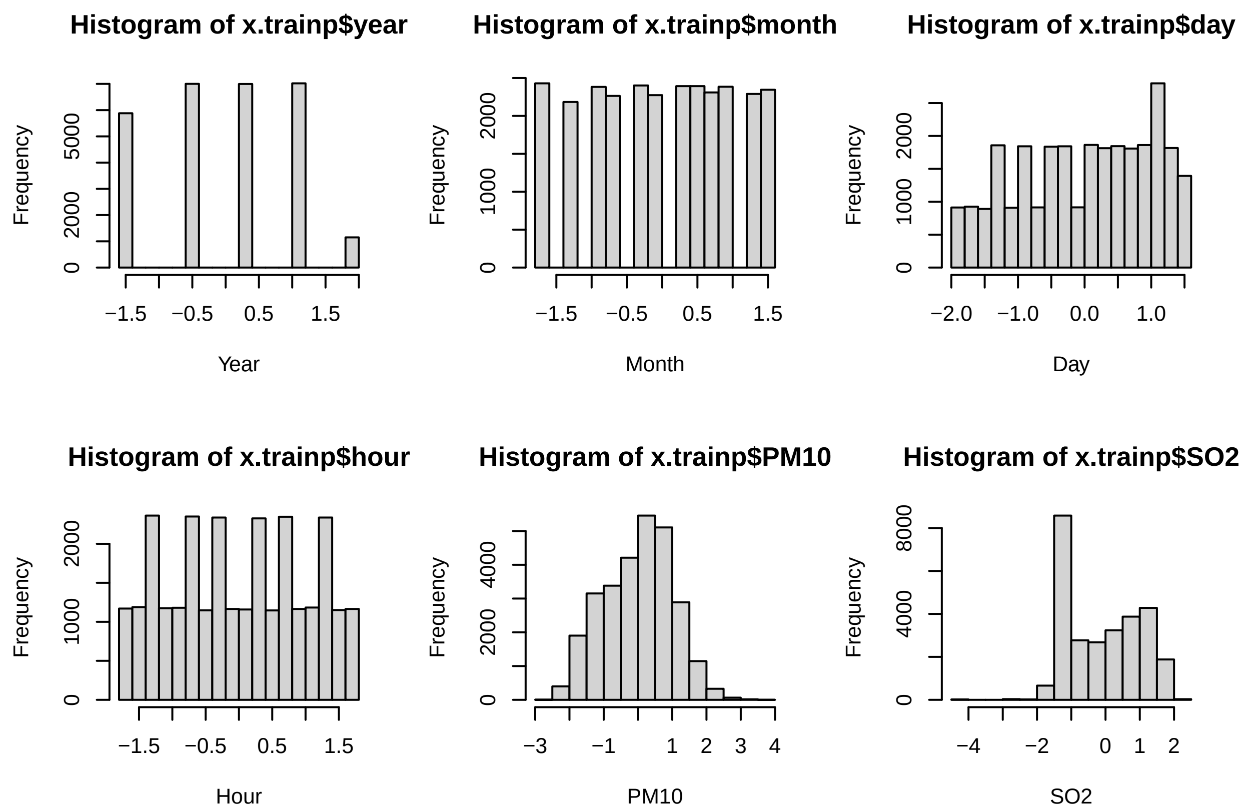

```
boxplot(x.trainp$year,xlab = "Year")
boxplot(x.trainp$month, xlab = "Month")
boxplot(x.trainp$day, xlab = "Day")
boxplot(x.trainp$hour, xlab = "Hour")
boxplot(x.trainp$PM10, xlab = "PM10")
boxplot(x.trainp$SO2, xlab = "SO2")
```



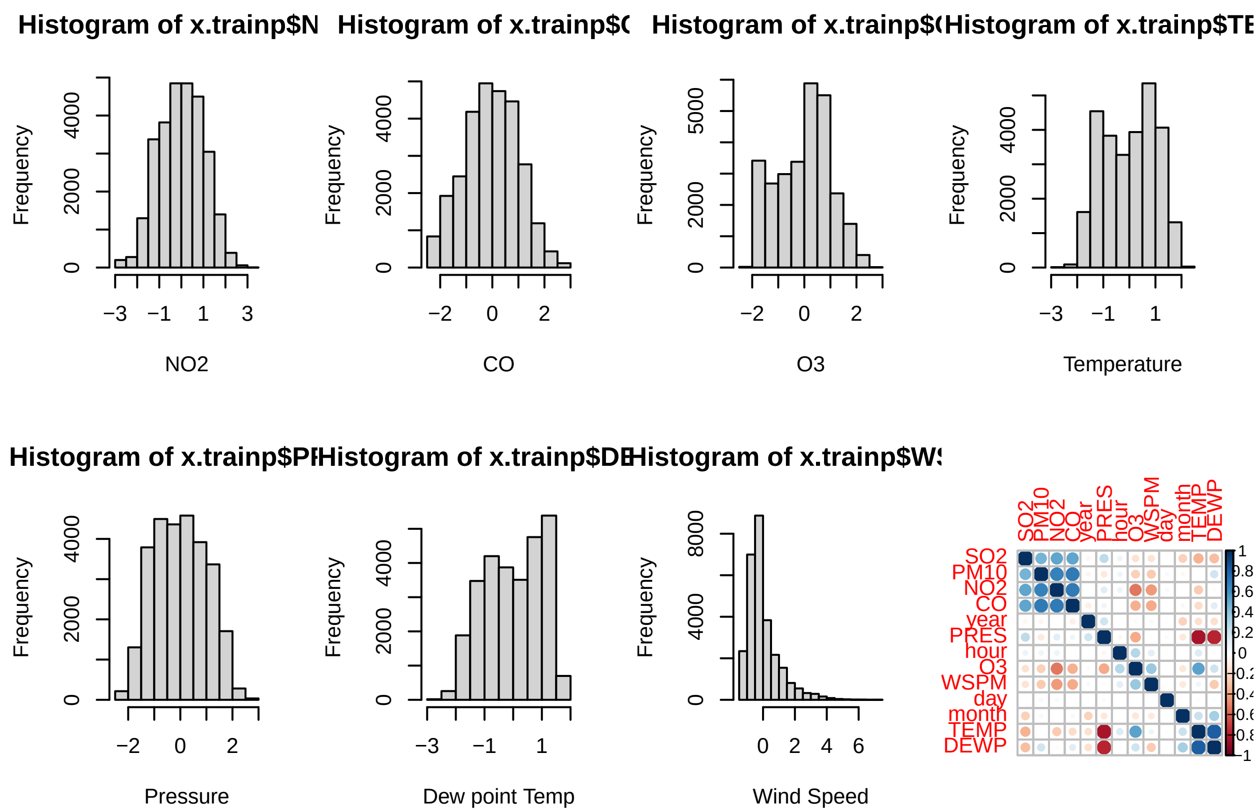

```r
par(mfrow = c(2,4))
boxplot(x.trainp$NO2, xlab = "NO2")
boxplot(x.trainp$CO, xlab = "CO")
boxplot(x.trainp$O3, xlab = "O3")
boxplot(x.trainp$TEMP, xlab = "Temperature")
boxplot(x.trainp$PRES, xlab = "Pressure")
boxplot(x.trainp$DEWP, xlab = "Dew point Temp")
boxplot(x.trainp$WSPM, xlab = "Wind Speed")
# visualizing distribution after transformations
par(mfrow = c(2,3))
```




```
hist(x.trainp$year,xlab = "Year")
hist(x.trainp$month, xlab = "Month")
hist(x.trainp$day, xlab = "Day")
hist(x.trainp$hour, xlab = "Hour")
hist(x.trainp$PM10, xlab = "PM10")
hist(x.trainp$SO2, xlab = "SO2")
```




```r
par(mfrow = c(2,4))
hist(x.trainp$NO2, xlab = "NO2")
hist(x.trainp$CO, xlab = "CO")
hist(x.trainp$O3, xlab = "O3")
hist(x.trainp$TEMP, xlab = "Temperature")
hist(x.trainp$PRES, xlab = "Pressure")
hist(x.trainp$DEWP, xlab = "Dew point Temp")
hist(x.trainp$WSPM, xlab = "Wind Speed")
# visualizing correlation
x.corr <- cor(x.trainp)
corrplot(x.corr, order = "hclust")
```

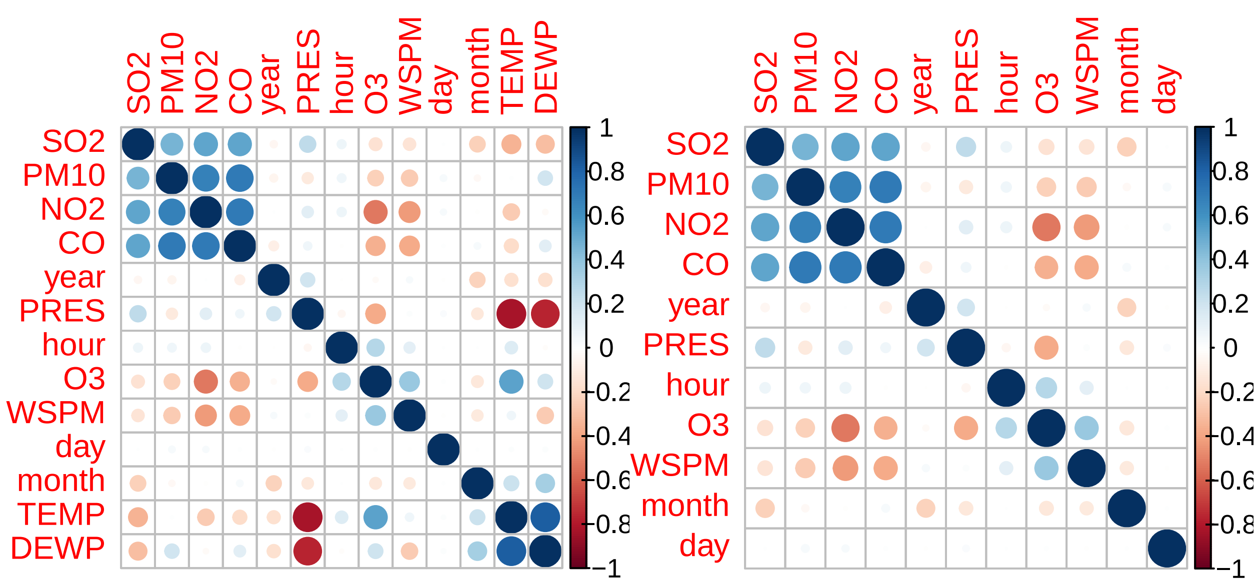



```r
hCorr <- findCorrelation(x.corr, cutoff = 0.75, exact = TRUE)
x.trainpc <- x.trainp[, -hCorr]
x.testpc <- x.testp[, -hCorr]
x.corrCheck <- cor(x.trainpc)
x.corrCheck
```

```
##                year        month          day          hour        PM10
## year    1.0000000000 -0.220995074 -0.004555605 -0.0003806929 -0.05659725
## month  -0.2209950737  1.000000000  0.004912318  0.0004323410 -0.03266403
## day    -0.0045556049  0.004912318  1.000000000  0.0015220176  0.03788677
## hour   -0.0003806929  0.000432341  0.001522018  1.0000000000  0.06790871
## PM10   -0.0565972501 -0.032664030  0.037886774  0.0679087075  1.00000000
## SO2    -0.0455255923 -0.233431501  0.001768788  0.0724217371  0.46332563
## NO2     0.0005176724 -0.007179976  0.031387206  0.0778285408  0.67247614
## CO     -0.0874638065  0.037719054  0.007406187 -0.0045958308  0.70132058
## O3     -0.0216861358 -0.129249680  0.007955603  0.2874547673 -0.23517020
## PRES    0.1909912799 -0.122466226  0.026467936 -0.0421577507 -0.11540840
## WSPM    0.0310687386 -0.117742917 -0.005800847  0.1139853998 -0.25470725
##                SO2          NO2           CO           O3        PRES
## year   -0.045525592  0.0005176724 -0.087463807 -0.021686136  0.19099128
## month  -0.233431501 -0.0071799758  0.037719054 -0.129249680 -0.12246623
## day     0.001768788  0.0313872056  0.007406187  0.007955603  0.02646794
## hour    0.072421737  0.0778285408 -0.004595831  0.287454767 -0.04215775
## PM10    0.463325633  0.6724761422  0.701320580 -0.235170198 -0.11540840
## SO2     1.000000000  0.5292395310  0.526082806 -0.154902642  0.25283278
## NO2     0.529239531  1.0000000000  0.701677378 -0.524489511  0.12005221
```

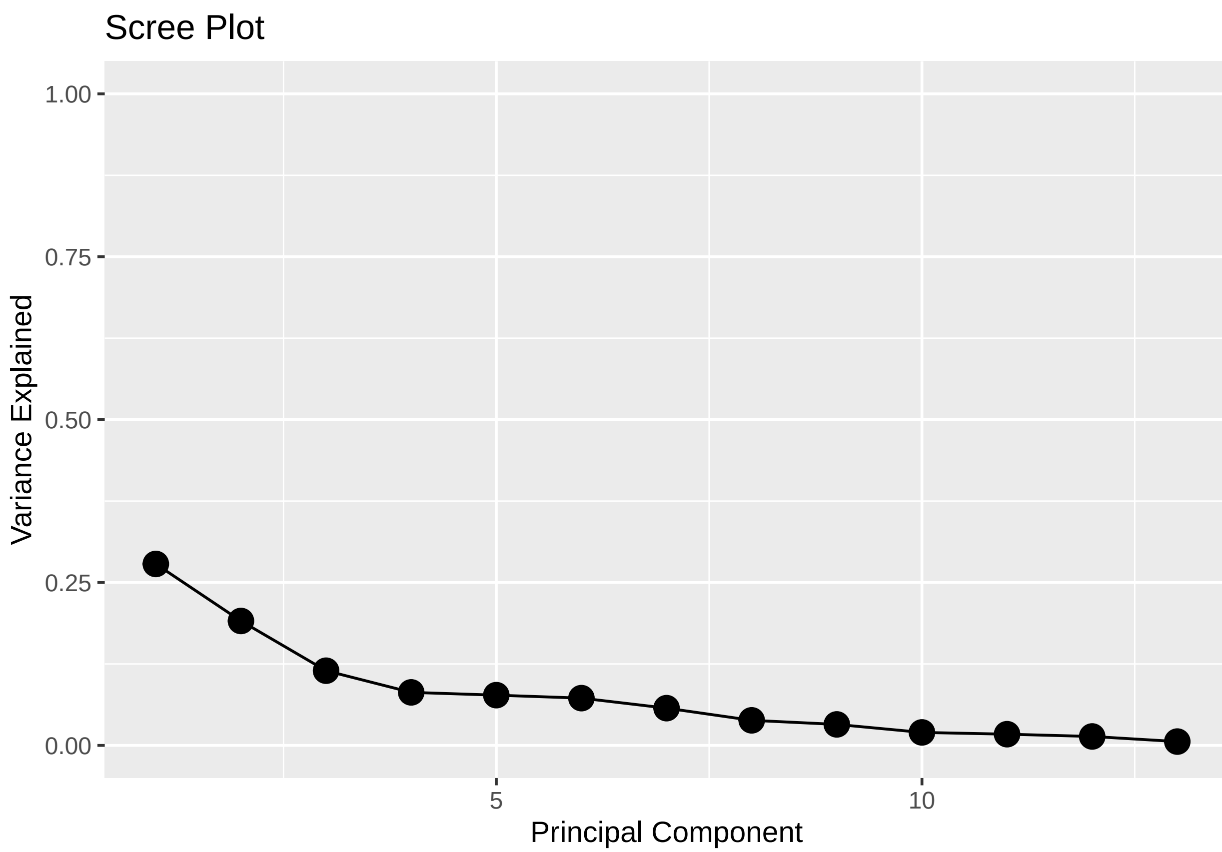
```
## CO     0.526082806  0.7016773783  1.000000000 -0.354901956  0.06123205
## O3    -0.154902642 -0.5244895112 -0.354901956  1.000000000 -0.37528681
## PRES   0.252832784  0.1200522100  0.061232054 -0.375286809  1.00000000
## WSPM  -0.143645492 -0.4282418879 -0.378932137  0.378000593  0.01706782
##              WSPM
## year   0.031068739
## month -0.117742917
## day   -0.005800847
## hour   0.113985400
## PM10  -0.254707254
## SO2   -0.143645492
## NO2   -0.428241888
## CO    -0.378932137
## O3     0.378000593
## PRES   0.017067816
## WSPM   1.000000000
```

```
par(mfrow = c(1,2))
corrplot(x.corr, order = "hclust")
corrplot(x.corrCheck, order = "hclust")
```



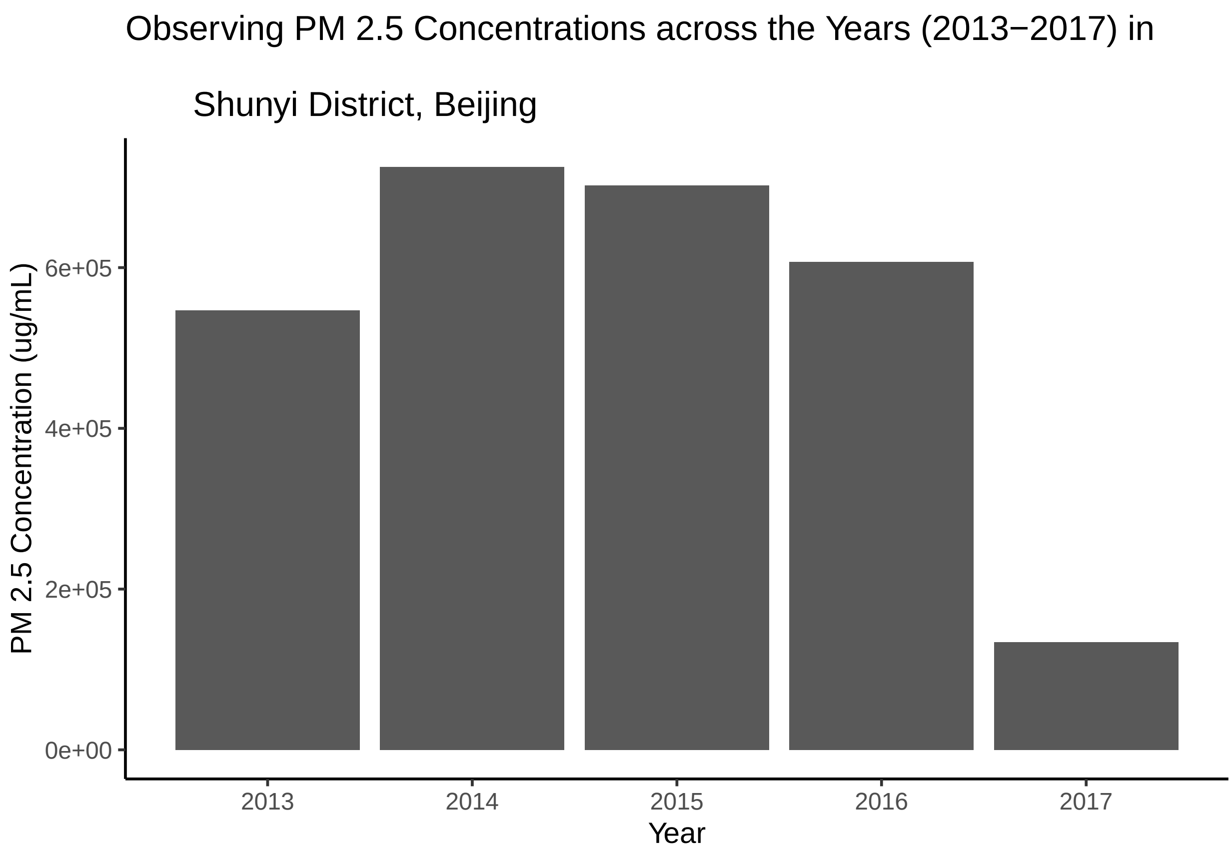

```
# PCA
pca.x <- prcomp(x.train, center = TRUE, scale. = TRUE)
variance = pca.x$sdev^2 / sum(pca.x$sdev^2)
# variance
```

```
qplot(c(1:13), variance) +
  geom_line() +
  geom_point(size=4)+
  xlab("Principal Component") +
  ylab("Variance Explained") +
  ggtitle("Scree Plot") +
  ylim(0, 1)
```

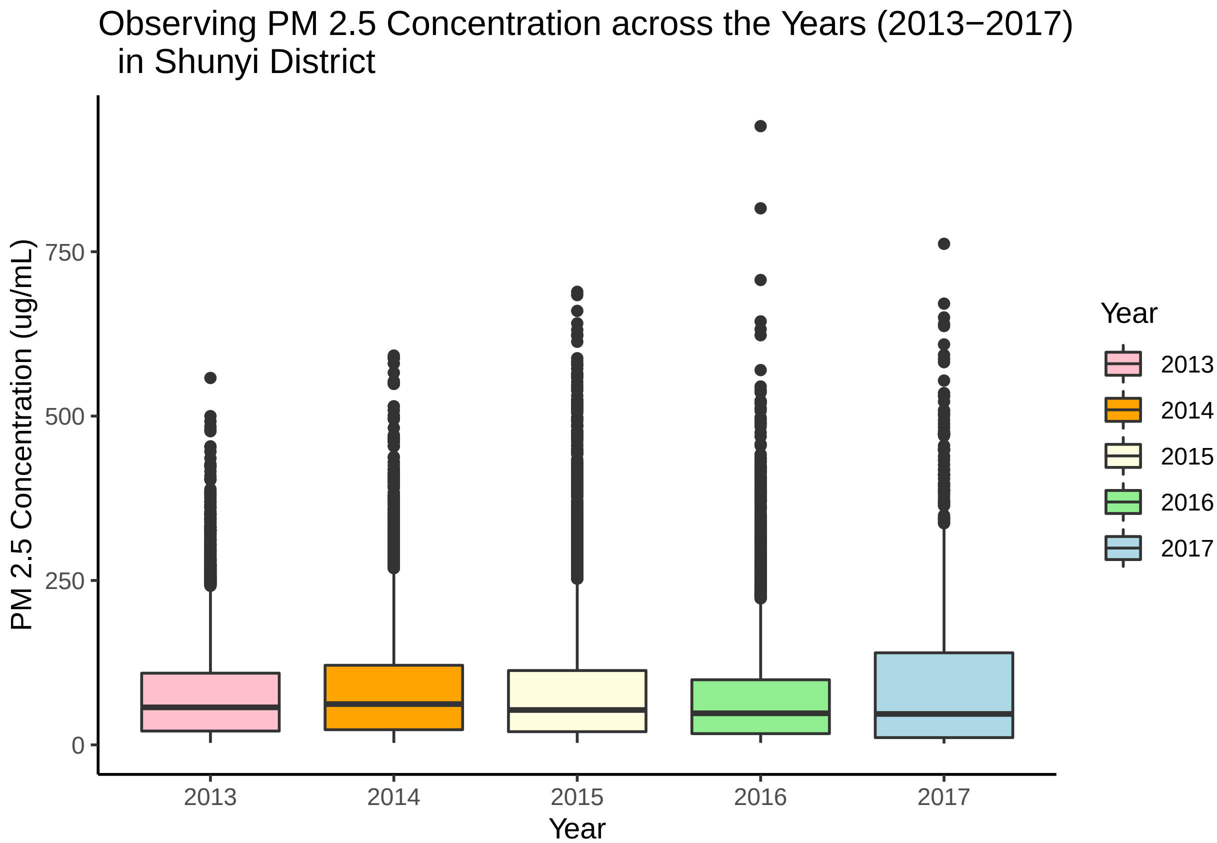


Exploratory Data Analysis

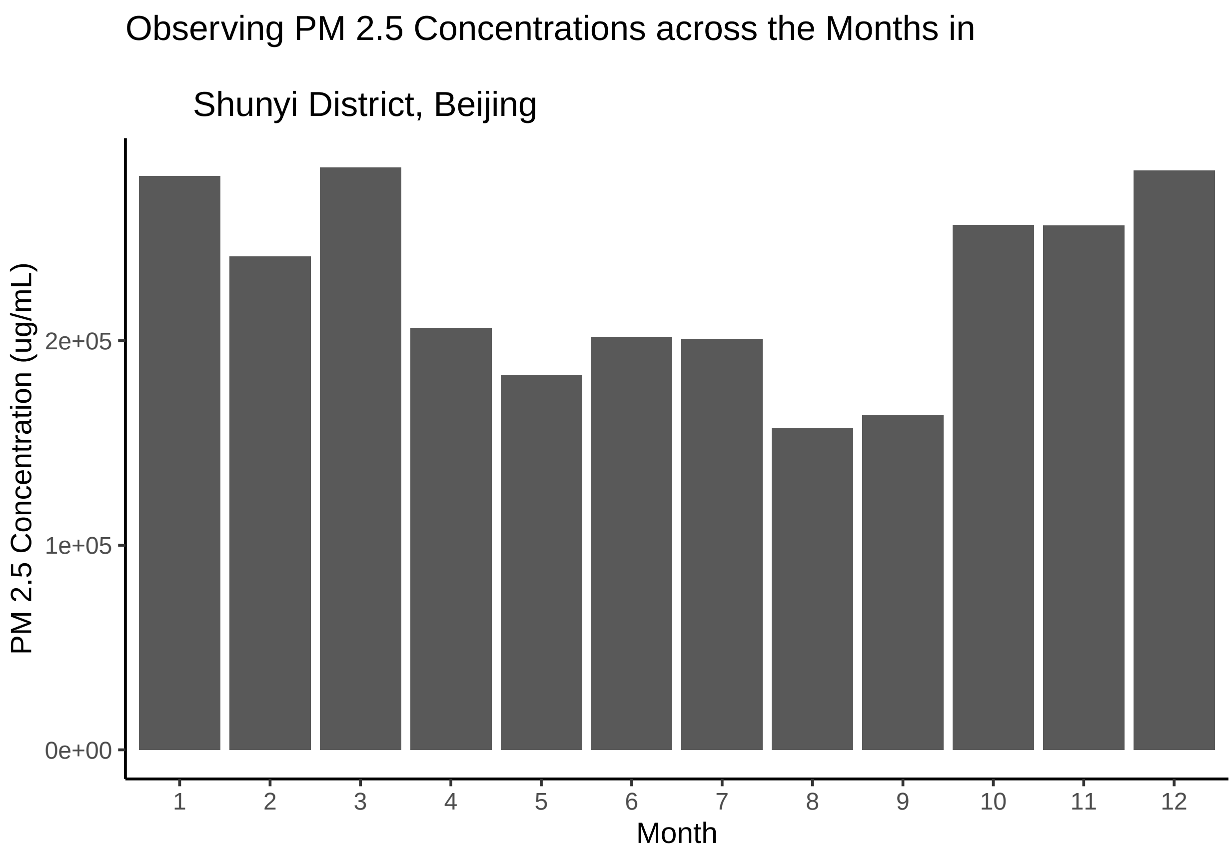
```
# these visualizations are observing data before it was pre-processed
# Observing PM 2.5 Concentrations across the Years (2013-2017)
ggplot(data = air, aes(x=year,y=PM2.5)) +
geom_bar(stat = "identity") +
theme_minimal() +
  labs(x = "Year", y = "PM 2.5 Concentration (ug/mL)",
       title =
         "Observing PM 2.5 Concentrations across the Years (2013-2017) in \n
       Shunyi District, Beijing", hjust = 0.5) +
  theme_classic()
```




```
# Boxplot PM 2.5 Concentrations across the Years (2013-2017)
ggplot(data = air, aes(x=year,y=PM2.5)) +
geom_boxplot(aes(fill=factor(year), fill = year)) +
theme_minimal() +
  scale_fill_manual(name = "Year", labels = c("2013",
                                               "2014",
                                               "2015",
                                               "2016",
                                               "2017"),
          values = c("pink","orange", "light yellow","light green",
                     "light blue")) +
  labs(x = "Year", y = "PM 2.5 Concentration (ug/mL)", title =
  "Observing PM 2.5 Concentration across the Years (2013-2017)
  in Shunyi District", adj = 0.5) +
  theme_classic()
```

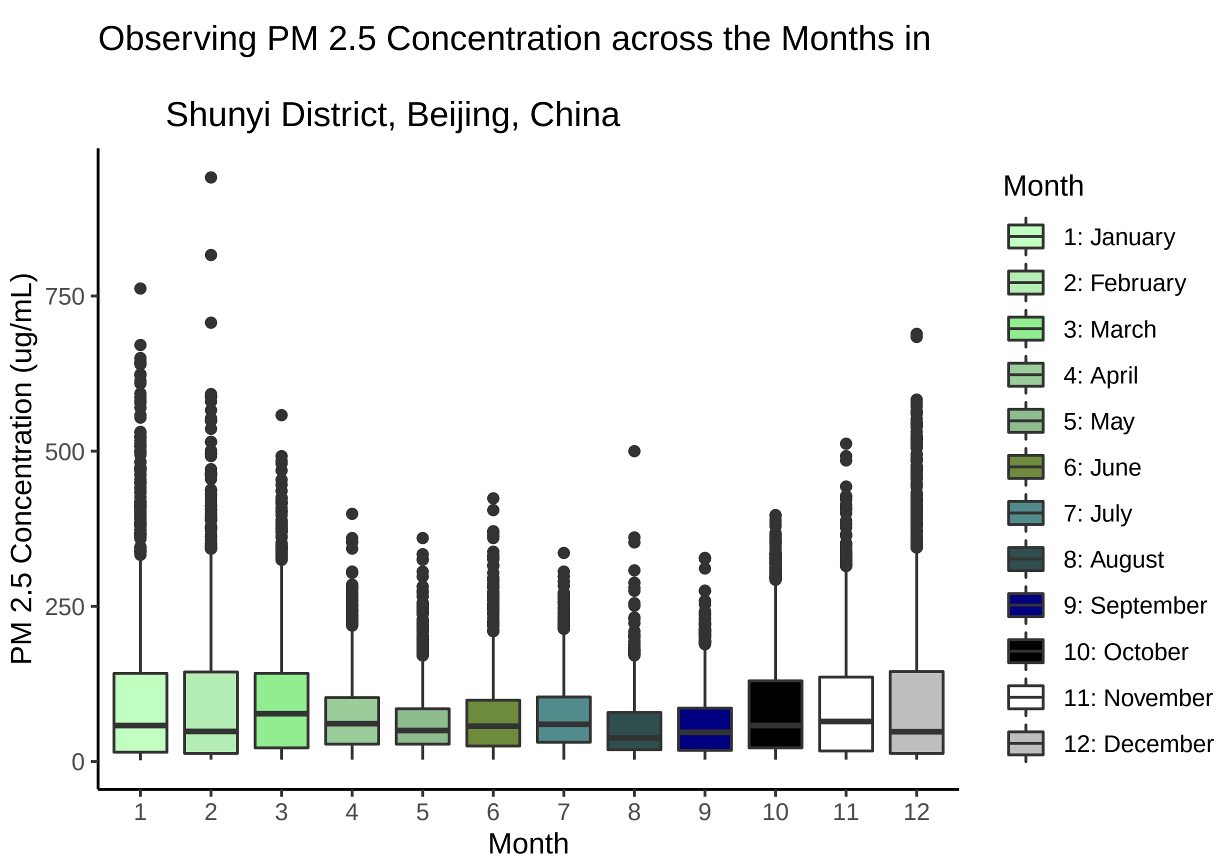



```
# Observing PM 2.5 Concentrations across the Months
ggplot(data = air, aes(x=month,y=PM2.5)) +
geom_bar(stat = "identity") +
theme_minimal() +
  labs(x = "Month", y = "PM 2.5 Concentration (ug/mL)",
       title = "Observing PM 2.5 Concentrations across the Months in \n
       Shunyi District, Beijing",
       adj = 0.5) +
   scale_x_discrete(name = "Month",
                    limits = c("1", "2", "3", "4", "5", "6","7", "8", "9", "10",
                               "11", "12")) +
  theme_classic()
```




```
# Boxplot of PM2.5 Concentrations per month
new_data = cbind(x.train, y.train)
ggplot(data = new_data, aes(x=month,y=y.train)) +
geom_boxplot(aes(fill=factor(month), fill = month)) +
theme_minimal() +
scale_color_manual(name = "Month", labels = c("1: January",
                                                      "2: February",
                                                      "3: March",
                                                      "4: April",
                                                "5: May",
                                                "6: June",
                                                "7: July",
                                                "8: August",
                                                "9: September",
                                                "10: October",
                                               "11: November",
                                               "12: December"),
                     values = c("darkseagreen1","darkseagreen2", "light green",
                                "darkseagreen3",
                                "darkseagreen",
                                "darkolivegreen4",
                                "darkslategray4", "darkslategray", "navy",
                                "black", "white", "grey")) +
  scale_fill_manual(name = "Month", labels = c("1: January",
                                                      "2: February",
                                                      "3: March",
```

```
                                                        "4: April",
                                                  "5: May",
                                                  "6: June",
                                                  "7: July",
                                                  "8: August",
                                                  "9: September",
                                                  "10: October",
                                                 "11: November",
                                                 "12: December"),
                    values = c("darkseagreen1","darkseagreen2", "light green",
                                "darkseagreen3",
                                "darkseagreen",
                                "darkolivegreen4",
                                "darkslategray4", "darkslategray", "navy",
                                "black", "white", "grey")) +
scale_x_discrete(name = "Month",
                 limits = c("1", "2", "3", "4", "5", "6","7", "8", "9", "10",
                            "11", "12")) +
labs(x = "Month", y = "PM 2.5 Concentration (ug/mL)",
     title =
       "Observing PM 2.5 Concentration across the Months in \n
     Shunyi District, Beijing, China",
     adj = 0.5) +
theme_classic()
```



Preliminary Models, Hyperparameter Tuning, and Model Evaluations

```
# OLS
# Using as a base model
set.seed(100)
indx <- createFolds(y.train, returnTrain = TRUE)
ctrl <- trainControl(method = "cv", index = indx)
pcrTune2 <- train(x = x.trainpc, y = y.train,
                  method = "lm",trControl = ctrl)
pcrTune2
```

```
## Linear Regression
##
## 28053 samples
##    11 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 25247, 25248, 25247, 25248, 25247, 25248, ...
## Resampling results:
##
##   RMSE     Rsquared   MAE
##   45.3015  0.6951456  31.54367
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
summary(pcrTune2)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -191.63  -27.21   -7.13   19.06  701.48
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  80.0284     0.2706 295.792  < 2e-16 ***
## year          2.6996     0.2863   9.428  < 2e-16 ***
## month        -1.8366     0.2968  -6.187 6.20e-10 ***
## day          -2.1772     0.2716  -8.017 1.12e-15 ***
## hour         -2.2280     0.2998  -7.431 1.11e-13 ***
## PM10         55.0953     0.4327 127.332  < 2e-16 ***
## SO2          -4.9880     0.3710 -13.444  < 2e-16 ***
## NO2          -4.0910     0.5030  -8.133 4.37e-16 ***
## CO           24.7482     0.4461  55.477  < 2e-16 ***
## O3            3.9193     0.4011   9.770  < 2e-16 ***
## PRES          8.1566     0.3370  24.206  < 2e-16 ***
## WSPM         -2.4262     0.3163  -7.671 1.76e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 45.32 on 28041 degrees of freedom
## Multiple R-squared:  0.695,  Adjusted R-squared:  0.6948
## F-statistic:  5808 on 11 and 28041 DF,  p-value: < 2.2e-16
```

```
# testResults
rfImp_OLS <- varImp(pcrTune2, scale = T)
rfImp_OLS
```

```
## lm variable importance
##
##       Overall
## PM10  100.000
## CO     40.686
## PRES   14.874
## SO2     5.990
## O3      2.957
## year    2.675
## NO2     1.606
## day     1.510
## WSPM    1.225
## hour    1.026
## month   0.000
```

```
fp_predict <- predict(pcrTune2, x.testpc)
postResample(fp_predict, y.test)
```

```
##      RMSE  Rsquared       MAE
## 43.249089  0.699256 30.864738
```

```
# PLS
set.seed(100)
indx <- createFolds(y.train, returnTrain = TRUE)
ctrl <- trainControl(method = "cv", index = indx)
pcrTune3 <- train(x = x.train, y = y.train,
                 method = "pls",
                 preProcess=c("center","scale"),
                 tuneGrid = expand.grid(ncomp = 1:14),
                 trControl = ctrl)
pcrTune3
```

```
## Partial Least Squares
##
## 28053 samples
##    13 predictor
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 25247, 25248, 25247, 25248, 25247, 25248, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE      Rsquared   MAE
```

```
##    1      43.96552  0.7124556  30.58076
##    2      38.29961  0.7817798  26.32104
##    3      34.12245  0.8267487  21.20946
##    4      32.99449  0.8380107  20.06535
##    5      32.50372  0.8427737  19.53363
##    6      32.37996  0.8439909  19.58367
##    7      32.32736  0.8445200  19.37568
##    8      32.28204  0.8449567  19.58626
##    9      32.27998  0.8449769  19.58984
##   10      32.27954  0.8449811  19.58751
##   11      32.27959  0.8449806  19.58824
##   12      32.27955  0.8449811  19.58853
##   13      32.27957  0.8449809  19.58874
##   14      32.27957  0.8449809  19.58874
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 10.
```

```
summary(pcrTune3)
```

```
## Data:    X dimension: 28053 13
##  Y dimension: 28053 1
## Fit method: oscorespls
## Number of components considered: 10
## TRAINING: % variance explained
##           1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X           24.13    44.07    52.89    61.34    66.79    71.91    76.93
## .outcome    71.27    78.19    82.70    83.83    84.31    84.43    84.48
##           8 comps  9 comps  10 comps
## X           78.15    84.91     87.18
## .outcome    84.52    84.52     84.52
```

```
fp_predict1 <- predict(pcrTune3, x.test)

postResample(fp_predict, y.test)
```

```
##      RMSE  Rsquared       MAE
## 43.249089  0.699256 30.864738
```

```
rfImp_PLS <- varImp(pcrTune3, scale = T)
rfImp_PLS
```

```
## pls variable importance
##
##       Overall
## PM10  100.000
## CO     81.674
## NO2    68.047
## SO2    50.017
## WSPM   30.027
## O3     16.971
## TEMP   15.615
```

```
## DEWP   13.557
## PRES    3.205
## month   3.140
## hour    1.995
## year    1.151
## day     0.000
```

```
# Random Forest
rfmodel <- randomForest(x = x.train, y = y.train,importance=TRUE,ntrees=500)

getRMSE <- function(x,y) {
  sqrt(sum((x-y)^2)/length(x))
}

testResults <- data.frame(obs = y.test,
                          rfmodel = predict(rfmodel, x.test))

getRMSE(testResults$obs, testResults$rfmodel)
```

```
## [1] 21.33502
```

```
fp_predict2 <- predict(rfmodel , x.testp)
# fp_predict2 (commented out because there were too many predictions)
summary(fp_predict2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   24.56   29.02   35.00   33.04   35.52   39.55
```

```
postResample(fp_predict2, y.test)
```

```
##         RMSE     Rsquared          MAE
## 9.121772e+01 6.878555e-05 5.896752e+01
```

```
rfImp_RF <- varImp(rfmodel, scale = T)
rfImp_RF
```

```
##         Overall
## year   39.32902
## month  26.88878
## day    68.77043
## hour   49.77910
## PM10  134.12958
## SO2    41.97516
## NO2    34.03051
## CO     61.72128
## O3     37.31964
## TEMP   41.96578
## PRES   39.57512
## DEWP   41.20180
## WSPM   34.20973
```

```r
# Elastic Net
enetGrid <- expand.grid(lambda = c(0, 0.01, .1),
                        fraction = seq(.05, 1, length = 20))
set.seed(100)
enetTune <- train(x = x.trainp, y = y.train,
                  method = "enet",
                  tuneGrid = enetGrid,
                  trControl = ctrl,
                  preProc = c("center", "scale"))
enetTune
```

```
## Elasticnet
##
## 28053 samples
##    13 predictor
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 25247, 25248, 25247, 25248, 25247, 25248, ...
## Resampling results across tuning parameters:
##
##   lambda  fraction  RMSE      Rsquared   MAE
##   0.00    0.05      76.69624  0.6442943  55.56063
##   0.00    0.10      71.62933  0.6442943  51.12443
##   0.00    0.15      66.85991  0.6442943  46.84017
##   0.00    0.20      62.45627  0.6442943  42.72756
##   0.00    0.25      58.45161  0.6627185  38.92585
##   0.00    0.30      54.86191  0.6728596  35.49089
##   0.00    0.35      51.77449  0.6780405  33.04523
##   0.00    0.40      49.28420  0.6806879  31.67756
##   0.00    0.45      47.48534  0.6819712  31.22058
##   0.00    0.50      46.41344  0.6853695  31.23297
##   0.00    0.55      45.84108  0.6890681  31.41194
##   0.00    0.60      45.56278  0.6922093  31.40713
##   0.00    0.65      45.35071  0.6949265  31.32230
##   0.00    0.70      45.17407  0.6972292  31.22923
##   0.00    0.75      45.02111  0.6992095  31.15339
##   0.00    0.80      44.89670  0.7007998  31.10540
##   0.00    0.85      44.80235  0.7019902  31.08550
##   0.00    0.90      44.73611  0.7028189  31.09045
##   0.00    0.95      44.69741  0.7033019  31.11632
##   0.00    1.00      44.68455  0.7034688  31.16275
##   0.01    0.05      76.84702  0.6442943  55.69120
##   0.01    0.10      71.91566  0.6442943  51.37780
##   0.01    0.15      67.26138  0.6442943  47.20666
##   0.01    0.20      62.94578  0.6442943  43.19495
##   0.01    0.25      59.00899  0.6622618  39.47508
##   0.01    0.30      55.45352  0.6725819  36.04051
##   0.01    0.35      52.35663  0.6778734  33.45819
##   0.01    0.40      49.80427  0.6805915  31.90402
##   0.01    0.45      47.88388  0.6819215  31.25982
##   0.01    0.50      46.66686  0.6835158  31.26665
##   0.01    0.55      45.97784  0.6878727  31.36382
##   0.01    0.60      45.65257  0.6908838  31.49907
```

```
##   0.01    0.65      45.42778  0.6937940  31.40977
##   0.01    0.70      45.24464  0.6961925  31.32397
##   0.01    0.75      45.08759  0.6982443  31.24948
##   0.01    0.80      44.95463  0.6999680  31.19262
##   0.01    0.85      44.85013  0.7013123  31.16243
##   0.01    0.90      44.77282  0.7023052  31.15562
##   0.01    0.95      44.72064  0.7029802  31.16703
##   0.01    1.00      44.69370  0.7033439  31.20023
##   0.10    0.05      77.52474  0.6442943  56.27742
##   0.10    0.10      73.20986  0.6442943  52.51900
##   0.10    0.15      69.09193  0.6442943  48.86199
##   0.10    0.20      65.21091  0.6449154  45.32252
##   0.10    0.25      61.63186  0.6640972  42.03086
##   0.10    0.30      58.30457  0.6736883  38.87298
##   0.10    0.35      55.27383  0.6785310  35.95402
##   0.10    0.40      52.59111  0.6809628  33.67057
##   0.10    0.45      50.31232  0.6821044  32.17746
##   0.10    0.50      48.49461  0.6825292  31.40031
##   0.10    0.55      47.19136  0.6825468  31.22433
##   0.10    0.60      46.41736  0.6832965  31.47757
##   0.10    0.65      45.96037  0.6867363  31.67839
##   0.10    0.70      45.76135  0.6889784  31.97151
##   0.10    0.75      45.61883  0.6909958  31.98278
##   0.10    0.80      45.48748  0.6928652  31.94907
##   0.10    0.85      45.36863  0.6945581  31.91679
##   0.10    0.90      45.28298  0.6958064  31.90855
##   0.10    0.95      45.21964  0.6967780  31.92103
##   0.10    1.00      45.17449  0.6975326  31.94838
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were fraction = 1 and lambda = 0.
```

```
enet_predict <- predict(enetTune, x.testp)
# enet_predict (commented out because there were too many predictions)
summary(enet_predict)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -115.65   27.65   82.33   80.03  129.27  300.80
```

```
postResample(enet_predict, y.test)
```

```
##       RMSE   Rsquared        MAE
## 42.9263146  0.7038906 30.6430285
```

```
rfImp_EN <- varImp(enetTune, scale = T)
rfImp_EN
```

```
## loess r-squared variable importance
##
##         Overall
## PM10  1.000e+02
## CO    7.661e+01
```

```
## NO2   5.191e+01
## SO2   2.566e+01
## WSPM  1.086e+01
## O3    8.651e+00
## TEMP  2.483e+00
## DEWP  1.871e+00
## month 2.562e-01
## hour  3.357e-02
## PRES  1.489e-02
## year  5.978e-03
## day   0.000e+00
```

```
resamp <- resamples(list(OLS = pcrTune2, PLS = pcrTune3,Enet=enetTune))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: OLS, PLS, Enet
## Number of resamples: 10
##
## MAE
##          Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## OLS  30.60395 31.03170 31.55598 31.54367 31.88867 32.88581    0
## PLS  18.57821 19.43927 19.57927 19.58751 19.94144 20.36607    0
## Enet 30.25447 30.66429 31.24254 31.16275 31.52767 32.35359    0
##
## RMSE
##          Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## OLS  41.91173 44.64342 45.53493 45.30150 46.22377 48.12999    0
## PLS  28.73939 31.79096 32.83348 32.27954 33.09557 33.99087    0
## Enet 41.58058 44.19342 44.83082 44.68455 45.55638 47.42750    0
##
## Rsquared
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## OLS  0.6746818 0.6904134 0.6944912 0.6951456 0.7026644 0.7166186    0
## PLS  0.8179351 0.8355709 0.8462767 0.8449811 0.8557245 0.8662906    0
## Enet 0.6840361 0.7009725 0.7027909 0.7034688 0.7105767 0.7214876    0
```