ADS 503 Final Project

Group1: Sean Torres, George Garcia, and Anusia Edward

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

```r
sum(is.na(air)) # checking for NAs
```

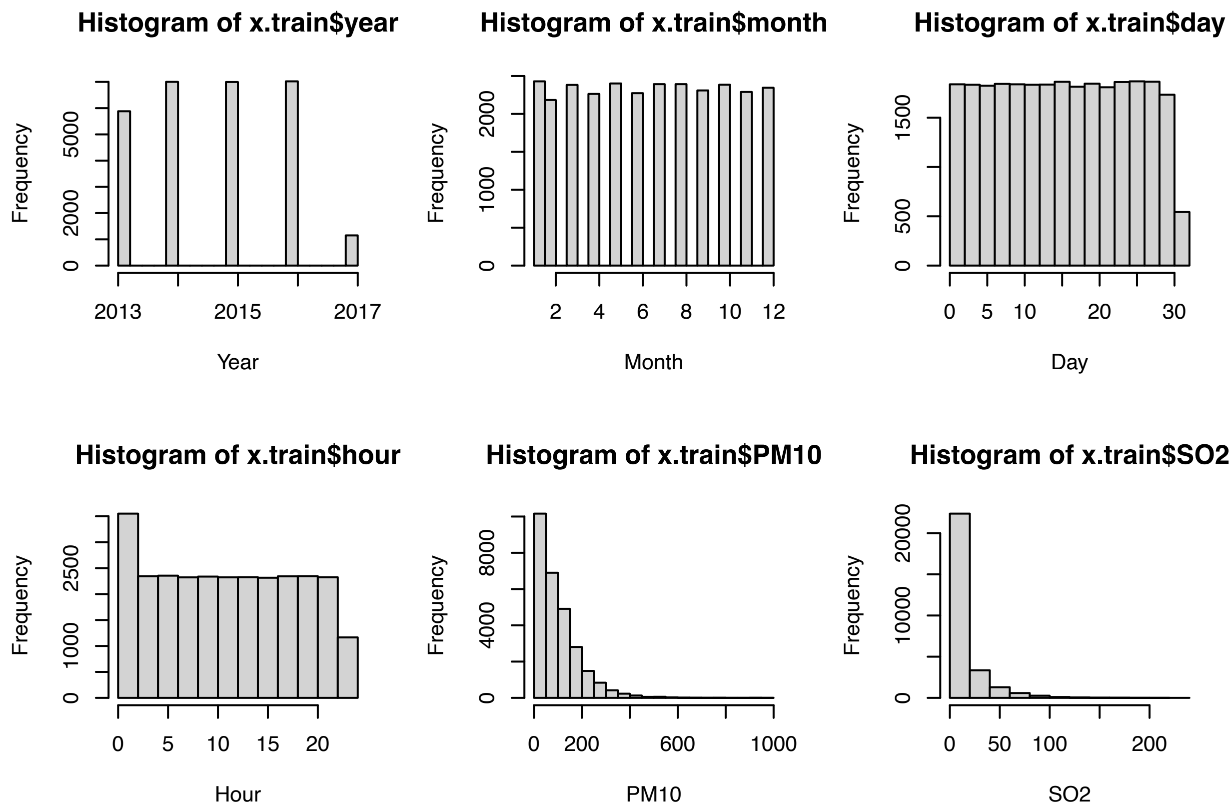
```
## [1] 8040
```

```r
air.knn <- kNN(air) # using 5KNN to impute missing values
# removing the imputation truth variables that were added
air.knn <- subset(air.knn, select = year:station)
sum(is.na(air.knn)) # double checking NAs
```

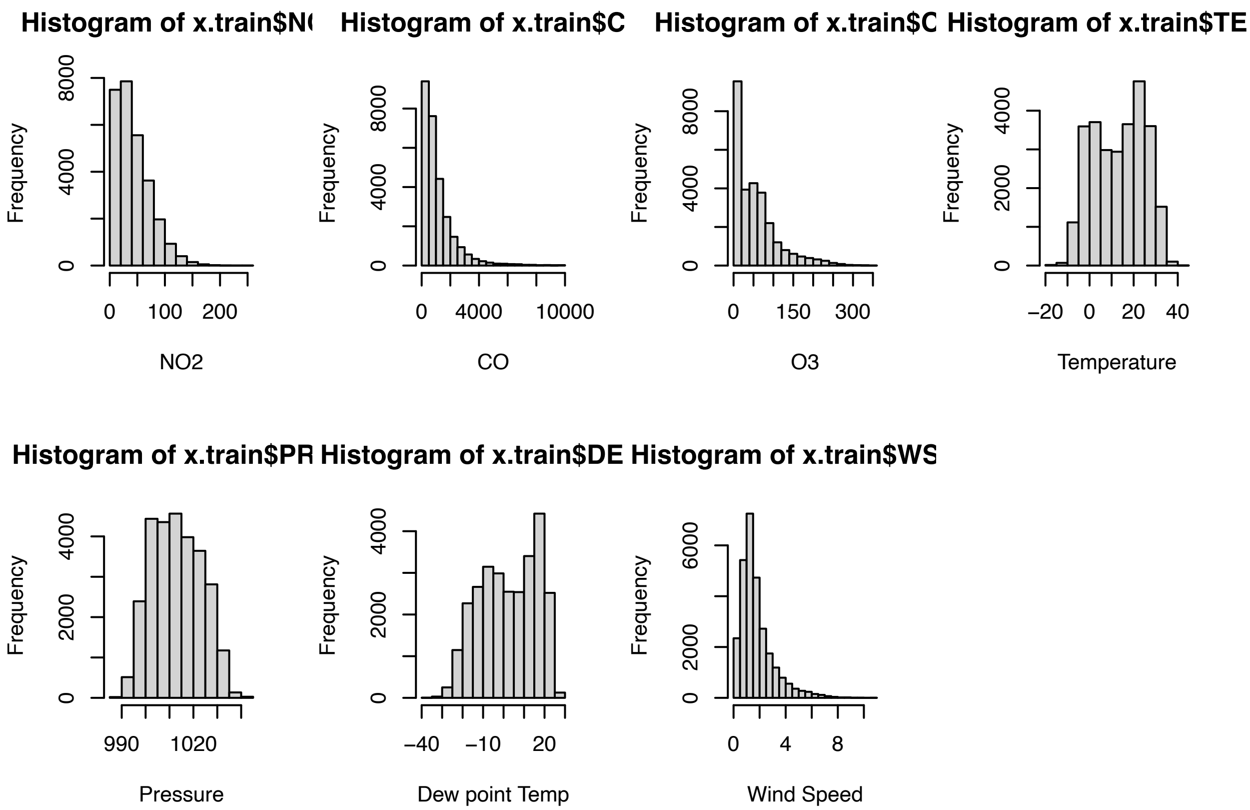
```
## [1] 0
```

```r
# splitting predictors and outcome variables
x <- subset(air.knn, select = -c(PM2.5))
y <- subset(air.knn, select = PM2.5)
library(caret)
nZV.x <- nearZeroVar(x)# checking near zero variance
x <- x[, -nZV.x]# removal of nZV
# splitting the data before preprocessing to avoid data leakage
library(caret)
set.seed(1)
trainset <- createDataPartition(air.knn$PM2.5, p = 0.8, list = FALSE)
x.train <- x[trainset, ]
y.train <- y[trainset, ]
y.train1 <- as.data.frame(y.train)
x.test <- x[-trainset, ]
y.test <- y[-trainset, ]
y.test1 <- as.data.frame(y.test)
```

```r
# checking 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")
```

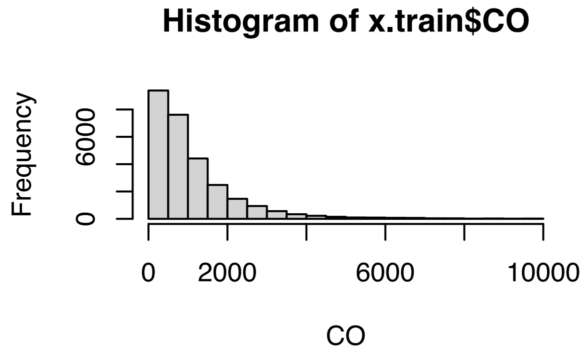


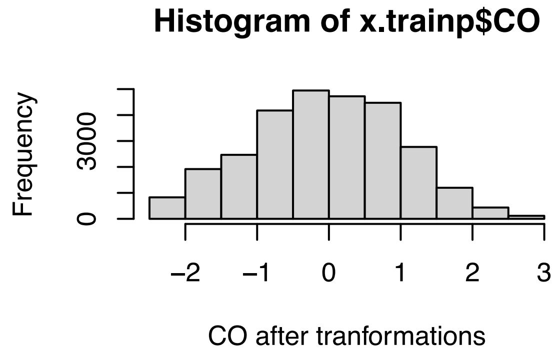

```r
par(mfrow = c(2,4))
hist(x.train$NO2, xlab = "NO2")
hist(x.train$CO, xlab = "CO")
hist(x.train$O3, xlab = "O3")
hist(x.train$TEMP, xlab = "Temperature")
hist(x.train$PRES, xlab = "Pressure")
hist(x.train$DEWP, xlab = "Dew point Temp")
hist(x.train$WSPM, xlab = "Wind Speed")
```

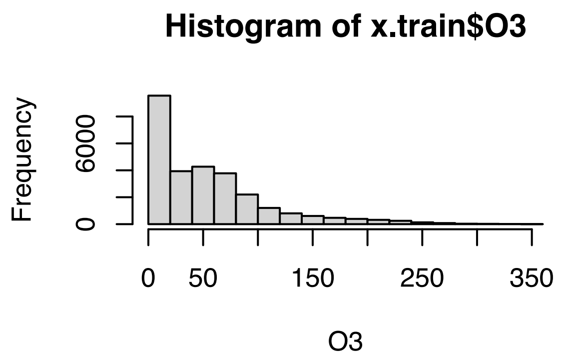


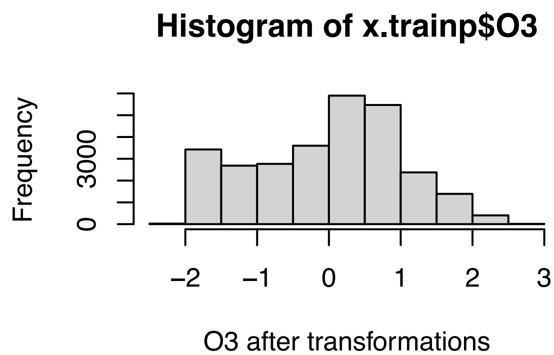

```r
# box-cox, center, scaling
trans <- preProcess(x.train,
                    method = c("BoxCox", "center", "scale"))
x.trainp <- predict(trans, x.train)
trans1 <- preProcess(y.train1,
                    method = c("BoxCox", "center", "scale"))
y.trainp <- predict(trans1, y.train1)
trans2 <- preProcess(x.test,
                    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)

# visualization of the transformations
par(mfrow = c(2,2))
hist(x.train$CO, xlab = "CO")
hist(x.trainp$CO, xlab = "CO after tranformations")
hist(x.train$O3, xlab = "O3")
hist(x.trainp$O3, xlab = "O3 after transformations")
```

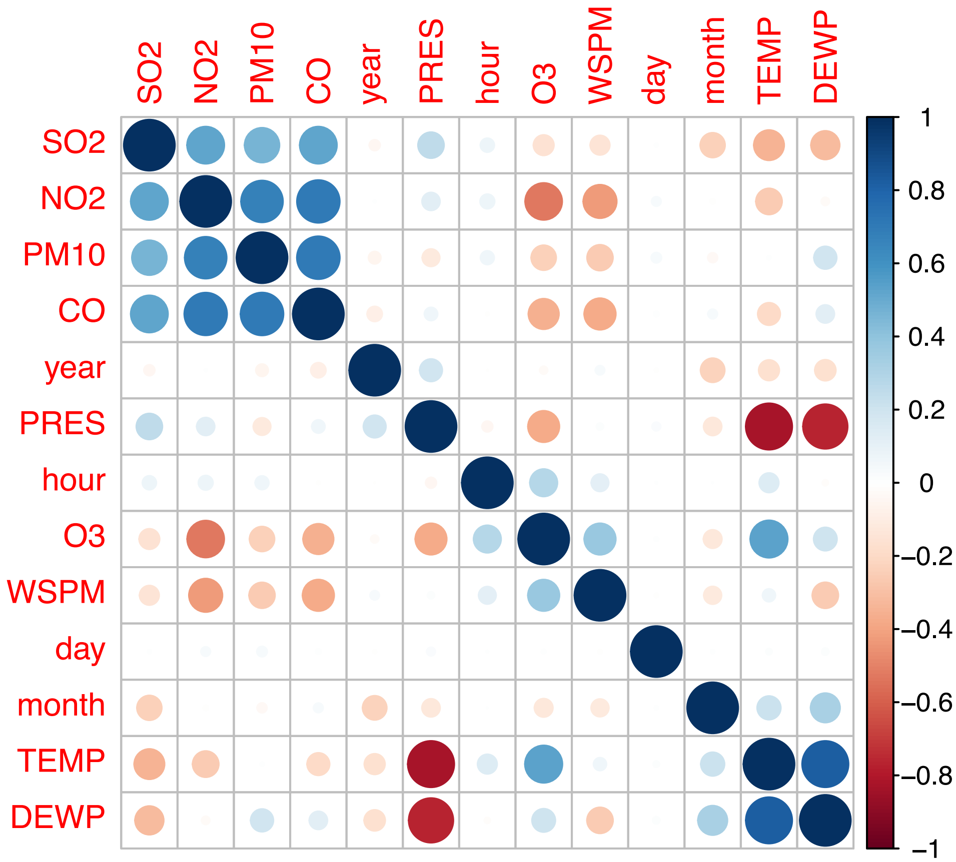









```
# checking for correlations
x.corr <- cor(x.trainp)
library(corrplot)
corrplot(x.corr, order = "hclust")
```

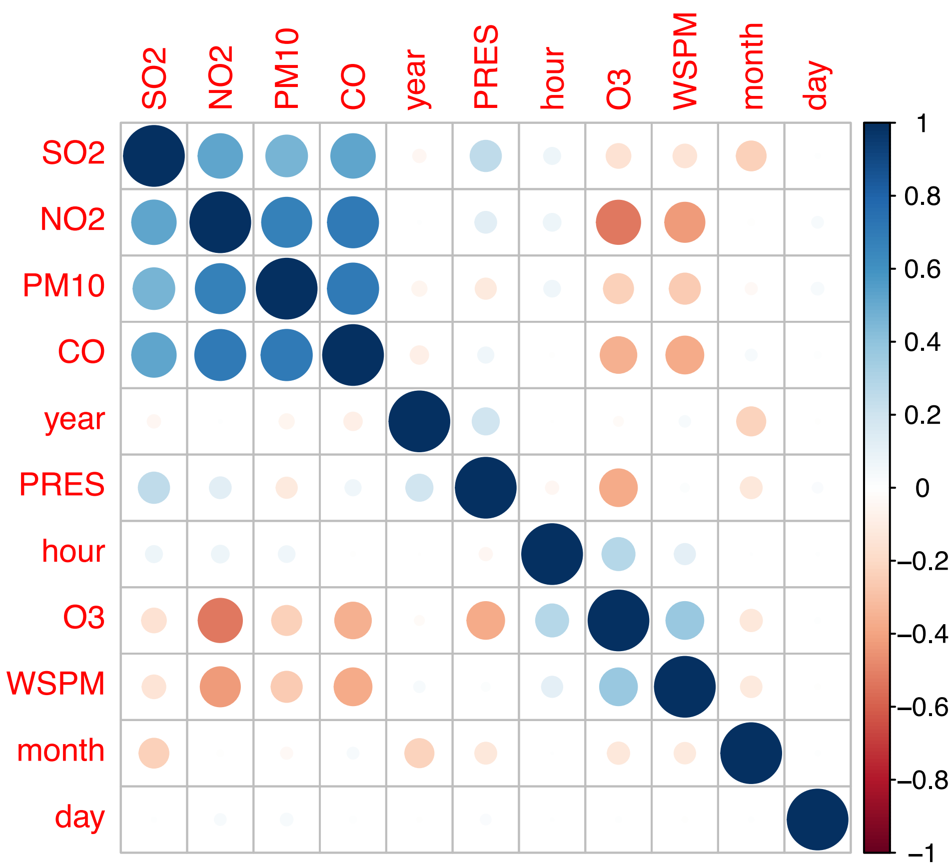



```r
hCorr <- findCorrelation(x.corr, cutoff = 0.75, exact = TRUE)
x.trainpc <- x.trainp[, -hCorr]
x.testpc <- x.testp[, -hCorr]
x.corrCheck <- cor(x.trainpc)
x.corrCheck
```

```
##                year        month          day          hour        PM10
## year   1.0000000000 -0.220995074 -0.004555605 -0.0003806929 -0.05636036
## month -0.2209950737  1.000000000  0.004912318  0.0004323410 -0.03482742
## day   -0.0045556049  0.004912318  1.000000000  0.0015220176  0.03849737
## hour  -0.0003806929  0.000432341  0.001522018  1.0000000000  0.06906342
## PM10  -0.0563603642 -0.034827423  0.038497373  0.0690634242  1.00000000
## SO2   -0.0427719109 -0.232968230  0.004591885  0.0718949090  0.46344592
## NO2    0.0021436699 -0.008830707  0.032289563  0.0786462009  0.67218506
## CO    -0.0851369005  0.032977567  0.009516887 -0.0037392558  0.70514562
## O3    -0.0223801562 -0.127280613  0.006558264  0.2868054160 -0.23603793
## PRES   0.1911976270 -0.122487232  0.026622130 -0.0420661905 -0.11398468
## WSPM   0.0310248765 -0.117616029 -0.005961203  0.1139495422 -0.25361462
##               SO2          NO2           CO           O3        PRES
## year  -0.042771911  0.002143670 -0.085136901 -0.022380156  0.19119763
## month -0.232968230 -0.008830707  0.032977567 -0.127280613 -0.12248723
## day    0.004591885  0.032289563  0.009516887  0.006558264  0.02662213
## hour   0.071894909  0.078646201 -0.003739256  0.286805416 -0.04206619
## PM10   0.463445923  0.672185063  0.705145619 -0.236037927 -0.11398468
## SO2    1.000000000  0.528683552  0.528485484 -0.157764142  0.25635066
## NO2    0.528683552  1.000000000  0.700884908 -0.522652765  0.12084783
## CO     0.528485484  0.700884908  1.000000000 -0.354476400  0.06435137
```

```
## O3    -0.157764142 -0.522652765 -0.354476400  1.000000000 -0.37581756
## PRES   0.256350660  0.120847832  0.064351370 -0.375817555  1.00000000
## WSPM  -0.143968581 -0.427182939 -0.379807062  0.378164028  0.01716655
##               WSPM
## year   0.031024877
## month -0.117616029
## day   -0.005961203
## hour   0.113949542
## PM10  -0.253614619
## SO2   -0.143968581
## NO2   -0.427182939
## CO    -0.379807062
## O3     0.378164028
## PRES   0.017166551
## WSPM   1.000000000
```

```r
corrplot(x.corrCheck, order = "hclust")
```

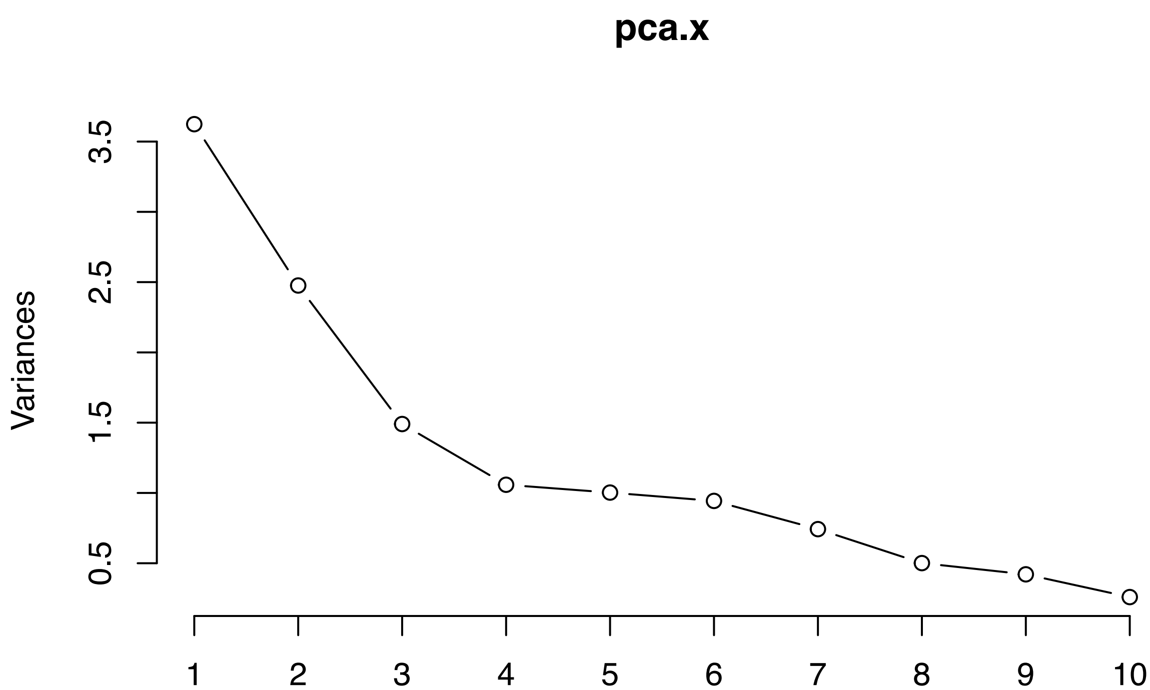



```r
# pca to determine the effective dimensions of the data
pca.x <- prcomp(x.train, center = TRUE, scale. = TRUE)
summary(pca.x)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     1.9036 1.5735 1.2208 1.02873 1.00136 0.97126 0.86177
## Proportion of Variance 0.2787 0.1905 0.1146 0.08141 0.07713 0.07257 0.05713
## Cumulative Proportion  0.2787 0.4692 0.5839 0.66526 0.74240 0.81496 0.87209
##                            PC8     PC9    PC10    PC11    PC12    PC13
```

```
## Standard deviation     0.70773 0.64884 0.50916 0.4757 0.42296 0.27669
## Proportion of Variance 0.03853 0.03238 0.01994 0.0174 0.01376 0.00589
## Cumulative Proportion  0.91062 0.94300 0.96294 0.9804 0.99411 1.00000
```

```
plot(pca.x, type = "l")
```

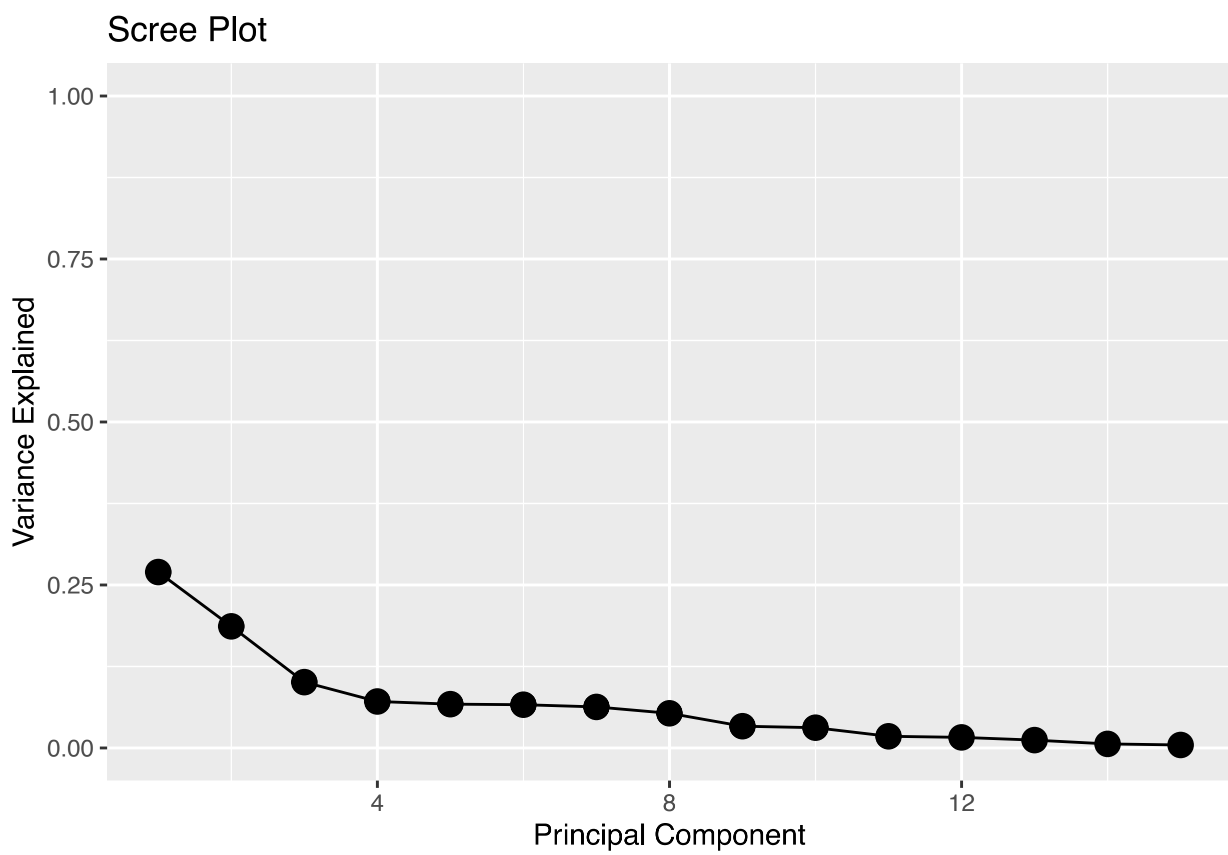



```
df = subset(air.knn, select = -c(station) )
pca <- prcomp(df, scale = TRUE)
pca
```

```
## Standard deviations (1, .., p=15):
##  [1] 2.0118640 1.6728418 1.2307788 1.0342707 1.0040115 0.9977254 0.9718079
##  [8] 0.8932577 0.7077300 0.6830616 0.5184660 0.4936256 0.4253049 0.3062092
## [15] 0.2628297
##
## Rotation (n x k) = (15 x 15):
##               PC1         PC2          PC3          PC4          PC5
## year   0.034148147 -0.15541204  0.157996068 -0.680292034  0.117554185
## month -0.044008501  0.12443403 -0.469222084  0.472384103 -0.088560722
## day    0.004701964  0.01080084 -0.010578203  0.124239220  0.764191678
## hour  -0.020367615  0.08437926  0.437330286  0.429385577 -0.040413673
## PM2.5  0.376502882  0.29639289  0.109763789 -0.079238662 -0.018963594
## PM10   0.364162461  0.29835035  0.155142346 -0.030892657  0.007393279
## SO2    0.331185633  0.03328032  0.258500114  0.150840798 -0.015429473
## NO2    0.407333448  0.13991551 -0.043405405  0.023852027  0.031515253
## CO     0.405745513  0.18936761  0.024247248  0.001892429 -0.057097404
## O3    -0.265851228  0.20325988  0.470690895  0.045117727  0.032983019
## TEMP  -0.307953813  0.43403417  0.062850951 -0.052406616  0.053075631
## PRES   0.229678831 -0.45559752 -0.021178389  0.136799515  0.001724717
## DEWP  -0.177321472  0.50160668 -0.193185278 -0.135848980  0.029608069
## RAIN  -0.033089004  0.04217899 -0.005805618 -0.109107247 -0.611047934
## WSPM  -0.176224369 -0.17047988  0.441343127  0.172726878 -0.098214369
##               PC6         PC7          PC8          PC9         PC10
```

```
## year  -0.12787793 -0.547301210  0.179248734  0.152408340 -0.298389805
## month -0.09422859 -0.367465272  0.429129330  0.296096609 -0.254438452
## day   -0.61120437  0.150381035  0.007364509 -0.016094456 -0.006346385
## hour  -0.10423200 -0.603682658 -0.340604771 -0.204227656  0.091668491
## PM2.5 -0.01363162  0.029119502  0.252407672  0.006969717  0.285069390
## PM10  -0.01226416  0.041349211  0.256530377 -0.125751523  0.216130419
## SO2    0.06851345  0.288267518 -0.273535705  0.405298678 -0.622473742
## NO2   -0.01342260 -0.164543259 -0.171081443 -0.326958419 -0.257375261
## CO    -0.02381849 -0.025678431  0.185532681  0.150458770  0.028807797
## O3     0.01578325  0.003863143  0.081916063  0.576135400  0.241941619
## TEMP   0.03630566 -0.009831329 -0.015917309 -0.041833474 -0.171089395
## PRES  -0.07284076 -0.121120653  0.070995889  0.248248746  0.243269839
## DEWP  -0.01600532 -0.039092810 -0.037345989 -0.014973877 -0.077564473
## RAIN  -0.75879529  0.130745712 -0.116640357  0.024753347  0.005508721
## WSPM  -0.04069220  0.177733622  0.610424331 -0.375919427 -0.319725952
##              PC11        PC12         PC13        PC14         PC15
## year   0.06441018 -0.06058159  0.070116331 -0.00691414  0.005730137
## month  0.14628423  0.06999090  0.153157112  0.02446333 -0.024353553
## day   -0.03808482 -0.01436902  0.042709097  0.01522694 -0.008567630
## hour  -0.04241084 -0.25648651  0.065213830  0.02827172  0.033844045
## PM2.5  0.19329886 -0.19068145 -0.005574507  0.62546875 -0.375083636
## PM10   0.44251478 -0.03614761  0.142607659 -0.56773012  0.288652696
## SO2    0.17606069 -0.22876667  0.004803740  0.02907216  0.028995437
## NO2    0.01307697  0.70883432 -0.257136445  0.11218181  0.008282685
## CO    -0.82254505 -0.09137811  0.016466794 -0.22373026 -0.035851839
## O3    -0.02349488  0.48752304 -0.032371785  0.07932857  0.142553285
## TEMP   0.05752187 -0.03471922 -0.300117254 -0.37782056 -0.661243073
## PRES   0.12533205 -0.09626044 -0.721669520 -0.15539160 -0.069001674
## DEWP  -0.04866123 -0.27195403 -0.486422248  0.19491896  0.547676161
## RAIN   0.03414150  0.04649830  0.001893423 -0.02329054 -0.028667476
## WSPM  -0.08466494 -0.01928072 -0.173876113  0.09790283  0.094108273
```

```r
variance = pca$sdev^2 / sum(pca$sdev^2)
#variance
library(ggplot2)
qplot(c(1:15), variance) +
  geom_line() +
  geom_point(size=4)+
  xlab("Principal Component") +
  ylab("Variance Explained") +
  ggtitle("Scree Plot") +
  ylim(0, 1)
```




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

```
## Linear Regression
##
## 28053 samples
##    13 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 25247, 25248, 25247, 25248, 25247, 25248, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   43.98981  0.7126459  30.94438
##
## 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
## -203.52  -26.31   -7.14   18.66  682.88
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  80.0284     0.2627 304.671  < 2e-16 ***
## year          2.5329     0.2783   9.102  < 2e-16 ***
## month        -0.6449     0.3038  -2.123   0.0338 *
## day          -2.0005     0.2641  -7.575 3.70e-14 ***
## hour         -1.2060     0.2945  -4.095 4.24e-05 ***
## PM10         55.6881     0.4267 130.498  < 2e-16 ***
## SO2          -5.3414     0.3828 -13.952  < 2e-16 ***
## NO2          -3.3492     0.4895  -6.842 7.94e-12 ***
## CO           22.2220     0.4795  46.346  < 2e-16 ***
## O3            8.6353     0.4296  20.101  < 2e-16 ***
## TEMP        -20.1808     0.7808 -25.847  < 2e-16 ***
## PRES          0.8997     0.5036   1.787   0.0740 .
## DEWP          9.6484     0.7186  13.426  < 2e-16 ***
## WSPM         -0.9219     0.3361  -2.743   0.0061 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 43.99 on 28039 degrees of freedom
## Multiple R-squared:  0.7125, Adjusted R-squared:  0.7124
## F-statistic:  5345 on 13 and 28039 DF,  p-value: < 2.2e-16
```

```r
#testResults

# rfImp <- varImp(pcrTune2, scale = FALSE)
#rfImp

fp_predict <- predict(pcrTune2, x.testp)

postResample(fp_predict, y.test)
```

```
##       RMSE   Rsquared        MAE
## 41.9917804  0.7164962 30.3784025
```

```r
#Taking account of RMSE and Rsqr values OLS seems to be the better model.
# Although it tied with pls ols is the simpler model.
```

```r
# try to reduce features using pls
set.seed(100)
indx <- createFolds(y.train, returnTrain = TRUE)
ctrl <- trainControl(method = "cv", index = indx)
pcrTune3 <- train(x = x.trainp, y = y.train,
                 method = "pls",
                 tuneGrid = expand.grid(ncomp = 1:14),
                 trControl = ctrl)
pcrTune3
```

```
## Partial Least Squares
##
## 28053 samples
```

```
##    13 predictor
##
## No pre-processing
## 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     53.13895  0.5805025  37.49803
##    2     48.98620  0.6435108  34.50358
##    3     45.72558  0.6894527  32.22521
##    4     44.48332  0.7061492  31.32279
##    5     44.12880  0.7107961  31.06912
##    6     44.06693  0.7116164  31.03347
##    7     44.02033  0.7122380  30.94315
##    8     44.00305  0.7124613  30.96422
##    9     43.99285  0.7126046  30.94465
##   10     43.99052  0.7126378  30.94738
##   11     43.98999  0.7126439  30.94673
##   12     43.98983  0.7126457  30.94519
##   13     43.98981  0.7126459  30.94438
##   14     43.98981  0.7126459  30.94438
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 13.
```

```
summary(pcrTune3)
```

```
## Data:    X dimension: 28053 13
##  Y dimension: 28053 1
## Fit method: oscorespls
## Number of components considered: 13
## TRAINING: % variance explained
##           1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X           25.35    42.92    54.40     62.9    68.20    74.16    77.50
## .outcome    58.02    64.33    68.93     70.6    71.07    71.15    71.21
##           8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X           81.41    85.52     88.60     92.79     98.43    100.00
## .outcome    71.23    71.25     71.25     71.25     71.25     71.25
```

```
fp_predict <- predict(pcrTune3, x.testp)

postResample(fp_predict, y.test)
```

```
##       RMSE   Rsquared        MAE
## 41.9917804  0.7164962 30.3784025
```

```
#rfImp <- varImp(pcrTune3, scale = FALSE)
#rfImp
```

```
# great for large data decision trees will be having better average accuracy.
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
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] 18.68842
```

```
fp_predict <- predict(rfmodel , x.testp)

postResample(fp_predict, y.test)
```

```
##         RMSE     Rsquared          MAE
## 92.612027006  0.004136611 59.925981786
```

```
#rfImp <- varImp(rfmodel, scale = FALSE)
#rfImp
```

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

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: OLS, PLS
## Number of resamples: 10
##
## MAE
##         Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## OLS 30.04526 30.42321 31.03142 30.94438 31.19767 32.05851    0
## PLS 30.04526 30.42321 31.03142 30.94438 31.19767 32.05851    0
##
## RMSE
##         Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## OLS 41.33369 43.13942 44.35575 43.98981 44.68207 46.74327    0
## PLS 41.33369 43.13942 44.35575 43.98981 44.68207 46.74327    0
##
## Rsquared
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## OLS 0.6926605 0.7110892 0.7164937 0.7126459 0.7181176 0.7250153    0
## PLS 0.6926605 0.7110892 0.7164937 0.7126459 0.7181176 0.7250153    0
```