Crime and Communities

Group Member 1 Name: Edward Zamora **Group Member 1 SID:** 3032137148

The crime and communities dataset contains crime data from communities in the United States. The data combines socio-economic data from the 1990 US Census, law enforcement data from the 1990 US LEMAS survey, and crime data from the 1995 FBI UCR. More details can be found at https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime+Unnormalized.

The dataset contains 125 columns total; $p = 124$ predictive and 1 target (ViolentCrimesPerPop). There are $n = 1994$ observations. These can be arranged into an $n \times p = 1994 \times 127$ feature matrix $\mathbf{X}$, and an $n \times 1 = 1994 \times 1$ response vector $\mathbf{y}$ (containing the observations of ViolentCrimesPerPop).

Once downloaded (from bCourses), the data can be loaded as follows.

```
library(readr)
library(randomForest)
```

```
## randomForest 4.6-14

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

```
CC <- read_csv("crime_and_communities_data.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double()
## )

## See spec(...) for full column specifications.
```

```
print(dim(CC))
```

```
## [1] 1994  125
```

```
y <- CC$ViolentCrimesPerPop
X <- subset(CC, select = -c(ViolentCrimesPerPop))
```

Dataset exploration

In this section, you should provide a thorough exploration of the features of the dataset. Things to keep in mind in this section include:

- Which variables are categorical versus numerical?
- What are the general summary statistics of the data? How can these be visualized?
- Is the data normalized? Should it be normalized?
- Are there missing values in the data? How should these missing values be handled?
- Can the data be well-represented in fewer dimensions?

```
str(X)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    1994 obs. of  124 variables:
##  $ population          : num  11980 23123 29344 16656 140494 ...
##  $ householdsize       : num  3.1 2.82 2.43 2.4 2.45 2.6 2.45 2.46 2.62 2.54 ...
##  $ racepctblack        : num  1.37 0.8 0.74 1.7 2.51 ...
##  $ racePctWhite        : num  91.8 95.6 94.3 97.3 95.7 ...
##  $ racePctAsian        : num  6.5 3.44 3.43 0.5 0.9 1.47 0.4 1.25 0.92 0.77 ...
```

```
##  $ racePctHisp          : num  1.88 0.85 2.35 0.7 0.95 ...
##  $ agePct12t21          : num  12.5 11 11.4 12.6 18.1 ...
##  $ agePct12t29          : num  21.4 21.3 25.9 25.2 32.9 ...
##  $ agePct16t24          : num  10.9 10.5 11 12.2 20 ...
##  $ agePct65up           : num  11.3 17.2 10.3 17.6 13.3 ...
##  $ numbUrban            : num  11980 23123 29344 0 140494 ...
##  $ pctUrban             : num  100 100 100 0 100 100 100 100 100 100 ...
##  $ medIncome            : num  75122 47917 35669 20580 21577 ...
##  $ pctWWage             : num  89.2 79 82 68.2 75.8 ...
##  $ pctWFarmSelf         : num  1.55 1.11 1.15 0.24 1 0.39 0.67 2.93 0.86 1.54 ...
##  $ pctWInvInc           : num  70.2 64.1 55.7 39 41.1 ...
##  $ pctWSocSec           : num  23.6 35.5 22.2 39.5 29.3 ...
##  $ pctWPubAsst          : num  1.03 2.75 2.94 11.71 7.12 ...
##  $ pctWRetire           : num  18.4 22.9 14.6 18.3 14.1 ...
##  $ medFamInc            : num  79584 55323 42112 26501 27705 ...
##  $ perCapInc            : num  29711 20148 16946 10810 11878 ...
##  $ whitePerCap          : num  30233 20191 17103 10909 12029 ...
##  $ blackPerCap          : num  13600 18137 16644 9984 7382 ...
##  $ indianPerCap         : num  5725 0 21606 4941 10264 ...
##  $ AsianPerCap          : num  27101 20074 15528 3541 10753 ...
##  $ OtherPerCap          : num  5115 5250 5954 2451 7192 ...
##  $ HispPerCap           : num  22838 12222 8405 4391 8104 ...
##  $ NumUnderPov          : num  227 885 1389 2831 23223 ...
##  $ PctPopUnderPov       : num  1.96 3.98 4.75 17.23 17.78 ...
##  $ PctLess9thGrade      : num  5.81 5.61 2.8 11.05 8.76 ...
##  $ PctNotHSGrad         : num  9.9 13.72 9.09 33.68 23.03 ...
##  $ PctBSorMore          : num  48.2 29.9 30.1 10.8 20.7 ...
##  $ PctUnemployed        : num  2.7 2.43 4.01 9.86 5.72 4.85 8.19 4.18 8.39 7.19 ...
##  $ PctEmploy            : num  64.5 62 69.8 54.7 59 ...
##  $ PctEmplManu          : num  14.7 12.3 15.9 31.2 14.3 ...
##  $ PctEmplProfServ      : num  28.8 29.3 21.5 27.4 26.8 ...
##  $ PctOccupManu         : num  5.49 6.39 8.79 26.76 14.72 ...
##  $ PctOccupMgmtProf     : num  50.7 37.6 32.5 22.7 23.4 ...
##  $ MalePctDivorce       : num  3.67 4.23 10.1 10.98 11.4 ...
##  $ MalePctNevMarr       : num  26.4 28 25.8 28.1 33.3 ...
##  $ FemalePctDiv         : num  5.22 6.45 14.76 14.47 14.46 ...
##  $ TotalPctDiv          : num  4.47 5.42 12.55 12.91 13.04 ...
##  $ PersPerFam           : num  3.22 3.11 2.95 2.98 2.89 3.14 2.95 3 3.11 2.99 ...
##  $ PctFam2Par           : num  91.4 86.9 78.5 64 71.9 ...
##  $ PctKids2Par          : num  90.2 85.3 78.8 62.4 69.8 ...
##  $ PctYoungKids2Par     : num  95.8 96.8 92.4 65.4 79.8 ...
##  $ PctTeen2Par          : num  95.8 86.5 75.7 67.4 75.3 ...
##  $ PctWorkMomYoungKids  : num  44.6 51.1 66.1 59.6 63 ...
##  $ PctWorkMom           : num  58.9 62.4 74.2 70.3 70.5 ...
##  $ NumKidsBornNeverMar  : num  31 43 164 561 1511 ...
##  $ PctKidsBornNeverMar  : num  0.36 0.24 0.88 3.84 1.58 1.18 4.66 1.64 4.71 2.47 ...
##  $ NumImmig             : num  1277 1920 1468 339 2091 ...
##  $ PctImmigRecent       : num  8.69 5.21 16.42 13.86 21.33 ...
##  $ PctImmigRec5         : num  13 8.65 23.98 13.86 30.56 ...
##  $ PctImmigRec8         : num  21 13.3 32.1 15.3 38 ...
##  $ PctImmigRec10        : num  30.9 22.5 35.6 15.3 45.5 ...
##  $ PctRecentImmig       : num  0.93 0.43 0.82 0.28 0.32 1.05 0.11 0.47 0.72 0.53 ...
##  $ PctRecImmig5         : num  1.39 0.72 1.2 0.28 0.45 1.49 0.2 0.67 1.07 1.05 ...
##  $ PctRecImmig8         : num  2.24 1.11 1.61 0.31 0.57 2.2 0.25 0.93 1.63 1.66 ...
```

```
##  $ PctRecImmig10        : num  3.3 1.87 1.78 0.31 0.68 2.55 0.29 1.07 2.31 1.94 ...
##  $ PctSpeakEnglOnly     : num  85.7 87.8 93.1 95 96.9 ...
##  $ PctNotSpeakEnglWell  : num  1.37 1.81 1.14 0.56 0.6 0.6 0.28 0.43 2.51 0.81 ...
##  $ PctLargHouseFam      : num  4.81 4.25 2.97 3.93 3.08 5.08 3.85 2.59 6.7 3.66 ...
##  $ PctLargHouseOccup    : num  4.17 3.34 2.05 2.56 1.92 3.46 2.55 1.54 4.1 2.51 ...
##  $ PersPerOccupHous     : num  2.99 2.7 2.42 2.37 2.28 2.55 2.36 2.32 2.45 2.42 ...
##  $ PersPerOwnOccHous    : num  3 2.83 2.69 2.51 2.37 2.89 2.42 2.77 2.47 2.5 ...
##  $ PersPerRentOccHous   : num  2.84 1.96 2.06 2.2 2.16 2.09 2.27 1.91 2.44 2.31 ...
##  $ PctPersOwnOccup      : num  91.5 89 64.2 58.2 57.8 ...
##  $ PctPersDenseHous     : num  0.39 1.01 2.03 1.21 2.11 1.47 1.9 1.67 6.14 3.41 ...
##  $ PctHousLess3BR       : num  11.1 23.6 47.5 45.7 53.2 ...
##  $ MedNumBR             : num  3 3 3 3 2 3 2 2 2 2 ...
##  $ HousVacant           : num  64 240 544 669 5119 ...
##  $ PctHousOccup         : num  98.4 97.2 95.7 91.2 91.8 ...
##  $ PctHousOwnOcc        : num  91 84.9 57.8 54.9 55.5 ...
##  $ PctVacantBoarded     : num  3.12 0 0.92 2.54 2.09 1.41 6.39 0.45 5.64 2.77 ...
##  $ PctVacMore6Mos       : num  37.5 18.33 7.54 57.85 26.22 ...
##  $ MedYrHousBuilt       : num  1959 1958 1976 1939 1966 ...
##  $ PctHousNoPhone       : num  0 0.31 1.55 7 6.13 ...
##  $ PctWOFullPlumb       : num  0.28 0.14 0.12 0.87 0.31 0.28 0.49 0.19 0.33 0.3 ...
##  $ OwnOccLowQuart       : num  215900 136300 74700 36400 37700 ...
##  $ OwnOccMedVal         : num  262600 164200 90400 49600 53900 ...
##  $ OwnOccHiQuart        : num  326900 199900 112000 66500 73100 ...
##  $ OwnOccQrange         : num  111000 63600 37300 30100 35400 60400 26100 39200 38800 41400 ...
##  $ RentLowQ             : num  685 467 370 195 215 463 186 241 192 234 ...
##  $ RentMedian           : num  1001 560 428 250 280 ...
##  $ RentHighQ            : num  1001 672 520 309 349 ...
##  $ RentQrange           : num  316 205 150 114 134 361 139 146 177 142 ...
##  $ MedRent              : num  1001 627 484 333 340 ...
##  $ MedRentPctHousInc    : num  23.8 27.6 24.1 28.7 26.4 24.4 26.3 25.2 29.6 23.8 ...
##  $ MedOwnCostPctInc     : num  21.1 20.7 21.7 20.6 17.3 20.8 15.1 20.7 19.4 17.1 ...
##  $ MedOwnCostPctIncNoMtg: num  14 12.5 11.6 14.5 11.7 12.5 12.2 12.8 13 12.9 ...
##  $ NumInShelters        : num  11 0 16 0 327 0 21 125 43 1 ...
##  $ NumStreet            : num  0 0 0 0 4 0 0 15 4 0 ...
##  $ PctForeignBorn       : num  10.66 8.3 5 2.04 1.49 ...
##  $ PctBornSameState     : num  53.7 77.2 44.8 88.7 64.3 ...
##  $ PctSameHouse85       : num  65.3 71.3 36.6 56.7 42.3 ...
##  $ PctSameCity85        : num  78.1 90.2 61.3 90.2 70.6 ...
##  $ PctSameState85       : num  89.1 96.1 82.8 96.2 85.7 ...
##  $ LemasSwornFT         : num  NA NA NA NA NA NA NA NA 198 NA ...
##   [list output truncated]
```

The first part of my regression analysis is to make sure there are no abnormalities in the data that should be accounted for. Evaluating the structure of my data, I can see whether or not there are any categorical variables that must be handled before a quantitaive analysis is performed. From the summary above, no such variables exist.

```r
colSums(!is.na(X))[colSums(!is.na(X))!=1994]
```

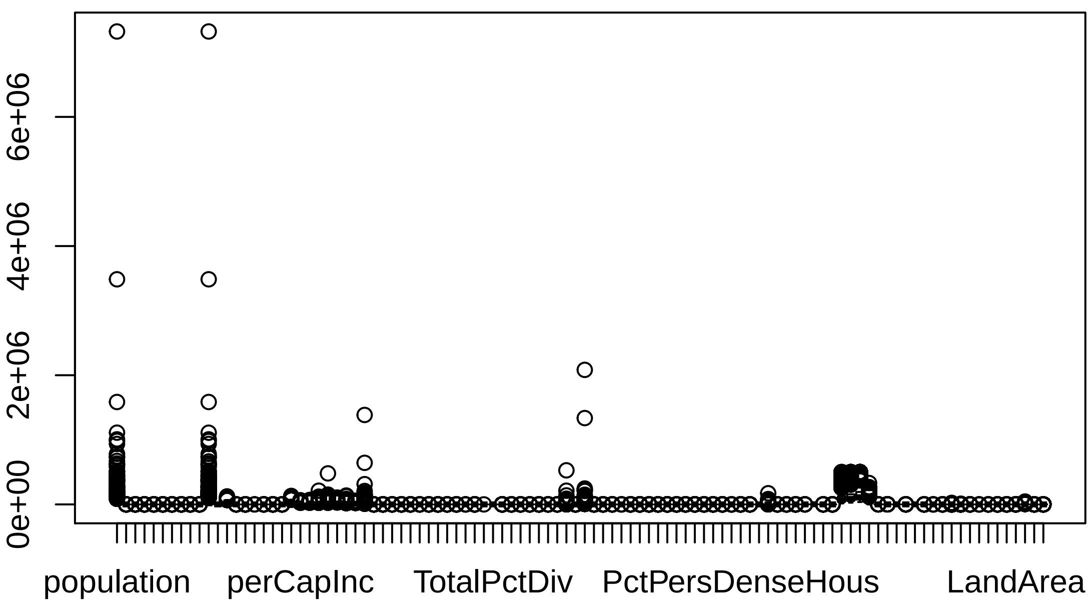
```
##         OtherPerCap        LemasSwornFT      LemasSwFTPerPop
##                1993                 319                  319
##   LemasSwFTFieldOps LemasSwFTFieldPerPop        LemasTotalReq
##                 319                  319                  319
##    LemasTotReqPerPop     PolicReqPerOffic           PolicPerPop
##                 319                  319                  319
```

```
##   RacialMatchCommPol        PctPolicWhite        PctPolicBlack
##                  319                  319                  319
##         PctPolicHisp        PctPolicAsian        PctPolicMinor
##                  319                  319                  319
##  OfficAssgnDrugUnits     NumKindsDrugsSeiz     PolicAveOTWorked
##                  319                  319                  319
##            PolicCars        PolicOperBudg  LemasPctPolicOnPatr
##                  319                  319                  319
##  LemasGangUnitDeploy       PolicBudgPerPop
##                  319                  319
```

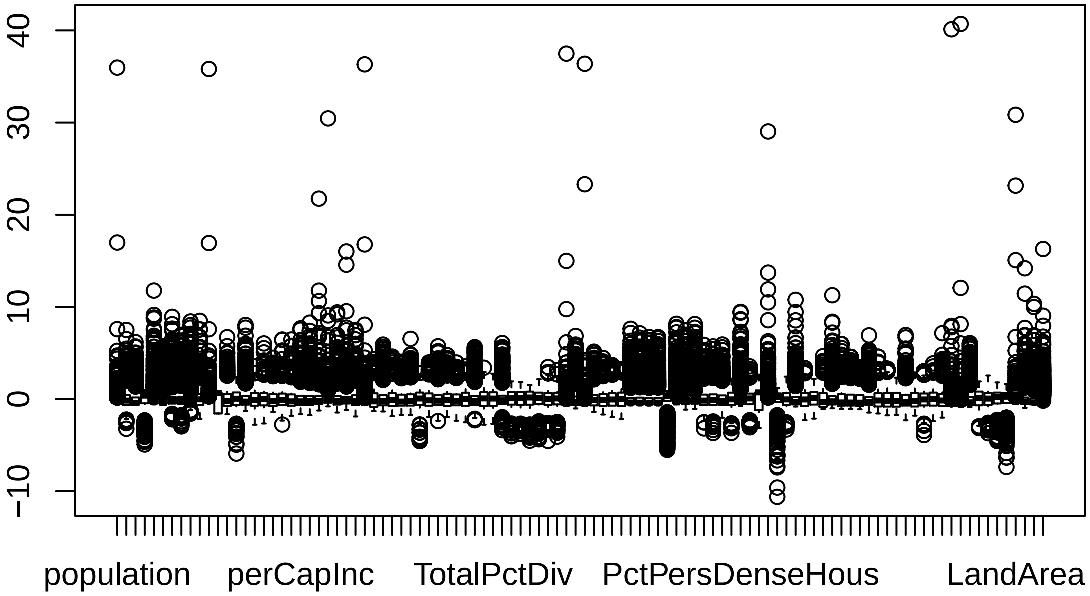
```r
#without na columns
X <- as.matrix(X[,colSums(is.na(X))<2])
X[is.na(X)] <- 0
```

The next part of fixing my data before analysis is handling na values. There are several ways that I considered handling these values: removing data points with na values, removing predictors with na values, or setting all na values to 0. As you can see in the data above depicting the number of non na values present in each column that has at least one na value, there are a significant number of rows containing some na value so removing those data points is not a viable option. Similarly, since these values are so pervasive within these columns, setting them to 0 would offer me not much benefit in the way of prediction, therefore I chose to omit the predictors entirely. However, one predictor column only had one missing value. In this case, rather than completely omitting this predictor, I replaced the missing value with 0 since it would likely add little distortion.

```r
boxplot(X)
```




```r
X <- scale(X)
boxplot(X)
```



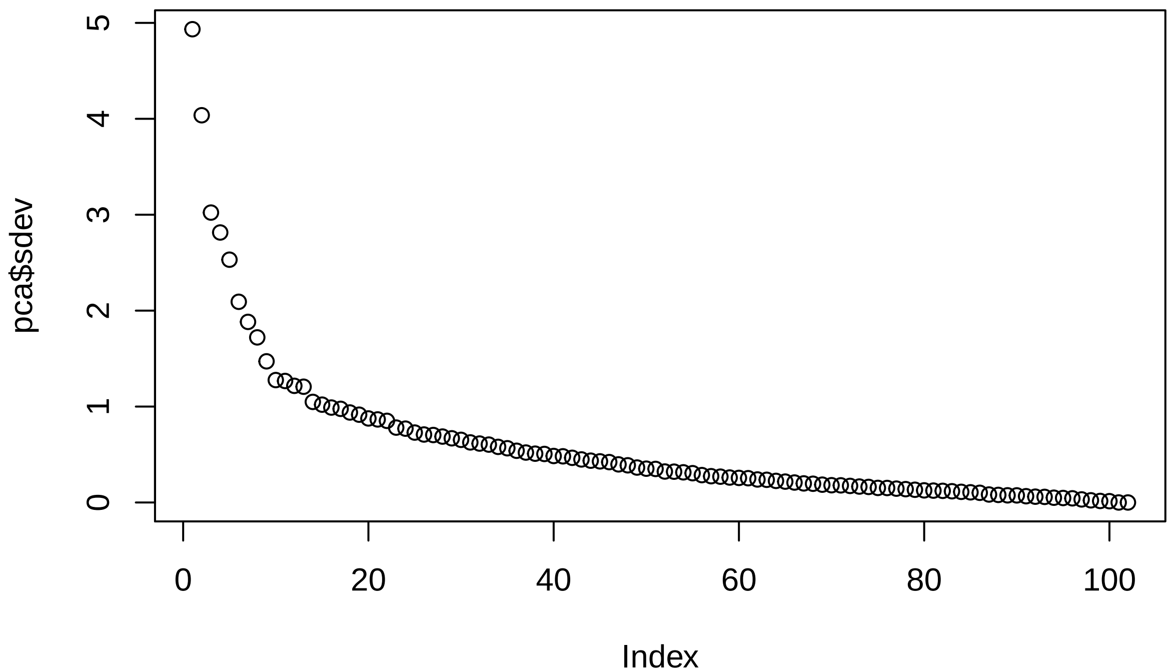
I then decided whether or not the data needed to be scaled. In looking at the labels of the data, there appeared to be a clear difference in values as some predictors involved perentages while others did not. To verify that scaling was necessary, I plotted the boxplots of each predictor side by side and determined that several of the predictor values were inherently much larger than others and would therefore skew the data. Once I scaled the data, the boxplots normalized and I proceeded with my analysis.

PCA

```
pca <- prcomp(X,scale. = FALSE,center = FALSE)
pca$sdev
```

```
##   [1] 4.933966e+00 4.037117e+00 3.022384e+00 2.814665e+00 2.531148e+00
##   [6] 2.092214e+00 1.882740e+00 1.721134e+00 1.471477e+00 1.276029e+00
##  [11] 1.266702e+00 1.216174e+00 1.206858e+00 1.048662e+00 1.019971e+00
##  [16] 9.895396e-01 9.763544e-01 9.381271e-01 9.152965e-01 8.761313e-01
##  [21] 8.657901e-01 8.509163e-01 7.806852e-01 7.698657e-01 7.288822e-01
##  [26] 7.085773e-01 7.032001e-01 6.871827e-01 6.690717e-01 6.527919e-01
##  [31] 6.261974e-01 6.142116e-01 6.034178e-01 5.796782e-01 5.651996e-01
##  [36] 5.389977e-01 5.205935e-01 5.088156e-01 5.057099e-01 4.845228e-01
##  [41] 4.812565e-01 4.657192e-01 4.486103e-01 4.357900e-01 4.286152e-01
##  [46] 4.208886e-01 3.972042e-01 3.874750e-01 3.640864e-01 3.525126e-01
##  [51] 3.495562e-01 3.226580e-01 3.211362e-01 3.142963e-01 3.053390e-01
##  [56] 2.854050e-01 2.738319e-01 2.691632e-01 2.599795e-01 2.561993e-01
##  [61] 2.529884e-01 2.397755e-01 2.363357e-01 2.249431e-01 2.173898e-01
##  [66] 2.080696e-01 1.987329e-01 1.952266e-01 1.857402e-01 1.800555e-01
##  [71] 1.784718e-01 1.727762e-01 1.659636e-01 1.619957e-01 1.518980e-01
##  [76] 1.510932e-01 1.443785e-01 1.389872e-01 1.327120e-01 1.262549e-01
##  [81] 1.235916e-01 1.213072e-01 1.170675e-01 1.110362e-01 1.057859e-01
##  [86] 1.009247e-01 8.311773e-02 7.890379e-02 7.452075e-02 7.402223e-02
##  [91] 6.522195e-02 6.051562e-02 5.819841e-02 4.971982e-02 4.678245e-02
##  [96] 4.340760e-02 3.193385e-02 2.233961e-02 1.564423e-02 1.363806e-02
## [101] 2.191121e-15 1.103233e-15
```

```r
plot(pca$sdev)
```



Before regressing, I verified whether or not it was possible to reduce the dimensions of my data in any significant way. To do so, I performed PCA on my data and then determined the proper nomber of PCs to utilize. I cosidered multiple ways to determine the number of PCs from the data such as finding where the "elbow" of my standard deviation graph was, but I decided to use a threshhold value of .7 instead since more stringent determinations explained much less variance. In this method, the 27 PCs explained almost 70% of the data's variance.

Understanding PCA

```
##
##           agePct12t21           agePct12t29           agePct16t24
##                     1                     1                     1
##            agePct65up           AsianPerCap           blackPerCap
##                     1                     1                     3
##          FemalePctDiv            HispPerCap           indianPerCap
##                     1                     2                     2
##              LandArea  LemasPctOfficDrugUn        MalePctDivorce
##                     2                     4                     1
##         MalePctNevMarr             MedNumBR      MedOwnCostPctInc
##                     1                     2                     2
## MedOwnCostPctIncNoMtg     MedRentPctHousInc        MedYrHousBuilt
##                     2                     2                     2
##           OtherPerCap          OwnOccMedVal          OwnOccQrange
##                     2                     1                     1
##      PctBornSameState           PctEmplManu        PctHousLess3BR
##                     2                     1                     1
##        PctHousNoPhone          PctHousOccup          PctImmigRec5
##                     1                     2                     1
##    PctKidsBornNeverMar       PctLargHouseFam      PctLargHouseOccup
##                     1                     2                     1
```

```
##          PctNotHSGrad      PctOccupMgmtProf        PctPopUnderPov
##                     1                     1                     1
##        PctSameState85     PctSpeakEnglOnly              pctUrban
##                     1                     1                     4
##         PctUsePubTrans     PctVacantBoarded           pctWFarmSelf
##                     1                     1                     3
##        PctWOFullPlumb            PctWorkMom   PctWorkMomYoungKids
##                     5                     1                     1
##            pctWRetire            pctWSocSec            PersPerFam
##                     1                     1                     1
##      PersPerOccupHous     PersPerOwnOccHous    PersPerRentOccHous
##                     1                     1                     1
##               PopDens          racePctAsian          racepctblack
##                     1                     1                     1
##          racePctWhite            RentQrange           TotalPctDiv
##                     2                     1                     1
```

Now that the PCs have been calculated, it would be helpful to interperet them, at least generally. Looking at a table of the predictors with the top 3 higest coefficients in each of the first 27 PCs, we can see that some factors are weighted higher in determining the number of violent crimes in a given setting. It appears that urban populations tend to be the main sites of these crime with most being related to drugs in some way. Additionally, there appears to be a higher tendency for these crimes to be perpetrated by Hispanic and Black individuals.

Regression task

In this section, you should use the techniques learned in class to develop a model to predict ViolentCrimesPerPop using the 124 features (or some subset of them) stored in **X**. Remember that you should try several different methods, and use model selection methods to determine which model is best. You should also be sure to keep a held-out test set to evaluate the performance of your model.

```r
train <- sample(1994,1994*.7)
test <- c(1:1994)[-train][sample(1994*.3,1994*.15)]
validation <- c(1:1994)[-c(train,test)]
```

Before I began my actual regression, I performed a 3 way split of the data: the training set to fit my models, the validation set to choose which model performed best, and a test set to estimate the true error rate of my data. In my evaluations, I utilized the standard MSE value to determine which models were most accurate.

```r
X_train <- X[train,]
y_train <- y[train]
```

OLS

```r
ols_mod <- lm(y_train~.,data.frame(X_train,y_train))

predictions <- predict(ols_mod,data.frame(X[validation,],y[validation]))
```

```
## Warning in predict.lm(ols_mod, data.frame(X[validation, ], y[validation])):
## prediction from a rank-deficient fit may be misleading
```

```r
mean((predictions-y[validation])^2)
```

```
## [1] 179047.4
```

The first model I attempted was an ordinary least squares approach of the data on all of the predictors offered. However, a negative effect of this model is that I learned nothing about what predictors were most important. In comparison to the models used later, the OLS was fairly accurate in its predictions.

PCR

```
pca <- prcomp(X_train,scale. = FALSE,center = FALSE)

pcaols_mod <- lm(y_train~.,data.frame(pca$x[,1:27],y_train))

pcapredict <- predict(pcaols_mod ,data.frame(X[validation,]%*%pca$rotation[,1:27],y[validation]))
mean((pcapredict-y[validation])^2)
```

```
## [1] 146375.8
```

The second method I attempted was PCR using an OLS approach for the main regression. Using the 27 PCs I determined to be of greatest importance in the exploratory analysis, I found the PCR to be a noticeable improvement upon the normal OLS method when predicting on our validation data.

PLS

CV

```
##  [1]  3  4  8  5  6  9 10  7  2  1
```

The third method I used was a PLS regression on the training data. In order to determine the number of PLS components optimal for my regression, I used a 5 fold cross validation to find that the MSE was minimized using only 3 components. (In my cv, I have r only extending to 10. This is because the cv process was very computationally heavy and the MSE for higher values significantly increased.)

```
x <- as.matrix(X_train)
Y <- as.matrix(y_train)
r <- 3
z <- matrix(nrow = nrow(x) ,ncol = r) #components
w <- matrix(0,nrow = ncol(x),ncol = r) #wights
b <- matrix(0,nrow = 1, ncol = r) #coefficients
p <- matrix(0,nrow = ncol(x), ncol = r) #loadings
for (h in 1:r){
  w[,h] <- (t(x)%*%Y)/(t(Y)%*%Y)[1]
  w[,h] <- w[,h]/sqrt(sum(w[,h]^2))
  z[,h] <- x%*%w[,h]
  p[,h] <- t(x)%*%z[,h]/(t(z[,h])%*%z[,h])[1]
  b[,h] <- t(Y)%*%z[,h]/(t(z[,h])%*%z[,h])[1]
  x <- x-z[,h]%*%t(p[,h])
  Y <- Y-t(b[,h]%*%z[,h])
}
```

MSE

```
## [1] 502747.9
```

PLS offered no significant advantage in comparison to PCR and even OLS when evaluated on the validation data. In fact, it performed much worse than all the models constructed on the training data so it is not useful for prediction in this instance.

Random Forest

```
## [1] 133526.7 133337.2
```

```
forest.crime =randomForest(y_train~.,crime,mtry=11,ntree=300)
mean((predict(forest.crime,data.frame(X,y)[validation,])-y[validation])^2)
```

```
## [1] 136847.2
```

The last parametric model I attempted was a random forest regression. I used a random forest in place of a normal decision tree to aid its prediction values. According to standard practice, I set the number of random predictors that the model would look at to be the square root of the total number of predictors. However, I found that, since the number calculated wasn't an integer, I would perform cross validation to determine which value should be used. Once determined, I modeled a random forest based on my training data and predicted using the validation data. This method offered the lowest MSE with a competitive runtime in comparison to the previous models.

Understanding Random Forest

```
##  [1] "PctKids2Par"         "PctKidsBornNeverMar" "PctFam2Par"
##  [4] "racePctWhite"        "NumKidsBornNeverMar" "racepctblack"
##  [7] "PctTeen2Par"         "pctWInvInc"          "PctYoungKids2Par"
## [10] "NumUnderPov"
```

Based on the values returned by the random forest, the top 10 most important fators in violent crimes can be seen above. Interestingly, family life appears to play a significant part in determining the number of violent crimes in a population as several factors pertain to number of children or size of the family. As in the the PCA, the number of black individuals in a population is a contributing factor to crime. More interesting, the number of white people plays an even more significant part. I assume this did not appear in the PCs as I was only filtering for positive values, or values that increased arrests, therefore it could be assumed that areas of more white people have less arrests. Still, this is not to say tht white individuals commit more crimes, simply that they ae arrested less.

Non Parametric Method

```
kNNR <-function(z,k){
  yhat =c()
  for(i in 1:nrow(z)){
    ix =sort(rowSums((pca$x[,1:27]-matrix(rep(z[i,], nrow(pca$x[,1:27])),
                                          nrow = nrow(pca$x[,1:27]),
                                          byrow = TRUE))^2), index.return = TRUE)$ix

    ix = ix[1:k]
    ynn = y_train[ix]
    yhat =c(yhat,mean(ynn))
    }
  return(yhat)
}

z=X[validation,]%*%pca$rotation[,1:27]
yhat =kNNR(z,k=10)
sum((yhat-y[validation])^2)
```

```
## [1] 47357674
```

The last method I attempted was a nonparametric method of K nearest neighbors. In this method, I simply averaged out the the response values of the closest neighbors of the point I was attempting to predict for. Using cross validation, I set the number of neighbors to 10, though this method only offered a slight improvement to the MSE in comparison to the worst model, the PLS regression, and so this too was not a viable option.

Based on the resulting MSEs, it appears that the Random Forest is the best model in terms of both prediction accuracy and computation time for the given data. The final prediction accuracy can be finally calculated using the constructed model and testing data.

Final Evaluation Using Rndom Forest

```
mean((predict(forest.crime,data.frame(X,y)[test,])-y[test])^2)
```

```
## [1] 137257.1
```