Communities and Crime

Statistical Learning Final Exam Project

Caria Natascia, Cozzolino Claudia, Petrella Alfredo

June 20, 2020

Aim of the project

Predicting the rate of violent crime in a community can be particularly helpful in reducing the actual possibility of such crimes occurring. Socio-economic, environmental and demographic characteristics can be important predictors of the level of violent crime in a population. Therefore, determining which factors are the most influential can play a key role in understanding this complex phenomenon of crime.

This is what we are going to do with this project, by investigating the USA Communities and Crime Data Set, sourced from the UCI Dataset Repository[7].

Dataset Description

The dataset contains a total number of 2215 instances, each one associated to a community in the US, and 147 attributes which result from the combination of socio-economic data from 1990 US Census, law enforcement data from the 1990 US LEMAS survey and crimedata from the 1995 FBI UCR.[1]

Data Cleaning

```
# import dataset (missing values stored as "?" - imported as NA)
crimedata <- read.csv("../data/crimedata.csv", na.strings="?")
dim(crimedata)
```

```
## [1] 2215  147
```

Attributes information:

- 4 non-predictive (communityname, countyCode, communityCode, fold)
- 125 predictive
- 18 potential response (murders, murdPerPop, rapes, rapesPerPop, robberies, robbbPerPop, assaults, assaultPerPop, burglaries, burglPerPop, larcenies, larcPerPop, autoTheft, autoTheftPerPop, arsons, arsonsPerPop, ViolentCrimesPerPop, nonViolPerPop)

From UCI description we know that the variables communityname and state are nominal while the remaining are all numeric.

```
# check if variables communityname and state are stored as factors
is.factor(crimedata$communityname)
```

```
## [1] TRUE
```

```
is.factor(crimedata$state)
```

```
## [1] TRUE
```

```
# check number of numeric variables: 145 (= 147 - 2) expected
sum(sapply(crimedata, is.numeric))
```

[1] Attributes details in the appendix.

```
## [1] 145
```

Missing values

```
# inspect total NA
sum(is.na(crimedata))
```

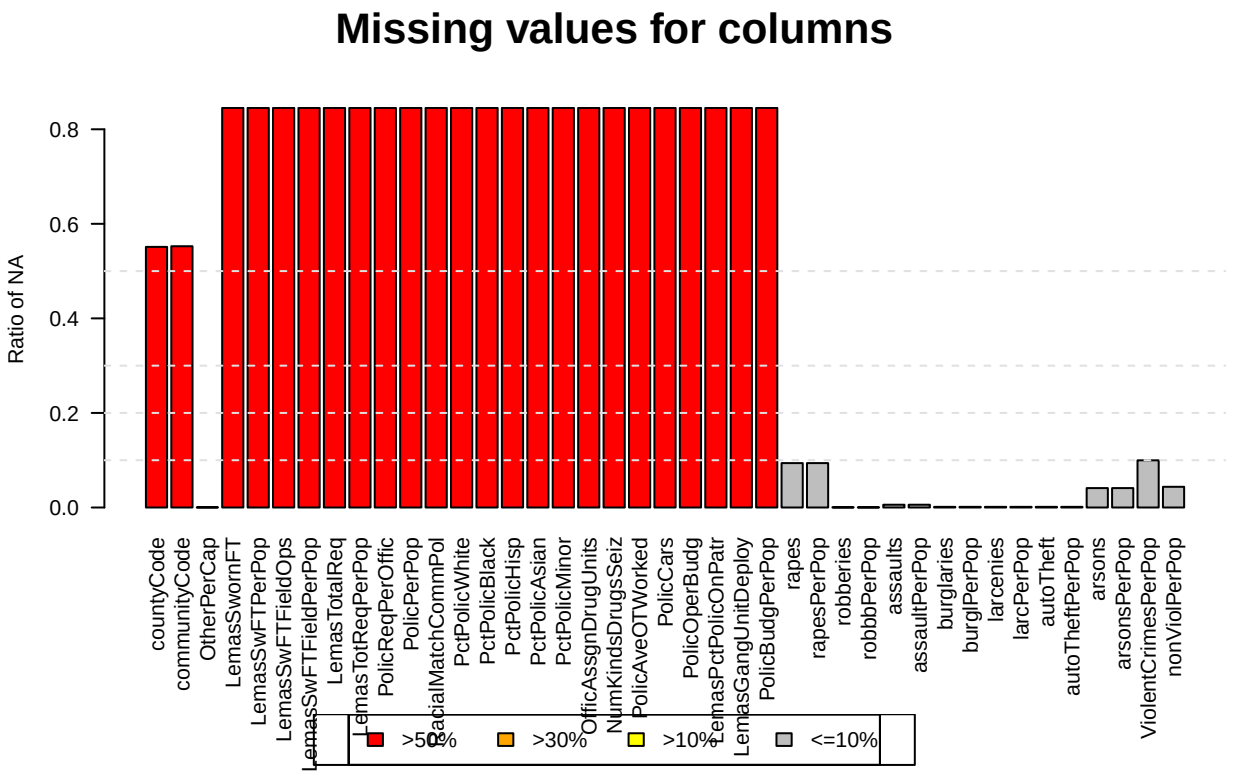
```
## [1] 44592
```

```
mean(is.na(crimedata))
```

```
## [1] 0.1369512
```

More than 10% of values is missing.

```
# dataframe with columns that present NA
nadf <- inspect.na(crimedata, hist=FALSE, summary=TRUE, byrow=FALSE, barplot=TRUE,
                   na.value = NA)
```



There are 41 columns with missing values, many of them with more than 50% of the data missing. Among them:

- 3 non-predictive (countyCode, communityCode, fold);
- many variables from US LEMAS dataset, including policing information;
- potential response (rapes, rapesPerPop, arsons, arsonsPerPop, ViolentCrimesPerPop, nonViolPerPop in particular)

From documentaion:

The per capita violent crimes variable was calculated using population and the sum of crime variables considered

violent crimes in the United States: murder, rape, robbery, and assault. There was apparently some controversy in some states concerning the counting of rapes. These resulted in missing values for rape, which resulted in missing values for per capita violent crime. Many of these omitted communities were from the midwestern USA (Minnesota, Illinois, and Michigan have many of these).

The per capita nonviolent crime variable was calculated using the sum of crime variables considered non-violent crimes in the United States: burglaries, larcenies, auto thefts and arsons. (There are many other types of crimes, these only include FBI 'Index Crimes').

Looking to the dataset, other suspecious values equal to 0.00 have been found:

```
zero_count <- function(x) sum(x==0, na.rm = TRUE)
zerodf <- lapply(crimedata, zero_count)[lapply(crimedata, zero_count)!=0]
names(zerodf)
```

```
##  [1] "racepctblack"        "numbUrban"           "pctUrban"
##  [4] "pctWFarmSelf"        "blackPerCap"         "indianPerCap"
##  [7] "AsianPerCap"         "OtherPerCap"         "HispPerCap"
## [10] "NumKidsBornNeverMar" "PctKidsBornNeverMar" "PctImmigRecent"
## [13] "PctImmigRec5"        "PctImmigRec8"        "PctImmigRec10"
## [16] "PctRecentImmig"      "PctRecImmig5"        "PctRecImmig8"
## [19] "PctRecImmig10"       "PctNotSpeakEnglWell" "PctVacantBoarded"
## [22] "PctHousNoPhone"      "PctWOFullPlumb"      "OwnOccQrange"
## [25] "RentQrange"          "NumInShelters"       "NumStreet"
## [28] "PctPolicBlack"       "PctPolicHisp"        "PctPolicAsian"
## [31] "PctPolicMinor"       "OfficAssgnDrugUnits" "PolicAveOTWorked"
## [34] "PctUsePubTrans"      "LemasGangUnitDeploy" "LemasPctOfficDrugUn"
## [37] "murders"             "murdPerPop"          "rapes"
## [40] "rapesPerPop"         "robberies"           "robbbPerPop"
## [43] "assaults"            "assaultPerPop"       "arsons"
## [46] "arsonsPerPop"        "ViolentCrimesPerPop"
```

Although it makes sense for some of these variables to take zero values, for others it seems unrealistic. However, we do not have any information about it in the documentation.

Therefore, columns with more than 50% NA are dropped and so are the variables fold (not useful), OwnOccQrange and RentQrange (obtained as the difference of other columns). We decide to keep the other columns with zero values due to their meaningfulness.

```
# find columns with > 50% NA
coltodrop <- as.vector(nadf$column_name[nadf$ratio_of_NA>0.50])

# add other columns to remove
coltodrop <- c(coltodrop, "fold", "OwnOccQrange", "RentQrange")
coltodrop
```

```
##  [1] "countyCode"           "communityCode"        "LemasSwornFT"
##  [4] "LemasSwFTPerPop"      "LemasSwFTFieldOps"    "LemasSwFTFieldPerPop"
##  [7] "LemasTotalReq"        "LemasTotReqPerPop"    "PolicReqPerOffic"
## [10] "PolicPerPop"          "RacialMatchCommPol"   "PctPolicWhite"
## [13] "PctPolicBlack"        "PctPolicHisp"         "PctPolicAsian"
## [16] "PctPolicMinor"        "OfficAssgnDrugUnits"  "NumKindsDrugsSeiz"
## [19] "PolicAveOTWorked"     "PolicCars"            "PolicOperBudg"
## [22] "LemasPctPolicOnPatr"  "LemasGangUnitDeploy"  "PolicBudgPerPop"
## [25] "fold"                 "OwnOccQrange"         "RentQrange"
```

```
# drop columns with > 50% NA and clearly redundant ones
cleandf <- crimedata[,!(names(crimedata) %in% coltodrop)]
```

Possible ways to handle missing values

The remaining variables with missing values are the ones related to the crimes and another one, the variable OtherPerCap (per capita income for people with 'other' heritage), which has only one missing value.

```
# remaining columns with NA
nadf <- inspect.na(cleandf, hist=FALSE, summary=TRUE, byrow=FALSE, barplot=FALSE,
                   na.value = NA)
nacolnames <- as.vector(nadf$column_name)
nacolnames
```

```
##  [1] "OtherPerCap"         "rapes"               "rapesPerPop"
##  [4] "robberies"           "robbbPerPop"         "assaults"
##  [7] "assaultPerPop"       "burglaries"          "burglPerPop"
## [10] "larcenies"           "larcPerPop"          "autoTheft"
## [13] "autoTheftPerPop"     "arsons"              "arsonsPerPop"
## [16] "ViolentCrimesPerPop" "nonViolPerPop"
```

Possible ways to handle the remaining missing values:

1. drop all raws with at least one missing value;
2. substitute a missing value with the average computed over the state a community belongs;
3. leave NA value and consider it as another category.

```
# rows with NA
narows <- inspect.na(cleandf, hist=FALSE, summary=TRUE, byrow=TRUE, barplot=FALSE,
                     na.value = NA)
dim(narows)[1]
```

```
## [1] 314
```

Since there are a lot of rows that have at least one missing values, we decide to proceed as in point 2.

```
# dataframe of the columns which still contain NA with the mean computed over the state
meandf <- aggregate(cleandf[,nacolnames], list(cleandf$state),
                    function(x) mean(x, na.rm = TRUE))
sum(is.na(meandf))
```

```
## [1] 12
```

The dataframe with the mean contains NA values, this means that there are States for which the value of a certain feature is zero for all communities that belong to them. This happens for IL, MI (no data of rapes, rapesPerPop, ViolentCrimesPerPop), KS, VT (no data of arsons, arsonsPerPop, nonViolPerPop). Moreover from documentation we know that also MN has a lot of missing values for rapes (59 out of 66):

```
mnNA = sum(is.na(cleandf[cleandf$state=="MN","rapes"]))
mnTot = length(cleandf[cleandf$state=="MN","rapes"])
round(c(mnNA, mnTot, mnNA/mnTot), 2)
```

```
## [1] 59.00 66.00  0.89
```

Therefore, data related to those states is removed.

```
rowtodrop <- as.numeric(rownames(cleandf[cleandf$state %in% c("IL","MI","MN","KS","VT"),]))
cleandf <- cleandf[!(rownames(cleandf) %in% rowtodrop),]
```

For the remaining variables with missing values, we substitute NA with the mean computed over the state.

```
# substitute a missing value with the average computed over the state
for(col in nacolnames) {
  match_table <- tapply(cleandf[[col]], cleandf$state, mean, na.rm=TRUE)
  NA_position <- which(is.na(cleandf[[col]]))
```

```
  cleandf[[col]][NA_position] <- match_table[cleandf$state[NA_position]]
}
```

Now cleandf does not contain NA.

```
sum(is.na(cleandf))
```

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

```
dim(cleandf)
```

```
## [1] 1996  120
```

The resulting dataframe consists of 1996 instances and 120 attributes.

Standardization

To avoid any bias due to the difference of the predictors content, the dataset must be somehow scaled. Different methods have been tested because of the presence of several outliers, such as the classical mean-standard deviation stardardization and the min-max normalization. In the latter case most of the information dropped lost for the previously mentioned thickness of the tails of the columns densities, while we'll see that the former one, together with a logarithmic transformation, gives far better results.

It is interesting to notice an additional detail: many columns of the dataset describe the same quantity for different categories. An attempt has been to jointly standardize such groups of variables, considering common values for the mean and the standard deviation, but the following results where basically equivalent to the first ones, so we decided not to overcomplicate the analysis.

```
standardization <- function(x) {
return ((x - mean(x)) / sd(x))
}

standf <- cleandf
standf[seq(3,dim(standf)[2])] <- lapply(standf[seq(3,dim(standf)[2])], standardization)
```

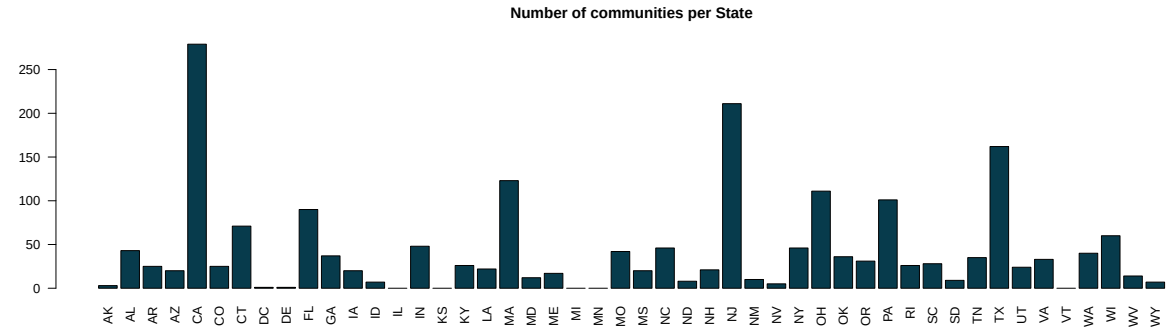
Exploratory Data Analysis

First of all, let us check how many communities are present in the dataset for each state.

```
sort(table(cleandf$state))
```

```
##
##  IL  KS  MI  MN  VT  DC  DE  AK  NV  ID  WY  ND  SD  NM  MD  WV  ME  AZ
##   0   0   0   0   0   1   1   3   5   7   7   8   9  10  12  14  17  20
##  IA  MS  NH  LA  UT  AR  CO  KY  RI  SC  OR  VA  TN  OK  GA  WA  MO  AL
##  20  20  21  22  24  25  25  26  26  28  31  33  35  36  37  40  42  43
##  NC  NY  IN  WI  CT  FL  PA  OH  MA  TX  NJ  CA
##  46  46  48  60  71  90 101 111 123 162 211 279
```

```
barplot(table(cleandf$state), las=2, col=Col[5], main = "Number of communities per State")
```



There are states represented by a single community and states with more than 200 observations available. The data may therefore seem unbalanced, but it must be taken into account that states have different geographical dimensions and each one consists of a different total number of communities.

Violent Crimes

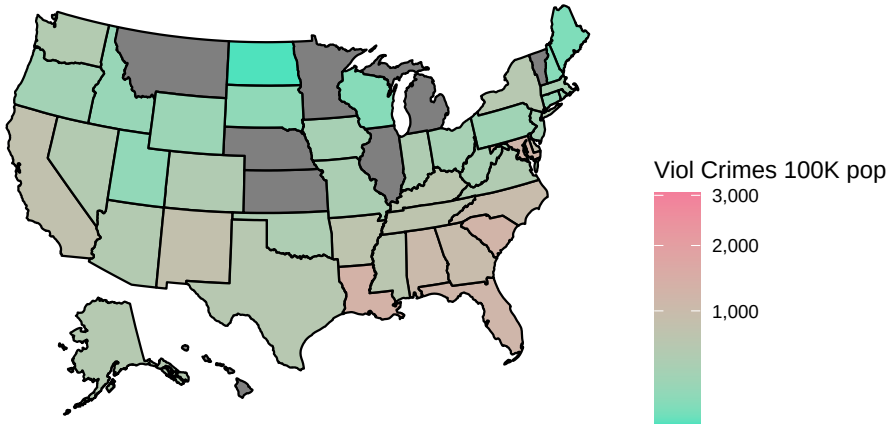
We begin our exploratory analysis by observing how the rate of violent crimes is distributed on average in the United States, based on the data we have available.

```
# Violent Crimes per 100K Population - State aggregation - Mean
vc_state <- aggregate(cleandf[, "ViolentCrimesPerPop"], list(cleandf$state), mean)
names(vc_state)[names(vc_state) == "Group.1"] <- "state"
names(vc_state)[names(vc_state) == "x"] <- "ViolentCrimesMean"
```

```
summary(vc_state$ViolentCrimesMean)
```

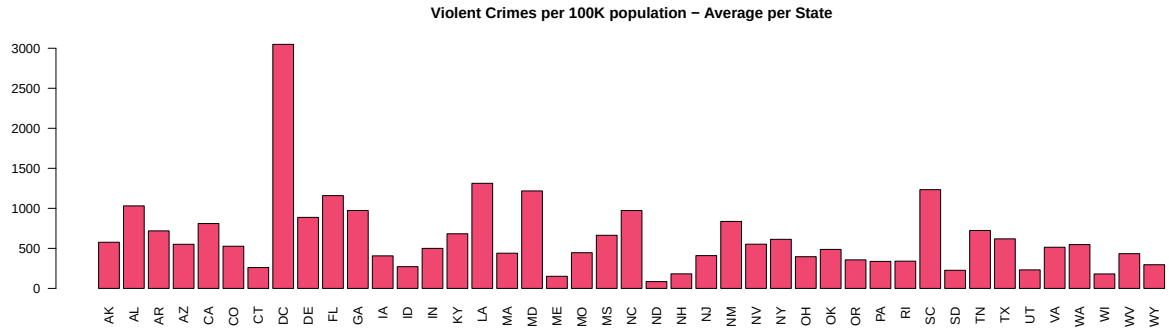
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   85.06  348.43  526.86  632.79  767.12 3048.38
```

```
par(mfrow=c(1,1))
plot_usmap(data = vc_state, values = "ViolentCrimesMean", color = "black") +
   scale_fill_continuous(trans="sqrt", low = Rgb[3], high = Rgb[1],
                         name = "Viol Crimes 100K pop", limits=c(85,3049),
                         label=scales::comma) +
  theme(legend.position = "right")
```



We can notice that the states with the highest average violent crimes rate are mostly located in the southeast. Moreover, the summary and the geographical plot suggest the presence of some outlier in the averages.

```
barplot(vc_state$ViolentCrimesMean, names.arg = vc_state$state, las=2, col=Col[1],
        main = "Violent Crimes per 100K population - Average per State")
```




```
vc_state[vc_state$ViolentCrimesMean == max(vc_state$ViolentCrimesMean),]
```

```
##   state ViolentCrimesMean
## 8    DC           3048.38
```

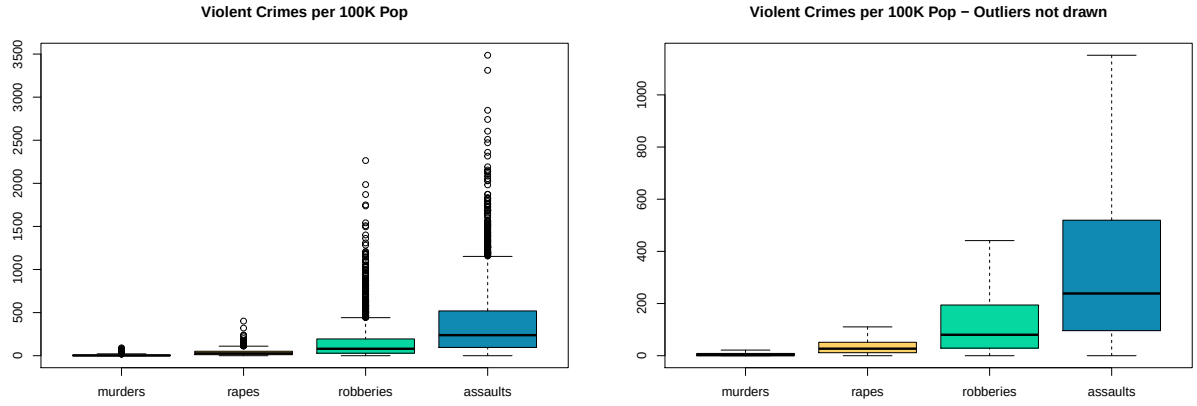
It is evident that the value associated to Washington D.C. is much larger than the others. However, this huge discrepancy may be due to the fact that we are dealing with the capital of the US, then it is interesting to maintain this value in the dataset.

Let us now look to the violent crimes independently. We recall that the crimes considered to be violent in the United States are murder, rape, robbery and assault.

```
violent_names <- c("murders", "rapes", "robberies", "assaults")
violPerPop_names <- c("murdPerPop", "rapesPerPop", "robbbPerPop", "assaultPerPop")

violent <- cleandf[, (colnames(cleandf) %in% violent_names)]
violPerPop <- cleandf[, (colnames(cleandf) %in% violPerPop_names)]
```

```
par(mfrow = c(1,2))
boxplot(violPerPop, outline = TRUE, col = Col[seq(1:4)],
        main = "Violent Crimes per 100K Pop",
        names = c("murders", "rapes", "robberies", "assaults"))
```

```
boxplot(violPerPop, outline = FALSE, col = Col[seq(1:4)],
        main = "Violent Crimes per 100K Pop - Outliers not drawn",
        names = c("murders", "rapes", "robberies", "assaults"))
```




```
par(mfrow=c(1,1))
```

From the boxplot it can be noticed that the most frequent violent crime is assault, as expected murders and rapes rapresent a minority.

Non Violent Crimes

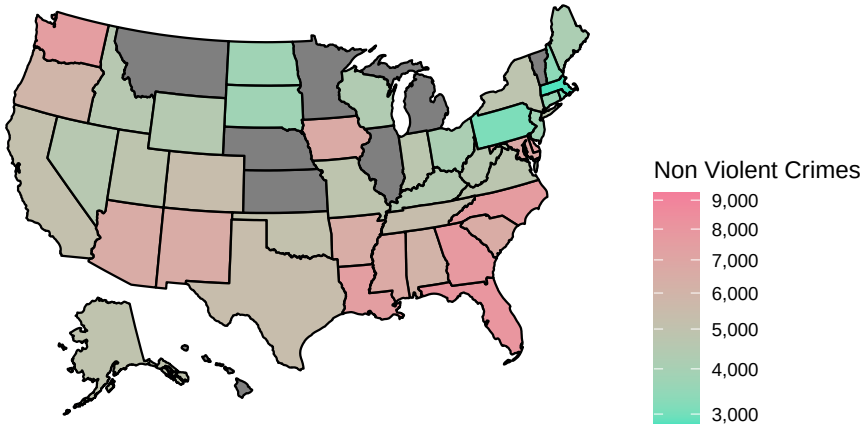
As before, we now investigate how the rate of non violent crimes is distributed on average in the United States, based on the data we have available.

```
# Non Violent Crimes per 100K Population - State aggregation - Mean
nvc_state <- aggregate(cleandf[, "nonViolPerPop"], list(cleandf$state), mean)
names(nvc_state)[names(nvc_state) == "Group.1"] <- "state"
names(nvc_state)[names(nvc_state) == "x"] <- "NonViolentCrimesMean"
```

```
summary(nvc_state$NonViolentCrimesMean)
```

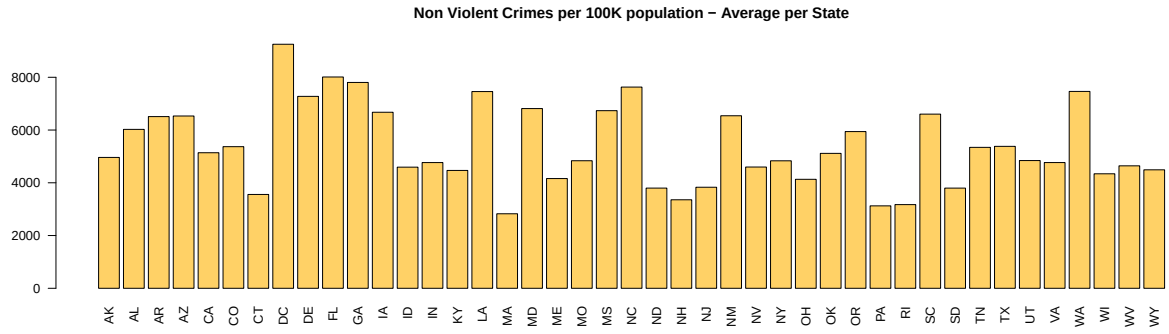
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2825    4405    4962    5384    6572    9252
```

```
par(mfrow=c(1,1))
plot_usmap(data = nvc_state, values = "NonViolentCrimesMean", color = "black") +
   scale_fill_continuous(trans="sqrt",low = Rgb[3], high = Rgb[1],
                         name = "Non Violent Crimes", limits=c(2825,9252),
                         label=scales::comma) +
  theme(legend.position = "right")
```



We can see that the south-eastern states have the highest average rates again. However, the crime rate is also very high in other areas. As expected, the non violent crimes are more spread all over the US than the violent ones.

```
barplot(nvc_state$NonViolentCrimesMean, names.arg = nvc_state$state, las=2, col=Col[2],
        main = "Non Violent Crimes per 100K population - Average per State")
```

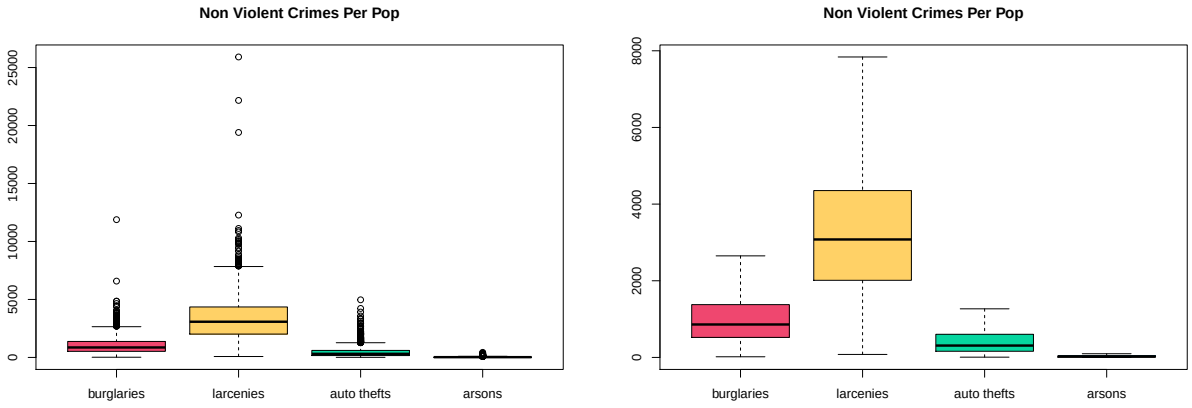


Also in this case, we can notice that the highest average value is associated to the Disctrict of Columbia. This leads us to believe that there may be a particular phenomenon that affects the crime rate or how it is recorded in the capital region.

The crimes considered as non-violent in the United States are burglaries, larcenies, auto thefts and arsons.

```
nonViol_names <- c("burglaries", "larcenies", "autoTheft", "arsons")
nonViolPerPop_names <- c("burglPerPop", "larcPerPop", "autoTheftPerPop", "arsonsPerPop")

nonViol <- cleandf[, (colnames(cleandf) %in% nonViol_names)]
nonViolPerPop <- cleandf[, (colnames(cleandf) %in% nonViolPerPop_names)]
```

```
par(mfrow = c(1,2))
boxplot(nonViolPerPop, outline = TRUE, col = Col[seq(1:4)],
        main="Non Violent Crimes Per Pop",
        names = c("burglaries", "larcenies", "auto thefts", "arsons"))
boxplot(nonViolPerPop, outline = FALSE, col = Col[seq(1:4)],
        main="Non Violent Crimes Per Pop",
        names = c("burglaries", "larcenies", "auto thefts", "arsons"))
```




```
par(mfrow=c(1,1))
```

Among the non violent crimes, in general the most frequent are larcenies.

Inspecting variables relation

Let us analyze which variables are more correlated (correlation>0.5) with the variable of violent crimes.

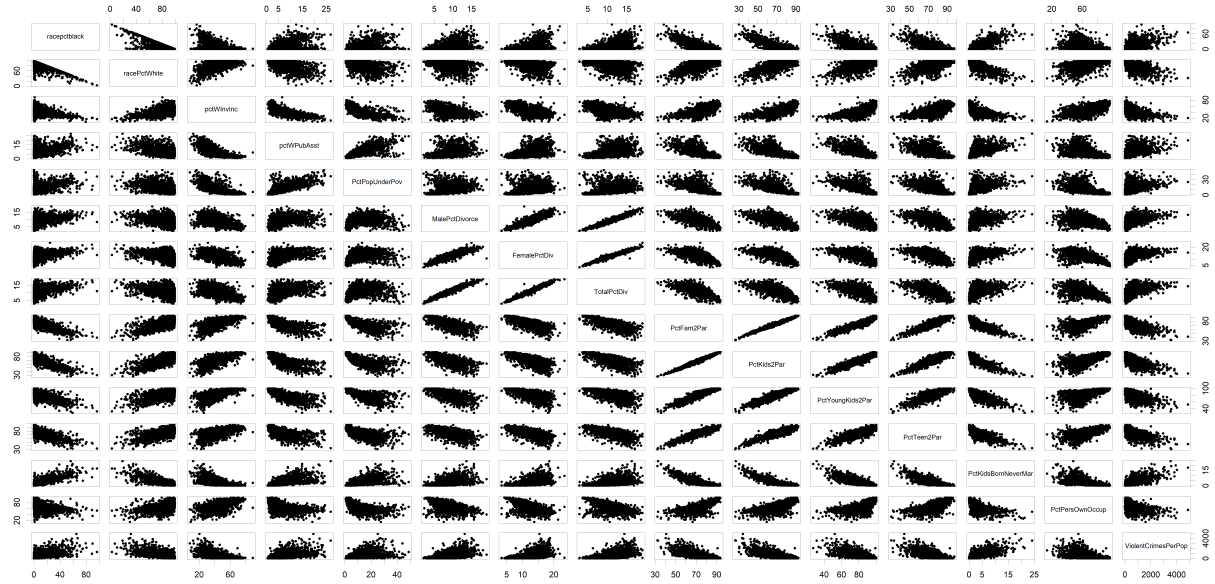
```
coltodrop <- c(1,2, seq(103,118), 120)
viol.corrdf <- cleandf[,-coltodrop]

# correlation matrix
viol.cm <- cor(viol.corrdf, use='complete.obs')

# violent crimes correlation
ViolCrimes.corr <- viol.cm[,dim(viol.cm)[2]]
ViolCrimes.names <- names((ViolCrimes.corr)>0.5)

sort(ViolCrimes.corr[abs(ViolCrimes.corr)>0.5])
```

```
##         PctKids2Par          PctFam2Par        racePctWhite
##           -0.7252333          -0.6963388          -0.6750131
##    PctYoungKids2Par         PctTeen2Par          pctWInvInc
##           -0.6560994          -0.6535612          -0.5570812
##     PctPersOwnOccup      PctPopUnderPov      MalePctDivorce
##           -0.5075462           0.5035348           0.5095005
##         TotalPctDiv        FemalePctDiv         pctWPubAsst
##           0.5355533           0.5362600           0.5600062
##        racepctblack PctKidsBornNeverMar ViolentCrimesPerPop
##           0.6260588           0.7344058           1.0000000
```

It seems that the variables more correlated with violent crimes are also correlated with each other. For instance, the attributes PctFam2Par, PctYoungKids2Par, PctKids2Par, PctTeen2Par all refer to the family structure, in particular to the percentage of families (with kids with different age) that are headed by two parents. Probably, also the variables TotalPctDiv, FemalePctDiv and MalePctDiv in addition to being trivially correlated with each other, are correlated with the previous ones. Moreover, we observe a positive correlation between the percentage of African-American in the population and violent crimes; on the other side, there is a negative correlation with the percentage of Caucasian-American. In general we can say that the family structure, the marital status, the economic status (pctWInvInc, PctPopUnderPov, PctWPubAsst) and the ethnicity (racepctWhite, racepctblack) are 4 main clusters stricly correlated with Violent Crimes.

Let us investigate if and how these clusters are cross-correlated through a scatterplot.

```
pairs(cleandf[,colnames(cleandf)%in% ViolCrimes.names], pch = 20)
```



The plot confirms what already said above: the variables related to the family structure pairly show an evident linear dependence, the same holds for the variables related to the marital status.

Moreover, we can observe that there exists also a relationship between ethnicity and family structure:

- there is a negative correlation between the percentage of African-American and the percentage of families (with kids differently aged) headed by two parents;
- there is a strong positive correlation between the percentage of African-American and the percentage of kids born from never married parents.

In addition, the plot shows a negative correlation between the percentages of Caucasian and African; therefore, all the previuos considerations should be done conversely for the percentage of population that is Caucasian.

This is confirmed by the correlation values shown below:

```
african.corr <- viol.cm[,3]
sort(african.corr[abs(african.corr)>0.5])
```

```
##        racePctWhite         PctKids2Par          PctFam2Par
##           -0.8037199          -0.7320774          -0.7006218
##         PctTeen2Par    PctYoungKids2Par ViolentCrimesPerPop
##           -0.6918957          -0.6573207           0.6260588
## PctKidsBornNeverMar        racepctblack
##            0.8053995           1.0000000
```

```
caucasian.corr <- viol.cm[,4]
sort(caucasian.corr[abs(caucasian.corr)>0.5])
```

```
##        racepctblack PctKidsBornNeverMar ViolentCrimesPerPop
##           -0.8037199          -0.7991972          -0.6750131
##    PctPersDenseHous         pctWPubAsst     PctLargHouseFam
##           -0.5917818          -0.5852270          -0.5432253
##      PctPopUnderPov       PctUnemployed     PctPersOwnOccup
##           -0.5332129          -0.5188358           0.5055375
```

```
##          pctWInvInc    PctYoungKids2Par         PctTeen2Par
##           0.5938626           0.6032863           0.6187954
##         PctFam2Par         PctKids2Par        racePctWhite
##           0.6429738           0.7031175           1.0000000
```

The above considerations could lead us to think that in general African American families have a more fragile structure that could be a risk factor for the exposure of young people to illegality and violence, or, on the other hand, they could highlight the presence of the well known, but always tricky, confounding effect, about which we'll go into details further. Along with that, another fundamental factor has to be considered: a pretty famous social phenomenon, known as the *crack epidemic*, took place in the USA during the past 80s and 90s, period in which our data have been gathered. Shortly speaking, faced with dropping prices of cocaine powder, drug dealers made a decision to convert it to "crack", a solid smokeable form of cocaine, that could be sold in smaller quantities, to more people. It was cheap, simple to produce, ready to use, and highly profitable for dealers to develop, which made the new product spread rapidly and made the homicide rate for young black males more than doubled, due to the fact that the distribution of the drug to the end-user occurred mainly in low-income inner city neighborhoods: this gave many inner-city residents the opportunity to move up the "economic ladder" in a drug market, at the cost of violence to establish territorial control. Moreover, this habits caused a long term effect in the society, estending the phenomenon over its actual time period[2,3,4,5].

We carry now the same analysis in order to understand which variables are more correlated (correlation>0.5) with the variable of non violent crimes.

```
coltodrop <- c(1,2, seq(103,119))
nonViol.corrdf <- cleandf[,-coltodrop]

# correlation matrix
nonViol.cm <- cor(nonViol.corrdf, use='complete.obs')

# violent crimes correlation
nonViol.corr <- nonViol.cm[,dim(nonViol.cm)[2]]

sort(nonViol.corr[abs(nonViol.corr)>0.5])
```

```
##         PctKids2Par          PctFam2Par         PctTeen2Par
##           -0.6490054          -0.6420788          -0.6033113
##    PctYoungKids2Par      PctPopUnderPov PctKidsBornNeverMar
##           -0.5959762           0.5001029           0.5338598
##      MalePctDivorce        FemalePctDiv         TotalPctDiv
##           0.5750248           0.5883801           0.5956666
##       nonViolPerPop
##           1.0000000
```

We can observe that the most correlated variables are still those related to the family structure, the marital status and poverty.

Multiple Linear Regression

For almost the entire project, we decided to focus the analyzes on the ViolentCrimePerPop variable only, i.e. the rate of the sum of the violent crimes, according to the US laws, per population. In particular our main task is to predict this response from the socio-economic and demographic attributes per community provided by the dataset using a multiple linear regression model.

For this purpose, we firstly remove non predictive variables, such as the community name and all the other crimes columns except for the ViolentCrimesPerPop response. Note that in this first study we also discarded the categorical attribute state, assuming it as not influential.

```
coltodrop <- c(1,2, seq(103,120)[-17]) # -17 keeps ViolentCrimesPerPop
df <- standf[,-coltodrop]
```

We then fit our multiple linear regression model and show statistics and plot about the residuals.

```
reg.out <- lm(ViolentCrimesPerPop~., data=df)
```

```
se  <- summary(reg.out)$sigma                  # se
rsq <- summary(reg.out)$r.squared              # R^2
adrsq <- summary(reg.out)$adj.r.squared        # adj R^2
cat("RSE:", round(se,2), "\n")
```

```
## RSE: 0.59
```

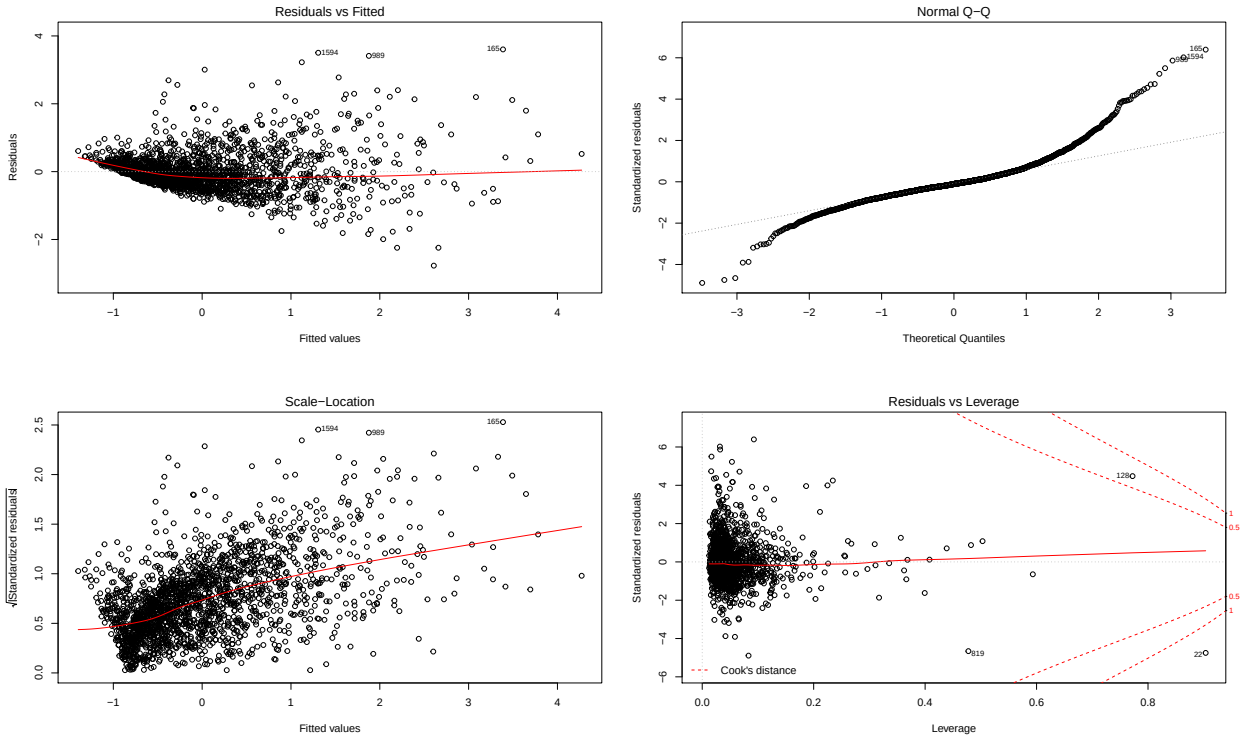
```
cat("R^2:", round(rsq,2), "\n")
```

```
## R^2: 0.67
```

```
cat("adjusted R^2:", round(adrsq,2))
```

```
## adjusted R^2: 0.65
```

```
par(mfrow=c(2,2))
plot(reg.out)
```




```
par(mfrow=c(1,1))
```

To analyze in detail the results obtained from this first simple model, we recall the necessary theoretical assumptions that must be verified.

Assessing Model Assumptions:

The model assumptions are:

- Linearity of the response-predictor relationships;
- Homeschedasticity of the error terms: $\text{Var}(\epsilon_i) = \sigma^2$.

To check the conditions listed above we use the residual plots provided by lm.

Linearity

The plot of residuals versus fitted values shows a little pattern, however it seems to indicate that there are linear associations in the data and that the errors are uncorrelated.

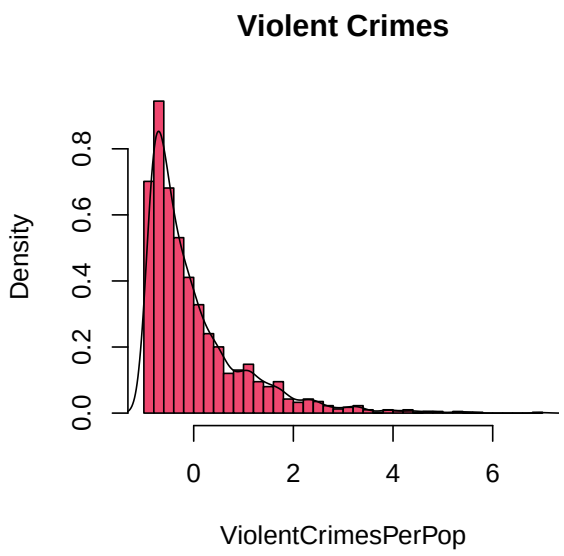
Homoschedasticity

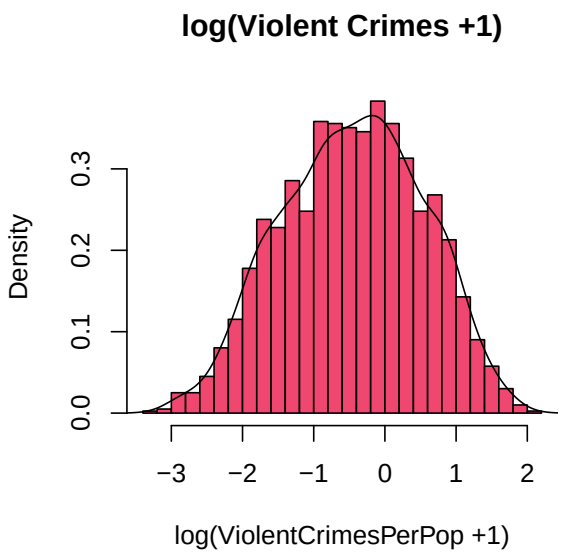
The presence of a funnel shape in the residual plot suggests that the error terms do not have a constant variance. One possible solution is to transform the response variable Y using a concave function such as $\log(Y)$ or $\sqrt{Y}$.

With a trial and error approach we came out with the best transformation: $\log(Y) + 1$, of which the resulting density is shown the following figures.

```
par(mfrow=c(1,2))
hist(df$ViolentCrimesPerPop, prob=TRUE, breaks=30, col=Col[1], main="Violent Crimes",
     xlab = "ViolentCrimesPerPop")
lines(density(df$ViolentCrimesPerPop))
```

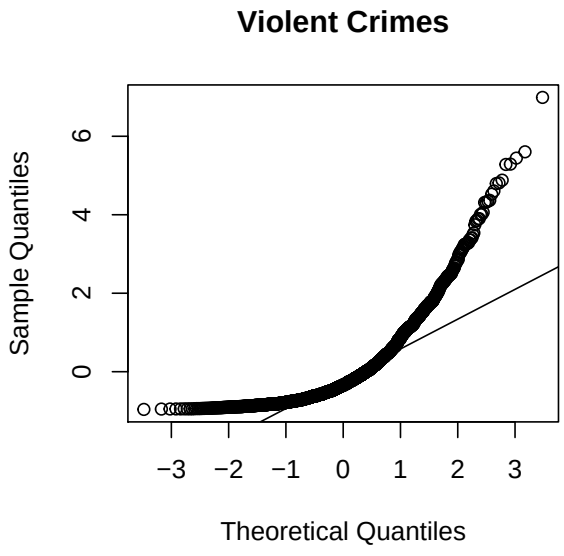
```r
hist(log(df$ViolentCrimesPerPop +1), prob=TRUE, breaks=30, col=Col[1],
     main="log(Violent Crimes +1)", xlab = "log(ViolentCrimesPerPop +1)")
lines(density(log(df$ViolentCrimesPerPop +1)))
```

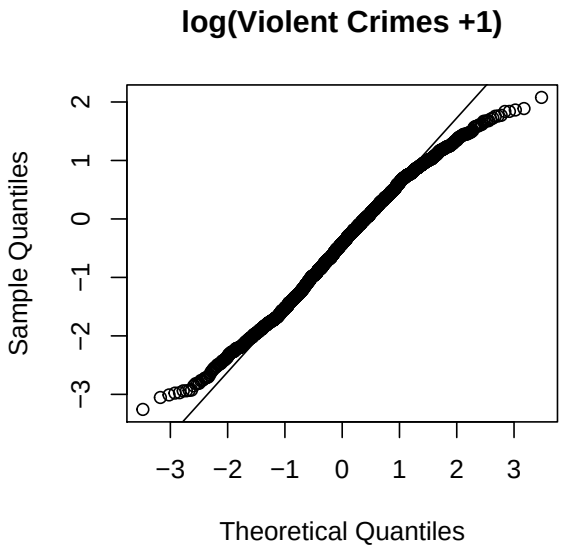





```r
par(mfrow=c(1,1))
```

```r
par(mfrow=c(1,2))
qqnorm(df$ViolentCrimesPerPop, main="Violent Crimes")
qqline(df$ViolentCrimesPerPop)

qqnorm(log(df$ViolentCrimesPerPop +1), main="log(Violent Crimes +1)")
qqline(log(df$ViolentCrimesPerPop +1))
```





```
par(mfrow=c(1,1))
```

At this point we fitted again the linear model using the transformed response.

```
reg.out2 <- lm(log(ViolentCrimesPerPop+1)~., data=df)
```

```
se  <- summary(reg.out2)$sigma                  # se
rsq <- summary(reg.out2)$r.squared              # R^2
adrsq <- summary(reg.out2)$adj.r.squared        # adj R^2
cat("RSE:", round(se,2), "\n")
```

```
## RSE: 0.56
```

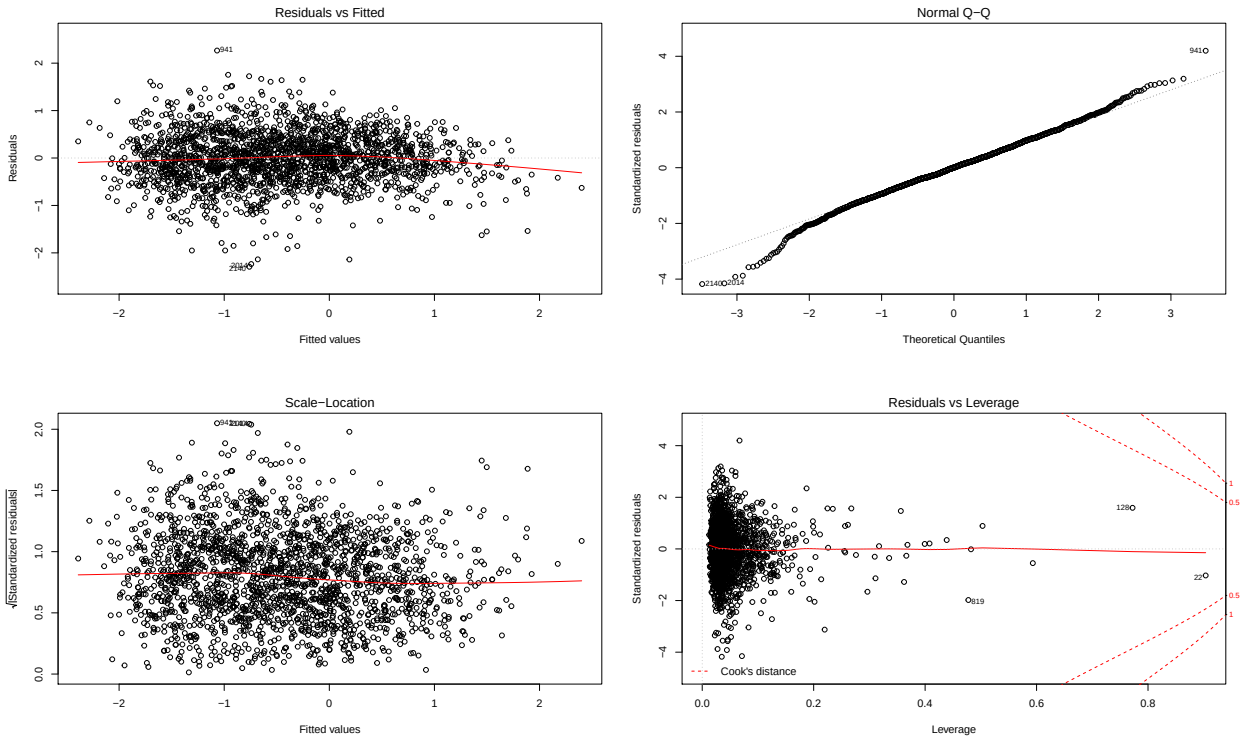
```
cat("R^2:", round(rsq,2), "\n")
```

```
## R^2: 0.7
```

```
cat("adjusted R^2:", round(adrsq,2))
```

```
## adjusted R^2: 0.68
```

```
par(mfrow=c(2,2))
plot(reg.out2)
```




```
par(mfrow=c(1,1))
```

The plots of residuals versus fitted values shows that such a transformation leads to a reduction in heteroscedasticity, moreover it managed to considerably improve the statistic measures.

Other Possible Problems

Outliers

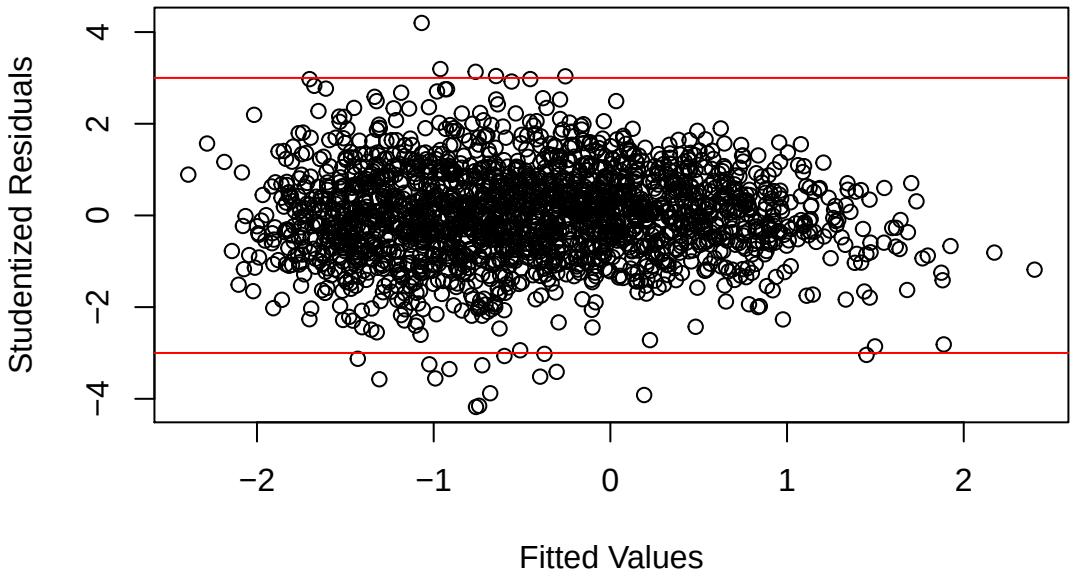
The residual plot identifies some outliers. However, it can be difficult to decide how large a residual needs to be before we consider the point to be an outlier. To address this problem, instead of plotting the residuals, we can plot the studentized residuals, computed by dividing each residual e_i by its estimated standard error. Observations whose studentized residuals are greater than 3 in absolute value are possible outliers.

Note that the empirical motivation for the value equal to 3 is that the Studentized Residuals are approximated by a $N(0,1)$. The probability to observe a value greater than 3 is then 0.001349898[1].

```
1 - pnorm(3)
```

```
## [1] 0.001349898
```

```
plot(predict(reg.out2), rstandard(reg.out2), xlab="Fitted Values", ylab = "Studentized Residuals")

abline(h=3, col = "red")
abline(h=-3, col = "red")
```




```
out <- names(rstandard(reg.out2)[(abs(rstandard(reg.out2)) > 3)])

standf[out,c(1,2)]
```

```
##               communityname state
## 260          Greenfieldtown    MA
## 386             Harvardtown    MA
## 444                Troycity    AL
## 821           Greenwoodcity    MS
## 822    ClevelandHeightscity    OH
## 941  LaCanadaFlintridgecity    CA
```

```
## 961           Hartsellecity    AL
## 991         Moundsvillecity    WV
## 994     SaratogaSpringscity    NY
## 1197            Norwalkcity    OH
## 1223             Warrentown    RI
## 1397           Spencercity    IA
## 1431          Marshallcity    MO
## 1489          Saralandcity    AL
## 1886           Milfordtown    MA
## 2009            Vernoncity    TX
## 2014         Ogdensburgcity    NY
## 2024        Arkadelphiacity    AR
## 2140            Oswegocity    NY
## 2177          Greenwoodcity    IN
```

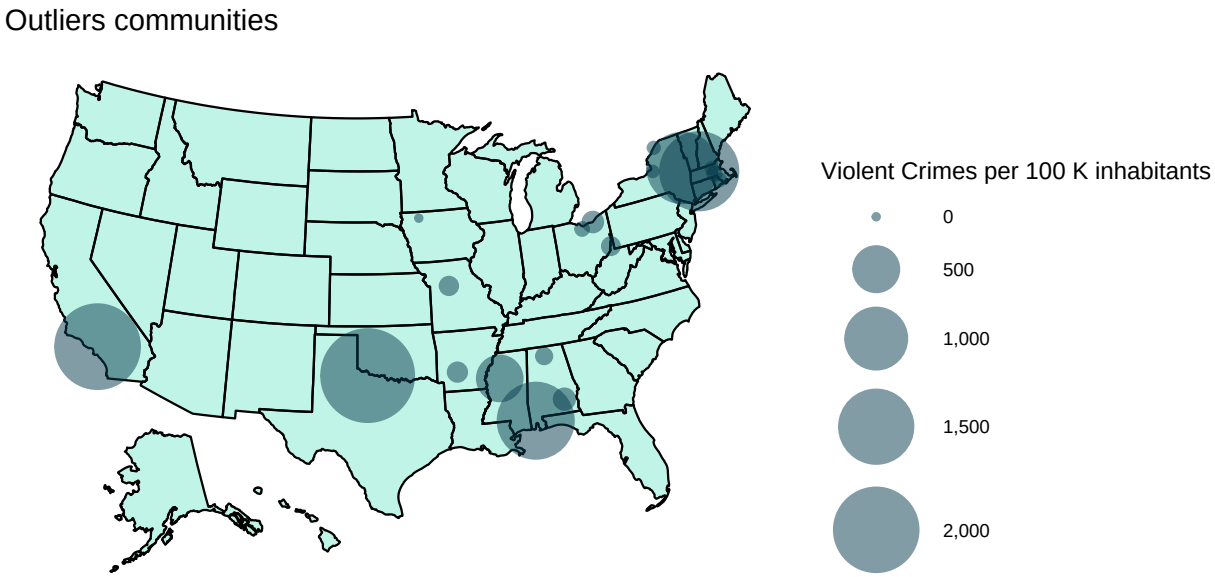
```
cityout<-standf$communityname[rownames(standf) %in% out]
```

```
citycord <- read.csv("../data/cities.csv")

# Rename column
colnames(citycord)[colnames(citycord) == "city"] <- "communityname"
colnames(citycord)[colnames(citycord) == "latitude"] <- "lat"
colnames(citycord)[colnames(citycord) == "longitude"] <- "lon"

citycord<-merge(citycord,cleandf,by=c("state","communityname"))
```

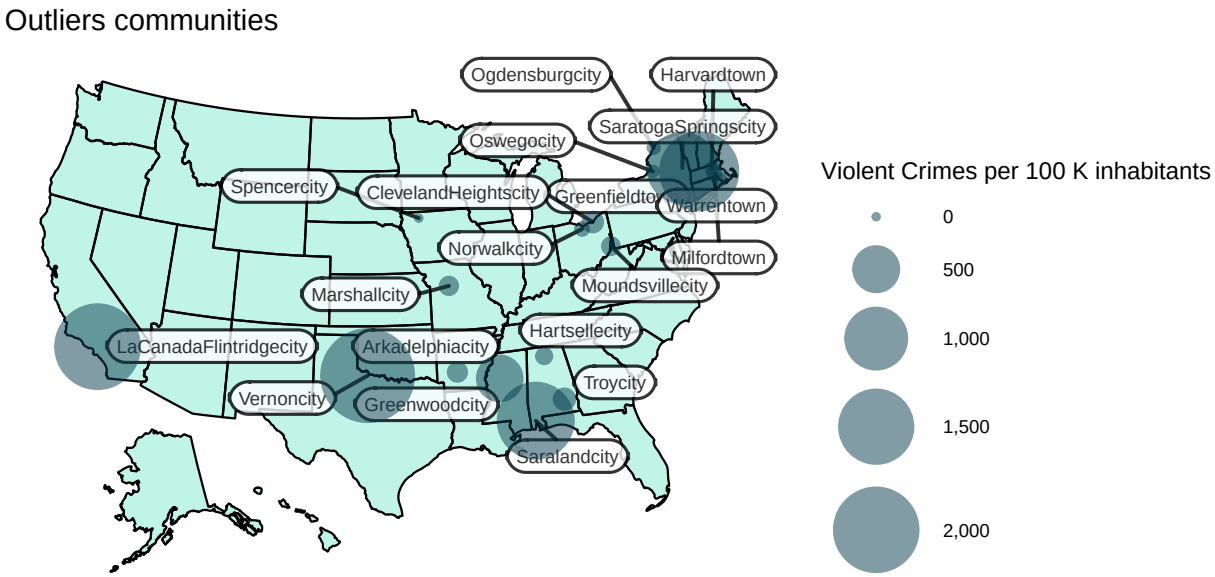
```
dfplot <- citycord[citycord$communityname %in% cityout, c("lon","lat","ViolentCrimesPerPop","communityna
```

Using this selection criterion, 20 communities are to be considered outliers. In a study like the one we are conducting, in which socio-economic, environmental and demographic information is fundamental, it might be interesting to investigate where these cities are located geographically. The following map shows the outliers communities with respect the ViolentCrimesPerPop variable. It can be noticed that the large majiority are clusterized in the state of NY and in the southern regions.

```
cities_t <- usmap_transform(dfplot)

#without city labels
plot_usmap(fill = "#06D6A0", alpha = 0.25) +
  geom_point(data = cities_t,
             aes(x = lon.1, y = lat.1, size = ViolentCrimesPerPop),
             color = "#073B4C", alpha = 0.5) +
  scale_size_continuous(range = c(1, 16),
                        label = scales::comma) +
  labs(title = "Outliers communities",
       size = "Violent Crimes per 100 K inhabitants") +
  theme(legend.position = "right")
```




```
#with city labels
plot_usmap(fill = "#06D6A0", alpha = 0.25) +
  ggrepel::geom_label_repel(data = cities_t,
             aes(x = lon.1, y = lat.1, label = communityname),
             size = 2.5, alpha = 0.8,
             label.r = unit(0.5, "lines"), label.size = 0.55,
             segment.color = "black", segment.size = 0.7,
             seed = 1002) +
  geom_point(data = cities_t,
             aes(x = lon.1, y = lat.1, size = ViolentCrimesPerPop),
             color = "#073B4C", alpha = 0.5) +
  scale_size_continuous(range = c(1, 16),
                        label = scales::comma) +
  labs(title = "Outliers communities",
       size = "Violent Crimes per 100 K inhabitants") +
  theme(legend.position = "right")
```



The results of the linear regressor fitted without the outliers are shown below.

```
# regression without outliers
reg.out3 <- lm(log(ViolentCrimesPerPop+1)~., data=df[!(rownames(df) %in% out),])
```

```
se  <- summary(reg.out3)$sigma                  # se
rsq <- summary(reg.out3)$r.squared              # R^2
adrsq <- summary(reg.out3)$adj.r.squared        # adj R^2
cat("RSE:", round(se,2), "\n")
```

```
## RSE: 0.52
```

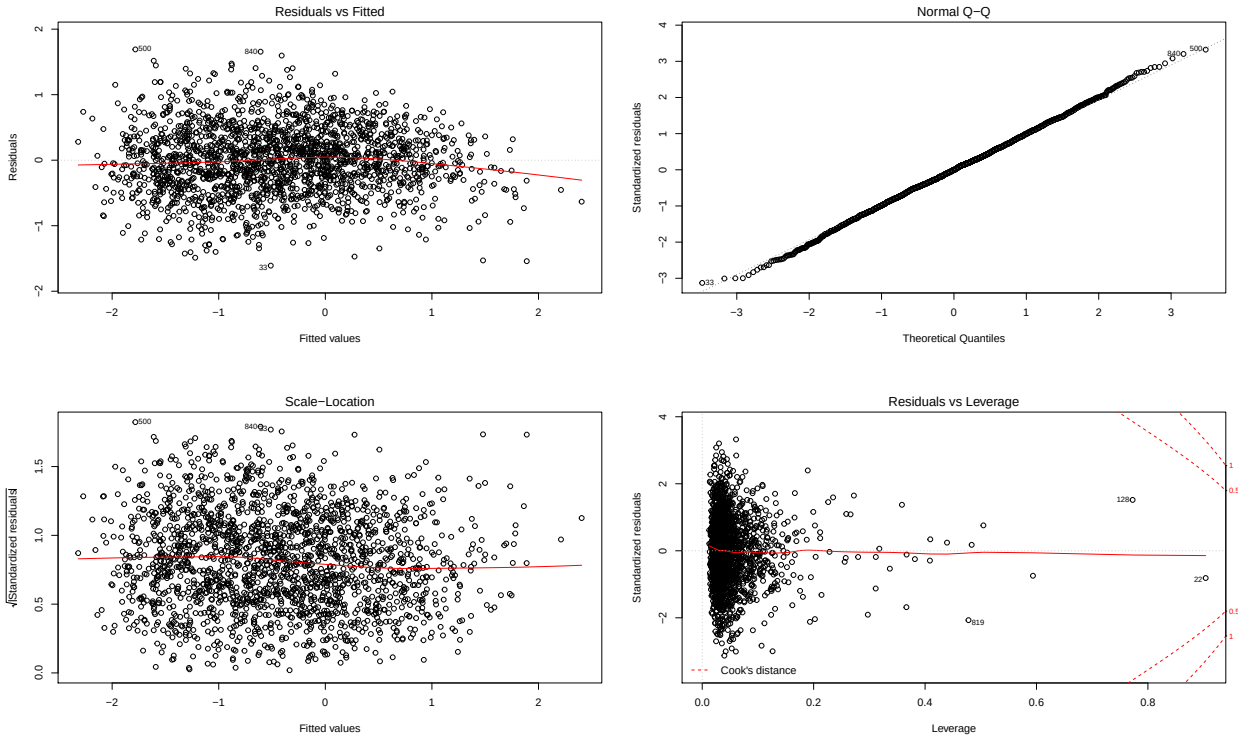
```
cat("R^2:", round(rsq,2), "\n")
```

```
## R^2: 0.72
```

```
cat("adjusted R^2:", round(adrsq,2))
```

```
## adjusted R^2: 0.71
```

```
par(mfrow=c(2,2))
plot(reg.out3)
```



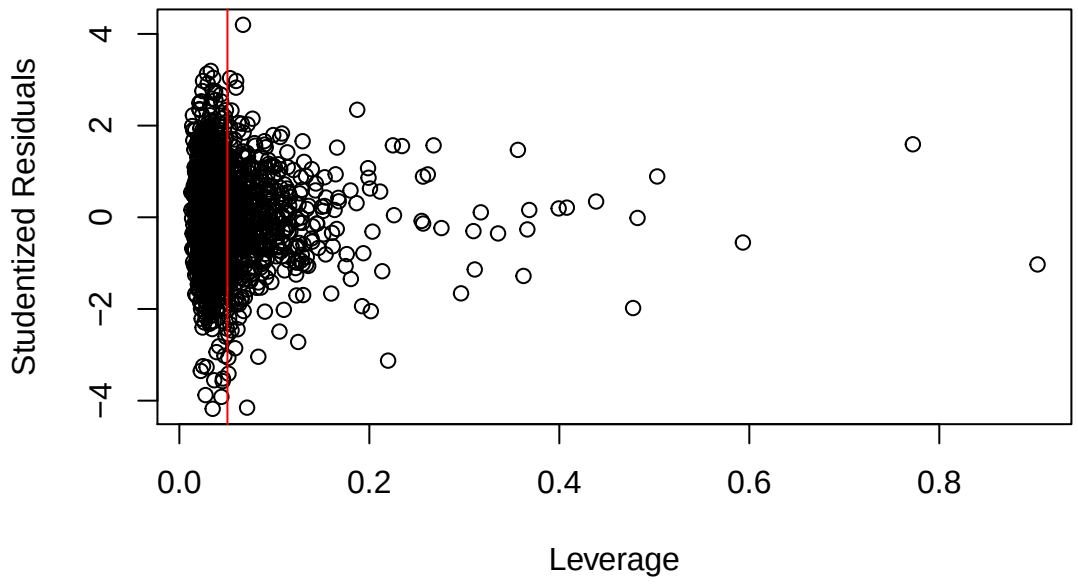

```
par(mfrow=c(1,1))
```

As expected, the RSE is smaller: 0.56 when the outliers are included versus 0.52 when they are removed. Nevertheless, since the RSE is used to compute all confidence intervals and p-values, this can have implications for the interpretation of the fit, as usual then, care should be taken in the decision of taking or dropping outliers. For the purpose of this work we have decided not to remove them, not only to avoid the risk of losing statistical information, but because they seem to have a geographical meaning and dropping them could make the study less consistent.

High Leverage Points

A second problem when dealing with regression is the presence of high leverage points. In order to quantify an observations leverage, we compute the leverage statistic. If a given observation has a leverage statistic that greatly exceeds $(p+1)/n$, then we may suspect that the corresponding point has high leverage.

```
hv <- hatvalues(reg.out2)
plot(hv, rstandard(reg.out2), xlab="Leverage", ylab = "Studentized Residuals")

p <- dim(df)[2]-1
n <- dim(df)[1]
abline(v=(p+1)/n, col = "red")
```



The previous figure shows that a lot of points fall to the right of the critical value, in order not to lose too much data we will consider high leverage only the points above $3(p+1)/n$ 8. Using this new rule it appears that 56 communities are to be considered high leverage, also in this case we decided to investigate the corresponding geographic locations through the map. In this particular case, they are drawn according to the total population.

```
lev <- names(hv[hv>3*(p+1)/n])
standf[lev,c(1,2)]
```

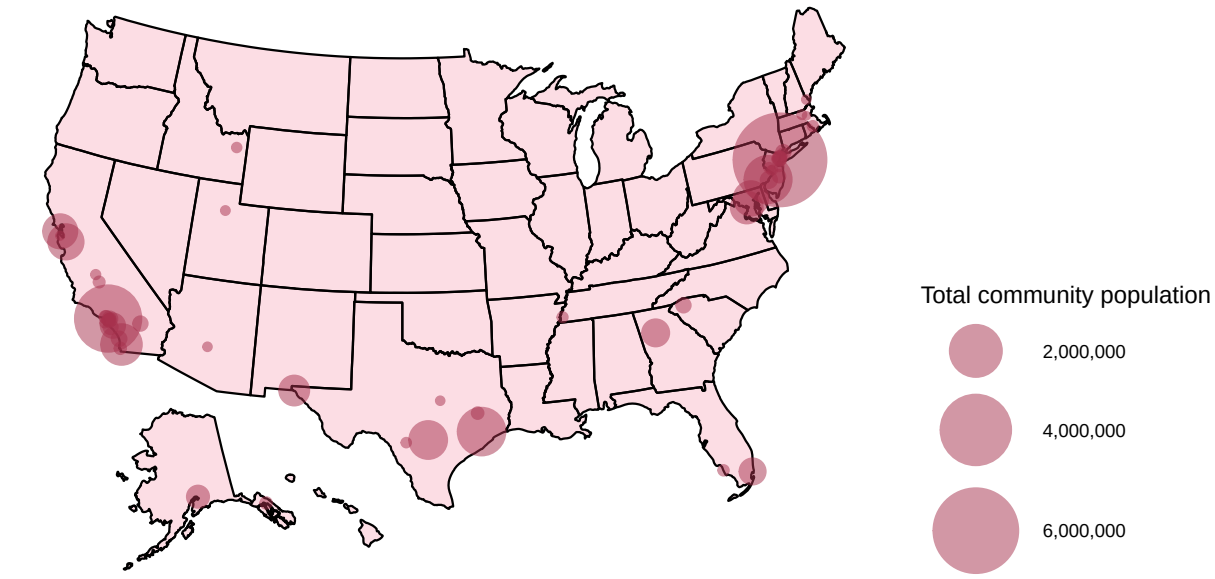
```
##                      communityname state
## 22                     NewYorkcity    NY
## 52                 BeverlyHillscity    CA
## 56                 Philadelphiacity    PA
## 94                  SouthJordancity    UT
## 115               Princetonborough    NJ
## 128                 LosAngelescity    CA
## 151                 UnionCitycity    NJ
## 210                  Coronadocity    CA
## 235                Raritantownship    NJ
## 331                 Greenvillecity    SC
## 370                 Greenwichtown    CT
## 386                   Harvardtown    MA
## 426                   Hobokencity    NJ
## 442            ParadiseValleytown    AZ
## 446                Holmdeltownship    NJ
## 455               SanFranciscocity    CA
## 594                 Huntsvillecity    TX
## 598                  SanMarinocity    CA
## 626         PalosVerdesEstatescity    CA
## 687                       ElPasocity    TX
```

```
## 707             WestHollywoodcity    CA
## 720                   Rexburgcity    ID
## 721                    Camdencity    NJ
## 749                   SanJosecity    CA
## 756                   Maywoodcity    CA
## 786              MontereyParkcity    CA
## 819                   Houstoncity    TX
## 831                Baltimorecity    MD
## 900                   Naplescity    FL
## 992                Anchoragecity    AK
## 1020                Somersettown    MA
## 1137                Corcorancity    CA
## 1144                  Uvaldecity    TX
## 1247                  Cudahycity    CA
## 1263                     Ryecity    NY
## 1331             BellGardenscity    CA
## 1348              FortLeeborough    NJ
## 1375                  Juneaucity    AK
## 1430                SanDiegocity    CA
## 1434              SanAntoniocity    TX
## 1479 TwentyninePalms-MorongoValle    CA
## 1582              Washingtoncity    DC
## 1585            Hopewelltownship    NJ
## 1606                 Atlantacity    GA
## 1621          Manchestertownship    NJ
## 1639              Gatesvillecity    TX
## 1709                Carlsbadcity    CA
## 1844                  Delanocity    CA
## 1875                   Miamicity    FL
## 1905                  Durhamtown    NH
## 1985             WestNewYorktown    NJ
## 1997              Millingtoncity    TN
## 2001                  Westontown    MA
## 2081                SantaAnacity    CA
## 2106            Hillsboroughtown    CA
## 2107          HuntingtonParkcity    CA
```

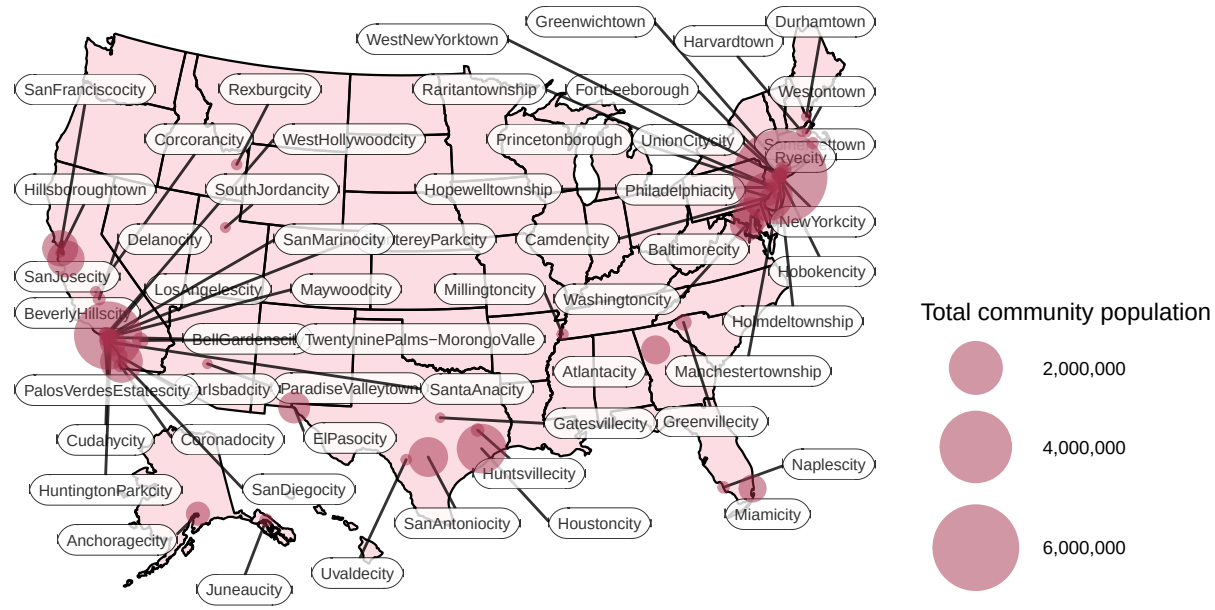
```
citylev<-standf$communityname[rownames(standf) %in% lev]
```

```
dfplot<-citycord[citycord$communityname %in% citylev,c("lon","lat","communityname","state","population")]
```

```
cities_t <- usmap_transform(dfplot)


#without city labels
plot_usmap(fill = "#f26b8b", alpha = 0.23) +
  geom_point(data = cities_t,
             aes(x = lon.1, y = lat.1, size = population),
             color = "#a7314d", alpha = 0.5) +
  scale_size_continuous(range = c(1, 16),
                        label = scales::comma) +
  labs(title = "Leverage communities",
       size = "Total community population") +
  theme(legend.position = "right")
```




```
#with city labels
plot_usmap(fill = "#f26b8b", alpha = 0.23) +
  ggrepel::geom_label_repel(data = cities_t,
             aes(x = lon.1, y = lat.1, label = communityname),
             size = 2, alpha = 0.8,
             label.r = unit(0.5, "lines"), label.size = 0.01,
             segment.color = "black", segment.size = 0.5,
             seed = 1002) +
  geom_point(data = cities_t,
             aes(x = lon.1, y = lat.1, size = population),
             color = "#a7314d", alpha = 0.5) +
  scale_size_continuous(range = c(1, 16),
                        label = scales::comma) +
  labs(title = "Leverage communities",
       size = "Total community population") +
  theme(legend.position = "right")
```



It is interesting to note that these cities belong in many cases to the most populous states in the US such as NY, CA, NJ, PA, TX, FL or to the the least inhabited such as AK and UT. Moreover, as for the outliers, the large majiority of them are clusterized in particular regions or states (CA, NY, NJ).

The results of the linear regressor fitted without the high leverage points are shown below.

```
# regression without leverage points
reg.out4 <- lm(log(ViolentCrimesPerPop+1)~., data=df[!(rownames(df) %in% lev),])

se  <- summary(reg.out4)$sigma                  # se
rsq <- summary(reg.out4)$r.squared              # R^2
adrsq <- summary(reg.out4)$adj.r.squared        # adj R^2
cat("RSE:", round(se,2), "\n")
```

```
## RSE: 0.56
```

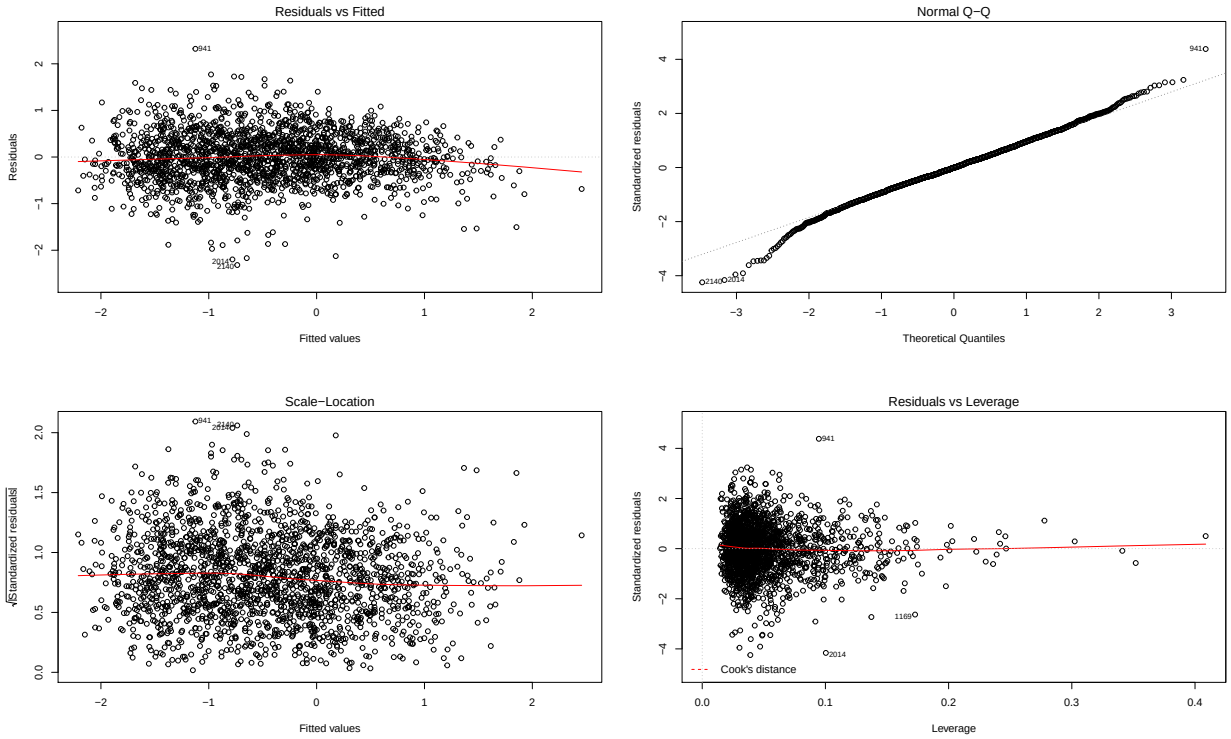
```
cat("R^2:", round(rsq,2), "\n")
```

```
## R^2: 0.69
```

```
cat("adjusted R^2:", round(adrsq,2))
```

```
## adjusted R^2: 0.67
```

```
par(mfrow=c(2,2))
plot(reg.out4)
```




```
par(mfrow=c(1,1))
```

The removal of these problematic points does not seem to significantly change the model. As a matter of fact, the RSE remains the same, while the adjusted R^2 decreases. Taking everything into account, we decided to keep these rows of the dataset avoiding to lose information.

Collinearity

Finally, a possible problem, in particular dealing with a large number of variables as in our case, is that of collinearity. It refers to the situation in which two or more predictor variables are closely related to one another. The presence of collinearity can pose problems in the regression context, since it can be difficult to separate out the individual effects of collinear variables on the response.

In addition, since collinearity reduces the accuracy of the estimates of the regression coefficients, it causes the standard error for $\hat{\beta}_j$ to grow. Recall that the t-statistic for each predictor is calculated by dividing $\hat{\beta}_j$ by its standard error. Consequently, collinearity results in a decline in the t-statistic. As a result, in the presence of collinearity, we may fail to reject $H_0 : \beta_j = 0$. This means that the power of the hypothesis test is reduced by collinearity.

To avoid such a situation, it is desirable to identify and address potential collinearity problems while fitting the model. A simple way to detect collinearity is to look at the correlation matrix of the predictors.

Correlation insight

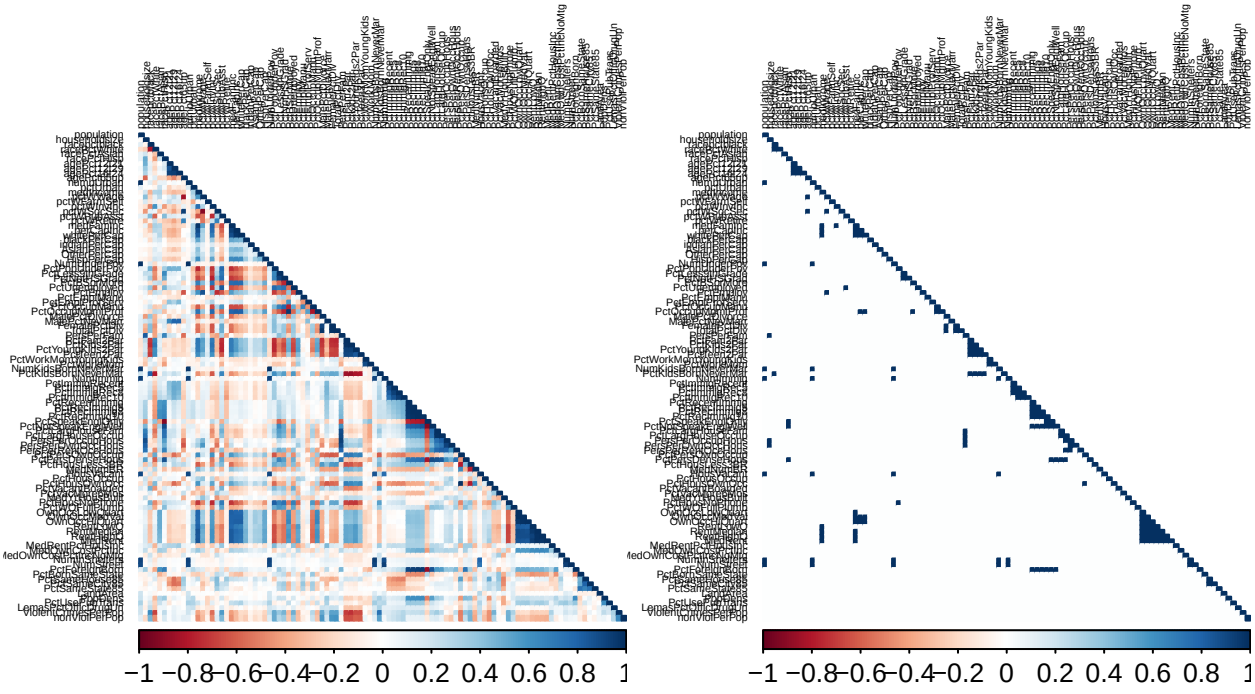
In the following few lines of code we perform a first manual skimming of strongly dependent variables removing, between the most correlated ones, the more redundant and meaningless. It would have been possible to use a simple for loop or an *ad hoc* package, such as *caret*, to filter out some of the most correlated columns to clean up and resize the dataset, but due to the possibly problematic interactions among variables we talked above

about, we preferred to follow a more guided path. In the analysis just the numerical predicting variables and the two main target variables are kept, to highlight eventual correlation-related issues.

```
# restrict to the predictive numeric attributes
coltodrop <- c(1,2, seq(103,118))
corrdf <- cleandf[,-coltodrop]

# correlation matrix
cm <- cor(corrdf, use='complete.obs')

par(mfrow=c(1,2))
# correlation matrix plot
corrplot(cm, method='color', type='lower', tl.pos='l', tl.col='black', tl.cex=0.35) # order=!!!

# correlation thrashold
threshold <- 0.8
# only strongly correlated attributes highlighted
cma <- abs(cm) > threshold
# number of strong correlations
(sum(cma) - dim(corrdf)[2]) / 2
```

```
## [1] 141
```

```
# filtered correlation matrix plot
corrplot(cma, method='color', type='lower', tl.pos='l', tl.col='black', tl.cex=0.35)
```




```
par(mfrow=c(1,1))
```

```r
# correlation tradeoff
threshold <- 0.55
# only strongly correlated attributes
cma <- abs(cm) > threshold
# strongly correlated attributes to "ViolentCrimesPerPop"
names(cma[101,][cma[101,]])
```

```
##  [1] "racepctblack"        "racePctWhite"        "pctWInvInc"
##  [4] "pctWPubAsst"         "PctFam2Par"          "PctKids2Par"
##  [7] "PctYoungKids2Par"    "PctTeen2Par"         "PctKidsBornNeverMar"
## [10] "ViolentCrimesPerPop" "nonViolPerPop"
```

With a threshold of 0.8 no response variable is significanly correlated to any predictor, while at 0.7 "PctKids2Par" and "PctKidsBornNeverMar" appear to be correlated to "ViolentCrimesPerPop". Only when we lower it to 0.55, due to the mutual correlation between the previously mantioned predictors and the new ones, socio-cultural factors show up, highlighting a probable confounding effect hidden between the variables. It is in fact kindly that, in case of non-American-born people, due to law differences between states and logistic problems, a huge number of such families results in analogous conditions to the ones actually without at least one parent, which is a leading condition among the ones positively correlated to the response.

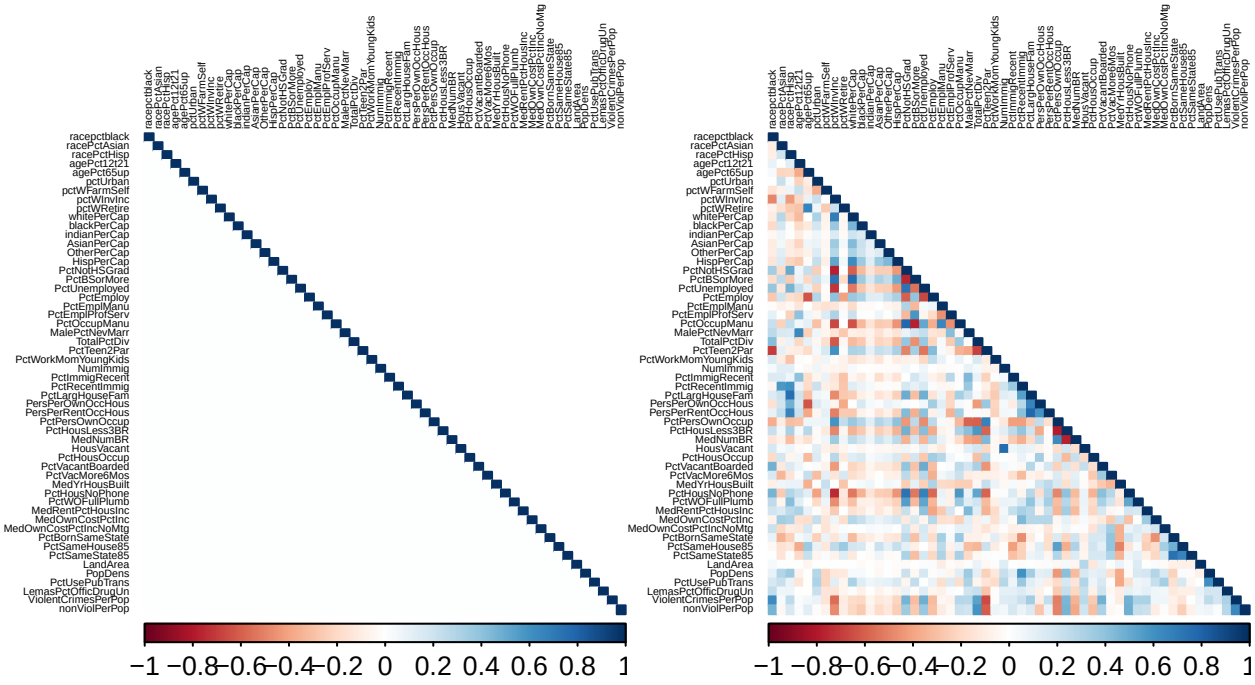
With this in mind, thanks to the following few lines of code, we handle the more correlated variables, trying, at the same time, to preserve as many columns as possible to preserve the original structure of our dataset.

When faced with the problem of collinearity, there are two simple solutions. The first is to drop one of the problematic variables from the regression. This can usually be done without much compromise to the regression fit, since the presence of collinearity implies that the information is redundant. The second solution is to combine the collinear variables together into a single predictor. We decided to proceed in the first way, thus eliminating the redundant columns resulting from the study above, in particular considering variables with correlarion above 0.8. Some relevant examples of columns dropped in the resulting process are:

- "population", due to the presence of both absolute and percent measurement of many variables;
- "agePct16t24" and other redundant columns computed on overlapping time periods;
- "OwnOccHiQuart" and other statistics strictly related to other ones (such as the median).

```r
# columns with correlation with more meaningful ones higher than threshold in absolute value

threshold <- 0.8

rem9 <- c(
"population","agePct16t24","numbUrban","pctWSocSec","medFamInc","perCapInc",
"NumUnderPov","PctLess9thGrade","PctOccupMgmtProf","MalePctDivorce","FemalePctDiv","PctFam2Par",
"PctKids2Par" ,"NumKidsBornNeverMar", "PctImmigRec5","PctImmigRec10",
"PctRecImmig5","PctRecImmig8","PctRecImmig10","PctSpeakEnglOnly" ,
"PctLargHouseOccup","PersPerOccupHous","PctHousOwnOcc","OwnOccLowQuart",
"OwnOccHiQuart","RentLowQ","RentHighQ","MedRent",
"NumInShelters","NumStreet","PctForeignBorn")

rem8 <- c(rem9,
"householdsize","racePctWhite","agePct12t29","medIncome",
"pctWWage","pctWPubAsst","PctPopUnderPov","PersPerFam",
"PctYoungKids2Par","PctWorkMom","PctKidsBornNeverMar", "PctImmigRec8",
"PctNotSpeakEnglWell", "PctPersDenseHous","OwnOccMedVal","RentMedian",
"PctSameCity85")

rem7 <- c(rem8,
"pctWInvInc","PctNotHSGrad","PctBSorMore","PctLargHouseFam","PctHousLess3BR",
```

```
"MedNumBR","PctSameState85" )


corrdf_ind <- corrdf[,!(colnames(corrdf) %in% rem8)]

# final correlation matrix and filter
cm_ind <- cor(corrdf_ind, use='complete.obs')
cma_ind <- abs(cm_ind) > threshold

par(mfrow=c(1,2))
# resulting filtered correlation matrix plot
corrplot(cma_ind, method='color', type='lower', tl.pos='l', tl.col='black', tl.cex=0.35)

# final correlation matrix plot
corrplot(cm_ind, method='color', type='lower', tl.pos='l', tl.col='black', tl.cex=0.35)
```




```
par(mfrow=c(1,1))
```

Let's now fit a linear model on the new obtained dataframe to see how it performs.

```
corrdf <- df[,!(colnames(df) %in% rem8)]

reg.out5 <- lm(log(ViolentCrimesPerPop+1)~., data=corrdf)
#summary(reg.out5)

se  <- summary(reg.out5)$sigma                  # se
rsq <- summary(reg.out5)$r.squared              # R^2
```

```
adrsq <- summary(reg.out5)$adj.r.squared          # adj R^2
cat("RSE:", round(se,2), "\n")
```

```
## RSE: 0.58
```

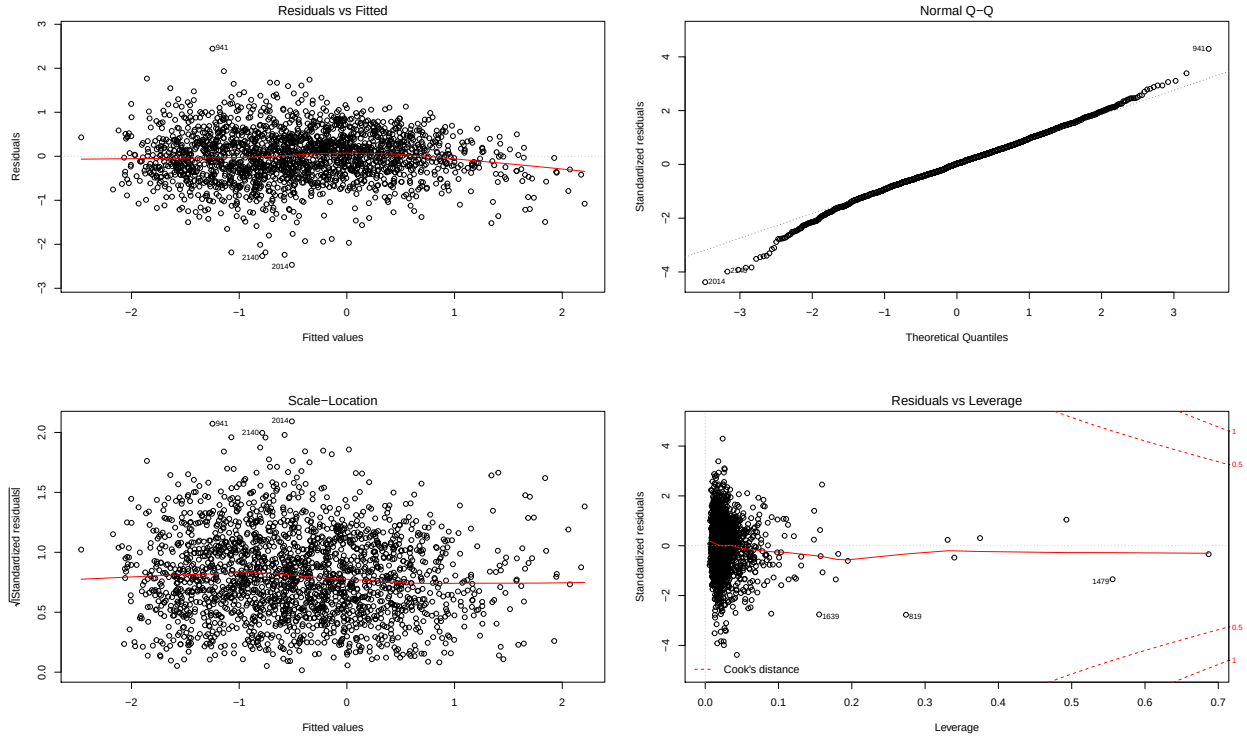
```
cat("R^2:", round(rsq,2), "\n")
```

```
## R^2: 0.67
```

```
cat("adjusted R^2:", round(adrsq,2))
```

```
## adjusted R^2: 0.66
```

```
par(mfrow=c(2,2))
plot(reg.out5)
```




```
par(mfrow=c(1,1))
```

It is possible to notice that the model fitted from the remained 52 predictors performs better than the model obtained with all the 100 ones: the RSE decreases from 0.59 to 0.58 while the adjusted R^2 increases from 0.65 to 0.66. This demonstrates the importance of careful analysis of the collinearity problem, eliminating redundant information not only simplifies the model, but makes it more robust.

From this point on, only the 52 variables selected in this section will be considered for the rest of the study.

Variables selection

With the removal of highly correlated columns, the dataset has decreased from 100 to 52 numeric predictor features. Even if we almost halved the dimension, p, the number of variables is still large. As we will see, many of these variables are irrelevant and not associated with the response. Performing attribute selection is then a necessary step in order to find a good and representative subset of significative features, avoiding then to include useless information that makes our model unnecessarily complex. In this section different techniques for feature selection are presented and tested such as exhaustive and greedy searches, PCA and shrinkage methods.

Before to go into the details, note that from now on the models are trained not on the full dataset but only on a fraction (80% of the rows). The remaining part is then used to estimate the test error. Results are given in terms of MSE (Mean Squared Error) and adjusted R^2 to ensure comparability between models built with different numbers of predictors.

Train - Test split

```
# Train-Test samples
train.sample <- as.numeric(sample(rownames(corrdf), 0.8*dim(corrdf)[1]))
test.sample <- as.numeric(setdiff(rownames(corrdf), train.sample))
```

```
X = subset(corrdf, select = - c(ViolentCrimesPerPop))
X.train <- subset(X, (row.names(X) %in% train.sample))
X.test <- subset(X, (row.names(X) %in% test.sample))

Y <- log(corrdf["ViolentCrimesPerPop",drop=FALSE] +1)

Y.train <- unlist(subset(Y, (row.names(Y) %in% train.sample)))
Y.test <- unlist(subset(Y, (row.names(Y) %in% test.sample)))
```

Full model

We firstly compute the test error on the full model in order to have beseline values. For "full" here we mean the linear regression model using all the standardized columns left from the correlation investigation which aims to predict the log-transformed variable ViolentCrimesPerPop.

```
full.mod <- lm(Y.train~., data=X.train)
summary(full.mod)
```

```
##
## Call:
## lm(formula = Y.train ~ ., data = X.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.43464 -0.35770 -0.00405  0.35640  2.37569
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)           -4.504e-01  1.460e-02 -30.849  < 2e-16 ***
## racepctblack           2.312e-01  2.959e-02   7.814 1.02e-14 ***
## racePctAsian           7.321e-02  2.013e-02   3.637 0.000285 ***
## racePctHisp            1.152e-01  3.413e-02   3.375 0.000755 ***
## agePct12t21           -7.387e-02  3.678e-02  -2.009 0.044755 *
## agePct65up             1.556e-01  5.384e-02   2.891 0.003900 **
```

```
## pctUrban               9.471e-02  2.017e-02   4.694 2.91e-06 ***
## pctWFarmSelf          -7.597e-03  1.843e-02  -0.412 0.680255
## pctWInvInc            -3.411e-01  5.186e-02  -6.576 6.59e-11 ***
## pctWRetire             4.539e-03  2.687e-02   0.169 0.865896
## whitePerCap            5.275e-02  3.829e-02   1.378 0.168475
## blackPerCap           -1.727e-02  2.132e-02  -0.810 0.418037
## indianPerCap          -1.965e-02  1.974e-02  -0.996 0.319569
## AsianPerCap            1.461e-02  1.638e-02   0.892 0.372709
## OtherPerCap            2.671e-02  1.575e-02   1.696 0.090106 .
## HispPerCap            -3.073e-02  2.086e-02  -1.473 0.141058
## PctNotHSGrad           3.675e-02  4.976e-02   0.739 0.460224
## PctBSorMore            2.805e-02  5.283e-02   0.531 0.595486
## PctUnemployed          3.351e-03  3.317e-02   0.101 0.919533
## PctEmploy              1.147e-01  4.845e-02   2.366 0.018088 *
## PctEmplManu           -7.356e-03  3.023e-02  -0.243 0.807750
## PctEmplProfServ        1.866e-02  3.226e-02   0.579 0.562988
## PctOccupManu          -2.282e-02  4.604e-02  -0.496 0.620145
## MalePctNevMarr         1.113e-01  4.086e-02   2.725 0.006502 **
## TotalPctDiv            1.507e-01  4.347e-02   3.466 0.000543 ***
## PctTeen2Par           -4.537e-03  3.323e-02  -0.137 0.891434
## PctWorkMomYoungKids   -3.750e-02  2.069e-02  -1.813 0.070079 .
## NumImmig               6.123e-02  7.862e-02   0.779 0.436189
## PctImmigRecent         1.638e-02  2.000e-02   0.819 0.412979
## PctRecentImmig        -9.202e-02  3.564e-02  -2.582 0.009920 **
## PctLargHouseFam        3.566e-02  4.713e-02   0.757 0.449313
## PersPerOwnOccHous     -7.715e-02  5.214e-02  -1.480 0.139114
## PersPerRentOccHous     1.065e-01  4.157e-02   2.563 0.010483 *
## PctPersOwnOccup       -8.251e-02  4.622e-02  -1.785 0.074407 .
## PctHousLess3BR        -1.404e-02  4.459e-02  -0.315 0.752951
## MedNumBR              -1.562e-02  2.401e-02  -0.650 0.515547
## HousVacant             6.637e-02  3.702e-02   1.793 0.073192 .
## PctHousOccup          -4.854e-02  2.193e-02  -2.213 0.027045 *
## PctVacantBoarded       1.408e-02  2.201e-02   0.640 0.522552
## PctVacMore6Mos        -4.251e-02  2.229e-02  -1.907 0.056681 .
## MedYrHousBuilt        -5.531e-02  2.980e-02  -1.856 0.063671 .
## PctHousNoPhone         2.500e-02  3.519e-02   0.710 0.477599
## PctWOFullPlumb        -4.123e-02  1.957e-02  -2.107 0.035294 *
## MedRentPctHousInc      5.733e-02  2.271e-02   2.524 0.011687 *
## MedOwnCostPctInc      -2.994e-02  2.672e-02  -1.120 0.262714
## MedOwnCostPctIncNoMtg -3.497e-02  2.070e-02  -1.689 0.091384 .
## PctBornSameState      -1.161e-01  3.642e-02  -3.187 0.001466 **
## PctSameHouse85        -4.785e-02  4.089e-02  -1.170 0.242091
## PctSameState85         1.392e-01  3.556e-02   3.915 9.43e-05 ***
## LandArea               4.331e-02  2.192e-02   1.976 0.048371 *
## PopDens               -6.398e-06  2.544e-02   0.000 0.999799
## PctUsePubTrans        -3.510e-02  2.540e-02  -1.382 0.167158
## LemasPctOfficDrugUn    4.346e-02  1.594e-02   2.726 0.006481 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5758 on 1543 degrees of freedom
## Multiple R-squared:  0.6683, Adjusted R-squared:  0.6572
## F-statistic: 59.79 on 52 and 1543 DF,  p-value: < 2.2e-16
```

```r
# prediction
n <- dim(X.test)[1]
p <- length(full.mod$coefficients) - 1
f.pred = predict(full.mod, X.test)
rss <- sum((Y.test - f.pred)^2)            # Residual Sum of Squares
ess <- sum((f.pred - mean(Y.test))^2)      # Explained Sum of Squares
tss <- ess + rss                           # Total Sum of Squares
r2 <- 1 - rss/tss                          # R Squared statistic
adjr2 <- 1 - (1-r2)*((n-1)/(n-p-1))        # adjusted R square

mse <- rss/n                               # Mean Squared Error
cat("MSE: ", round(mse,2), "\n")
```

```
## MSE:  0.39
```

```r
cat("adjR^2: ", round(adjr2,2), "\n")
```

```
## adjR^2:  0.62
```

As expected, from the summary it can be noticed that many of the 52 predictors are not relevant for the prediction of the violent crimes rate, making us think that there exists a better subset of variables able to lower the MSE and increase the adjusted R^2.

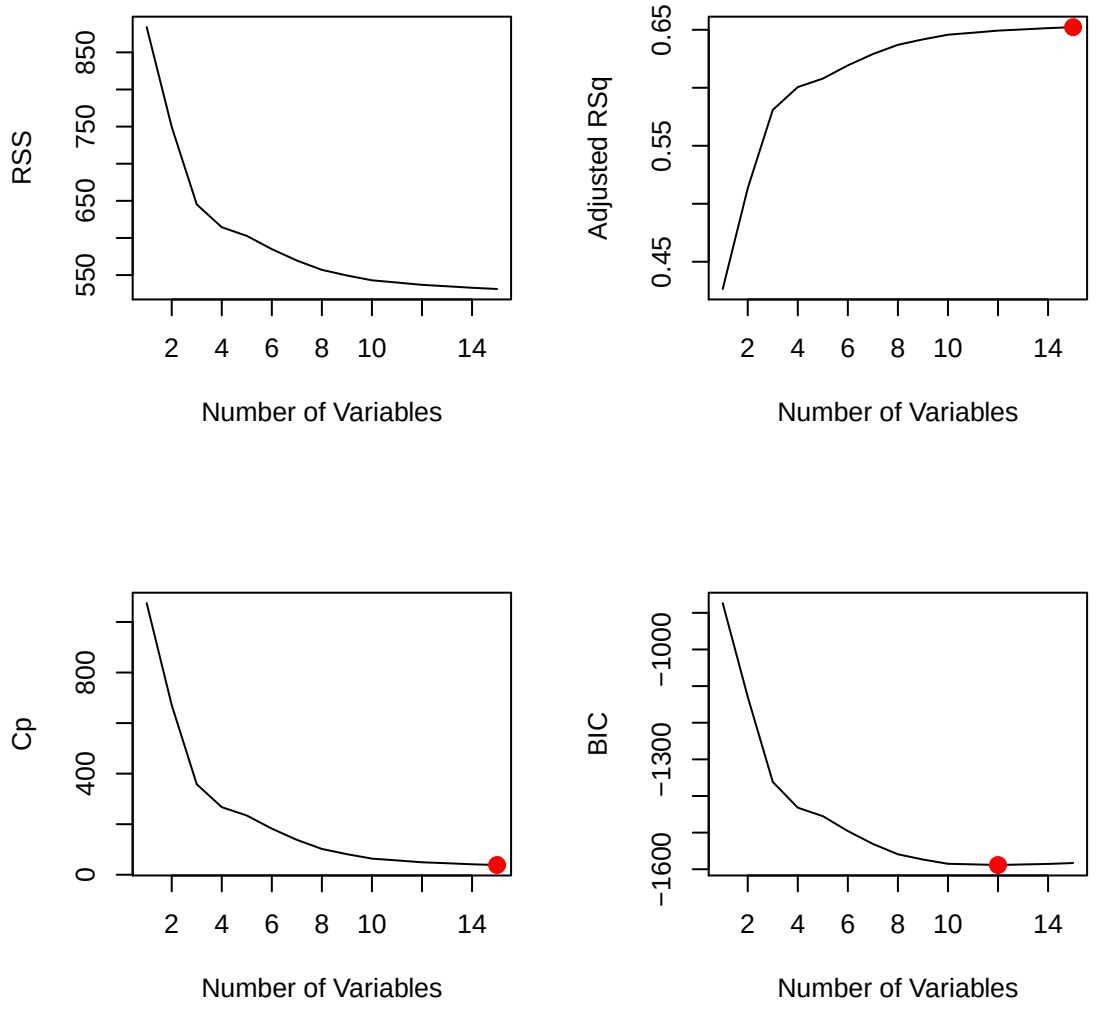
Best subset selection

The first approach we tried is the Best Subset Selection method, an intuitively simple strategy which fits all possible models with at most $k = 1, 2, \ldots, p$ attributes chosen between all the p available and then select the best according to different measures such as C_p, BIC and adjusted R^2. It is easy to understand that this exhaustive search has exponential time in p, making it not very efficient for our case. For this reason we performed the method fixing the maximum number of features to 15 instead of 52. Note that even with this big reduction the function is very slow (10 minutes running time).

```r
regfit.full <- regsubsets(Y.train~., nvmax = 15, really.big=T, data= X.train)
reg.summary <- summary(regfit.full)
```

The plots of the evaluation measures C_p, BIC and adjusted R^2 in function of the number of variables are shown below with a red star in corrispondence of the optima.

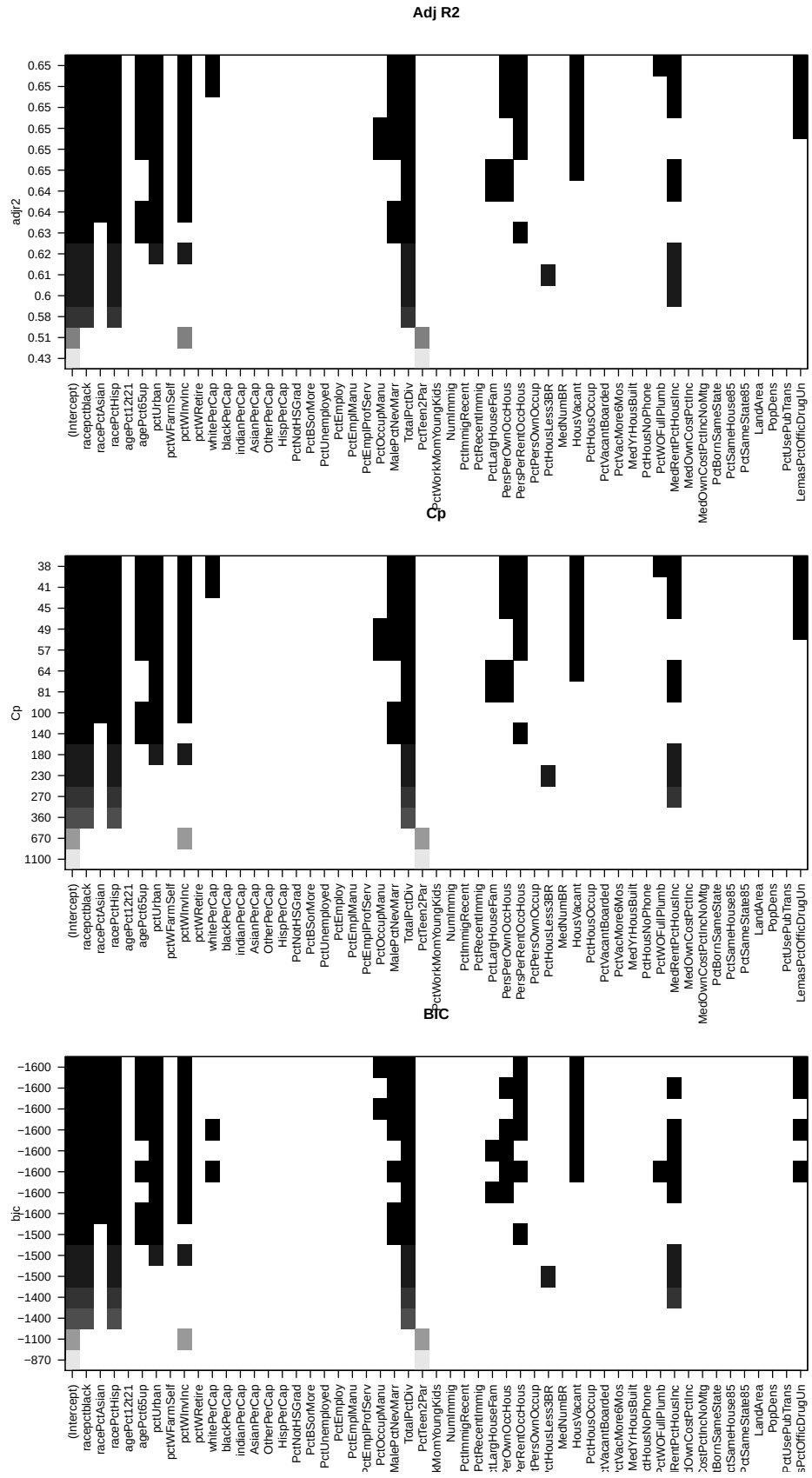
```r
par(mfrow=c(2,2))
plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")

plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="l")
mad<-which.max(reg.summary$adjr2)
points(mad,reg.summary$adjr2[mad], col="red",cex=2,pch=20)

plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",type='l')
mcp<-which.min(reg.summary$cp)
points(mcp,reg.summary$cp[mcp],col="red",cex=2,pch=20)

plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type='l')
mb<-which.min(reg.summary$bic)
points(mb,reg.summary$bic[mb],col="red",cex=2,pch=20)
```




```
par(mfrow=c(1,1))
```

In addition, graphical representations of the best subsect selected for each value of k $\in [1, 15]$ are plotted. Note that the intensity of the colours on the gray scale corresponds to the level of significance.

```
par(mfrow=c(3,1))
plot(regfit.full,scale="adjr2", main="Adj R2")
plot(regfit.full,scale="Cp", main="Cp")
plot(regfit.full,scale="bic", main="BIC")
```


Adj R2
adjr2
0.65
0.65
0.65
0.65
0.65
0.65
0.64
0.64
0.63
0.62
0.61
0.6
0.58
0.51
0.43
Cp
38
41
45
49
57
64
81
100
140
180
230
270
360
670
1100
BIC
−1600
−1600
−1600
−1600
−1600
−1600
−1600
−1600
−1500
−1500
−1500
−1400
−1400
−1100
−870
(Intercept)
racepctblack
racePctAsian
racePctHisp
agePct12t21
agePct65up
pctUrban
pctWFarmSelf
pctWInvInc
pctWRetire
whitePerCap
blackPerCap
indianPerCap
AsianPerCap
OtherPerCap
HispPerCap
PctNotHSGrad
PctBSorMore
PctUnemployed
PctEmploy
PctEmplManu
PctEmplProfServ
PctOccupManu
MalePctNevMarr
TotalPctDiv
PctTeen2Par
PctWorkMomYoungKids
NumImmig
PctImmigRecent
PctRecentImmig
PctLargHouseFam
PersPerOwnOccHous
PersPerRentOccHous
PctPersOwnOccup
PctHousLess3BR
MedNumBR
HousVacant
PctHousOccup
PctVacantBoarded
PctVacMore6Mos
MedYrHousBuilt
PctHousNoPhone
PctWOFullPlumb
MedRentPctHousInc
MedOwnCostPctInc
MedOwnCostPctIncNoMtg
PctBornSameState
PctSameHouse85
PctSameState85
LandArea
PopDens
PctUsePubTrans
LemasPctOfficDrugUn

Finally, the best models selected according to C_p, BIC and adjusted R^2 the are then tested.

```
# predict with adjR2 best
coef <- coef(regfit.full,mad) #coeff of the best model suggested by adjR2

#best subsect of var
as.data.frame(coef)
```

```
##                              coef
## (Intercept)          -0.45115336
## racepctblack          0.21837452
## racePctAsian          0.06375129
## racePctHisp           0.14529384
## agePct65up            0.13368986
## pctUrban              0.10429331
## pctWInvInc           -0.27669141
## whitePerCap           0.05817629
## MalePctNevMarr        0.10912970
## TotalPctDiv           0.29783161
## PersPerOwnOccHous    -0.07046266
## PersPerRentOccHous    0.11567598
## HousVacant            0.10716295
## PctWOFullPlumb       -0.03875542
## MedRentPctHousInc     0.05118911
## LemasPctOfficDrugUn   0.04617855
```

```
advar <- names(coef)[-1]

n <- dim(X.test)[1]
p <- mad
ad.pred=as.matrix(X.test[,advar])%*%coef[advar] + coef["(Intercept)"]
rss <- sum((Y.test - ad.pred)^2)       # Residual Sum of Squares
ess <- sum((ad.pred - mean(Y.test))^2) # Explained Sum of Squares
tss <- ess + rss                           # Total Sum of Squares
r2 <- 1 - rss/tss                          # R Squared statistic
adjr2 <- 1 - (1-r2)*((n-1)/(n-p-1))        # adjusted R square

mse <- rss/n                               # Mean Squared Error
cat("MSE: ", round(mse,2), "\n")
```

```
## MSE:  0.36
```

```
cat("adjR^2: ", round(adjr2,2), "\n")
```

```
## adjR^2:  0.66
```

```
# predict with Cp best
coef<-coef(regfit.full,mcp) #coeff of the best model suggested by Cp

#best subsect of var
as.data.frame(coef)
```

```
##                              coef
## (Intercept)          -0.45115336
## racepctblack          0.21837452
## racePctAsian          0.06375129
## racePctHisp           0.14529384
```

```
## agePct65up           0.13368986
## pctUrban             0.10429331
## pctWInvInc          -0.27669141
## whitePerCap          0.05817629
## MalePctNevMarr       0.10912970
## TotalPctDiv          0.29783161
## PersPerOwnOccHous   -0.07046266
## PersPerRentOccHous   0.11567598
## HousVacant           0.10716295
## PctWOFullPlumb      -0.03875542
## MedRentPctHousInc    0.05118911
## LemasPctOfficDrugUn  0.04617855
```

```
cpvar<-names(coef)[-1]

n <- dim(X.test)[1]
p <- mcp

cp.pred=as.matrix(X.test[,cpvar])%*%coef[cpvar] + coef["(Intercept)"]
rss <- sum((Y.test - cp.pred)^2)       # Residual Sum of Squares
ess <- sum((cp.pred - mean(Y.test))^2) # Explained Sum of Squares
tss <- ess + rss                       # Total Sum of Squares
r2 <- 1 - rss/tss                      # R Squared statistic
adjr2 <- 1 - (1-r2)*((n-1)/(n-p-1))    # adjusted R square

mse <- rss/n                           #Mean Squared Error
cat("MSE: ", round(mse,2), "\n")
```

```
## MSE:  0.36
```

```
cat("adjR^2: ", round(adjr2,2), "\n")
```

```
## adjR^2:  0.66
```

```
# predict with BIC best
coef<-coef(regfit.full,mb) #coeff of the best model suggested by BIC

#best subsect of var
as.data.frame(coef)
```

```
##                          coef
## (Intercept)         -0.45192946
## racepctblack         0.21927724
## racePctAsian         0.07136262
## racePctHisp          0.11868134
## agePct65up           0.19132246
## pctUrban             0.09608300
## pctWInvInc          -0.24238948
## PctOccupManu        -0.06698617
## MalePctNevMarr       0.14156452
## TotalPctDiv          0.34923257
## PersPerRentOccHous   0.12961038
## HousVacant           0.10268842
## LemasPctOfficDrugUn  0.04790166
```

```
bvar<-names(coef)[-1]
```

```
n <- dim(X.test)[1]
p <- mb

b.pred=as.matrix(X.test[,bvar])%*%coef[bvar] + coef["(Intercept)"]

rss <- sum((Y.test - b.pred)^2)             # Residual Sum of Squares
ess <- sum((b.pred - mean(Y.test))^2)       # Explained Sum of Squares
tss <- ess + rss                            # Total Sum of Squares
r2 <- 1 - rss/tss                           # R Squared statistic
adjr2 <- 1 - (1-r2)*((n-1)/(n-p-1))         # adjusted R square

mse <- rss/n                                # Mean Squared Error
cat("MSE: ", round(mse,2), "\n")
```

```
## MSE:  0.36
```

```
cat("adjR^2: ", round(adjr2,2), "\n")
```

```
## adjR^2:  0.65
```

The results are very good considering that we reduced p from 52 to at most 15, with all the three choices this simplified models perfom better than the full both in terms of MSE and adjusted R^2.

Greedy search algorithms

The Best subset method is very time expensive when searching between large number of feautures as in our application, moreover only models with a small k can be investigated in feasible time. This limitations motivates us to abandone exact and exhaustive approaches and move to greedy searches like backward and forward selection.

Backward selection

Backward selection begins with the full least squares model containing all p predictors, and then iteratively removes the least useful predictor, one-at-a-time.

```
back.mod <- step(full.mod, steps=53,  trace=0, direction="backward")
summary(back.mod)
```

```
##
## Call:
## lm(formula = Y.train ~ racepctblack + racePctAsian + racePctHisp +
##     agePct12t21 + agePct65up + pctUrban + pctWInvInc + whitePerCap +
##     OtherPerCap + HispPerCap + PctEmploy + PctEmplProfServ +
##     MalePctNevMarr + TotalPctDiv + PctWorkMomYoungKids + PctRecentImmig +
##     PersPerOwnOccHous + PersPerRentOccHous + PctPersOwnOccup +
##     HousVacant + PctHousOccup + PctVacMore6Mos + MedYrHousBuilt +
##     PctWOFullPlumb + MedRentPctHousInc + MedOwnCostPctIncNoMtg +
##     PctBornSameState + PctSameState85 + LandArea + PctUsePubTrans +
##     LemasPctOfficDrugUn, data = X.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.4537 -0.3557  0.0026  0.3551  2.3422
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)           -0.45105    0.01444 -31.247  < 2e-16 ***
## racepctblack           0.23875    0.02480   9.628  < 2e-16 ***
## racePctAsian           0.06839    0.01902   3.595 0.000334 ***
## racePctHisp            0.12901    0.02926   4.409 1.11e-05 ***
## agePct12t21           -0.05605    0.03340  -1.678 0.093516 .
## agePct65up             0.14334    0.04061   3.530 0.000428 ***
## pctUrban               0.09344    0.01816   5.146 3.00e-07 ***
## pctWInvInc            -0.33706    0.04407  -7.649 3.52e-14 ***
## whitePerCap            0.05556    0.02849   1.950 0.051323 .
## OtherPerCap            0.02835    0.01558   1.820 0.068933 .
## HispPerCap            -0.03263    0.02031  -1.606 0.108386
## PctEmploy              0.08277    0.03785   2.187 0.028903 *
## PctEmplProfServ        0.04365    0.02251   1.939 0.052631 .
## MalePctNevMarr         0.11005    0.03490   3.153 0.001645 **
## TotalPctDiv            0.17868    0.03669   4.870 1.23e-06 ***
## PctWorkMomYoungKids   -0.03570    0.01967  -1.815 0.069697 .
## PctRecentImmig        -0.07674    0.02862  -2.681 0.007418 **
## PersPerOwnOccHous     -0.07384    0.03414  -2.163 0.030695 *
## PersPerRentOccHous     0.12330    0.03022   4.080 4.74e-05 ***
## PctPersOwnOccup       -0.10009    0.03673  -2.725 0.006496 **
## HousVacant             0.08954    0.03079   2.908 0.003690 **
## PctHousOccup          -0.05164    0.02042  -2.529 0.011542 *
## PctVacMore6Mos        -0.04249    0.02105  -2.019 0.043681 *
## MedYrHousBuilt        -0.04339    0.02389  -1.816 0.069563 .
## PctWOFullPlumb        -0.03323    0.01850  -1.796 0.072685 .
## MedRentPctHousInc      0.05175    0.02029   2.550 0.010859 *
## MedOwnCostPctIncNoMtg -0.04501    0.01890  -2.382 0.017328 *
## PctBornSameState      -0.10023    0.03455  -2.901 0.003767 **
## PctSameState85         0.11702    0.03159   3.704 0.000219 ***
## LandArea               0.04170    0.02127   1.961 0.050111 .
## PctUsePubTrans        -0.03296    0.02212  -1.490 0.136300
## LemasPctOfficDrugUn    0.04610    0.01575   2.926 0.003484 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.574 on 1564 degrees of freedom
## Multiple R-squared:  0.6659, Adjusted R-squared:  0.6593
## F-statistic: 100.6 on 31 and 1564 DF,  p-value: < 2.2e-16
```

```r
# prediction
n <- dim(X.test)[1]
p <- length(back.mod$coefficients) - 1
bw.pred = predict(back.mod, X.test)
rss <- sum((Y.test - bw.pred)^2)            # Residual Sum of Squares
ess <- sum((bw.pred - mean(Y.test))^2)      # Explained Sum of Squares
tss <- ess + rss                            # Total Sum of Squares
r2 <- 1 - rss/tss                           # R Squared statistic
adjr2 <- 1 - (1-r2)*((n-1)/(n-p-1))         # adjusted R square

mse <- rss/n                                # Mean Squared Error
cat("MSE: ", round(mse,2), "\n")
```

```
## MSE:  0.36
```

```r
cat("adjR^2: ", round(adjr2,2), "\n")
```

```
## adjR^2:  0.64
```

Forward selection

Forward stepwise selection begins with the empty model containing no predictors, and then adds one-at-a-time the most significative predictor.

```r
empty.mod <- lm(Y.train~1, data=X.train)

forw.mod <- step(empty.mod, steps=53,  trace=0, scope=list(lower=formula(empty.mod), upper=formula(full
summary(forw.mod)
```

```
##
## Call:
## lm(formula = Y.train ~ PctTeen2Par + pctWInvInc + pctUrban +
##     PctBornSameState + TotalPctDiv + racepctblack + racePctHisp +
##     MedRentPctHousInc + PersPerOwnOccHous + racePctAsian + PctLargHouseFam +
##     HousVacant + LemasPctOfficDrugUn + PctSameState85 + PctPersOwnOccup +
##     PctRecentImmig + whitePerCap + PctWOFullPlumb + agePct65up +
##     PersPerRentOccHous + MalePctNevMarr + LandArea + PctEmploy +
##     MedOwnCostPctIncNoMtg + PctHousOccup + MedYrHousBuilt + PctVacMore6Mos,
##     data = X.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.34927 -0.35853 -0.00355  0.35409  2.38553
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -0.450832   0.014465 -31.168  < 2e-16 ***
## PctTeen2Par             -0.006847   0.032381  -0.211 0.832566
## pctWInvInc              -0.320209   0.043017  -7.444 1.61e-13 ***
## pctUrban                 0.097642   0.017867   5.465 5.38e-08 ***
## PctBornSameState        -0.089076   0.034264  -2.600 0.009419 **
## TotalPctDiv              0.174175   0.037893   4.596 4.64e-06 ***
## racepctblack             0.230591   0.026006   8.867  < 2e-16 ***
## racePctHisp              0.136959   0.029158   4.697 2.87e-06 ***
## MedRentPctHousInc        0.063651   0.019966   3.188 0.001461 **
## PersPerOwnOccHous       -0.112937   0.047500  -2.378 0.017543 *
## racePctAsian             0.073032   0.019103   3.823 0.000137 ***
## PctLargHouseFam          0.054540   0.043050   1.267 0.205384
## HousVacant               0.081871   0.030148   2.716 0.006688 **
## LemasPctOfficDrugUn      0.046637   0.015798   2.952 0.003204 **
## PctSameState85           0.105600   0.031202   3.384 0.000731 ***
## PctPersOwnOccup         -0.103795   0.036449  -2.848 0.004461 **
## PctRecentImmig          -0.085964   0.028898  -2.975 0.002978 **
## whitePerCap              0.067197   0.025842   2.600 0.009401 **
## PctWOFullPlumb          -0.040192   0.018459  -2.177 0.029599 *
## agePct65up               0.114259   0.040070   2.852 0.004408 **
## PersPerRentOccHous       0.090239   0.034219   2.637 0.008444 **
## MalePctNevMarr           0.058637   0.031952   1.835 0.066673 .
## LandArea                 0.043060   0.021283   2.023 0.043222 *
## PctEmploy                0.055465   0.029552   1.877 0.060724 .
```

```
## MedOwnCostPctIncNoMtg -0.038629   0.018086  -2.136 0.032842 *
## PctHousOccup          -0.049893   0.020462  -2.438 0.014868 *
## MedYrHousBuilt        -0.046274   0.023020  -2.010 0.044586 *
## PctVacMore6Mos        -0.039524   0.020983  -1.884 0.059806 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5753 on 1568 degrees of freedom
## Multiple R-squared:  0.6635, Adjusted R-squared:  0.6577
## F-statistic: 114.5 on 27 and 1568 DF,  p-value: < 2.2e-16
```

```
# prediction
n <- dim(X.test)[1]
p <- length(forw.mod$coefficients) - 1
fw.pred = predict(forw.mod, X.test)
rss <- sum((Y.test - fw.pred)^2)            # Residual Sum of Squares
ess <- sum((fw.pred - mean(Y.test))^2)      # Explained Sum of Squares
tss <- ess + rss                            # Total Sum of Squares
r2 <- 1 - rss/tss                           # R Squared statistic
adjr2 <- 1 - (1-r2)*((n-1)/(n-p-1))         # adjusted R square

mse <- rss/n                                # Mean Squared Error
cat("MSE: ", round(mse,2), "\n")
```

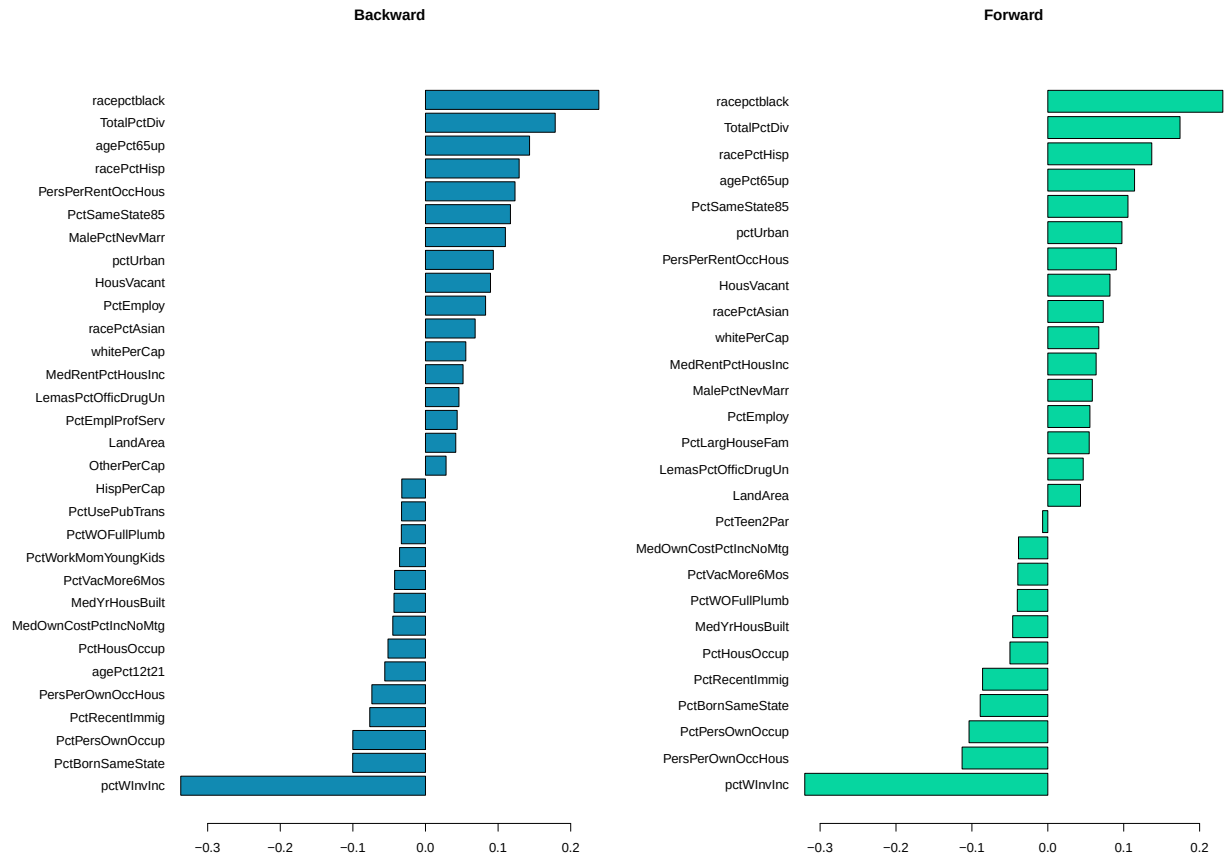
```
## MSE:  0.36
```

```
cat("adjR^2: ", round(adjr2,2), "\n")
```

```
## adjR^2:  0.65
```

These greedy algorithms managed in few seconds to achieve quality solutions in terms of meaningfulness and reduction of the variables, MSE and adjusted R^2 scores. Note that the accuracy in MSE is equal to the one of super time expensive best subset methods, backward and forward strategy proved then to be very efficient in our quite large application.

```
# backward and forward estimated coefficients comparison
par(mfrow=c(1,2), mar=c(5,12,6,1)+.1)
barplot(sort(back.mod$coefficients[!(names(back.mod$coefficients) %in% "(Intercept)")]),
        horiz = TRUE, las=1, col=Col[4], main ="Backward")

barplot(sort(forw.mod$coefficients[!(names(forw.mod$coefficients) %in% "(Intercept)")]),
        horiz = TRUE, las=1, col=Col[3], main ="Forward")
```



Shrinkage methods

The shrinkage methods represent a valid alternative to the subset selection methods seen above, they allow to reduce the complexity of the model and moreover they perform a regularization role. While backward and forward elimination use least squares to fit a linear model that contains a subset of the predictors, the shrinkage methods fit a model containing all p predictors using a technique that shrinks the coefficient estimates towards zero.

RIDGE and LASSO regression are implemented below.

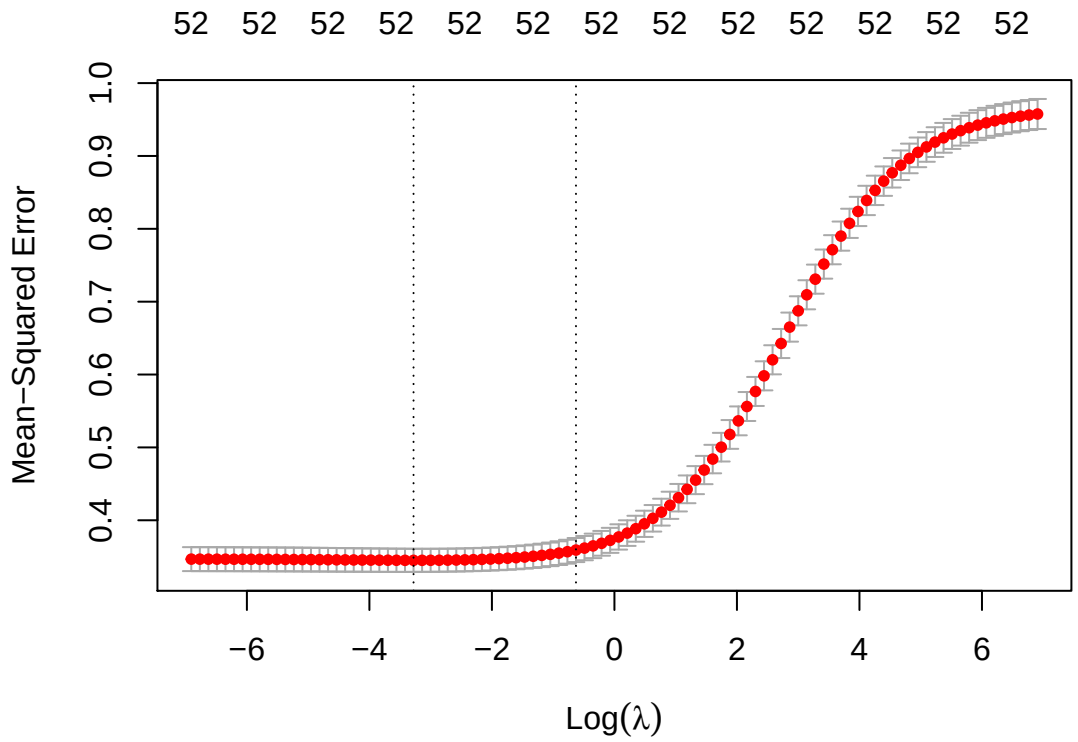
```
# convert dataframes to matrixes
X.train.mat <- as.matrix(X.train)
X.test.mat <- as.matrix(X.test)
Y.train.mat <- as.matrix(Y.train)
Y.test.mat <- as.matrix(Y.test)
```

Ridge Regression with Cross-Validation[II]

In RIDGE regression the estimated coefficients are obtained by minimizing $RSS + \lambda \sum_{j=1}^{p-1} \beta_j^2$. In order to determine the best value of the regularization parameter we perform a 10-fold cross-validation on 100 values of λ in the interval $[10^{-3}, 10^3]$.

```
# lambda values to try
lambda.val <- 10^seq(3, -3, length=100)
```

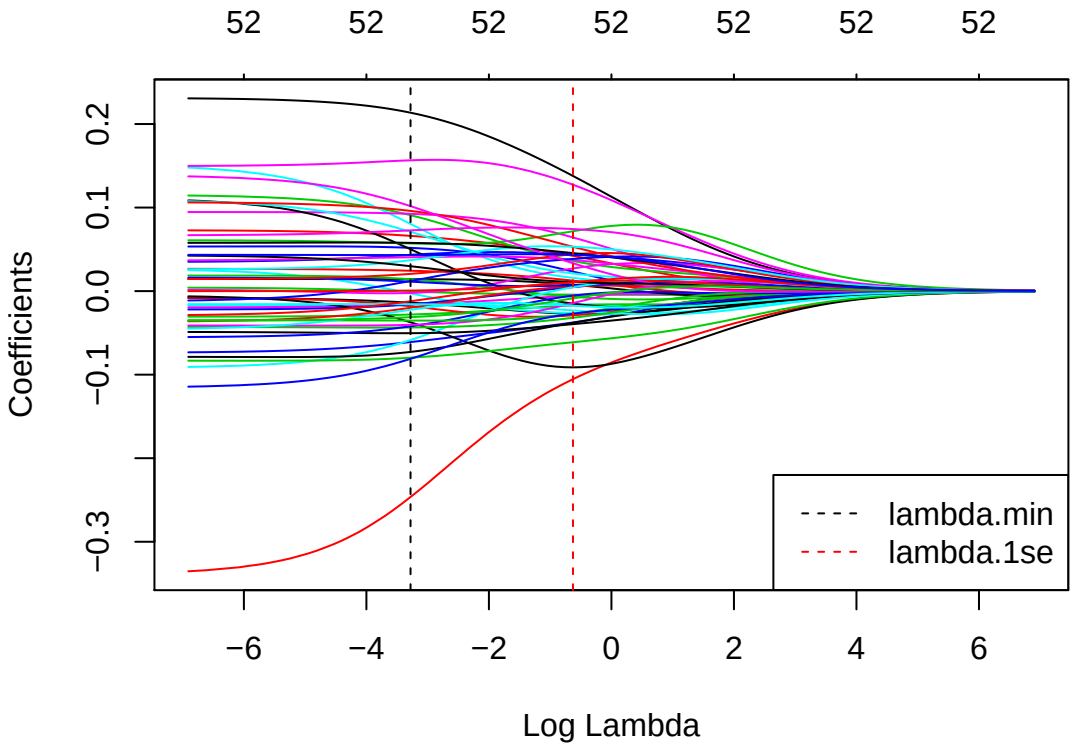
```
# ridge regression with 10-fold cross-validation on training set to choose lambda
ridge.mod.cv <- cv.glmnet(X.train.mat, Y.train.mat, alpha=0,
                          lambda=lambda.val, nfolds = 10)
plot(ridge.mod.cv)
```




```
# best cross-validated lambda
lam.min <- ridge.mod.cv$lambda.min
lam.1se <- ridge.mod.cv$lambda.1se
```

From the model we obtain the value of λ.min which gives the lowest cross-validation error and the value of λ.1se which has error within 1 standard error of the best model. Using the value of λ.1se results in a model that is slightly simpler than the best model and whose accuracy is comparable with the best model.

```
# ridge regression on the entire train set
ridge.mod <- glmnet(X.train.mat, Y.train.mat, alpha=0, lambda=lambda.val)
plot(ridge.mod, xvar = "lambda")
abline(v=log(lam.min), lty=2, col=1)
abline(v=log(lam.1se), lty=2, col=2)
legend("bottomright", c("lambda.min", "lambda.1se"), lty=c(2,2), col = c(1,2))
```



RIDGE regression decreases model complexity while keeping all variables in the model by shrinking the coefficients of low-variance components more than those of highvariance components. The more lambda increases, the more the coefficients are shrunk towards zero.

Let us compare the coefficients which result from the choice of λ.min and λ.1se.

```
# ridge regression estimated coefficients for lambda.min
ridge.coef <- as.matrix(coef(ridge.mod.cv,s=lam.min))
ridge.coef <- as.data.frame(ridge.coef)
colnames(ridge.coef) <- c("Coefficients")

# most relevant predictors
subset.data.frame(ridge.coef, abs(Coefficients)>0.05)
```

```
##                   Coefficients
## (Intercept)         -0.45044552
## racepctblack         0.21354645
## racePctAsian         0.06612264
## racePctHisp          0.09087443
## agePct12t21         -0.06094878
## agePct65up           0.08037638
## pctUrban             0.09246961
## pctWInvInc          -0.24642101
## whitePerCap          0.05177054
## PctEmploy            0.05011212
## MalePctNevMarr       0.07098134
## TotalPctDiv          0.15657227
```

```
## NumImmig              0.05702770
## PctRecentImmig       -0.05854656
## PersPerOwnOccHous    -0.07276336
## PersPerRentOccHous    0.09598975
## PctPersOwnOccup      -0.07972499
## HousVacant            0.07270244
## PctHousOccup         -0.05037006
## MedRentPctHousInc     0.05776009
## PctBornSameState     -0.08103421
## PctSameState85        0.10169350
```

```
# ridge regression estimated coefficients for lambda.1se
ridge.coef <- as.matrix(coef(ridge.mod.cv,s=lam.1se))
ridge.coef <- as.data.frame(ridge.coef)
colnames(ridge.coef) <- c("Coefficients")

# most relevant predictors
subset.data.frame(ridge.coef, abs(Coefficients)>0.05)
```

```
##                     Coefficients
## (Intercept)          -0.45040356
## racepctblack          0.13784753
## pctUrban              0.06426888
## pctWInvInc           -0.10541701
## TotalPctDiv           0.12735756
## PctTeen2Par          -0.09126449
## NumImmig              0.07165006
## PersPerRentOccHous    0.05304393
## PctPersOwnOccup      -0.06120459
## HousVacant            0.07428241
## PctHousNoPhone        0.05309455
```

```
par(mfrow=c(1,2), mar=c(5,12,6,1)+.1)
ridge.coef.lam.min <- coef(ridge.mod.cv,s=lam.min)
barplot(sort(ridge.coef.lam.min[!(rownames(ridge.coef.lam.min) %in% "(Intercept)"),]),
        horiz = TRUE, las=1, col=Col[5], main ="RIDGE coefficients - lambda.min")

ridge.coef.lam.1se <- coef(ridge.mod.cv,s=lam.1se)
barplot(sort(ridge.coef.lam.1se[!(rownames(ridge.coef.lam.1se) %in% "(Intercept)"),]),
        horiz = TRUE, las=1, col=Col[1], main ="RIDGE coefficients - lambda.1se")
```


RIDGE coefficients − lambda.mi
racepctblack
TotalPctDiv
PctSameState85
PersPerRentOccHous
pctUrban
racePctHisp
agePct65up
HousVacant
MalePctNevMarr
racePctAsian
MedRentPctHousInc
NumImmig
whitePerCap
PctEmploy
PctNotHSGrad
LemasPctOfficDrugUn
PctLargHouseFam
PctHousNoPhone
LandArea
OtherPerCap
PctVacantBoarded
AsianPerCap
PctEmplProfServ
PctHousLess3BR
PctImmigRecent
PopDens
pctWRetire
PctUnemployed
PctBSorMore
MedOwnCostPctInc
pctWFarmSelf
MedNumBR
PctOccupManu
blackPerCap
indianPerCap
PctEmplManu
PctWorkMomYoungKids
PctSameHouse85
HispPerCap
PctUsePubTrans
MedOwnCostPctIncNoMtg
PctTeen2Par
PctWOFullPlumb
MedYrHousBuilt
PctVacMore6Mos
PctHousOccup
PctRecentImmig
agePct12t21
PersPerOwnOccHous
PctPersOwnOccup
PctBornSameState
pctWInvInc
−0.2
−0.1
0.0
0.1
0.2
RIDGE coefficients − lambda.1s
racepctblack
TotalPctDiv
HousVacant
NumImmig
pctUrban
PctHousNoPhone
PersPerRentOccHous
MedRentPctHousInc
PctVacantBoarded
PctNotHSGrad
LemasPctOfficDrugUn
racePctAsian
PctHousLess3BR
PctLargHouseFam
racePctHisp
PctSameState85
PctUnemployed
whitePerCap
MalePctNevMarr
agePct65up
OtherPerCap
LandArea
MedOwnCostPctInc
PopDens
AsianPerCap
PctImmigRecent
PctWorkMomYoungKids
PctOccupManu
PctRecentImmig
pctWRetire
MedYrHousBuilt
indianPerCap
PctEmplProfServ
PctUsePubTrans
PctWOFullPlumb
PctEmploy
blackPerCap
HispPerCap
PctSameHouse85
MedNumBR
PctBSorMore
PctBornSameState
MedOwnCostPctIncNoMtg
pctWFarmSelf
PctEmplManu
PctVacMore6Mos
agePct12t21
PersPerOwnOccHous
PctHousOccup
PctPersOwnOccup
PctTeen2Par
pctWInvInc
−0.10
0.00
0.05
0.10

Since the model with λ.1se would trivially perform worse, we test the model associated to λ.min.

```
# prediction on test set
Y.hat <- predict(ridge.mod.cv, newx = X.test.mat, s=lam.min)

n <- dim(X.test)[1]
p <- dim(ridge.coef)[1] -1 #remove intercept

rss <- sum((Y.test.mat - Y.hat)^2)              # Residual Sum of Squares
ess <- sum((Y.hat - mean(Y.test.mat))^2)        # Explained Sum of Squares
tss <- ess + rss                                # Total Sum of Squares
r2 <- 1 - rss/tss                               # R Squared statistic
adjr2 <- 1 - (1-r2)*((n-1)/(n-p-1))             # adjusted R square

mse <- rss/n                                    # Mean Squared Error
cat("MSE: ", round(mse,2), "\n")
```

```
## MSE:  0.38
```

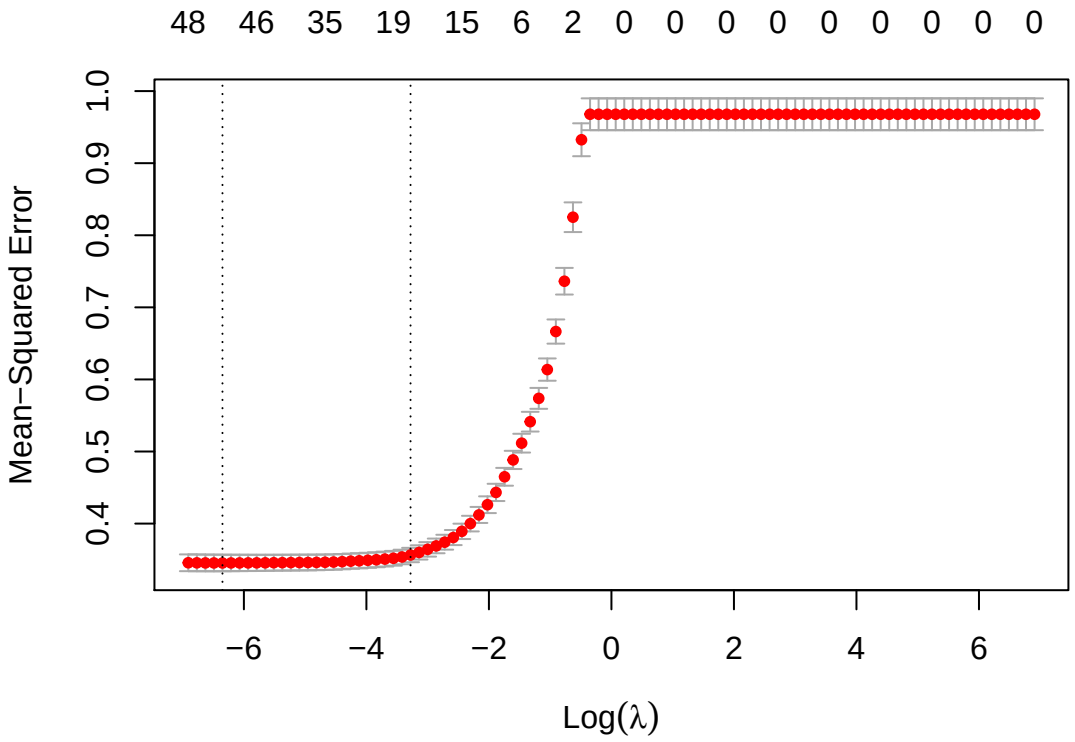
```
cat("adjR^2: ", round(adjr2,2), "\n")
```

```
## adjR^2:  0.62
```

Lasso Regression with Cross-Validation[III]

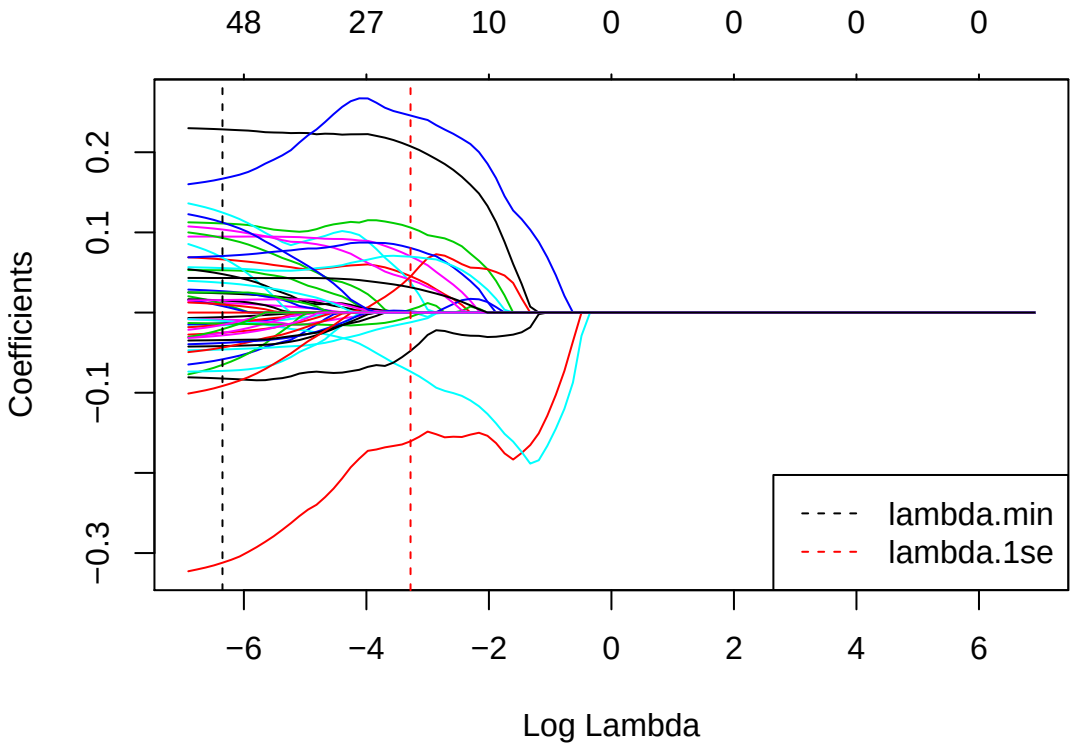
In LASSO regression the estimated coefficients are obtained by minimizing $RSS + \lambda \sum_{j=1}^{p-1} |\beta_j|$). In order to determine the best value of the regularization parameter we use the same approach as before and we perform a 10-fold cross-validation on 100 values of λ in the interval $[10^{-3}, 10^3]$.

```
# lambda values to try
lambda.val <- 10^seq(3, -3, length=100)

# lasso regression with 10-fold cross-validation on training set to choose lambda
lasso.mod.cv <- cv.glmnet(X.train.mat, Y.train.mat, alpha=1,
                          lambda=lambda.val, nfolds = 10)
plot(lasso.mod.cv)
```




```
# best cross-validated lambda
lam.min <- lasso.mod.cv$lambda.min
lam.1se <- lasso.mod.cv$lambda.1se
```

```
# lasso regression on the entire train set
lasso.mod <- glmnet(X.train.mat, Y.train.mat, alpha=1, lambda=lambda.val)
plot(lasso.mod, xvar = "lambda")
abline(v=log(lam.min), lty=2, col=1)
abline(v=log(lam.1se), lty=2, col=2)
legend("bottomright", c("lambda.min", "lambda.1se"), lty=c(2,2), col = c(1,2))
```



Unlike RIDGE regression, for high values of λ, many coefficients are exactly zeroed under LASSO.

Let us compare the coefficients which result from the choice of λ.min and λ.1se.

```
# lasso regression estimated coefficients for lambda.min
lasso.coef <- as.matrix(coef(lasso.mod.cv,s=lam.min))
lasso.coef <- as.data.frame(lasso.coef)
colnames(lasso.coef) <- c("Coefficients")

# number of coefficents different from zero
p.min <- sum(abs(lasso.coef$Coefficients)>0)-1  #remove intercept
p.min
```

```
## [1] 48
```

```
# most relevant predictors
subset.data.frame(lasso.coef, abs(Coefficients)>0.05)
```

```
##                   Coefficients
## (Intercept)         -0.45075564
## racepctblack         0.22849484
## racePctAsian         0.06636637
## racePctHisp          0.11118227
## agePct12t21         -0.05830133
## agePct65up           0.12478316
## pctUrban             0.09476707
## pctWInvInc          -0.31201843
## whitePerCap          0.05276338
```

```
## PctEmploy             0.06844107
## MalePctNevMarr        0.09303905
## TotalPctDiv           0.16703018
## PctRecentImmig       -0.06517466
## PersPerOwnOccHous    -0.07275021
## PersPerRentOccHous    0.10359902
## PctPersOwnOccup      -0.08227470
## HousVacant            0.07057062
## MedRentPctHousInc     0.05549684
## PctBornSameState     -0.09156281
## PctSameState85        0.11231977
```

```
# lasso regression estimated coefficients for lambda.1se
lasso.coef <- as.matrix(coef(lasso.mod.cv,s=lam.1se))
lasso.coef <- as.data.frame(lasso.coef)
colnames(lasso.coef) <- c("Coefficients")

# number of coefficents different from zero
p.1se <- sum(abs(lasso.coef$Coefficients)>0)-1  #remove intercept
p.1se
```

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

```
# most relevant predictors
subset.data.frame(lasso.coef, abs(Coefficients)>0.05)
```

```
##                    Coefficients
## (Intercept)         -0.45319289
## racepctblack         0.20740203
## racePctHisp          0.10603509
## pctUrban             0.07017883
## pctWInvInc          -0.16056132
## TotalPctDiv          0.24579851
## PctTeen2Par         -0.07357947
## HousVacant           0.08055867
## MedRentPctHousInc    0.06972198
```

LASSO regression with λ.1se results in a simpler model with less variables.

```
par(mfrow=c(1,2), mar=c(5,12,6,1)+.1)
lasso.coef.lam.min <- coef(lasso.mod.cv,s=lam.min)
barplot(sort(lasso.coef.lam.min[!(rownames(lasso.coef.lam.min) %in% "(Intercept)"),]),
        horiz = TRUE, las=1, col=Col[5], main ="LASSO coefficients - lambda.min")

lasso.coef.lam.1se <- coef(lasso.mod.cv,s=lam.1se)
barplot(sort(lasso.coef.lam.1se[!(rownames(lasso.coef.lam.1se) %in% "(Intercept)"),]),
        horiz = TRUE, las=1, col=Col[1], main ="LASSO coefficients - lambda.1se")
```


LASSO coefficients − lambda.mi
racepctblack
TotalPctDiv
agePct65up
PctSameState85
racePctHisp
PersPerRentOccHous
pctUrban
MalePctNevMarr
HousVacant
PctEmploy
racePctAsian
MedRentPctHousInc
whitePerCap
NumImmig
LemasPctOfficDrugUn
LandArea
PctLargHouseFam
PctHousNoPhone
OtherPerCap
PctVacantBoarded
PctEmplProfServ
AsianPerCap
PctNotHSGrad
PctImmigRecent
PctBSorMore
PopDens
PctHousLess3BR
PctUnemployed
pctWRetire
pctWFarmSelf
PctEmplManu
PctTeen2Par
blackPerCap
MedNumBR
PctOccupManu
MedOwnCostPctInc
indianPerCap
PctSameHouse85
HispPerCap
PctWorkMomYoungKids
PctUsePubTrans
MedOwnCostPctIncNoMtg
PctWOFullPlumb
PctVacMore6Mos
MedYrHousBuilt
PctHousOccup
agePct12t21
PctRecentImmig
PersPerOwnOccHous
PctPersOwnOccup
PctBornSameState
pctWInvInc
−0.3
−0.1
0.0
0.1
0.2
LASSO coefficients − lambda.1s
TotalPctDiv
racepctblack
racePctHisp
HousVacant
pctUrban
MedRentPctHousInc
racePctAsian
PctHousLess3BR
PersPerRentOccHous
agePct65up
LemasPctOfficDrugUn
PctNotHSGrad
PctLargHouseFam
PctUsePubTrans
PopDens
LandArea
PctSameState85
PctSameHouse85
PctBornSameState
MedOwnCostPctIncNoMtg
MedOwnCostPctInc
PctWOFullPlumb
PctHousNoPhone
MedYrHousBuilt
PctVacMore6Mos
PctVacantBoarded
PersPerOwnOccHous
PctRecentImmig
PctImmigRecent
NumImmig
PctWorkMomYoungKids
MalePctNevMarr
PctOccupManu
PctEmplProfServ
PctEmplManu
PctEmploy
PctUnemployed
PctBSorMore
HispPerCap
OtherPerCap
AsianPerCap
indianPerCap
blackPerCap
whitePerCap
pctWRetire
pctWFarmSelf
agePct12t21
MedNumBR
PctHousOccup
PctPersOwnOccup
PctTeen2Par
pctWInvInc
−0.1
0.0
0.1
0.2

We test the model with λ.min.

```
# prediction on test set
Y.hat <- predict(lasso.mod.cv, newx = X.test.mat, s=lam.min)

n <- dim(X.test)[1]
p <- p.min

rss <- sum((Y.test.mat - Y.hat)^2)              # Residual Sum of Squares
ess <- sum((Y.hat - mean(Y.test.mat))^2)        # Explained Sum of Squares
tss <- ess + rss                                # Total Sum of Squares
r2 <- 1 - rss/tss                               # R Squared statistic
adjr2 <- 1 - (1-r2)*((n-1)/(n-p-1))             # adjusted R square

mse <- rss/n                                    # Mean Squared Error
cat("MSE: ", round(mse,2), "\n")
```

```
## MSE:  0.37
```

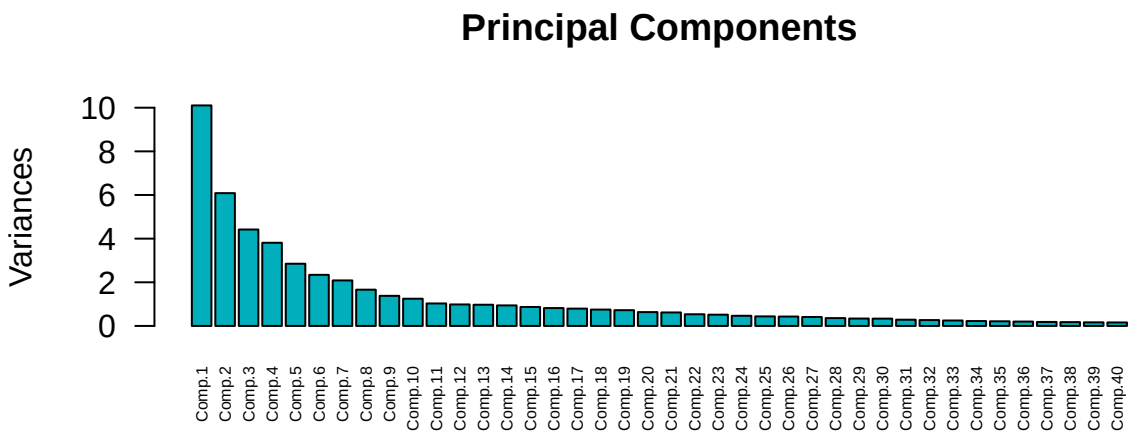
```
cat("adjR^2: ", round(adjr2,2), "\n")
```

```
## adjR^2:  0.63
```

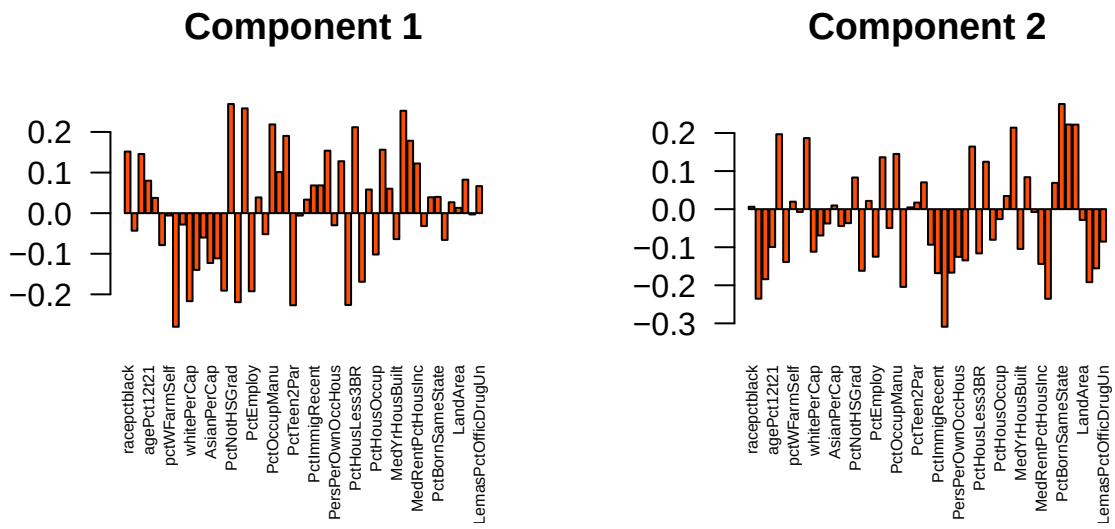
Principal Component Analysis[IV]

To gather the most information possible in relatively few columns we implemented the Principal Component Analysis. In particular the strength of this method resides in the fact that, starting from a variance maximization unconstrained problem, we go on selecting orthogonal components given by proper linear combinations of the original columns, both scaling the dimensionality of the problem and mitigating the effects related to the covariance between the original variables.

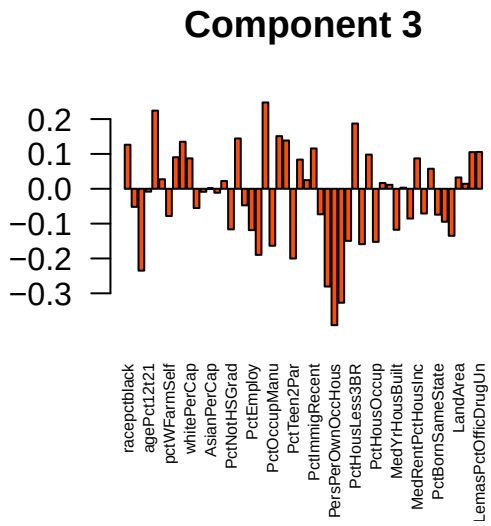
```
# PCA computation
pc <- princomp(X) # cor=TRUE to obtain it from the correlation matrix

# str(pc)
# first k principal components
k <- 40
plot(pc, npcs=k, cex.names=0.5,las=2, col="#00AFBB", main='Principal Components')
```

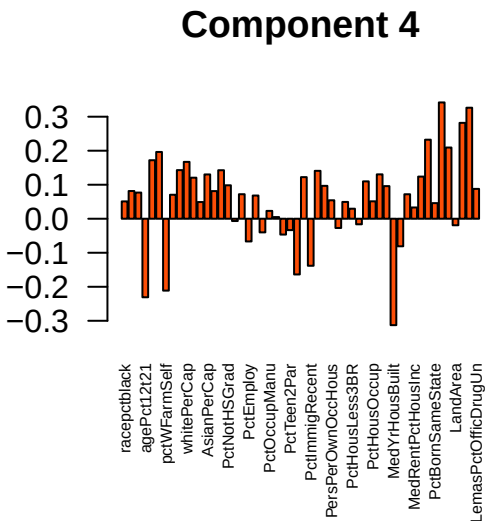



```
par(mfrow=c(1,2))
barplot(pc$loadings[,1], cex.names=0.5,las=2, col=  "#FC4E07", main='Component 1')
barplot(pc$loadings[,2], cex.names=0.5,las=2, col=  "#FC4E07", main='Component 2')
```

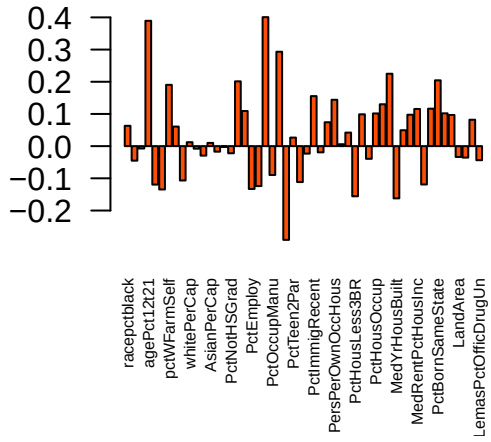


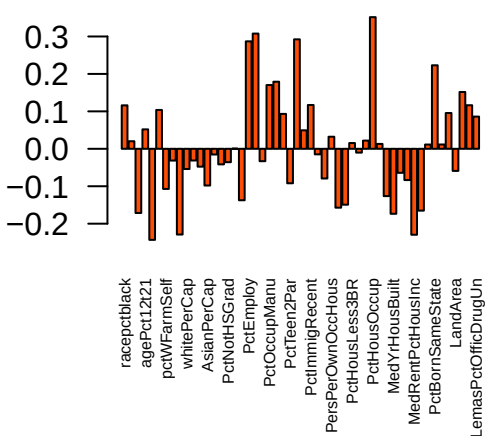

```
barplot(pc$loadings[,3], cex.names=0.5,las=2, col=  "#FC4E07", main='Component 3')
barplot(pc$loadings[,4], cex.names=0.5,las=2, col=  "#FC4E07", main='Component 4')
```



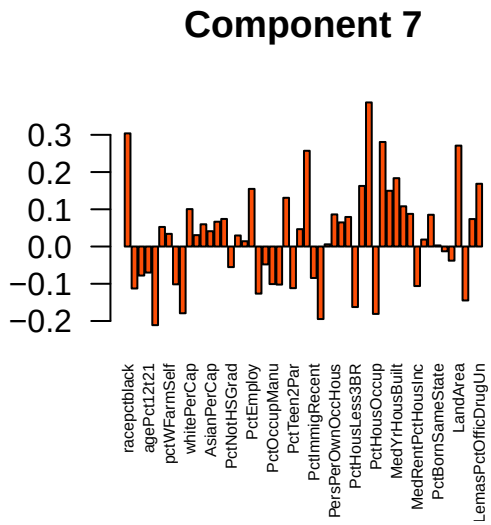


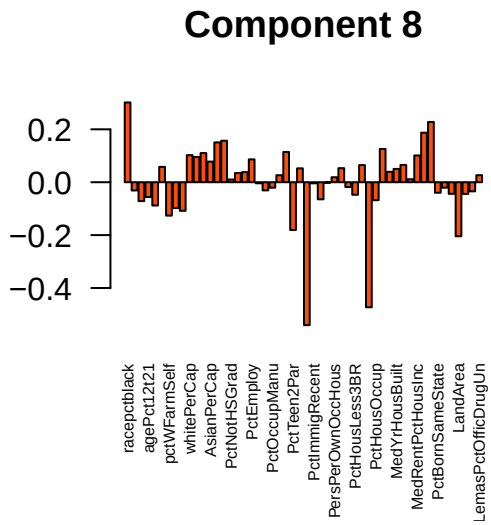

```
barplot(pc$loadings[,5], cex.names=0.5,las=2, col=  "#FC4E07", main='Component 5')
barplot(pc$loadings[,6], cex.names=0.5,las=+2, col=  "#FC4E07", main='Component 6')
```



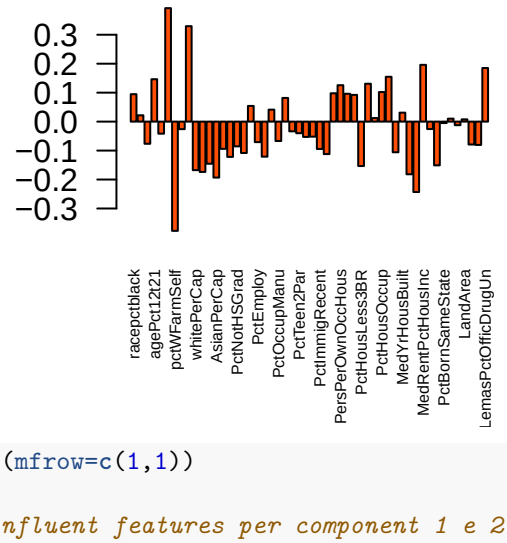


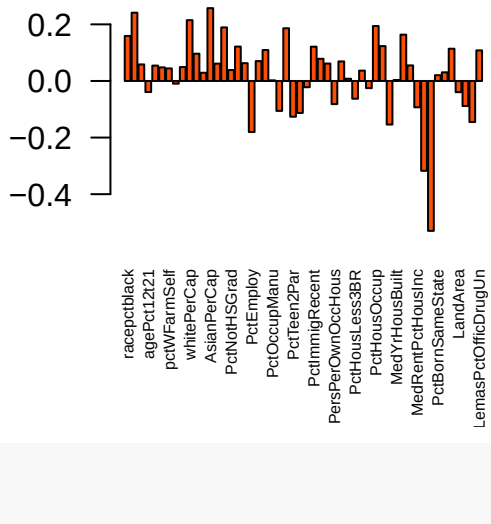

```
barplot(pc$loadings[,7], cex.names=0.5,las=2, col=  "#FC4E07", main='Component 7')
barplot(pc$loadings[,8], cex.names=0.5,las=2, col=  "#FC4E07", main='Component 8')
```






```
barplot(pc$loadings[,9], cex.names=0.5,las=2, col=  "#FC4E07", main='Component 9')
barplot(pc$loadings[,10], cex.names=0.5,las=2, col=  "#FC4E07", main='Component 10')
```





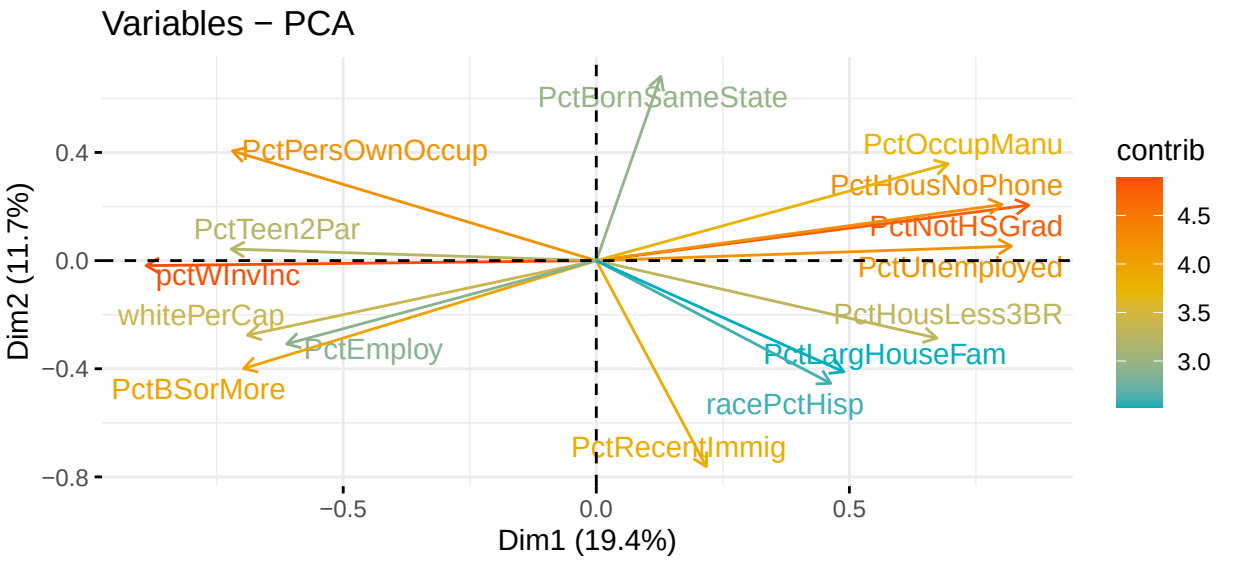

```
par(mfrow=c(1,1))

# influent features per component 1 e 2
infl1 <- colnames(corrdf[,-dim(corrdf)[2]])[abs(pc$loadings[,1])>0.25]
infl1
```

```
## [1] "pctWInvInc"     "PctNotHSGrad"   "PctUnemployed"  "PctHousNoPhone"
```

```
infl2 <- colnames(corrdf[,-dim(corrdf)[2]])[abs(pc$loadings[,2])>0.2]
infl2
```

```
## [1] "racePctAsian"     "MalePctNevMarr"   "PctRecentImmig"
## [4] "PctVacMore6Mos"   "MedOwnCostPctInc" "PctBornSameState"
## [7] "PctSameHouse85"   "PctSameState85"
```

```
fviz_pca_var(pc,
             col.var = "contrib", # Color by contributions to the PC
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = T,     # Avoid text overlapping
             select.var = list(contrib = 15)
             )
```

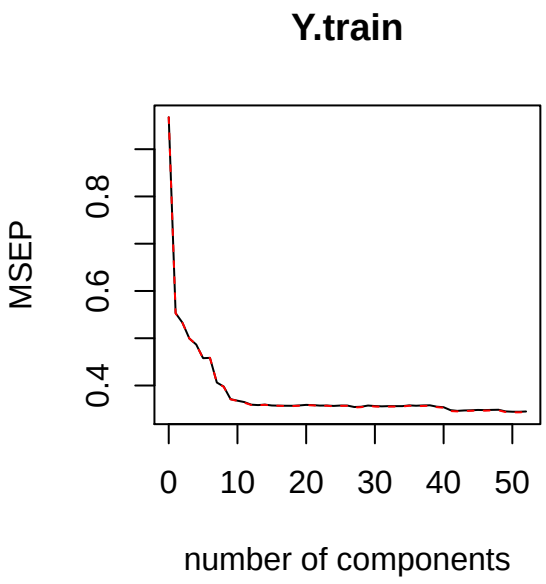


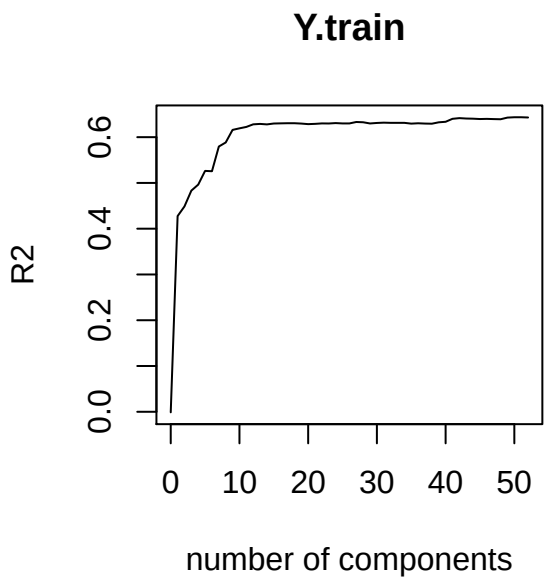
The principal limitation of PCA concerns interpretation, but it is possible to explore the resulting loadings to hopefully extrapolate the semantic area of the main components in each principal direction. Trying to interpret the main components of the first loading (corresponding to the maximum variance direction), for example, it seems to be mainly identified by the original columns related to the working condition of the householder(s). It is interesting to notice the presence of the column "PctHousNoPhone", the percentage of occupied housing units without phone, but one should notice that in 1990 it was a strong indication of lack of need to contact and be contacted, besides the economical factor. As for the second one, the geo-cultural factors are predominant.

PC regression

At this point we use a 10-fold cross validation on the training set to select the best tradeoff between the number of columns to drop and the performance of the model.

```
pcr.fit = pcr(Y.train~., data=X.train, validation = "CV") # k=10 by default
# summary(pcr.fit)

# plot of CV error
par(mfrow=c(1,2))
validationplot(pcr.fit, val.type = "MSEP")
validationplot(pcr.fit, val.type = "R2")
```






```
par(mfrow=c(1,1))
# looking at the plot, 11 could be a great choice
```

It shows up that great tradeoff is to train the final model using the first 11 principal components, and in this way we're able to reach a slightly better accuracy than the previous models, still keepeng low the dimension of the used dataset.

```
# prediction in the model above
n <- dim(X.test)[1]
p <- 11
pcr.pred = predict(pcr.fit, X.test, ncomp=11)
rss <- sum((Y.test - pcr.pred)^2)          # Residual Sum of Squares
ess <- sum((pcr.pred - mean(Y.test))^2)    # Explained Sum of Squares
tss <- ess + rss                           # Total Sum of Squares
r2 <- 1 - rss/tss                          # R Squared statistic
adjr2 <- 1 - (1-r2)*((n-1)/(n-p-1))        # adjusted R square

mse <- rss/n                               # Mean Squared Error
cat("MSE: ", round(mse,2), "\n")
```

```
## MSE:  0.35
```

```
cat("adjR^2: ", round(adjr2,2), "\n")
```

```
## adjR^2:  0.65
```

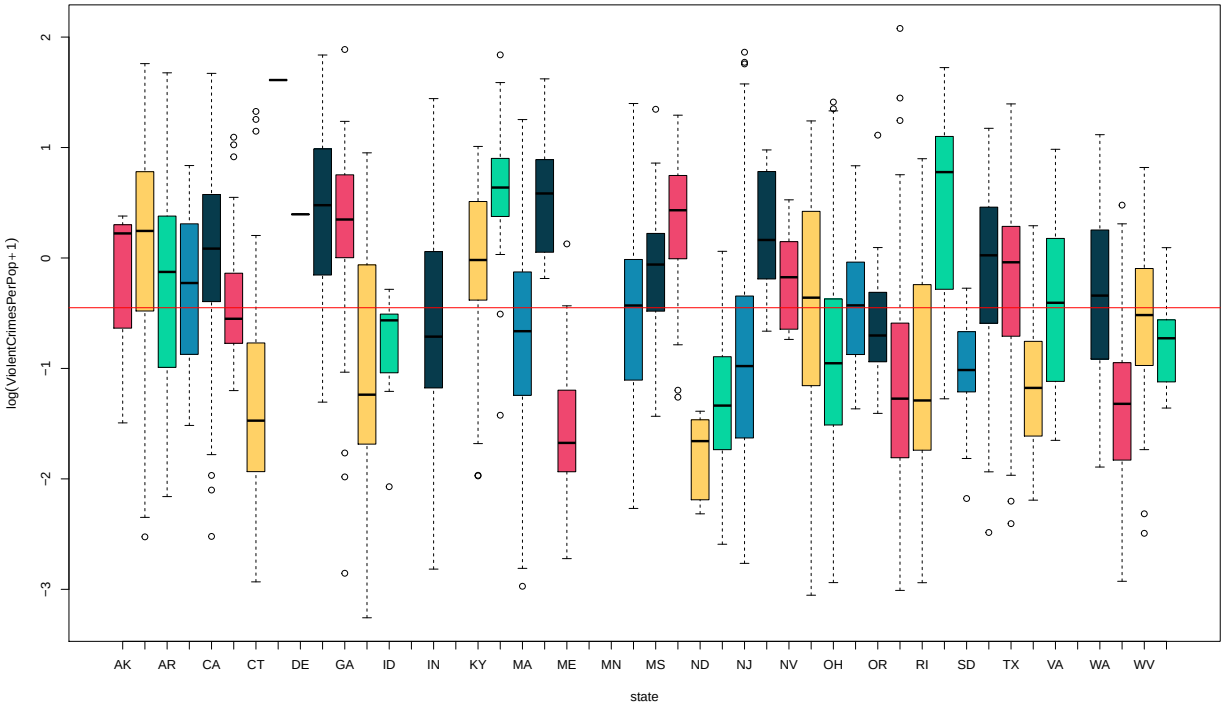
Other possible predictive models

In this section other models have been tested for the prediction of the violent and non violent rate var variables.

Add categorical predictor

Until now we assumed that the response variable ViolentCrimesPerPop has equal behaviour between the different states. Even if in practice this assumption considerably simplified the model, the hypothesis of similar distribution between different cities belonging to a vast territory such as the American one, characterized by a wide variety of geographic, cultural and social conditions, could be a too strong request. Below a side by side boxplot of ViolentCrimesPerPop between the different states is presented (using the standardized and log-transformed version of the data). The horizontal line draws the mean value between all data without distinction.

```
boxplot(log(standf$ViolentCrimesPerPop+1)~standf$state, xlab="state",
        ylab=expression(log(ViolentCrimesPerPop+1)), col= Col)
abline(h=mean(log(standf$ViolentCrimesPerPop+1)), col = "red")
```



The plot seems to confirm the doubts discussed above, therefore we tested the linear model considering the categorical variable State as factor.

```
XS = subset(corrdf, select = - c(ViolentCrimesPerPop))
XS["state"]=standf$state

is.factor(XS$state)
```

```
## [1] TRUE
```

```
XS.train <- subset(XS, (row.names(XS) %in% train.sample))
XS.test <- subset(XS, (row.names(XS) %in% test.sample))
```

```r
Y <- log(corrdf["ViolentCrimesPerPop",drop=FALSE] +1)

Y.train <- unlist(subset(Y, (row.names(Y) %in% train.sample)))
Y.test <- unlist(subset(Y, (row.names(Y) %in% test.sample)))

regc.out <- lm(Y.train~. , data=XS.train) #application of LR with a categorical variable
summary(regc.out)
```

```
##
## Call:
## lm(formula = Y.train ~ ., data = XS.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.2882 -0.3236  0.0000  0.3280  2.2839
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)           -0.1111685  0.5215870  -0.213 0.831251
## racepctblack           0.2490296  0.0322253   7.728 1.99e-14 ***
## racePctAsian           0.0611886  0.0212866   2.875 0.004103 **
## racePctHisp            0.0806493  0.0388384   2.077 0.038015 *
## agePct12t21           -0.0381374  0.0371304  -1.027 0.304530
## agePct65up             0.0682776  0.0533439   1.280 0.200760
## pctUrban               0.0845359  0.0203339   4.157 3.40e-05 ***
## pctWFarmSelf          -0.0156900  0.0196464  -0.799 0.424636
## pctWInvInc            -0.1114630  0.0651758  -1.710 0.087437 .
## pctWRetire             0.0314409  0.0278336   1.130 0.258826
## whitePerCap            0.0400097  0.0391152   1.023 0.306534
## blackPerCap           -0.0180752  0.0206014  -0.877 0.380423
## indianPerCap          -0.0330123  0.0188431  -1.752 0.079986 .
## AsianPerCap            0.0311589  0.0159049   1.959 0.050289 .
## OtherPerCap            0.0197386  0.0152254   1.296 0.195028
## HispPerCap            -0.0217430  0.0203593  -1.068 0.285708
## PctNotHSGrad           0.0934587  0.0562817   1.661 0.097012 .
## PctBSorMore           -0.0516907  0.0585526  -0.883 0.377482
## PctUnemployed          0.0034157  0.0339525   0.101 0.919880
## PctEmploy              0.1276962  0.0485272   2.631 0.008590 **
## PctEmplManu           -0.0282607  0.0309100  -0.914 0.360711
## PctEmplProfServ        0.0265512  0.0322615   0.823 0.410640
## PctOccupManu           0.0325895  0.0481007   0.678 0.498177
## MalePctNevMarr         0.0769907  0.0429761   1.791 0.073418 .
## TotalPctDiv            0.1680887  0.0442842   3.796 0.000153 ***
## PctTeen2Par           -0.0235768  0.0328447  -0.718 0.472976
## PctWorkMomYoungKids   -0.0426303  0.0213675  -1.995 0.046212 *
## NumImmig               0.0241438  0.0749743   0.322 0.747477
## PctImmigRecent         0.0158342  0.0195749   0.809 0.418698
## PctRecentImmig        -0.0600558  0.0347851  -1.726 0.084467 .
## PctLargHouseFam        0.0548290  0.0535080   1.025 0.305675
## PersPerOwnOccHous     -0.0256246  0.0534497  -0.479 0.631713
## PersPerRentOccHous     0.0250630  0.0416965   0.601 0.547877
## PctPersOwnOccup       -0.0999260  0.0475738  -2.100 0.035857 *
## PctHousLess3BR        -0.0227240  0.0451073  -0.504 0.614492
```

```
## MedNumBR              -0.0211034  0.0233329  -0.904 0.365903
## HousVacant             0.0799582  0.0370398   2.159 0.031031 *
## PctHousOccup          -0.0466124  0.0221946  -2.100 0.035881 *
## PctVacantBoarded       0.0208738  0.0214220   0.974 0.330010
## PctVacMore6Mos        -0.0142948  0.0225643  -0.634 0.526496
## MedYrHousBuilt        -0.0659490  0.0299805  -2.200 0.027978 *
## PctHousNoPhone         0.0284426  0.0381971   0.745 0.456613
## PctWOFullPlumb        -0.0260907  0.0190417  -1.370 0.170834
## MedRentPctHousInc      0.0593909  0.0228635   2.598 0.009479 **
## MedOwnCostPctInc      -0.0522752  0.0296434  -1.763 0.078025 .
## MedOwnCostPctIncNoMtg  0.0248968  0.0238916   1.042 0.297545
## PctBornSameState      -0.0217810  0.0468781  -0.465 0.642264
## PctSameHouse85        -0.0596972  0.0421078  -1.418 0.156480
## PctSameState85         0.0787045  0.0417197   1.887 0.059419 .
## LandArea               0.0005207  0.0295925   0.018 0.985963
## PopDens               -0.0123562  0.0252643  -0.489 0.624857
## PctUsePubTrans        -0.0154807  0.0253705  -0.610 0.541832
## LemasPctOfficDrugUn    0.0497240  0.0154102   3.227 0.001279 **
## stateAL               -0.1718398  0.5475747  -0.314 0.753702
## stateAR               -0.4727910  0.5455930  -0.867 0.386320
## stateAZ               -0.2257706  0.5385647  -0.419 0.675124
## stateCA               -0.0191014  0.5290509  -0.036 0.971203
## stateCO               -0.1537137  0.5347109  -0.287 0.773792
## stateCT               -0.7077304  0.5228240  -1.354 0.176046
## stateDC               -0.5379927  0.7921712  -0.679 0.497156
## stateDE                0.1095951  0.7590360   0.144 0.885214
## stateFL                0.2113257  0.5291317   0.399 0.689669
## stateGA               -0.6999490  0.5415656  -1.292 0.196399
## stateIA               -0.6594030  0.5413733  -1.218 0.223408
## stateID               -0.2511126  0.5622071  -0.447 0.655189
## stateIN               -0.6249305  0.5342384  -1.170 0.242283
## stateKY               -0.3046641  0.5455119  -0.558 0.576592
## stateLA               -0.1955446  0.5487393  -0.356 0.721627
## stateMA               -0.0960452  0.5321069  -0.180 0.856784
## stateMD               -0.0991069  0.5507448  -0.180 0.857216
## stateME               -1.2161772  0.5440277  -2.236 0.025531 *
## stateMO               -0.2029891  0.5319716  -0.382 0.702828
## stateMS               -0.7775438  0.5516593  -1.409 0.158905
## stateNC               -0.0926758  0.5429368  -0.171 0.864488
## stateND               -1.0742403  0.5823522  -1.845 0.065284 .
## stateNH               -0.6429524  0.5336261  -1.205 0.228442
## stateNJ               -0.5196547  0.5234166  -0.993 0.320961
## stateNM                0.1230246  0.5562935   0.221 0.825005
## stateNV               -0.2408284  0.5758187  -0.418 0.675834
## stateNY               -0.6602626  0.5298014  -1.246 0.212869
## stateOH               -0.5830871  0.5291522  -1.102 0.270670
## stateOK               -0.2021760  0.5358044  -0.377 0.705980
## stateOR               -0.4480493  0.5296891  -0.846 0.397759
## statePA               -0.6238180  0.5295559  -1.178 0.238982
## stateRI               -0.5833039  0.5361102  -1.088 0.276757
## stateSC               -0.0229768  0.5495147  -0.042 0.966653
## stateSD               -0.3912805  0.5629061  -0.695 0.487095
## stateTN               -0.1016610  0.5423505  -0.187 0.851337
## stateTX               -0.2513062  0.5302063  -0.474 0.635584
```

```
## stateUT               -0.5041611  0.5590104  -0.902 0.367265
## stateVA               -0.5998832  0.5325308  -1.126 0.260144
## stateWA               -0.2829157  0.5277676  -0.536 0.591996
## stateWI               -0.6803160  0.5220085  -1.303 0.192684
## stateWV               -0.5838214  0.5611710  -1.040 0.298339
## stateWY               -0.2411069  0.5730024  -0.421 0.673977
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5443 on 1501 degrees of freedom
## Multiple R-squared:  0.7117, Adjusted R-squared:  0.6936
## F-statistic: 39.42 on 94 and 1501 DF,  p-value: < 2.2e-16
```

From the results plotted by the summary, it seems that adding this information does not help too much the model, the State variable seem not very influential. The only exceptions are the Maine and North Dakota states (ME and ND) for which the information is more significant. Analyzing the position of these two territories it is interesting to note that they are both two northern states on the border with Canada, this precise geographical diversity could be the reason for our result.

Testing the model, as expected the MSE is slightly better than the equivalent model withot the categorical information, otherwise, the adjusted R^2 is smaller.

```
# prediction
pred = predict(regc.out, XS.test)

n <- dim(XS.test)[1]
p <- 94

rss <- sum((Y.test - pred)^2)       # Residual Sum of Squares
ess <- sum((pred - mean(Y.test))^2) # Explained Sum of Squares
tss <- ess + rss                    # Total Sum of Squares
r2 <- 1 - rss/tss                   # R Squared statistic
adjr2 <- 1 - (1-r2)*((n-1)/(n-p-1)) # adjusted R square

mse <- rss/n                        #Mean Squared Error
cat("MSE: ", round(mse,2), "\n")
```

```
## MSE:  0.34
```

```
cat("adjR^2: ", round(adjr2,2), "\n")
```

```
## adjR^2:  0.61
```

Given the contradictory results between the boxplot and the model, we decided to conduct a deeper study by performing ANOVA, Tukey's test and ANCOVA on our dataset. As expected, the ANOVA gives a very small p-value, confirming the theory of not all equal means. At this point hence the Tukey method for all pairwise mean comparisons is applied.

```
summary(aov.state <- aov(ViolentCrimesPerPop ~ state, data = standf))
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## state         42  420.3  10.006   12.41 <2e-16 ***
## Residuals   1953 1574.7   0.806
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

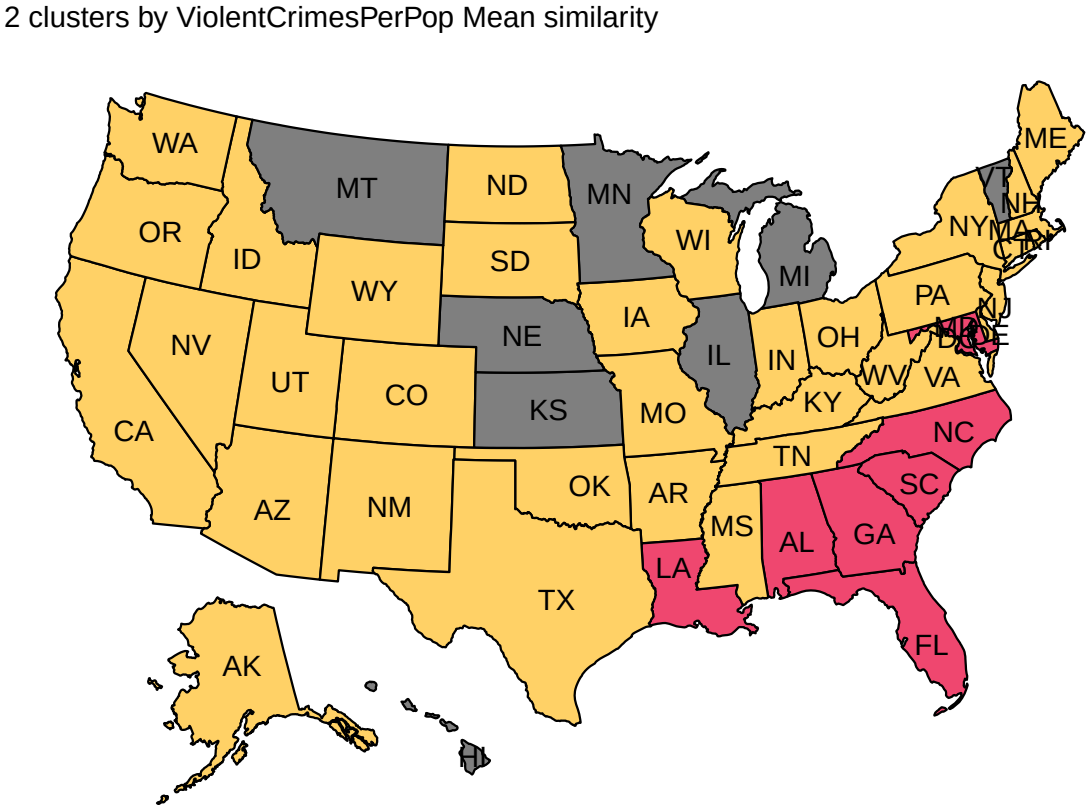
```
#reject all equal mean hyphothesis?
```

```r
result<-TukeyHSD(aov.state, "state", ordered = TRUE)

result<-as.data.frame(result$state)

head(result[result[,4]<0.05,],10) #where reject equal mean hyphothesis?
```

```
##           diff        lwr      upr        p adj
## NC-ND 1.447734 0.09648296 2.798984 1.760285e-02
## GA-ND 1.448349 0.07296693 2.823731 2.329075e-02
## AL-ND 1.541745 0.18352943 2.899961 6.279792e-03
## FL-ND 1.750999 0.44960236 3.052396 1.184642e-04
## MD-ND 1.846624 0.23656283 3.456685 5.139321e-03
## SC-ND 1.872313 0.45817997 3.286445 1.842662e-04
## LA-ND 2.001532 0.54517714 3.457887 6.462946e-05
## DC-ND 4.831302 1.08985879 8.572746 3.556852e-04
## CA-ME 1.074382 0.19316624 1.955598 1.417515e-03
## NC-ME 1.339425 0.33820432 2.340646 1.388763e-04
```

Examining all the pairs in which Tukey's test allows to reject the hypothesis, i.e. where the adjusted p-value is less than $\alpha = 0.05$, it can be concluded that the states can be grouped in two main cluster. Also in this case it is interesting to observe how the statistical test result can be interpreted geographically, the two clusers exactly divide the USin two regions, making us hypothesize that the variable of interest may have influences related to social and cultural aspects related to the position.

```r
cluster1 <- c("ND","ME","WI","NH","SD","UT","CT","ID","WY","PA","RI","OR","OH","IA",
              "NJ","WV","MA","MO","OK","IN","VA","CO","WA","AZ","NV","AK","NY","TX",
              "MS","KY","AR","TN","CA","NM","DE")
cluster2 <- c("NC","GA","AL","FL","MD","SC","LA","DC")

XS["Cluster"]=rep(1,1996)
XS$Cluster[XS$state %in% cluster2] <- 2

plot_usmap(labels = T,
    data = XS[c("state","Cluster")], values = "Cluster",  color = "black") +
    scale_fill_gradient(low = Col[2], high =  Col[1]) +
  theme(legend.position = "none")+
    labs(title = "2 clusters by ViolentCrimesPerPop Mean similarity")
```



The linear model considering a new categorical predictor, that factorize the cluster to which each state belongs, is then built. The cluster information is now significative, moreover the results on the test set show a decrease of the MSE and equal adjusted R^2 (compared to the equivalent model without categorical predictor).

```
XS = XS[,-53]
XS$Cluster <- as.factor(XS$Cluster)

is.factor(XS$Cluster)
```

```
## [1] TRUE
```

```
XS.train <- subset(XS, (row.names(XS) %in% train.sample))
XS.test <- subset(XS, (row.names(XS) %in% test.sample))


regc.out <- lm(Y.train~. , data=XS.train) #application of LR with a categorical variable
summary(regc.out)
```

```
##
## Call:
## lm(formula = Y.train ~ ., data = XS.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.49039 -0.35994 -0.00336  0.36299  2.34735
##
## Coefficients:
```

```
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -0.493478   0.016489 -29.927  < 2e-16 ***
## racepctblack            0.202469   0.029791   6.796 1.53e-11 ***
## racePctAsian            0.084676   0.020055   4.222 2.56e-05 ***
## racePctHisp             0.131586   0.033954   3.875 0.000111 ***
## agePct12t21            -0.070939   0.036446  -1.946 0.051790 .
## agePct65up              0.130709   0.053542   2.441 0.014748 *
## pctUrban                0.087069   0.020040   4.345 1.49e-05 ***
## pctWFarmSelf            0.006149   0.018435   0.334 0.738774
## pctWInvInc             -0.298980   0.051968  -5.753 1.05e-08 ***
## pctWRetire              0.023339   0.026850   0.869 0.384846
## whitePerCap             0.060021   0.037962   1.581 0.114064
## blackPerCap            -0.008198   0.021190  -0.387 0.698907
## indianPerCap           -0.021435   0.019559  -1.096 0.273284
## AsianPerCap             0.014938   0.016230   0.920 0.357528
## OtherPerCap             0.025287   0.015607   1.620 0.105380
## HispPerCap             -0.036156   0.020698  -1.747 0.080861 .
## PctNotHSGrad            0.012312   0.049505   0.249 0.803631
## PctBSorMore            -0.018676   0.053042  -0.352 0.724817
## PctUnemployed           0.024429   0.033090   0.738 0.460483
## PctEmploy               0.131580   0.048109   2.735 0.006309 **
## PctEmplManu             0.005871   0.030049   0.195 0.845134
## PctEmplProfServ         0.037526   0.032148   1.167 0.243278
## PctOccupManu           -0.029245   0.045630  -0.641 0.521678
## MalePctNevMarr          0.107772   0.040488   2.662 0.007853 **
## TotalPctDiv             0.166746   0.043178   3.862 0.000117 ***
## PctTeen2Par             0.013376   0.033094   0.404 0.686131
## PctWorkMomYoungKids    -0.044134   0.020533  -2.149 0.031759 *
## NumImmig                0.050884   0.077920   0.653 0.513837
## PctImmigRecent          0.019453   0.019823   0.981 0.326578
## PctRecentImmig         -0.091744   0.035315  -2.598 0.009470 **
## PctLargHouseFam         0.039149   0.046702   0.838 0.402007
## PersPerOwnOccHous      -0.060402   0.051751  -1.167 0.243325
## PersPerRentOccHous      0.094661   0.041250   2.295 0.021878 *
## PctPersOwnOccup        -0.090968   0.045822  -1.985 0.047292 *
## PctHousLess3BR         -0.009733   0.044191  -0.220 0.825706
## MedNumBR               -0.011889   0.023800  -0.500 0.617488
## HousVacant              0.071573   0.036692   1.951 0.051277 .
## PctHousOccup           -0.052097   0.021744  -2.396 0.016697 *
## PctVacantBoarded        0.017556   0.021817   0.805 0.421135
## PctVacMore6Mos         -0.038071   0.022103  -1.722 0.085182 .
## MedYrHousBuilt         -0.073569   0.029720  -2.475 0.013416 *
## PctHousNoPhone          0.031680   0.034894   0.908 0.364077
## PctWOFullPlumb         -0.043265   0.019393  -2.231 0.025827 *
## MedRentPctHousInc       0.063540   0.022531   2.820 0.004862 **
## MedOwnCostPctInc       -0.026385   0.026483  -0.996 0.319257
## MedOwnCostPctIncNoMtg -0.025949   0.020580  -1.261 0.207555
## PctBornSameState       -0.108223   0.036111  -2.997 0.002771 **
## PctSameHouse85         -0.065704   0.040647  -1.616 0.106202
## PctSameState85          0.162443   0.035494   4.577 5.10e-06 ***
## LandArea                0.039435   0.021731   1.815 0.069775 .
## PopDens                -0.001762   0.025214  -0.070 0.944309
## PctUsePubTrans         -0.021441   0.025288  -0.848 0.396634
## LemasPctOfficDrugUn     0.041878   0.015801   2.650 0.008122 **
```

```
## Cluster2               0.307533   0.056533   5.440 6.19e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5705 on 1542 degrees of freedom
## Multiple R-squared:  0.6746, Adjusted R-squared:  0.6634
## F-statistic: 60.31 on 53 and 1542 DF,  p-value: < 2.2e-16
```

```
# prediction
pred = predict(regc.out, XS.test)

n <- dim(XS.test)[1]
p <- 53

rss <- sum((Y.test - pred)^2)       # Residual Sum of Squares
ess <- sum((pred - mean(Y.test))^2) # Explained Sum of Squares
tss <- ess + rss                    # Total Sum of Squares
r2 <- 1 - rss/tss                   # R Squared statistic
adjr2 <- 1 - (1-r2)*((n-1)/(n-p-1)) # adjusted R square

mse <- rss/n                        #Mean Squared Error
cat("MSE: ", round(mse,2), "\n")
```

```
## MSE:  0.38
```

```
cat("adjR^2: ", round(adjr2,2), "\n")
```

```
## adjR^2:  0.62
```

Finally ANCOVA is performed, to asses if the two models differ between the two groups. The previous summary proves that the intercepts are different, now we want to test if also some of the slope coefficients vary between the two clusters. To this end the linear model exploiting also interaction effects between the Cluster variable and all the other is built.

```
#add interaction and main effect due to Cluster
regc.out <- lm(Y.train~.*Cluster , data=XS.train)
summary(regc.out)
```

```
##
## Call:
## lm(formula = Y.train ~ . * Cluster, data = XS.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.47552 -0.34738 -0.00083  0.35615  2.35546
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 -0.492814   0.016694 -29.521  < 2e-16 ***
## racepctblack                 0.215129   0.032589   6.601 5.65e-11 ***
## racePctAsian                 0.092604   0.021000   4.410 1.11e-05 ***
## racePctHisp                  0.143546   0.036566   3.926 9.04e-05 ***
## agePct12t21                 -0.050147   0.039016  -1.285 0.198891
## agePct65up                   0.139879   0.060182   2.324 0.020245 *
## pctUrban                     0.081137   0.021773   3.726 0.000201 ***
## pctWFarmSelf                 0.013068   0.019215   0.680 0.496576
## pctWInvInc                  -0.309658   0.056338  -5.496 4.55e-08 ***
```

```
## pctWRetire                    0.015848   0.030122   0.526 0.598871
## whitePerCap                   0.075715   0.040363   1.876 0.060872 .
## blackPerCap                   0.002024   0.021604   0.094 0.925358
## indianPerCap                 -0.026350   0.021831  -1.207 0.227613
## AsianPerCap                   0.019029   0.017901   1.063 0.287952
## OtherPerCap                   0.020299   0.016660   1.218 0.223256
## HispPerCap                   -0.037313   0.022109  -1.688 0.091674 .
## PctNotHSGrad                  0.032918   0.053273   0.618 0.536724
## PctBSorMore                  -0.026636   0.057769  -0.461 0.644805
## PctUnemployed                 0.037833   0.035577   1.063 0.287761
## PctEmploy                     0.139080   0.051507   2.700 0.007008 **
## PctEmplManu                   0.014736   0.032130   0.459 0.646566
## PctEmplProfServ               0.036343   0.035043   1.037 0.299857
## PctOccupManu                 -0.045473   0.048856  -0.931 0.352133
## MalePctNevMarr                0.085559   0.043838   1.952 0.051157 .
## TotalPctDiv                   0.185547   0.047149   3.935 8.69e-05 ***
## PctTeen2Par                   0.023798   0.036569   0.651 0.515297
## PctWorkMomYoungKids          -0.041868   0.021926  -1.910 0.056383 .
## NumImmig                      0.026666   0.086620   0.308 0.758235
## PctImmigRecent                0.034727   0.023320   1.489 0.136661
## PctRecentImmig               -0.112999   0.038168  -2.961 0.003120 **
## PctLargHouseFam               0.048503   0.049712   0.976 0.329381
## PersPerOwnOccHous            -0.060414   0.056106  -1.077 0.281750
## PersPerRentOccHous            0.084266   0.043933   1.918 0.055295 .
## PctPersOwnOccup              -0.097107   0.049812  -1.949 0.051425 .
## PctHousLess3BR               -0.010658   0.047934  -0.222 0.824076
## MedNumBR                     -0.011188   0.025825  -0.433 0.664910
## HousVacant                    0.071338   0.041084   1.736 0.082703 .
## PctHousOccup                 -0.058027   0.023052  -2.517 0.011932 *
## PctVacantBoarded              0.023306   0.024235   0.962 0.336358
## PctVacMore6Mos               -0.039500   0.023483  -1.682 0.092762 .
## MedYrHousBuilt               -0.072956   0.031770  -2.296 0.021793 *
## PctHousNoPhone               -0.016505   0.038639  -0.427 0.669320
## PctWOFullPlumb               -0.044552   0.021710  -2.052 0.040328 *
## MedRentPctHousInc             0.071923   0.023943   3.004 0.002710 **
## MedOwnCostPctInc             -0.027999   0.028608  -0.979 0.327891
## MedOwnCostPctIncNoMtg        -0.027460   0.021832  -1.258 0.208666
## PctBornSameState             -0.076437   0.040402  -1.892 0.058695 .
## PctSameHouse85               -0.063225   0.043569  -1.451 0.146952
## PctSameState85                0.137133   0.038832   3.531 0.000426 ***
## LandArea                      0.040480   0.022733   1.781 0.075171 .
## PopDens                      -0.003067   0.026283  -0.117 0.907131
## PctUsePubTrans               -0.005041   0.026859  -0.188 0.851149
## LemasPctOfficDrugUn           0.064304   0.020796   3.092 0.002024 **
## Cluster2                      0.225170   0.220512   1.021 0.307362
## racepctblack:Cluster2        -0.007369   0.115761  -0.064 0.949253
## racePctAsian:Cluster2         0.027745   0.374968   0.074 0.941025
## racePctHisp:Cluster2          0.249907   0.232428   1.075 0.282458
## agePct12t21:Cluster2         -0.337530   0.155115  -2.176 0.029712 *
## agePct65up:Cluster2          -0.206923   0.171605  -1.206 0.228081
## pctUrban:Cluster2             0.006175   0.068130   0.091 0.927791
## pctWFarmSelf:Cluster2        -0.161986   0.096727  -1.675 0.094210 .
## pctWInvInc:Cluster2           0.215233   0.269140   0.800 0.424008
## pctWRetire:Cluster2           0.027628   0.095874   0.288 0.773258
```

```
## whitePerCap:Cluster2           -0.168754   0.148925  -1.133 0.257335
## blackPerCap:Cluster2           -0.555678   0.192939  -2.880 0.004033 **
## indianPerCap:Cluster2           0.008441   0.055823   0.151 0.879825
## AsianPerCap:Cluster2            0.031298   0.047144   0.664 0.506864
## OtherPerCap:Cluster2            0.034010   0.055867   0.609 0.542774
## HispPerCap:Cluster2            -0.009366   0.080772  -0.116 0.907707
## PctNotHSGrad:Cluster2          -0.243740   0.216163  -1.128 0.259681
## PctBSorMore:Cluster2            0.129830   0.219461   0.592 0.554219
## PctUnemployed:Cluster2         -0.146644   0.122400  -1.198 0.231081
## PctEmploy:Cluster2             -0.372796   0.220084  -1.694 0.090496 .
## PctEmplManu:Cluster2           -0.172791   0.116466  -1.484 0.138122
## PctEmplProfServ:Cluster2       -0.119395   0.107265  -1.113 0.265850
## PctOccupManu:Cluster2           0.285224   0.185355   1.539 0.124066
## MalePctNevMarr:Cluster2         0.321957   0.156526   2.057 0.039871 *
## TotalPctDiv:Cluster2           -0.201605   0.148609  -1.357 0.175108
## PctTeen2Par:Cluster2           -0.065925   0.101889  -0.647 0.517710
## PctWorkMomYoungKids:Cluster2    0.069949   0.082907   0.844 0.398969
## NumImmig:Cluster2              -0.235758   0.280481  -0.841 0.400735
## PctImmigRecent:Cluster2        -0.067576   0.050256  -1.345 0.178949
## PctRecentImmig:Cluster2         0.124329   0.180123   0.690 0.490147
## PctLargHouseFam:Cluster2       -0.074506   0.289851  -0.257 0.797176
## PersPerOwnOccHous:Cluster2     -0.026542   0.198810  -0.134 0.893811
## PersPerRentOccHous:Cluster2     0.157059   0.175198   0.896 0.370149
## PctPersOwnOccup:Cluster2       -0.052142   0.174100  -0.299 0.764602
## PctHousLess3BR:Cluster2        -0.013976   0.163855  -0.085 0.932037
## MedNumBR:Cluster2              -0.009304   0.077113  -0.121 0.903979
## HousVacant:Cluster2             0.076499   0.132192   0.579 0.562880
## PctHousOccup:Cluster2           0.124628   0.100343   1.242 0.214423
## PctVacantBoarded:Cluster2       0.015544   0.070201   0.221 0.824796
## PctVacMore6Mos:Cluster2         0.055345   0.091142   0.607 0.543786
## MedYrHousBuilt:Cluster2         0.108487   0.132774   0.817 0.414015
## PctHousNoPhone:Cluster2         0.140221   0.129026   1.087 0.277317
## PctWOFullPlumb:Cluster2         0.030200   0.053488   0.565 0.572417
## MedRentPctHousInc:Cluster2     -0.146109   0.085980  -1.699 0.089465 .
## MedOwnCostPctInc:Cluster2       0.043653   0.095754   0.456 0.648535
## MedOwnCostPctIncNoMtg:Cluster2 -0.055267   0.088294  -0.626 0.531451
## PctBornSameState:Cluster2      -0.106217   0.159161  -0.667 0.504647
## PctSameHouse85:Cluster2         0.108799   0.169685   0.641 0.521501
## PctSameState85:Cluster2         0.085137   0.144484   0.589 0.555785
## LandArea:Cluster2              -0.023396   0.121594  -0.192 0.847449
## PopDens:Cluster2               -0.059164   0.162575  -0.364 0.715971
## PctUsePubTrans:Cluster2        -0.119070   0.114672  -1.038 0.299274
## LemasPctOfficDrugUn:Cluster2   -0.050373   0.034174  -1.474 0.140688
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5712 on 1490 degrees of freedom
## Multiple R-squared:  0.6848, Adjusted R-squared:  0.6626
## F-statistic: 30.83 on 105 and 1490 DF,  p-value: < 2.2e-16
```

```
# prediction
pred = predict(regc.out, XS.test)

n <- dim(XS.test)[1]
```

```r
p <- 105

rss <- sum((Y.test - pred)^2)       # Residual Sum of Squares
ess <- sum((pred - mean(Y.test))^2) # Explained Sum of Squares
tss <- ess + rss                          # Total Sum of Squares
r2 <- 1 - rss/tss                         # R Squared statistic
adjr2 <- 1 - (1-r2)*((n-1)/(n-p-1))       # adjusted R square

mse <- rss/n                              #Mean Squared Error
cat("MSE: ", round(mse,2), "\n")
```

```
## MSE:  0.38
```

```r
cat("adjR^2: ", round(adjr2,2), "\n")
```

```
## adjR^2:  0.56
```

The summary shows that there are significative slope changes for coefficients like agePct12t21, blackPerCap, pctWFarmSelf, PctEmploy, MalePctNevMarr and MedRentPctHousInc. Note that many of these are right among the more influential predictors emerged from the feature selection section. However this more complex model does not perform better than others in terms of MSE, as a matter of fact the adjusted R^2 is considerably smaller.

In conclusion, the contribution of the variable State, although it seems to have influences on the response, it does not seem to bring many improvements to the simplified model tested so far. Moreover not having considered it does not therefore seem to be problematic, making us believe that the model adopted is a good compromise between valid representation of data and simplicity.

Multiple Responses Linear Regression

We conclude the experimentation of continuous regression models testing the contemporary prediction of all the crime rates related response variables: murdPerPop, rapesPerPop, robbbPerPop, assaultPerPop, burglPerPop, larcPerPop, autoTheftPerPop, arsonsPerPop, ViolentCrimesPerPop and nonViolPerPop (with reasonings similar to those made above, logarithmic transformation was used for each of them). Note that this variables are correlated between them, and hence it is reasonably to expect that they depend on the same predictors, this is the main motivation to build such model, moreover training the regressor considering all of them could have benefit effect. As a matter of fact, the MSE results on the test are good for almost all the responses.

```r
# predict all the crime features per pop
X = subset(corrdf, select = - c(ViolentCrimesPerPop))
X.train <- subset(X, (row.names(X) %in% train.sample))
X.test <- subset(X, (row.names(X) %in% test.sample))


Ys<-cbind(log(standf["murdPerPop"]+1), log(standf["rapesPerPop"]+2),
          log(standf["robbbPerPop"]+1), log(standf["assaultPerPop"]+1),
          log(standf["burglPerPop"]+2), log(standf["larcPerPop"]+2),
          log(standf["autoTheftPerPop"]+1), log(standf["arsonsPerPop"]+1),
          log(standf["ViolentCrimesPerPop"]+1), log(standf["nonViolPerPop"]+2))

Ys.train<- as.matrix(subset(Ys, (row.names(Ys) %in% train.sample)))
Ys.test<- as.matrix(subset(Ys, (row.names(Ys) %in% test.sample)))

mm <- lm( Ys.train ~., data=X.train)
#summary(mm) #long output
```

```r
# predictions
n <- dim(XS.test)[1]
p <- 52

pred = predict(mm, X.test)
rss <- colSums((as.data.frame(Ys.test) - as.data.frame(pred))^2)# Residual Sum of Squares
ess <- colSums((pred - mean(as.matrix(Ys.test)))^2) # Explained Sum of Squares
tss <- ess + rss                          # Total Sum of Squares
r2 <- 1 - rss/tss                         # R Squared statistic
adjr2 <- 1 - (1-r2)*((n-1)/(n-p-1))       # adjusted R square

mse <- rss/n

as.data.frame(cbind(mse,adjr2))
```

```
##                           mse     adjr2
## murdPerPop          0.4075534 0.5639338
## rapesPerPop         0.1168448 0.7617004
## robbbPerPop         0.1782303 0.7430063
## assaultPerPop       0.4638742 0.5801796
## burglPerPop         0.0763295 0.8386650
## larcPerPop          0.1264317 0.7554891
## autoTheftPerPop     0.3382799 0.6339980
## arsonsPerPop        0.6726474 0.3554115
## ViolentCrimesPerPop 0.3858798 0.6781118
## nonViolPerPop       0.1089594 0.7855009
```

Binary classification models

The last section of our research is dedicated to binary classification. The capability of logistic regressor and KNN algorithm have been tested in the classification task among two level of ViolentCrimesPerPop: High and Low. In particular, we set a threshold in order to distinguish if a community has high level of crimes per population. Different options were possible for the value of this threshold such as considering the mean among all the US provided in 6, the mean or the median of our entire dataset.

For convenience in this simplistic study, we decided to proceed using the median of ViolentCrimesPerPop, the main reasons of this choice are that, first of all, it is more representative than the mean since the distribution is skewed, secondly this value split the data in balanced classes.

```r
#HLthreshold <- 0 # threshold is the mean (0 since standardized)

# threshold is the median (balanced classes)
HLthreshold <- median(standf$ViolentCrimesPerPop)

YF.train <- Y.train
YF.train[Y.train > HLthreshold] <- "High"
YF.train[!Y.train > HLthreshold] <- "Low"
YF.train <- as.factor(YF.train)

YF.test <- Y.test
YF.test[Y.test > HLthreshold] <- "High"
YF.test[!Y.test > HLthreshold]<- "Low"
YF.test <- as.factor(YF.test)
```

Logistic regression

The logistic regression is built below.

```
lreg.out<-glm(YF.train~., family = binomial, data=X.train)

# check the coding of ViolentCrimesFactor
contrasts(YF.train)
```

```
##      Low
## High   0
## Low    1
```

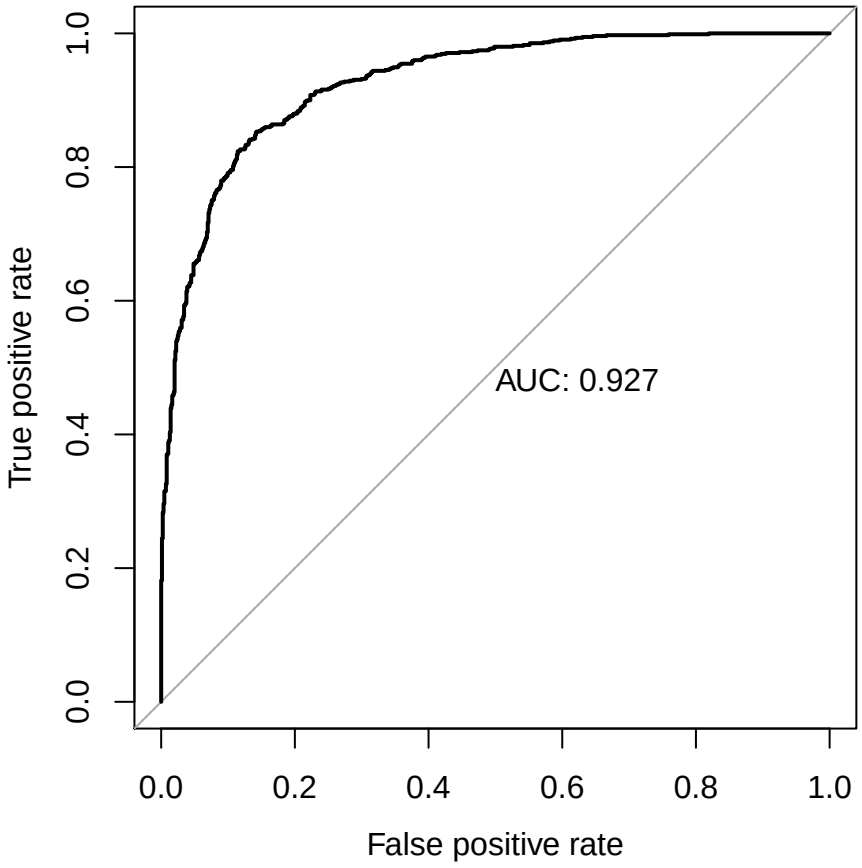
```
logistic.prob <- predict(lreg.out, type="response") # want probability
```

In order to select the best threshold for the probability, the ROC curve is plotted. The shape of the function and the AUC value confirm the acceptable quality of the model.

```
# ROC curve
# levels = controls (0's) as first element and  cases (1's) as second
roc.out <- roc(YF.train, logistic.prob, levels=c("Low", "High"), transpose = TRUE)

auc(roc.out)
```

```
## Area under the curve: 0.9272
```

```
plot(roc.out,  print.auc=TRUE, legacy.axes=TRUE, xlab="False positive rate",
     ylab="True positive rate")
```




```
# threshold that maximises the sum of sensitivity and specificity
rxy <- coords(roc.out, "best")
rxy
```

```
##   threshold specificity sensitivity
## 1 0.5459048   0.8571429   0.8531375
```

```
logistic.pred <- rep("High", 1596)
logistic.pred[logistic.prob>rxy[,1]] <- "Low"

#train accuracy
table(logistic.pred,YF.train)
```

```
##              YF.train
## logistic.pred High Low
##          High  639 121
##          Low   110 726
```

```
mean(logistic.pred==YF.train)
```

```
## [1] 0.8552632
```

```
#test accuracy
logistic.prob <- predict(lreg.out, X.test, type="response") # want probability
logistic.pred <- rep("High", 400)
```

```
logistic.pred[logistic.prob>rxy[,1]] <- "Low"

table(logistic.pred,YF.test)
```

```
##              YF.test
## logistic.pred High Low
##          High  164  30
##          Low    29 177
```

```
mean(logistic.pred==YF.test)
```

```
## [1] 0.8525
```

With this choice of the probability threshold, the model achieves good sensitivity and specificity and an accuracy on the test set of 85%.

KNN[V]

Finally we improved a completely non-parametric and non-linear approach: KNN. In this method no assumptions are made about the shape of the decision boundary, moreover differently from LDA or QDA, normality condition of the predictors are not necessary.

Strictly speaking, we first performed 10 folds cross validation to choose the best value for k, i.e. the number of closest training observations to consider to assign the majority class to a new point.

```
#10 fold cross validation
XY.train=as.data.frame(cbind(X.train,YF.train))
XY.train$YF.train<- as.factor(XY.train$YF.train)
is.factor(XY.train$YF.train)
```

```
## [1] TRUE
```

```
trControl <- trainControl(method  = "cv",number  = 10)

fit <- train(YF.train ~ .,
             method     = "knn",
             tuneGrid   = expand.grid(k = 1:10),
             trControl  = trControl,
             metric     = "Accuracy",
             data       = XY.train)

fit
```

```
## k-Nearest Neighbors
##
## 1596 samples
##   52 predictor
##    2 classes: 'High', 'Low'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1436, 1436, 1436, 1436, 1436, 1437, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.7625492  0.5232707
##    2  0.7543968  0.5067570
##    3  0.8039374  0.6054803
```

```
##    4  0.7932531  0.5841587
##    5  0.8120663  0.6222206
##    6  0.7995269  0.5972147
##    7  0.8033164  0.6046457
##    8  0.7957609  0.5897184
##    9  0.8070385  0.6122842
##   10  0.8076635  0.6136809
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

From the C.V., the best k is 5, then the model is performed with this value on the test set, returning an accuracy of 81.5%. Note that the performance is slightly poorer than the linear, confirming us the power of this simple and fully interpretable model even with a large dataset, with many possible correlated features, as the one under investigation.

```
# test accuracy
knn.pred <- knn(X.train, X.test, YF.train, k=5)
table(knn.pred,YF.test)
```

```
##         YF.test
## knn.pred High Low
##     High  158  39
##     Low    35 168
```

```
mean(knn.pred==YF.test)
```

```
## [1] 0.815
```

Results Discussion

Let's have a brief recap and see what we have learned. We started with an overall exploration of the dataset, handling, in the meanwhile, null and corrupt data, and choosing as interesting to predict among the others the proportion of violent crimes per 100K population. At this point we investigated the reletions between our target variable and the rest of the dataset, achieving the first meaningful result: the extraction, via EDA before and different statistical tools after, of a few topics which mainly determine the output of the models. In particular, the most significant clusters are identified, in descending influence order, by the family structure, the marital status, the economic status and the ethnicity of the population of each city. We then applied a multiple linear regression, and increasingly improved it checking for outliers, high leverage points, collinearity issues, and testing all kinds of variable selection methods comparing them and eventually cumulating the obtained results. We furthermore evaluated the influence of the categorical variable "state" on the prediction with ANOVA, Tukey's test and ANCOVA and we built a multiple responses L.R. to check how it would perform. At the end, for the sake of curiosity, we built two binary classification models, giving a threshold to split "ViolentCrimesPerPop" in the "High" and "Low" classes: a classical logistic regrassion and a KNN to test a non-linear and non-parametric approach.

As for the effectiveness of the model, it turned out that the binary classifiers have a rewarding accuracy of 80-85%, while the regression models reach barely sufficient results: in the Principal Component regression, even with a valuable adjusted R^2 of 65%, the logarithmic tranformation applied to the target variable led us to compute a MSLE of 0.35 on the test set, which says that the average ratio between the predicted value $+1$ and the true value $+1$ is $\sqrt{e^{\text{MSLE}}} = 1.81^{(\pm 1)}$. This is mainly due to the fact that a linear model (and in general a polinomial one, which gave similar results), is not enough to fit the complexity of the data, as we expected given the complexity of the task even from a social perspective. Furthermore, a huge limitation was the really poor quality of the LEMAS dataset, the one gathered from the police departments, and this didn't allow us to perform a parallel study to investigate a more local point of view.

At the end of the story, we tried to apply our knowledge in statistical learning to reach a satisfying level of comprehension of our data and the structures behind them, realizing that this is a task that goes beyond the concatenation of mathematical models, involving creativity, newness and dedication. The Communities and Crime data set has been a challenge for us, and only now that we know more about the data and the questions they may hide, we can eventually afford to apply deep learning models to see if the results can be improved.

Dataset Appendix

Attributes details

(125 predictive, 4 non-predictive, 18 potential responses)

- communityname: Community name - not predictive - for information only (string)
- state: US state (by 2 letter postal abbreviation)(nominal)
- countyCode: numeric code for county - not predictive, and many missing values (numeric)
- communityCode: numeric code for community - not predictive and many missing values (numeric)
- fold: fold number for non-random 10 fold cross validation, potentially useful for debugging, paired tests - not predictive (numeric - integer)

Socio-economic data from 1990 US Census

(Macro areas: race, age, income, employment, economic status, marital status, family structure, immigration data and home ownership)

- population: population for community: (numeric - expected to be integer)
- householdsize: mean people per household (numeric - decimal)
- racepctblack: percentage of population that is african american (numeric - decimal)
- racePctWhite: percentage of population that is caucasian (numeric - decimal)
- racePctAsian: percentage of population that is of asian heritage (numeric - decimal)
- racePctHisp: percentage of population that is of hispanic heritage (numeric - decimal)
- agePct12t21: percentage of population that is 12-21 in age (numeric - decimal)
- agePct12t29: percentage of population that is 12-29 in age (numeric - decimal)
- agePct16t24: percentage of population that is 16-24 in age (numeric - decimal)
- agePct65up: percentage of population that is 65 and over in age (numeric - decimal)
- numbUrban: number of people living in areas classified as urban (numeric - expected to be integer)
- pctUrban: percentage of people living in areas classified as urban (numeric - decimal)
- medIncome: median household income (numeric - may be integer)
- pctWWage: percentage of households with wage or salary income in 1989 (numeric - decimal)
- pctWFarmSelf: percentage of households with farm or self employment income in 1989 (numeric - decimal)
- pctWInvInc: percentage of households with investment / rent income in 1989 (numeric - decimal)
- pctWSocSec: percentage of households with social security income in 1989 (numeric - decimal)
- pctWPubAsst: percentage of households with public assistance income in 1989 (numeric - decimal)
- pctWRetire: percentage of households with retirement income in 1989 (numeric - decimal)
- medFamInc: median family income (differs from household income for non-family households) (numeric - may be integer)
- perCapInc: per capita income (numeric - decimal)
- whitePerCap: per capita income for caucasians (numeric - decimal)
- blackPerCap: per capita income for african americans (numeric - decimal)
- indianPerCap: per capita income for native americans (numeric - decimal)
- AsianPerCap: per capita income for people with asian heritage (numeric - decimal)
- OtherPerCap: per capita income for people with 'other' heritage (numeric - decimal)
- HispPerCap: per capita income for people with hispanic heritage (numeric - decimal)
- NumUnderPov: number of people under the poverty level (numeric - expected to be integer)
- PctPopUnderPov: percentage of people under the poverty level (numeric - decimal)
- PctLess9thGrade: percentage of people 25 and over with less than a 9th grade education (numeric - decimal)
- PctNotHSGrad: percentage of people 25 and over that are not high school graduates (numeric - decimal)
- PctBSorMore: percentage of people 25 and over with a bachelors degree or higher education (numeric - decimal)
- PctUnemployed: percentage of people 16 and over, in the labor force, and unemployed (numeric - decimal)

- PctEmploy: percentage of people 16 and over who are employed (numeric - decimal)
- PctEmplManu: percentage of people 16 and over who are employed in manufacturing (numeric - decimal)
- PctEmplProfServ: percentage of people 16 and over who are employed in professional services (numeric - decimal)
- PctOccupManu: percentage of people 16 and over who are employed in manufacturing (numeric - decimal)
- PctOccupMgmtProf: percentage of people 16 and over who are employed in management or professional occupations (numeric - decimal)
- MalePctDivorce: percentage of males who are divorced (numeric - decimal)
- MalePctNevMarr: percentage of males who have never married (numeric - decimal)
- FemalePctDiv: percentage of females who are divorced (numeric - decimal)
- TotalPctDiv: percentage of population who are divorced (numeric - decimal)
- PersPerFam: mean number of people per family (numeric - decimal)
- PctFam2Par: percentage of families (with kids) that are headed by two parents (numeric - decimal)
- PctKids2Par: percentage of kids in family housing with two parents (numeric - decimal)
- PctYoungKids2Par: percent of kids 4 and under in two parent households (numeric - decimal)
- PctTeen2Par: percent of kids age 12-17 in two parent households (numeric - decimal)
- PctWorkMomYoungKids: percentage of moms of kids 6 and under in labor force (numeric - decimal)
- PctWorkMom: percentage of moms of kids under 18 in labor force (numeric - decimal)
- NumKidsBornNeverMar: number of kids born to never married (numeric - expected to be integer)
- PctKidsBornNeverMar: percentage of kids born to never married (numeric - decimal)
- NumImmig: total number of people known to be foreign born (numeric - expected to be integer)
- PctImmigRecent: percentage of *immigrants* who immigated within last 3 years (numeric - decimal)
- PctImmigRec5: percentage of *immigrants* who immigated within last 5 years (numeric - decimal)
- PctImmigRec8: percentage of *immigrants* who immigated within last 8 years (numeric - decimal)
- PctImmigRec10: percentage of *immigrants* who immigated within last 10 years (numeric - decimal)
- PctRecentImmig: percent of *population* who have immigrated within the last 3 years (numeric - decimal)
- PctRecImmig5: percent of *population* who have immigrated within the last 5 years (numeric - decimal)
- PctRecImmig8: percent of *population* who have immigrated within the last 8 years (numeric - decimal)
- PctRecImmig10: percent of *population* who have immigrated within the last 10 years (numeric - decimal)
- PctSpeakEnglOnly: percent of people who speak only English (numeric - decimal)
- PctNotSpeakEnglWell: percent of people who do not speak English well (numeric - decimal)
- PctLargHouseFam: percent of family households that are large (6 or more) (numeric - decimal)
- PctLargHouseOccup: percent of all occupied households that are large (6 or more people) (numeric - decimal)
- PersPerOccupHous: mean persons per household (numeric - decimal)
- PersPerOwnOccHous: mean persons per owner occupied household (numeric - decimal)
- PersPerRentOccHous: mean persons per rental household (numeric - decimal)
- PctPersOwnOccup: percent of people in owner occupied households (numeric - decimal)
- PctPersDenseHous: percent of persons in dense housing (more than 1 person per room) (numeric - decimal)
- PctHousLess3BR: percent of housing units with less than 3 bedrooms (numeric - decimal)
- MedNumBR: median number of bedrooms (numeric - decimal)
- HousVacant: number of vacant households (numeric - expected to be integer)
- PctHousOccup: percent of housing occupied (numeric - decimal)
- PctHousOwnOcc: percent of households owner occupied (numeric - decimal)
- PctVacantBoarded: percent of vacant housing that is boarded up (numeric - decimal)
- PctVacMore6Mos: percent of vacant housing that has been vacant more than 6 months (numeric - decimal)
- MedYrHousBuilt: median year housing units built (numeric - may be integer)
- PctHousNoPhone: percent of occupied housing units without phone (in 1990, this was rare!) (numeric - decimal)
- PctWOFullPlumb: percent of housing without complete plumbing facilities (numeric - decimal)

- OwnOccLowQuart: owner occupied housing - lower quartile value (numeric - decimal)
- OwnOccMedVal: owner occupied housing - median value (numeric - decimal)
- OwnOccHiQuart: owner occupied housing - upper quartile value (numeric - decimal)
- OwnOccQrange: owner occupied housing - difference between upper quartile and lower quartile values (numeric - decimal)
- RentLowQ: rental housing - lower quartile rent (numeric - decimal)
- RentMedian: rental housing - median rent (Census variable H32B from file STF1A) (numeric - decimal)
- RentHighQ: rental housing - upper quartile rent (numeric - decimal)
- RentQrange: rental housing - difference between upper quartile and lower quartile rent (numeric - decimal)
- MedRent: median gross rent (Census variable H43A from file STF3A - includes utilities) (numeric - decimal)
- MedRentPctHousInc: median gross rent as a percentage of household income (numeric - decimal)
- MedOwnCostPctInc: median owners cost as a percentage of household income - for owners with a mortgage (numeric - decimal)
- MedOwnCostPctIncNoMtg: median owners cost as a percentage of household income - for owners without a mortgage (numeric - decimal)
- NumInShelters: number of people in homeless shelters (numeric - expected to be integer)
- NumStreet: number of homeless people counted in the street (numeric - expected to be integer)
- PctForeignBorn: percent of people foreign born (numeric - decimal)
- PctBornSameState: percent of people born in the same state as currently living (numeric - decimal)
- PctSameHouse85: percent of people living in the same house as in 1985 (5 years before) (numeric - decimal)
- PctSameCity85: percent of people living in the same city as in 1985 (5 years before) (numeric - decimal)
- PctSameState85: percent of people living in the same state as in 1985 (5 years before) (numeric - decimal)

Law enforcement data from the 1990 US LEMAS survey

- LemasSwornFT: number of sworn full time police officers (numeric - expected to be integer)
- LemasSwFTPerPop: sworn full time police officers per 100K population (numeric - decimal)
- LemasSwFTFieldOps: number of sworn full time police officers in field operations (on the street as opposed to administrative etc) (numeric - expected to be integer)
- LemasSwFTFieldPerPop: sworn full time police officers in field operations (on the street as opposed to administrative etc) per 100K population (numeric - decimal)
- LemasTotalReq: total requests for police (numeric - expected to be integer)
- LemasTotReqPerPop: total requests for police per 100K popuation (numeric - decimal)
- PolicReqPerOffic: total requests for police per police officer (numeric - decimal)
- PolicPerPop: police officers per 100K population (numeric - decimal)
- RacialMatchCommPol: a measure of the racial match between the community and the police force. High values indicate proportions in community and police force are similar (numeric - decimal)
- PctPolicWhite: percent of police that are caucasian (numeric - decimal)
- PctPolicBlack: percent of police that are african american (numeric - decimal)
- PctPolicHisp: percent of police that are hispanic (numeric - decimal)
- PctPolicAsian: percent of police that are asian (numeric - decimal)
- PctPolicMinor: percent of police that are minority of any kind (numeric - decimal)
- OfficAssgnDrugUnits: number of officers assigned to special drug units (numeric - expected to be integer)
- NumKindsDrugsSeiz: number of different kinds of drugs seized (numeric - expected to be integer)
- PolicAveOTWorked: police average overtime worked (numeric - decimal)
- LandArea: land area in square miles (numeric - decimal)
- PopDens: population density in persons per square mile (numeric - decimal)
- PctUsePubTrans: percent of people using public transit for commuting (numeric - decimal)
- PolicCars: number of police cars (numeric - expected to be integer)

- PolicOperBudg: police operating budget (numeric - may be integer)
- LemasPctPolicOnPatr: percent of sworn full time police officers on patrol (numeric - decimal)
- LemasGangUnitDeploy: gang unit deployed (numeric - integer - but really nominal - 0 means NO, 10 means YES, 5 means Part Time)
- LemasPctOfficDrugUn: percent of officers assigned to drug units (numeric - decimal)
- PolicBudgPerPop: police operating budget per population (numeric - decimal)

Crimedata from the 1995 FBI UCR

- murders: number of murders in 1995 (numeric - expected to be integer) potential GOAL attribute (to be predicted)
- murdPerPop: number of murders per 100K population (numeric - decimal) potential GOAL attribute (to be predicted)
- rapes: number of rapes in 1995 (numeric - expected to be integer) potential GOAL attribute (to be predicted)
- rapesPerPop: number of rapes per 100K population (numeric - decimal) potential GOAL attribute (to be predicted)
- robberies: number of robberies in 1995 (numeric - expected to be integer) potential GOAL attribute (to be predicted)
- robbbPerPop: number of robberies per 100K population (numeric - decimal) potential GOAL attribute (to be predicted)
- assaults: number of assaults in 1995 (numeric - expected to be integer) potential GOAL attribute (to be predicted)
- assaultPerPop: number of assaults per 100K population (numeric - decimal) potential GOAL attribute (to be predicted)
- burglaries: number of burglaries in 1995 (numeric - expected to be integer) potential GOAL attribute (to be predicted)
- burglPerPop: number of burglaries per 100K population (numeric - decimal) potential GOAL attribute (to be predicted)
- larcenies: number of larcenies in 1995 (numeric - expected to be integer) potential GOAL attribute (to be predicted)
- larcPerPop: number of larcenies per 100K population (numeric - decimal) potential GOAL attribute (to be predicted)
- autoTheft: number of auto thefts in 1995 (numeric - expected to be integer) potential GOAL attribute (to be predicted)
- autoTheftPerPop: number of auto thefts per 100K population (numeric - decimal) potential GOAL attribute (to be predicted)
- arsons: number of arsons in 1995 (numeric - expected to be integer) potential GOAL attribute (to be predicted)
- arsonsPerPop: number of arsons per 100K population (numeric - decimal) potential GOAL attribute (to be predicted)
- ViolentCrimesPerPop: total number of violent crimes per 100K popuation (numeric - decimal) GOAL attribute (to be predicted)
- nonViolPerPop: total number of non-violent crimes per 100K popuation (numeric - decimal) potential GOAL attribute (to be predicted)

Technical appendix

[I] Mean Squared Logarithmic Error

The logarithmic transformation $\tilde{y} = \log(y+1)$ applied to the target variable makes the MSE take the form $\frac{\sum(\log(y_T+1)-\log(y_P+1))^2}{n} = \frac{\sum(\log(\frac{y_T+1}{y_P+1}))^2}{n}$, resulting in a measure that only cares about the percentual difference between the true and the predicted values. Moreover, in our case we don't want large errors to be significantly more penalized than small ones, due to the presence of cities with significantly high values with respect to the mean making the range of the target value large. To read more about the Mean Squared Logarithmic Error, see [9].

[II] Ridge regression

Ridge regression is pretty similar to the MLR, except that the coefficients are not estimated by minimizing the usual $RSS = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2$ but the modified expression $RSS + \lambda\sum_{j=1}^{p}\beta_j^2$ where $\lambda \geq 0$ can be tuned in order to *shrink* the estimates of the β_i (but not the intercept!) to 0. To read more about Ridge regression, see [1], 215.

[III] Lasso

The lasso is an alternative to the Ridge regression that takes into account its main disadvantage: Ridge regression, in fact, shrinks all the coefficients towards 0 never pulling them down to actually 0. The only difference for the Lasso is the choice of an L^1 normalization function, which is known by theory to be a **sparse metric**, allowing the method to perform an automatic variable selection. To read more about LASSO regression, see [1], 219.

[IV] Principal Component Analysis

PCA is a technique for reducing the dimension of a n×p data matrix X. In our case, we apply it to our dataframe matrix, in order to perform, somehow, a variable selection. Somehow in the sense that the variables are not just selected, but linearly combined in new ones which are chosen orthogonal and sorted by the variance of the model they can explain, assuming it as a synonym of information. The first principal component direction, in particular, is that along which the observations vary the most, and so are the next ones among the left orthogonal directions left. To read more about the linear algebra involved in the PCA, see [10].

[V] K-Nearest Neighbors classifier

A KNN is a non-parametric and non-linear method which offers a good alternative when linear models are not suitable. In KNN classification, the output is a class membership. An object is assigned to the most common class among its K nearest neighbors, where K is a positive integer. Note that the results is strongly local, because it just depends on the nearest K training instances to the target point. To read more about KNN, see [1], 39.

References

[1] “An Introduction to Statistical Learning”, G. James, D. Witten, T. Hastie and R. Tibshirani, Springer, 2013.

[2] “DEA History Book, 1876–1990” (drug usage & enforcement), US Department of Justice, 1991, USDoJ.gov, webpage: DoJ-DEA-History-1985-1990.

[3] “Guns and Violence: The Enduring Impact of Crack Cocaine Markets on Young Black Males”, W.N. Evans, G. Garthwaite, T. Moore, 2018.

[4] “Measuring Crack Cocaine and Its Impact”, Fryer, Roland. Harvard University Society of Fellows: 3, 66. Retrieved January 4, 2016.

[5] “The New Jim Crow: Mass Incarceration in the Age of Colorblindness”, M. Alexander.

[6] http://www.disastercenter.com/crime/uscrime.htm

[7] https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime+Unnormalized

[8] https://online.stat.psu.edu/stat462/node/171/.

[9] https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/mean-squared-logarithmic-error-(msle)

[10] https://towardsdatascience.com/the-mathematics-behind-principal-component-analysis-fff2d7f4b643

Citation Request

U. S. Department of Commerce, Bureau of the Census, Census Of Population And Housing 1990 United States: Summary Tape File 1a & 3a (Computer Files),

U.S. Department Of Commerce, Bureau Of The Census Producer, Washington, DC and Inter-university Consortium for Political and Social Research Ann Arbor, Michigan. (1992)

U.S. Department of Justice, Bureau of Justice Statistics, Law Enforcement Management And Administrative Statistics (Computer File) U.S. Department Of Commerce, Bureau Of The Census Producer, Washington, DC and Inter-university Consortium for Political and Social Research Ann Arbor, Michigan. (1992)

U.S. Department of Justice, Federal Bureau of Investigation, Crime in the United States (Computer File) (1995)