# SL integration

*Gitteam*

*8/6/2020*

Importante: gli indici di riga del cleandf originale non coincidevano con quelli del nuovo (che vengono assegnati da 1 a 1996, nuova dimensione). Per coerenza ho preferito risalvarlo e importarlo con row.names=TRUE.

```
#write.csv(cleandf, "crimedata-cleaned.csv", row.names=TRUE)
# import cleandf dataset
cleandf <- read.csv("crimedata-cleaned.csv", row.names = 1) # use the first column as index
```

## Std/Norm

```
# Standardization using an home made function
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)
```

```
# Normalization using an home made function
normalization <- function(x) {
return ((x - min(x)) / (max(x) - min(x)))
}

normdf <- cleandf
normdf[seq(3,dim(normdf)[2])] <- lapply(cleandf[seq(3,dim(cleandf)[2])], normalization)
```

## High cov

```
# columns with correlation with more meaningful ones higher than threshold in absolute value
rem9 <- c(1,9,11,17,20,21,28,30,38,39,41,44,45,50,54,56,58,59,60,61,64,65,74,80,82,83,85,86,90,91,92)
rem8 <- c(rem9,2,4,8,13,14,18,29,43,46,49,51,55,62,69,81,84,95)
rem7 <- c(rem8,16,31,32,63,70,71,96)

cleandf <- cleandf[-(rem7+2)] # execute just once!
standf <- standf[-(rem7+2)] # execute just once!
normdf <- normdf[-(rem7+2)] # execute just once!
```

##Out

```
# Remove row if one of its value is
# 3 times greater than the upper interquartile bound or
# 3 times lower than the lower interquartile bound

is.out.IQR <- function(x){
  Q <- quantile(x, probs=c(.25, .75))
  iqr <- IQR(x)
  up <-  Q[2]+3*iqr # Upper bound
  low <- Q[1]-3*iqr # Lower bound
```

```r
  out <- (x < low | x > up)
  return(out)
}

is.out.sd <- function(x){
  sd <- sd(x)
  up <-  mean(x)+3*sd # Upper bound
  low <- mean(x)-3*sd # Lower bound
  out <- (x < low | x > up)
  return(out)
}


osds <- standf
osds[seq(3,dim(standf)[2])] <- lapply(standf[seq(3,dim(standf)[2])], is.out.IQR)

osds <- inspect.na(osds, hist=FALSE, byrow=TRUE, barplot=FALSE, na.value=TRUE)
rowtodrop_s <- osds$row_name

standf_nO <- standf[!(rownames(standf) %in% rowtodrop_s),]


osdn <- normdf
osdn[seq(3,dim(normdf)[2])] <- lapply(normdf[seq(3,dim(normdf)[2])], is.out.IQR)

osdn <- inspect.na(osdn, hist=FALSE, byrow=TRUE, barplot=FALSE, na.value=TRUE)
rowtodrop_n <- osdn$row_name

normdf_nO <- normdf[!(rownames(normdf) %in% rowtodrop_n),]
```

## Split in Train and Test / validation Folds

Remove non predictive variables: communityname, state, all crimes except variable ViolentCrimesPerPop

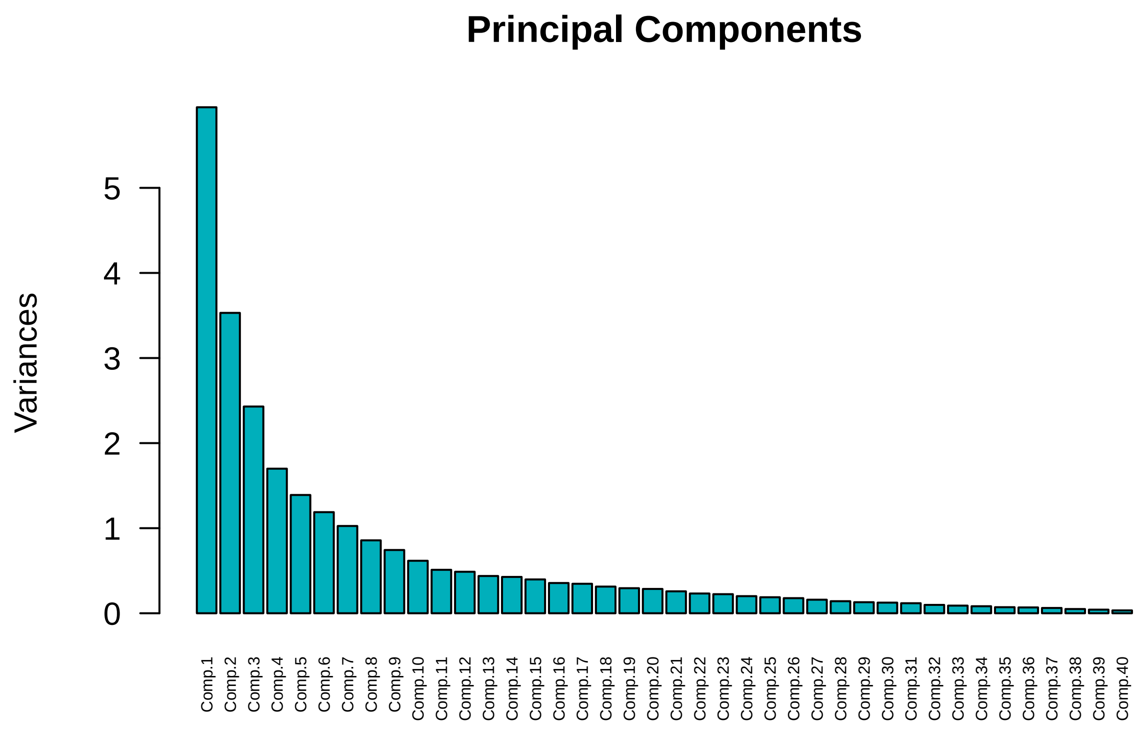
```r
coltodrop <- c(1,2,seq(48,63))
# da togliere dopo la scelta, adattando i nomi a seguire
df <- standf_nO # standf, normdf, standf_nO, normdf_nO
```

```r
train <- sample(rownames(cleandf), 1600) # leave as strings to select original indexes

Xy_train <- df[(rownames(df) %in% train),-coltodrop]
#y_train_V <- df[(rownames(df) %in% train),'ViolentCrimesPerPop']
#y_train_nV <- df[(rownames(df) %in% train),'nonViolPerPop']
Xy_test <- df[!(rownames(df) %in% train),-coltodrop]
#y_test_V <- df[!(rownames(df) %in% train),'ViolentCrimesPerPop']
#y_test_nV <- df[!(rownames(df) %in% train),'nonViolPerPop']
```

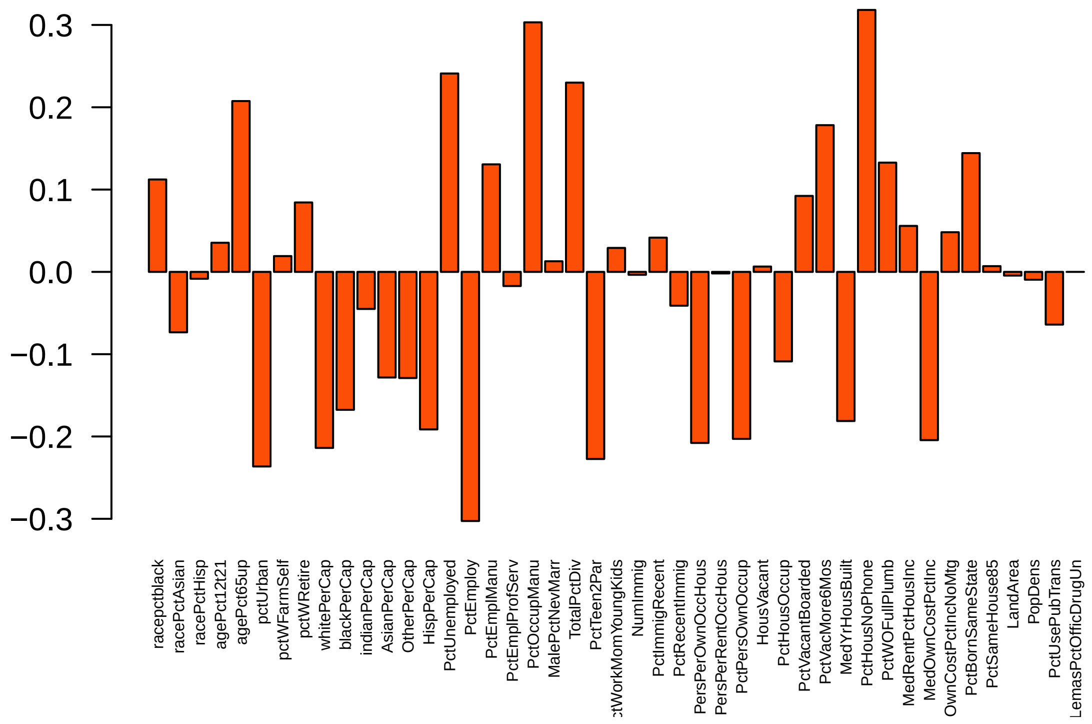
## PCA comparison

```r
# PCA computation
pc <- princomp(Xy_train[-c(46,47)]) # cor=TRUE to obtain it from the correlation matrix (use it for the

#str(pc)
```

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



```r
barplot(pc$loadings[,1], cex.names=0.5,las=2, col=  "#FC4E07")
```



```r
infl1 <- colnames(Xy_train[-c(46,47)])[abs(pc$loadings[,1])>0.15]
infl1
```

```
##  [1] "agePct65up"        "pctUrban"          "whitePerCap"
##  [4] "blackPerCap"       "HispPerCap"        "PctUnemployed"
##  [7] "PctEmploy"         "PctOccupManu"      "TotalPctDiv"
## [10] "PctTeen2Par"       "PersPerOwnOccHous" "PctPersOwnOccup"
## [13] "PctVacMore6Mos"    "MedYrHousBuilt"    "PctHousNoPhone"
## [16] "MedOwnCostPctInc"
```

As expected, normalizing the data as the researchers did in the online work (evitiamo di scriverlo o no?) totally messes up the information contained in the dataset, as it is clearly shown by the PCA. From now on we'll then go on with the classically standardized dataframe.

Multiple Linear Regression

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

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

```
## RSE: 0.4
```

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

```
## R^2: 0.57
```

```r
# mean squared error on train
y.pred <- predict(reg.out, newdata=Xy_train)
```

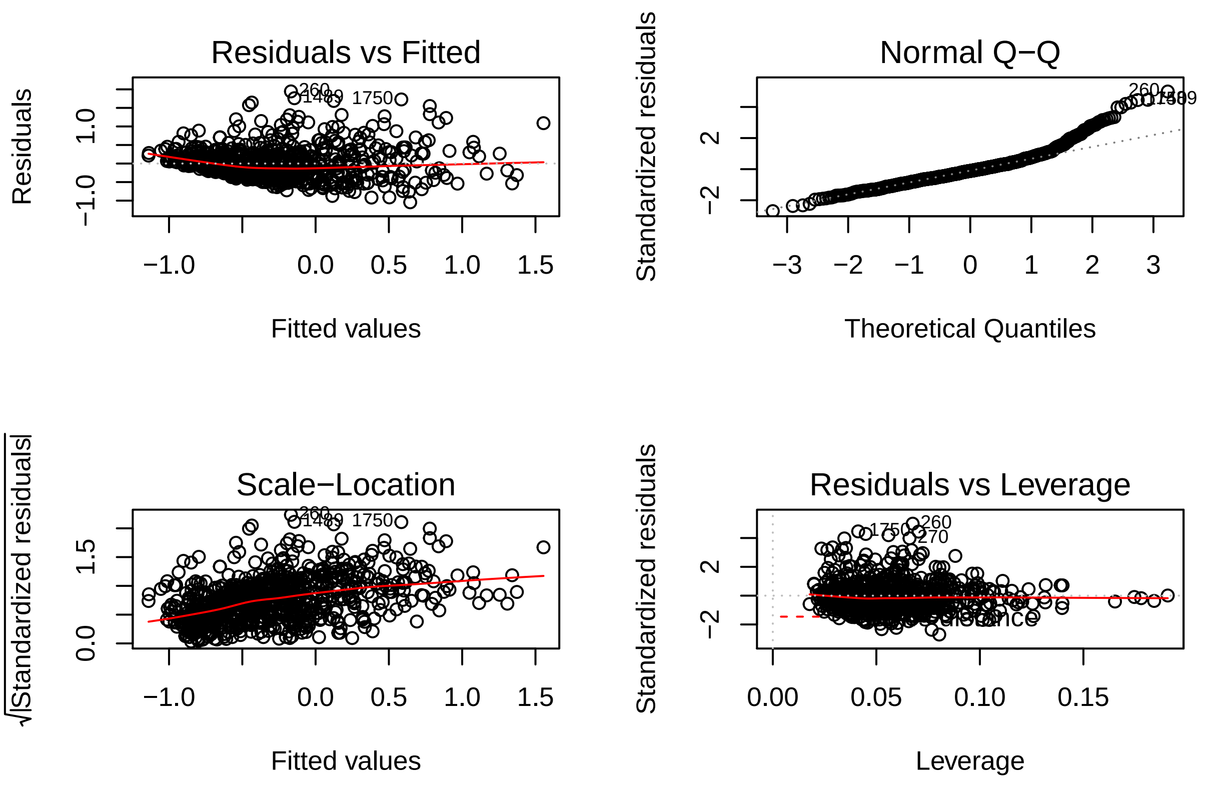
```
MSE <- mean((Xy_train$ViolentCrimesPerPop-y.pred)^2)
MSE
```

```
## [1] 0.1534414
```

```
# mean squared error on test
y.pred <- predict(reg.out, newdata=Xy_test)
MSE <- mean((Xy_test$ViolentCrimesPerPop-y.pred)^2)
MSE
```

```
## [1] 0.1042436
```

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



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

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

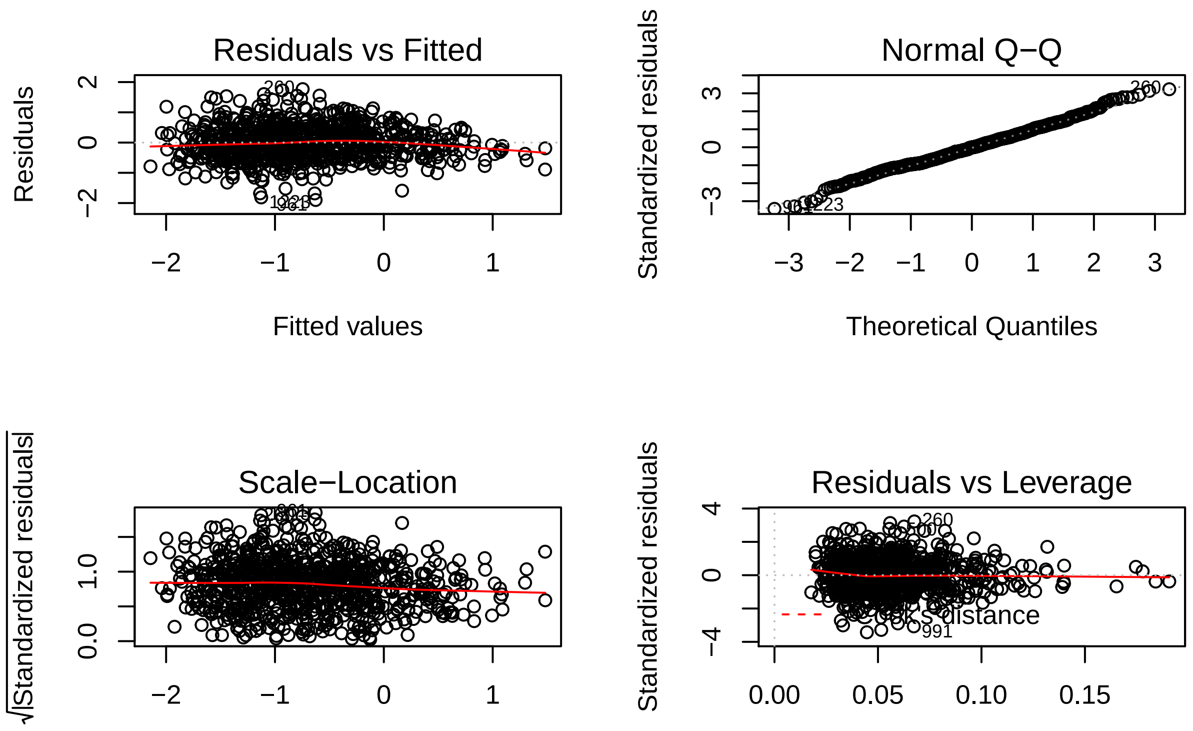
```
se  <- summary(reg.out2)$sigma        # se
rsq <- summary(reg.out2)$r.squared    # R^2
cat("RSE:", round(se,2), "\n")
```

```
## RSE: 0.57
```

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

```
## R^2: 0.58
```

```
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.

```
# mean squared error on train
y.pred <- predict(reg.out2, newdata=Xy_train)
MSE <- mean((log(Xy_train$ViolentCrimesPerPop+1)-y.pred)^2)
MSE
```

```
## [1] 0.3033649
```

```
# mean squared error on test
y.pred <- predict(reg.out2, newdata=Xy_test)
MSE <- mean((log(Xy_test$ViolentCrimesPerPop+1)-y.pred)^2)
MSE
```

```
## [1] 0.2439829
```