HomeWork 6

HomeWork 6

ISYE 6501

Question 9.1

Using the same crime data set uscrime.txt as in Question 8.2, apply Principal Component Analysis and then create a regression model using the first few principal components. Specify your new model in terms of the original variables (not the principal components), and compare its quality to that of your solution to Question 8.2. You can use the R function prcomp for PCA. (Note that to first scale the data, you can include scale. = TRUE to scale as part of the PCA function. Don't forget that, to make a prediction for the new city, you'll need to unscale the coefficients (i.e., do the scaling calculation in reverse)!)

```
library(kernlab)
library(kknn)
library(lattice)
library(ggplot2)
library(outliers)
library(dplyr)

library(leaps)
```

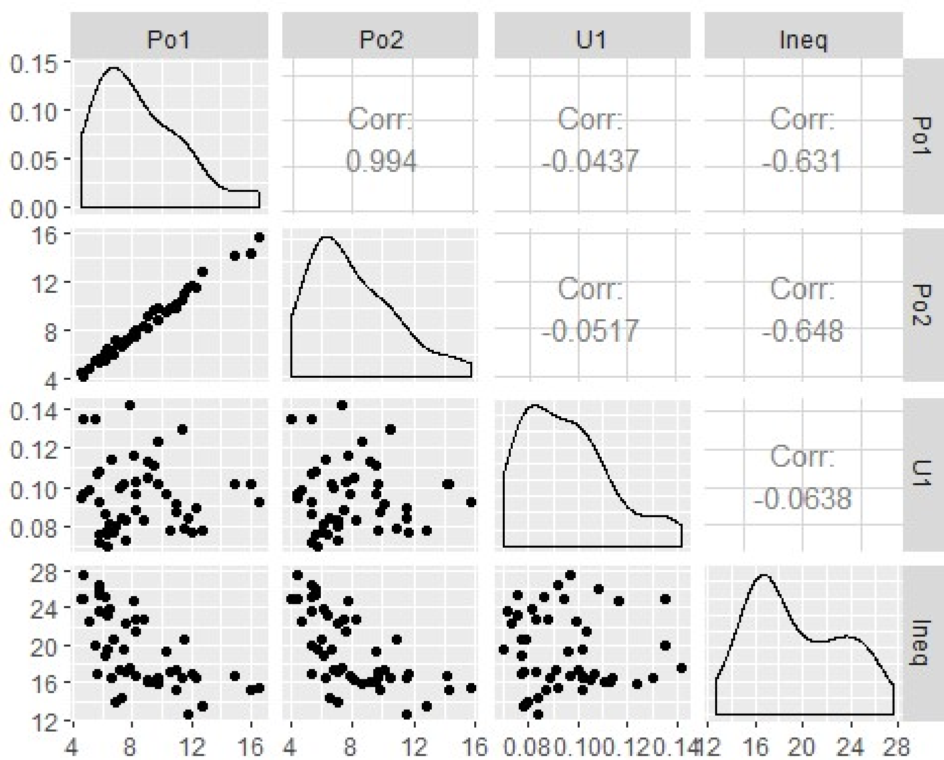
Next we will load the data and look at the data structure.

```
rm(list=ls())
uscrime <- read.table("uscrime.txt",stringsAsFactors = FALSE, header = TRUE)
head(uscrime)

##      M So   Ed  Po1  Po2    LF   M.F Pop   NW    U1  U2 Wealth Ineq     Pr
ob
## 1 15.1  1  9.1  5.8  5.6 0.510  95.0  33 30.1 0.108 4.1   3940 26.1 0.0846
02
## 2 14.3  0 11.3 10.3  9.5 0.583 101.2  13 10.2 0.096 3.6   5570 19.4 0.0295
99
## 3 14.2  1  8.9  4.5  4.4 0.533  96.9  18 21.9 0.094 3.3   3180 25.0 0.0834
01
## 4 13.6  0 12.1 14.9 14.1 0.577  99.4 157  8.0 0.102 3.9   6730 16.7 0.0158
01
## 5 14.1  0 12.1 10.9 10.1 0.591  98.5  18  3.0 0.091 2.0   5780 17.4 0.0413
99
## 6 12.1  0 11.0 11.8 11.5 0.547  96.4  25  4.4 0.084 2.9   6890 12.6 0.0342
01
##      Time Crime
## 1 26.2011   791
## 2 25.2999  1635
```

```
## 3 24.3006   578
## 4 29.9012  1969
## 5 21.2998  1234
## 6 20.9995   682
```

Lets examine the data for correlations using a visualisation.

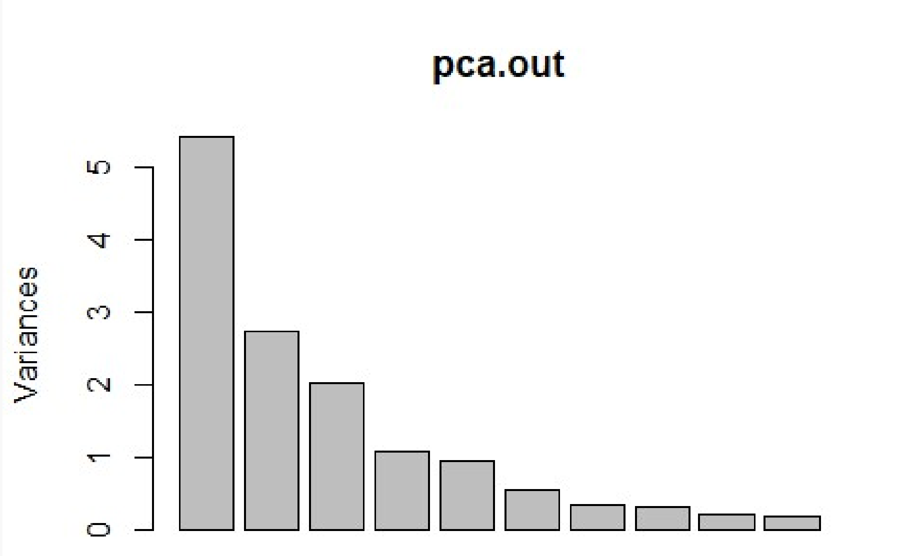
```
library (GGally)

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2

##
## Attaching package: 'GGally'

## The following object is masked from 'package:dplyr':
##
##     nasa

ggpairs(uscrime, columns = c("Po1","Po2","U1","Ineq"))
```



PCA is used when we have large number of features in the hundreds or thousands, and we cannot make a reasoable model with the limited number of data points we have. In the current data set PCA would be over kill and will probably lead to over fitting. PCA also has the added advantage of reducing correlation by giving higher priority to more relavent features. PCA doesnt perform well with binary features

```
uscrime$So <- NULL
#head(uscrime)
```

```
pca.out = prcomp(uscrime[,1:14],scale = TRUE) # prinicipal component analysis
model
pca.out$sdev   # gives standard deviations

##  [1] 2.32616392 1.65127377 1.41578394 1.03669703 0.96745274 0.74049025
##  [7] 0.56414623 0.54675066 0.44751244 0.42747346 0.35944524 0.31852077
## [13] 0.25159368 0.06802123

variance <- pca.out$sdev^2 #  get back eigenvalues

summary(pca.out)

## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC
7
## Standard deviation     2.3262 1.6513 1.4158 1.03670 0.96745 0.74049 0.5641
5
## Proportion of Variance 0.3865 0.1948 0.1432 0.07677 0.06685 0.03917 0.0227
3
## Cumulative Proportion  0.3865 0.5813 0.7244 0.80121 0.86806 0.90723 0.9299
6
##                            PC8    PC9    PC10    PC11    PC12    PC13    P
C14
## Standard deviation     0.54675 0.4475 0.42747 0.35945 0.31852 0.25159 0.06
802
## Proportion of Variance 0.02135 0.0143 0.01305 0.00923 0.00725 0.00452 0.00
033
## Cumulative Proportion  0.95132 0.9656 0.97867 0.98790 0.99515 0.99967 1.00
000

screeplot(pca.out)   # Scree plot shows variance explained per principal
```



From the grapgh and the summary looks like the first 5 Principal components have conciderably higher variance. We will use these 4 PCs in our regression model.

```
#Top 5 principal components
pca1 <- pca.out$x[,1:5]
#pca1

uscrimePC <- cbind(pca1, uscrime["Crime"])
#head(uscrimePC)
#Running a linear model
modelPCA <- lm(Crime~.,data = as.data.frame(uscrimePC))
summary(modelPCA)

##
## Call:
## lm(formula = Crime ~ ., data = as.data.frame(uscrimePC))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -439.96 -181.93    3.13  177.53  444.64
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  905.085     36.364  24.890  < 2e-16 ***
## PC1           76.750     15.802   4.857 1.77e-05 ***
## PC2          -57.648     22.260  -2.590   0.0132 *
## PC3           24.313     25.962   0.936   0.3545
## PC4            3.786     35.456   0.107   0.9155
## PC5         -235.831     37.994  -6.207 2.20e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 249.3 on 41 degrees of freedom
## Multiple R-squared:  0.6297, Adjusted R-squared:  0.5845
## F-statistic: 13.94 on 5 and 41 DF,  p-value: 5.685e-08
```

In the above model our adjusted R square is abysmal. Lets do some feature selection on the Principal components. We will use ols_subset to Select the subset of predictors have the largest R2 value or the smallest mean squared error.

```
#Get all principal components from pca output
pc_full <- pca.out$x[,]
head(pc_full)

##            PC1         PC2         PC3         PC4        PC5        PC6
## [1,] -3.893446 -1.29197714 -1.10138991  0.85371781 -0.2582188 -0.2220763
## [2,]  0.971018  0.69084709 -0.05783388 -0.36390197 -1.1311282  0.6335173
## [3,] -3.946974  0.08584861 -0.36356482  0.58382631  0.3690864 -0.7120597
## [4,]  3.951699 -2.29488126  0.22720214 -0.04741697 -1.7050714 -0.6698441
## [5,]  1.647039  1.44338543  1.25954528  0.64890563 -0.1162943  0.4327958
## [6,]  2.861367 -0.11765453  0.51823342  1.44813377  1.0732058  0.6906828
##              PC7         PC8         PC9        PC10       PC11         PC1
2
## [1,]  0.48212291  0.07191639 -0.46939569 -0.05399584 -0.2035171  0.0358941
7
## [2,] -0.20182701 -0.37733855 -0.24477852  0.02972945 -0.3578821 -0.1584916
8
```

```
## [3,]  0.01482101 -0.37633010  0.08792458 -0.40993363  0.1155926  0.3097035
2
## [4,] -0.74188485  1.42942026 -0.20222448 -0.01001613 -0.2187344 -0.1864198
0
## [5,] -0.28482158  0.38994226 -0.49201684 -0.22200390 -0.3052962  0.6775270
4
## [6,] -0.06932221 -0.70963951 -0.13164159  0.12061204  0.7156631 -0.0485135
4
```

Now that we have our PC values we need to reverse enginner to the coefficients to get the right features. Below is a work through of the math.

using a toy data set.

$$\frac{z}{\sigma_x \sigma_y} = \frac{a_x}{\sigma_y}\left(\frac{x-\mu_x}{\sigma_x}\right) + \frac{a_y}{\sigma_x}\left(\frac{y-\mu_y}{\sigma_y}\right) + \frac{I+\mu_x+\mu_y}{\sigma_x \sigma_y}$$

where a_x and a_y are x and y coefficients
I is the intercept.

$$z = a_x \sigma_x x_s + a_y \sigma_y y_s + I + \mu_x + \mu_y$$

where x_s and y_s are scaled data points

Now to apply the transformation where the rotation matrix is $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

$x' = x_s \cos\theta - y_s \sin\theta$
$y' = x_s \sin\theta + y_s \cos\theta$
here x' and y' are principle co

inversly ~~$x_s =$~~ $x_s = \cos\theta\, x' - \sin\theta\, y'$
$y_s = \sin\theta\, x' + \cos\theta\, y'$

$$z = a_x \sigma_x (\cos\theta\, x' - \sin\theta\, y') + a_y \sigma_y (\sin\theta\, x' + \cos\theta\, y') + I + \mu_x + \mu_y$$

Say from our Linear regression we get:
$z = a'_x x' + a'_y y' + I'$ where a'_x, a'_y and I' are based on PCs.

$a_x \sigma_x \cos\theta + a_y \sigma_y \sin\theta = a'_x$ and $-a_x \sigma_x \sin\theta + a_y \sigma_y \cos\theta = a'_y$

To rotate back and get a_x and a_y

$$\begin{bmatrix} \sigma_x a_x \\ \sigma_y a_y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} a'_x \\ a'_y \end{bmatrix}$$

to unscale the coefficients divide the result by Sd (σ)

In order to get the scaled coefficients in original factures we need to multipy the coefficient vector to the rotation matrix of the PCA. TO unscale the coefficient matrix we need to divide by standard deviation. For the intersept we only divide by the PC intercept by the sum of the means.

```
# Rotation vector for the first 5 PCs
rotation_vec <- pca.out$rotation[,1:5]
```

```r
# Coefficients of PCA model without the Intercept
PCA_coef <- modelPCA$coefficients[2:6]

scaled_coef <- PCA_coef%*% t(rotation_vec)
scaled_coef

##             M       Ed      Po1      Po2       LF      M.F      Pop
NW
## [1,] 64.14904 14.09834 116.4574 110.4209 76.17019 103.1387 63.33765 104.15
81
##           U1       U2   Wealth     Ineq      Prob     Time
## [1,] 3.83432 32.23305 29.57314 25.60389 -27.59591 26.85164
```

Now to get original coefficients and intercepts

```r
intercept <- modelPCA$coefficients[1]-sum(scaled_coef*sapply(uscrime[,1:14],m
ean)/pca.out$scale) #sum(sapply(uscrime[,1:15],mean))
intercept

## (Intercept)
##   -5726.224

unscaled_coef <- scaled_coef/pca.out$scale
unscaled_coef

##             M       Ed      Po1      Po2      LF      M.F      Pop        N
W
## [1,] 51.04305 12.60243 39.18621 39.49058 1884.85 35.00099 1.663664 10.1292
7
##            U1       U2     Wealth     Ineq      Prob     Time
## [1,] 212.6777 38.16618 0.03064862 6.417648 -1213.702 3.788914

matrix_data <- as.matrix(uscrime[,1:14])

# estimate is of the form estimate = aX +b
# where a are the coefficients and b is the intercept
estimates <- matrix_data %*% t(unscaled_coef) + intercept
SSE = sum((estimates - uscrime[,15])^2)
SStot = sum((uscrime[,15] - mean(uscrime[,15]))^2)
R2 <- 1 - SSE/SStot
R2

## [1] 0.6296769

# Create the test datapoint mannually ising the data.frame() function for the
new city
test_point <- data.frame(M = 14.0,Ed = 10.0 ,Po1 = 12.0 ,Po2 = 15.5,LF = 0.64
0 ,M.F = 94.0 ,Pop = 150 ,NW = 1.1 ,U1= 0.120 ,U2 = 3.6 ,Wealth = 3200,Ineq =
20.1,Prob = 0.04 ,Time = 39.0)

# Use the intercepts and coefficiets to make a prediction of Cirme in the new
city
matrix_test <- as.matrix(test_point)
```

```
prediction = matrix_test %*% t(unscaled_coef) + intercept
prediction

##          [,1]
## [1,] 1443.039
```

As we already know the crime for the new city from the previous Homework, out model is overestimating. As mentioned earlier PCA is ideally suited for large number of data points. With our 50 data points the model is over fitting. Compared to my previous model from 8.2 which gave a value of 728, this model is less accurate. The final prediction for test city using PCA is 1443 with an R-square of 0.629.

```
#Sanity check to see if our reverse PCA calculation was correct
#project new data onto PCA space and run model

pca_test <- scale(test_point, pca.out$center, pca.out$scale) %*% pca.out$rota
tion

pca_test <- as.data.frame (pca_test)
pca_test

##           PC1       PC2       PC3      PC4       PC5       PC6        PC7
## [1,] 1.161658 -2.841351 0.5694485 -1.04263 -1.166522 -2.191452 -0.4660637
##            PC8      PC9      PC10      PC11     PC12      PC13     PC14
## [1,] 0.9344984 0.227878 0.555688 -1.088542 3.504094 0.6951131 1.269102


predict(modelPCA,newdata=pca_test,type="response")

##        1
## 1443.039
```

We get the same response by projecting the data onto the PCA axis.