HW 6 ISYE

PG

2023-10-04

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)!

Answer 9.1

As always, we begin by clearing the work space and loading the dataset. Check to see if loaded and load the relevant libraries.

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

```
##      M So   Ed  Po1  Po2    LF   M.F Pop   NW    U1  U2 Wealth Ineq     Prob
## 1 15.1  1  9.1  5.8  5.6 0.510  95.0  33 30.1 0.108 4.1   3940 26.1 0.084602
## 2 14.3  0 11.3 10.3  9.5 0.583 101.2  13 10.2 0.096 3.6   5570 19.4 0.029599
## 3 14.2  1  8.9  4.5  4.4 0.533  96.9  18 21.9 0.094 3.3   3180 25.0 0.083401
## 4 13.6  0 12.1 14.9 14.1 0.577  99.4 157  8.0 0.102 3.9   6730 16.7 0.015801
## 5 14.1  0 12.1 10.9 10.1 0.591  98.5  18  3.0 0.091 2.0   5780 17.4 0.041399
## 6 12.1  0 11.0 11.8 11.5 0.547  96.4  25  4.4 0.084 2.9   6890 12.6 0.034201
##      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
```

The dataset is loaded correctly. Next, I will load the relevant libraries to examine the correlation in the data. This is a visualization of a subset of the correlation matrix of the data.

```
library(GGally)
```

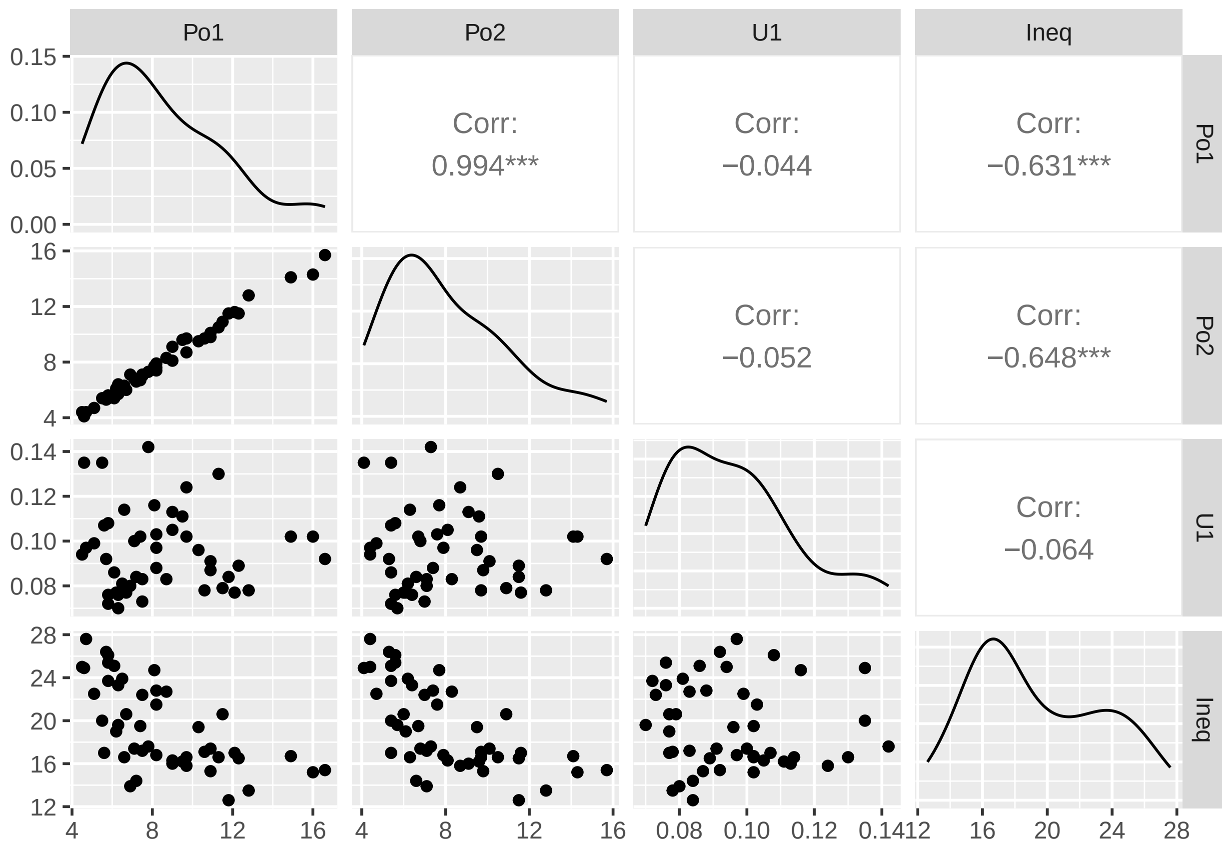
```
## Loading required package: ggplot2

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

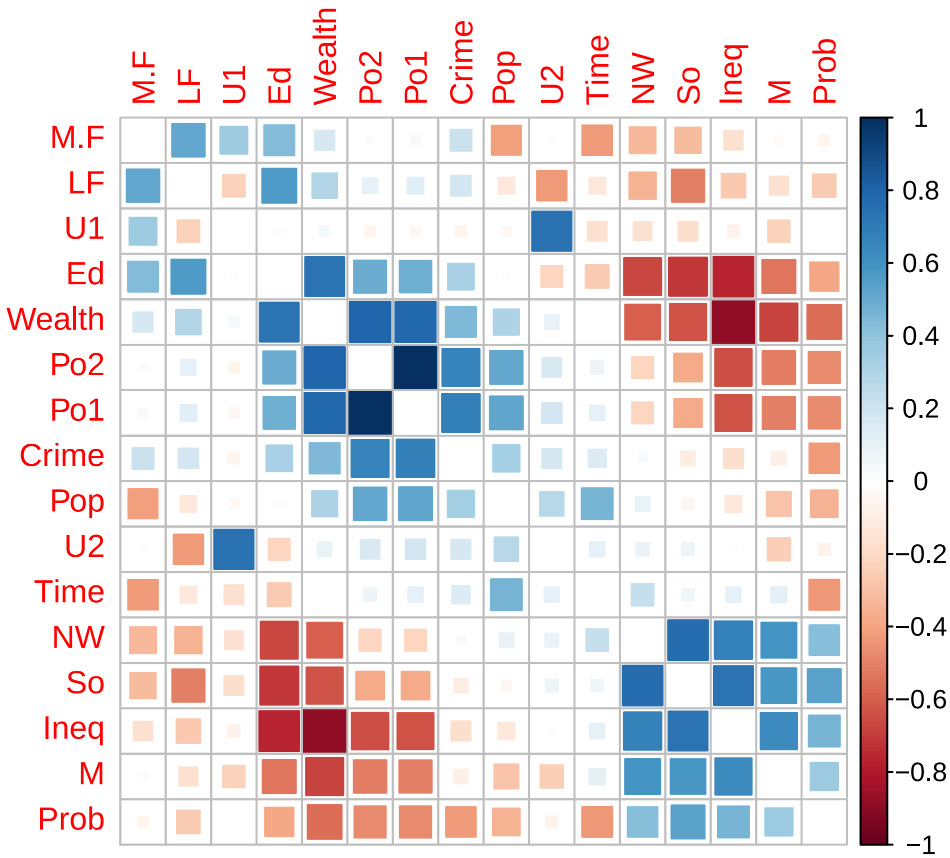
```r
library(corrplot)
```

```
## corrplot 0.92 loaded
```

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




```r
#Used AOE for angular order of eigenvectors
corrplot(cor(uscrime), method = 'square', order = 'AOE', diag = FALSE)
```



As seen from the figure above, we have multicolineraity situation. Thus, a reasonable approach would be to use Principal Component Analysis to reduce the correlation among predictors. This will help me come with a simpler model that has less chance of overfitting.

Next, running PCA on the matrix of SCALED predictors.

```
PCA <- prcomp(uscrime[,1:15], scale = TRUE)
summary(PCA)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.4534 1.6739 1.4160 1.07806 0.97893 0.74377 0.56729
## Proportion of Variance 0.4013 0.1868 0.1337 0.07748 0.06389 0.03688 0.02145
## Cumulative Proportion  0.4013 0.5880 0.7217 0.79920 0.86308 0.89996 0.92142
##                            PC8     PC9    PC10    PC11    PC12    PC13   PC14
## Standard deviation     0.55444 0.48493 0.44708 0.41915 0.35804 0.26333 0.2418
## Proportion of Variance 0.02049 0.01568 0.01333 0.01171 0.00855 0.00462 0.0039
## Cumulative Proportion  0.94191 0.95759 0.97091 0.98263 0.99117 0.99579 0.9997
##                           PC15
## Standard deviation     0.06793
## Proportion of Variance 0.00031
## Cumulative Proportion  1.00000
```

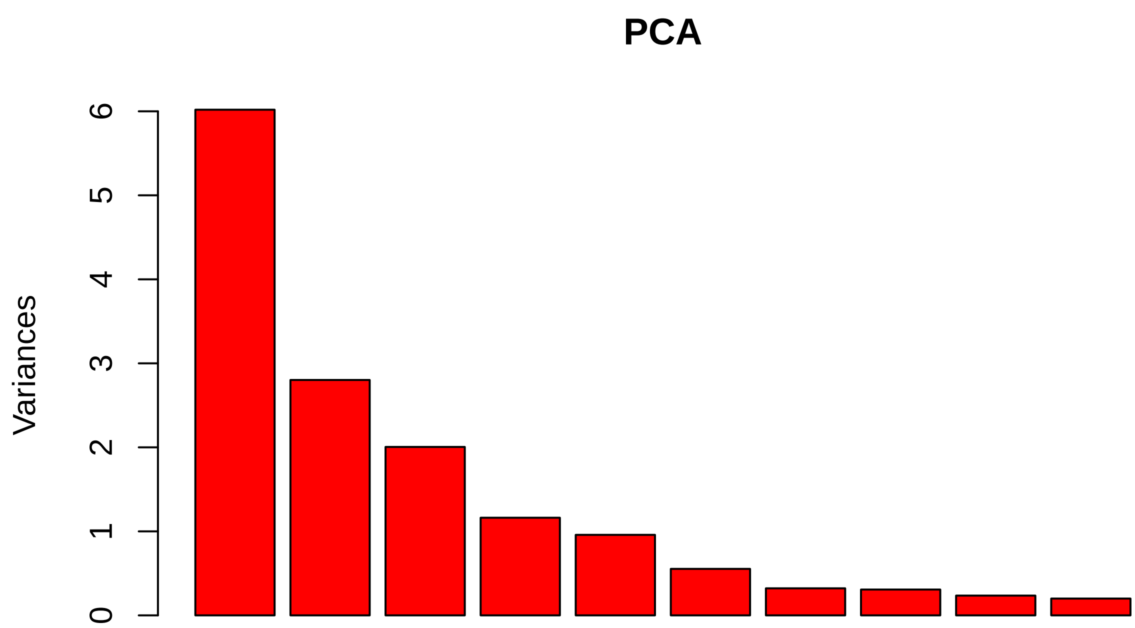
Get my matrix of eigenvectors as below using $rotation

```r
PCA$rotation
```

```
##                PC1         PC2           PC3         PC4         PC5
## M      -0.30371194  0.06280357  0.1724199946 -0.02035537 -0.35832737
## So     -0.33088129 -0.15837219  0.0155433104  0.29247181 -0.12061130
## Ed      0.33962148  0.21461152  0.0677396249  0.07974375 -0.02442839
## Po1     0.30863412 -0.26981761  0.0506458161  0.33325059 -0.23527680
## Po2     0.31099285 -0.26396300  0.0530651173  0.35192809 -0.20473383
## LF      0.17617757  0.31943042  0.2715301768 -0.14326529 -0.39407588
## M.F     0.11638221  0.39434428 -0.2031621598  0.01048029 -0.57877443
## Pop     0.11307836 -0.46723456  0.0770210971 -0.03210513 -0.08317034
## NW     -0.29358647 -0.22801119  0.0788156621  0.23925971 -0.36079387
## U1      0.04050137  0.00807439 -0.6590290980 -0.18279096 -0.13136873
## U2      0.01812228 -0.27971336 -0.5785006293 -0.06889312 -0.13499487
## Wealth  0.37970331 -0.07718862  0.0100647664  0.11781752  0.01167683
## Ineq   -0.36579778 -0.02752240 -0.0002944563 -0.08066612 -0.21672823
## Prob   -0.25888661  0.15831708 -0.1176726436  0.49303389  0.16562829
## Time   -0.02062867 -0.38014836  0.2235664632 -0.54059002 -0.14764767
##                 PC6         PC7         PC8         PC9        PC10        PC11
## M      -0.449132706 -0.15707378 -0.55367691  0.15474793 -0.01443093  0.39446657
## So     -0.100500743  0.19649727  0.22734157 -0.65599872  0.06141452  0.23397868
## Ed     -0.008571367 -0.23943629 -0.14644678 -0.44326978  0.51887452 -0.11821954
## Po1    -0.095776709  0.08011735  0.04613156  0.19425472 -0.14320978 -0.13042001
## Po2    -0.119524780  0.09518288  0.03168720  0.19512072 -0.05929780 -0.13885912
## LF      0.504234275 -0.15931612  0.25513777  0.14393498  0.03077073  0.38532827
## M.F    -0.074501901  0.15548197 -0.05507254 -0.24378252 -0.35323357 -0.28029732
## Pop     0.547098563  0.09046187 -0.59078221 -0.20244830 -0.03970718  0.05849643
## NW      0.051219538 -0.31154195  0.20432828  0.18984178  0.49201966 -0.20695666
## U1      0.017385981 -0.17354115 -0.20206312  0.02069349  0.22765278 -0.17857891
## U2      0.048155286 -0.07526787  0.24369650  0.05576010 -0.04750100  0.47021842
## Wealth -0.154683104 -0.14859424  0.08630649 -0.23196695 -0.11219383  0.31955631
## Ineq    0.272027031  0.37483032  0.07184018 -0.02494384 -0.01390576 -0.18278697
## Prob    0.283535996 -0.56159383 -0.08598908 -0.05306898 -0.42530006 -0.08978385
## Time   -0.148203050 -0.44199877  0.19507812 -0.23551363 -0.29264326 -0.26363121
##               PC12        PC13        PC14          PC15
## M       0.16580189  0.05142365  0.04901705 -0.0051398012
## So     -0.05753357  0.29368483 -0.29364512 -0.0084369230
## Ed      0.47786536 -0.19441949  0.03964277  0.0280052040
## Po1     0.22611207  0.18592255 -0.09490151  0.6894155129
## Po2     0.19088461  0.13454940 -0.08259642 -0.7200270100
## LF      0.02705134  0.27742957 -0.15385625 -0.0336823193
## M.F    -0.23925913 -0.31624667 -0.04125321 -0.0097922075
## Pop    -0.18350385 -0.12651689 -0.05326383 -0.0001496323
## NW     -0.36671707 -0.22901695  0.13227774  0.0370783671
## U1     -0.09314897  0.59039450 -0.02335942 -0.0111359325
## U2      0.28440496 -0.43292853 -0.03985736 -0.0073618948
## Wealth -0.32172821  0.14077972  0.70031840  0.0025685109
## Ineq    0.43762828  0.12181090  0.59279037 -0.0177570357
## Prob    0.15567100  0.03547596  0.04761011 -0.0293376260
## Time    0.13536989  0.05738113 -0.04488401 -0.0376754405
```

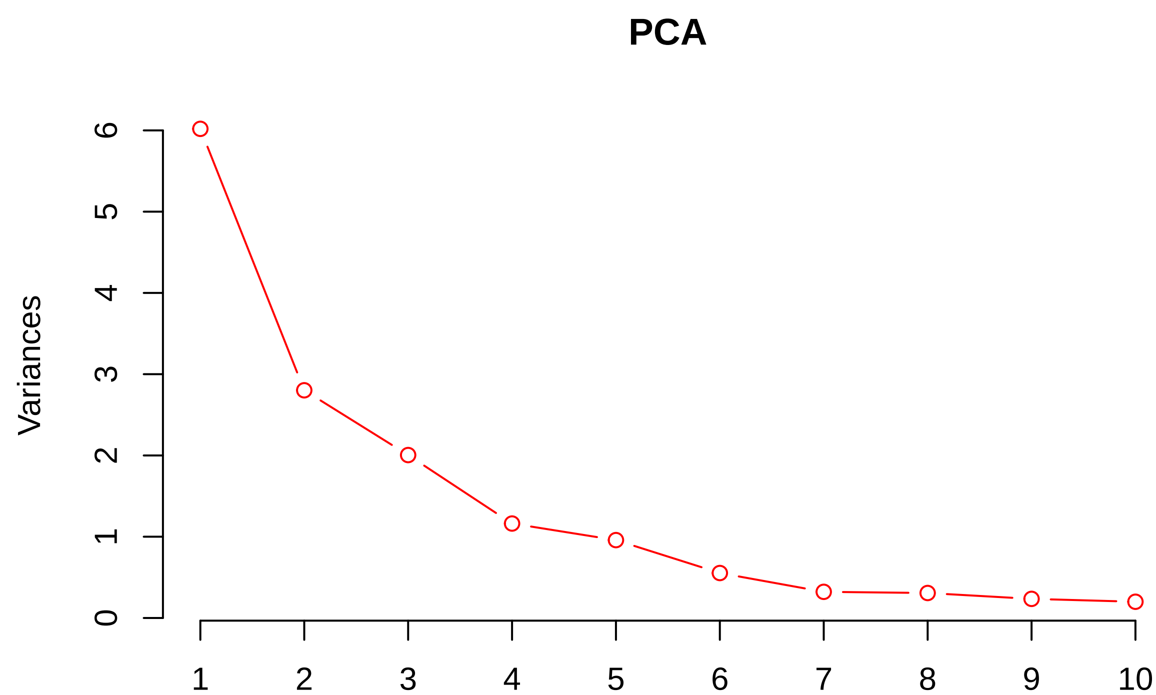
The screeplot function plots the variance of each of the principal components (where variance = pca$sdev^2) to help us decide on a number of principal components to use.

```r
screeplot(PCA, type = "barplot", col = "red")
```



The question wants us to use a few principal components. I am going to use the first four (4) as it encompasses the majority of the barplot. Here is another visualization of the same.

```r
screeplot(PCA, type = "lines", col = "red")
```



Next, I get my first 4 principal components

```
PC <- PCA$x[1:4]
PC
```

```
## [1] -4.199284  1.172663 -4.173725  3.834962
```

I now build a linear regression model with these first four principal components. Use the lm function for that purpose.

```
uscrimePC <- cbind(PC, uscrime[,16])
```

```
## Warning in cbind(PC, uscrime[, 16]): number of rows of result is not a multiple
## of vector length (arg 1)
```

```
modelPCA <- lm(V2~., data = as.data.frame(uscrimePC))
summary(modelPCA)
```

```
##
## Call:
## lm(formula = V2 ~ ., data = as.data.frame(uscrimePC))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -544.96 -275.72  -41.49  137.19 1009.04
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   940.20      55.69  16.881   <2e-16 ***
## PC             37.32      15.60   2.392    0.021 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 368.3 on 45 degrees of freedom
## Multiple R-squared:  0.1128, Adjusted R-squared:  0.09308
## F-statistic: 5.721 on 1 and 45 DF,  p-value: 0.021
```

The p-value is less than 0.05. Let us move forward with the prediction.

```r
#Performing the required mathematics
beta0 <- modelPCA$coefficients[1]
betas <- modelPCA$coefficients[2:5]
alpha <- PCA$rotation[,1:4] %*% betas  #performing matrix multiplication

#Recover the original alpha values and beta as given using the sapply function in R
mu <- lapply(uscrime[,1:15], mean)
sigma <- lapply(uscrime[,1:15], sd)
#origAlpha <- alpha/sigma
#origBeta0 <- beta0 - sum(alpha*mu/sigma)

#Calculate estimates
#estimates <- as.matrix(uscrime[,1:15]) %*% origAlpha + origBeta0

#Use estimates to observe the model accuracy
#SSE = sum((estimates - uscrime[,16])^2)
#SStot = sum((uscrime[,16] - mean(uscrime[,16]))^2)
#R2 <- 1 - SSE/SStot
#R2
#0.675
```

The R2 value is 0.675 - much less than last week's results.

Now applying the new city data from last week's homework to see what the predicted crime rate is.

```r
new_city <- data.frame(M= 14.0, So = 0, Ed = 10.0, Po1 = 12.0, Po2 = 15.5,
                    LF = 0.640, M.F = 94.0, Pop = 150, NW = 1.1, U1 = 0.120, U2 = 3.6, Wealth = 3200, In

pred_df <- data.frame(predict(PCA, new_city))
pred <- predict(modelPCA, pred_df)
```

```
## Warning: 'newdata' had 1 row but variables found have 4 rows
```

```r
pred
```

```
##         1         2         3         4
##  783.4868  983.9567  784.4406 1083.3081
```

I get a range of 4 values as predictions. The low is 783 while the high is 1083.3. All these predicted values are in the ballpark of what I hoped to predict given last week's predictions of 987.

Extension Ideas:

1. I see that the "So" column has binary data (1 vs 0). PCA doesn't generally work well with binary data. I would remove the binary factor when doing the PCA second time to see if I get better predictions.
2. I think my model could do better if I used more principal components. Maybe 6 or 7 (instead of 4 or 5) based on the screeplots.