Crime Rate Prediction using PCA and Regression

Alena Fedash

2022-09-30

Contents

Please find below my step-by-step solution in R with explanations, comments and conclusions for each step.

Step 0: Load the libraries

```
library(dplyr)
library(tidyverse)
library(dslabs)
library(data.table)
library(ggplot2)
library(plotly)
library(outliers)
library(qcc)
library(mctest)
library(ppcor)
library(car)
library(psych)
library(ggthemes)
library(corrplot)
library(DAAG)
library(GGally)
library(caret)
library(psych)
library(ggpubr)
```

Step 1: Load the dataset

```
data <- read.table("uscrime.txt",
                   header = TRUE,
                   stringsAsFactors = FALSE,
                   sep = "",
                   dec = ".")
head(data)
```

```
##      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
```

```
test_data <- 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,
                        Ineq = 20.1,
                        Prob = 0.040,
                        Time = 39.0)
test_data
```

```
##    M So Ed Po1  Po2   LF M.F Pop  NW   U1  U2 Wealth Ineq Prob Time
## 1 14  0 10  12 15.5 0.64  94 150 1.1 0.12 3.6   3200 20.1 0.04   39
```

Step 2: Basic Explorations

From previous analysis, we already know that there are no NA values in the data frame. Let's check a short summary for each variable:

```
describe.by(data)
```

```
## Warning: describe.by is deprecated.  Please use the describeBy function
```

```
## Warning in describeBy(x = x, group = group, mat = mat, type = type, ...): no
## grouping variable requested
```

```
##        vars  n    mean     sd  median trimmed     mad     min     max   range
```

```
## M         1 47   13.86   1.26   13.60   13.75    1.19   11.90   17.70    5.80
## So        2 47    0.34   0.48    0.00    0.31    0.00    0.00    1.00    1.00
## Ed        3 47   10.56   1.12   10.80   10.59    1.19    8.70   12.20    3.50
## Po1       4 47    8.50   2.97    7.80    8.21    2.82    4.50   16.60   12.10
## Po2       5 47    8.02   2.80    7.30    7.76    2.82    4.10   15.70   11.60
## LF        6 47    0.56   0.04    0.56    0.56    0.05    0.48    0.64    0.16
## M.F       7 47   98.30   2.95   97.70   98.02    1.93   93.40  107.10   13.70
## Pop       8 47   36.62  38.07   25.00   29.95   22.24    3.00  168.00  165.00
## NW        9 47   10.11  10.28    7.60    8.56    7.71    0.20   42.30   42.10
## U1       10 47    0.10   0.02    0.09    0.09    0.02    0.07    0.14    0.07
## U2       11 47    3.40   0.84    3.40    3.35    0.89    2.00    5.80    3.80
## Wealth   12 47 5253.83 964.91 5370.00 5286.67 1111.95 2880.00 6890.00 4010.00
## Ineq     13 47   19.40   3.99   17.60   19.28    3.56   12.60   27.60   15.00
## Prob     14 47    0.05   0.02    0.04    0.05    0.02    0.01    0.12    0.11
## Time     15 47   26.60   7.09   25.80   26.35    6.37   12.20   44.00   31.80
## Crime    16 47  905.09 386.76  831.00  863.05  314.31  342.00 1993.00 1651.00
##          skew kurtosis     se
## M        0.82     0.38   0.18
## So       0.65    -1.61   0.07
## Ed      -0.32    -1.15   0.16
## Po1      0.89     0.16   0.43
## Po2      0.84     0.01   0.41
## LF       0.27    -0.89   0.01
## M.F      0.99     0.65   0.43
## Pop      1.85     3.08   5.55
## NW       1.38     1.08   1.50
## U1       0.77    -0.13   0.00
## U2       0.54     0.17   0.12
## Wealth  -0.38    -0.61 140.75
## Ineq     0.37    -1.14   0.58
## Prob     0.88     0.75   0.00
## Time     0.37    -0.41   1.03
## Crime    1.05     0.78  56.42
```

Here we can see that the predictors have very different scales, even those that describe similar things (like U1 and U2 that describe unemployment have different scales (U1 displays 10% as 0.1, and U2 as 10.0)). From previous work on this data I also know that most predictors are moderately skewed (**between -1 and -0.5 or 0.5 and 1**) and have have platykurtic distribution(<3). I will visualize the predictors to get a general sense of the parameters:

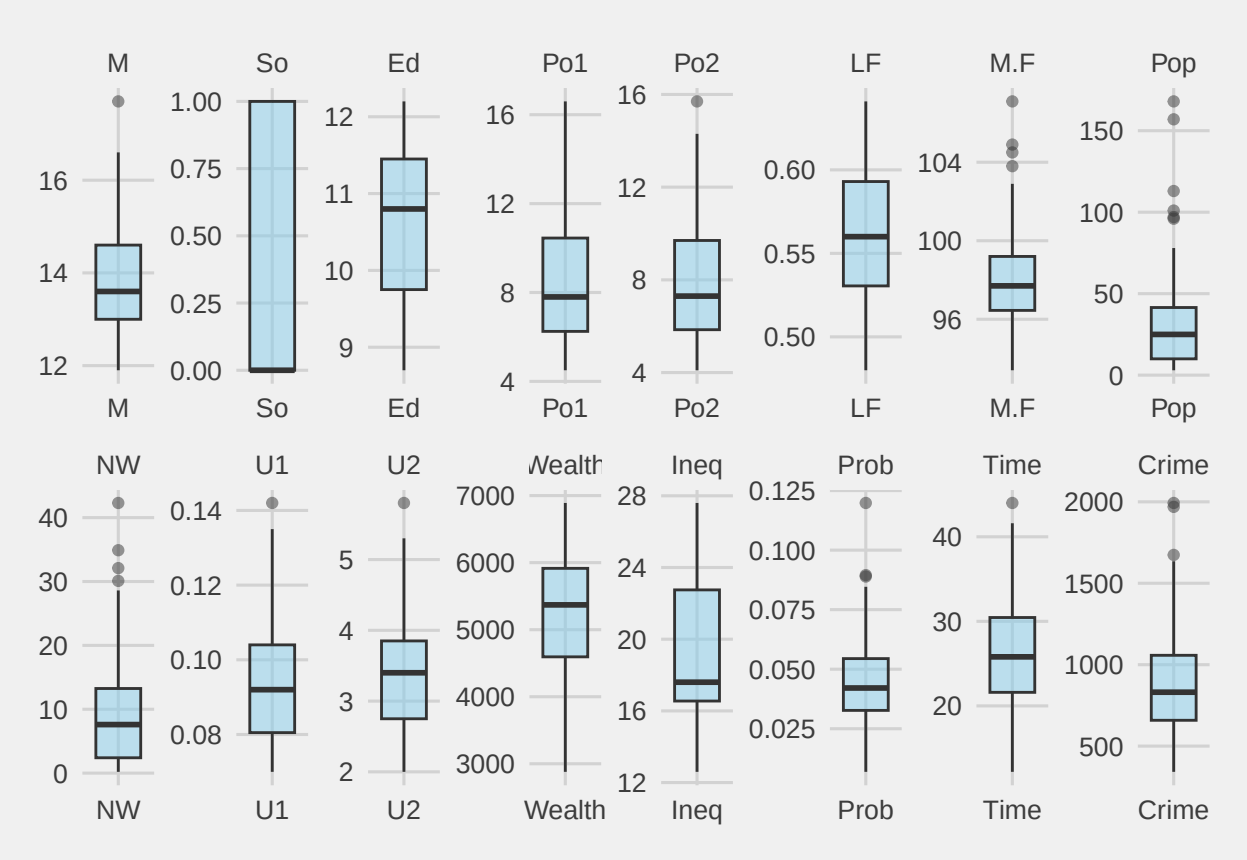
```
#melt data for easier visualization
melted<-melt(data)
```

```
## Warning in melt(data): The melt generic in data.table has been passed a
## data.frame and will attempt to redirect to the relevant reshape2 method; please
## note that reshape2 is deprecated, and this redirection is now deprecated as
## well. To continue using melt methods from reshape2 while both libraries are
## attached, e.g. melt.list, you can prepend the namespace like
## reshape2::melt(data). In the next version, this warning will become an error.
```

```
## No id variables; using all as measure variables
```

```
#boxplots
box_plots <- ggplot(melted,
                    aes(x=factor(variable), y=value))+
             geom_boxplot(alpha=.5, fill="skyblue")+
```

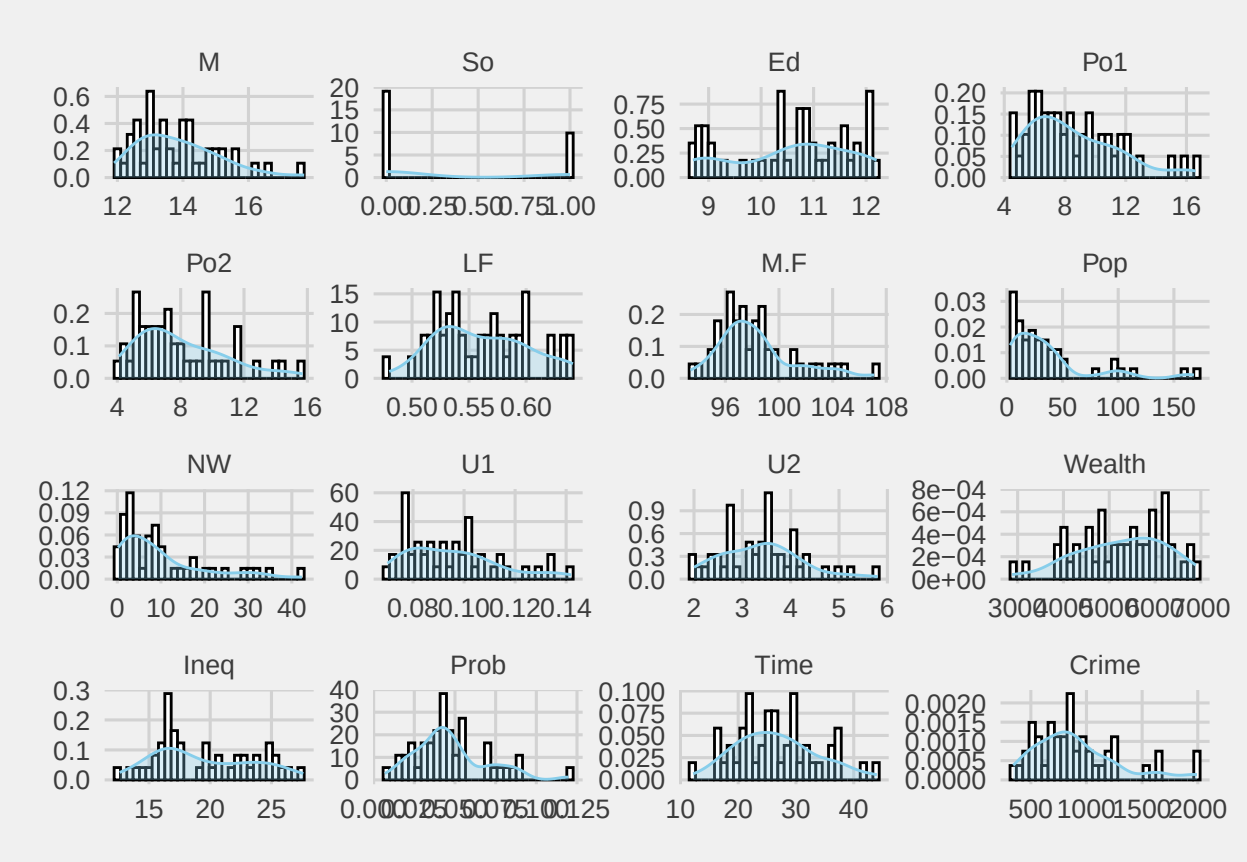
```
              facet_wrap(~variable, ncol=8, scale="free")+
              theme_fivethirtyeight()


box_plots
```




```
#distribution plots with density
hist_plots <- ggplot(melted,
                     aes(x=value))+
              geom_histogram(aes(y=..density..), colour="black", fill="white")+
              geom_density(alpha=.3, color="skyblue", fill="skyblue")+
              facet_wrap(~variable, scale="free")+
              theme_fivethirtyeight()
hist_plots
```

```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Boxplots clearly show skewness to the right for many parameters and that some of them might have extreme values / outliers. Histograms with a density curve for each variable support our assumption that most variables are right-skewed, and also most of the predictors are not normally distributed - the only normally distributed variable seems to be Time.

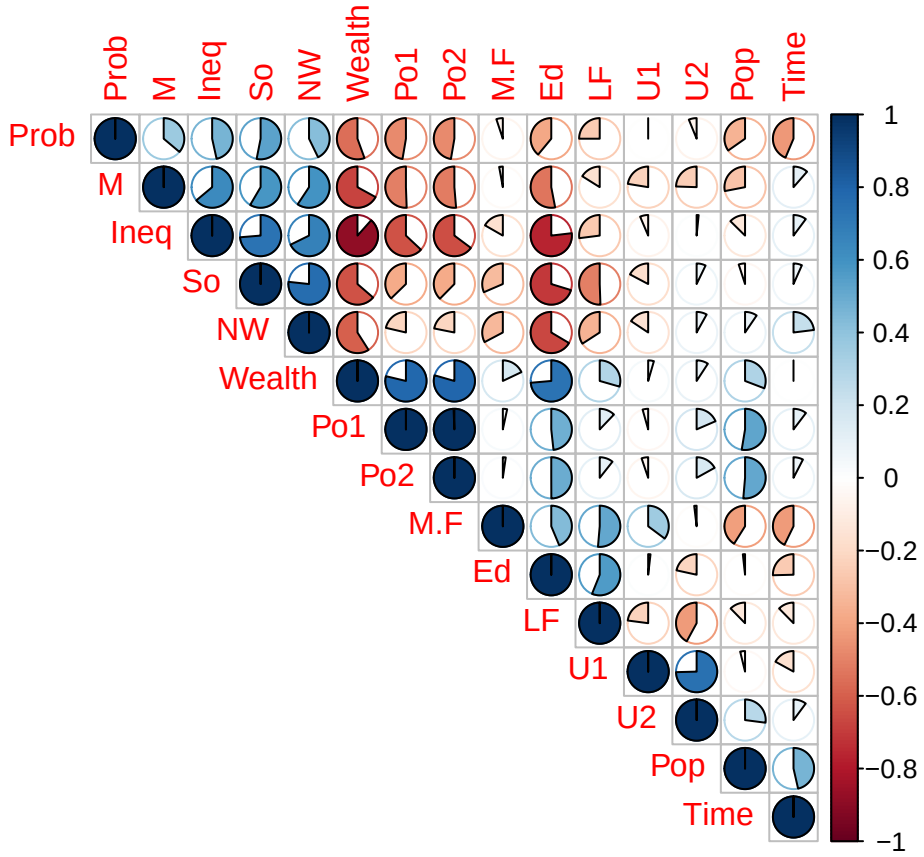
Step 3: Pairwise Correlation

Before PCA, it is important to understand whether we really need it - we need to check if there is strong pairwise correlation for some variables. Here are the assumptions I made (based on the description of what each variable shows):

- Po1 and Po2, as those are expenditures on police protection in two consequent years (1959, 1960)
- U1 and U2, since they show urban male unemployment rate for two age groups (14-24, 35-39)
- Wealth and Ineq, since wealth parameter of a family and it's inequality level seem to be describing the same parameter (inequality is probably (to a large extent) based on wealth)
- NW and Ineq, more inequality in states with higher percentage of non-white population
- NW and Ed, states with higher non-white population number might have less mean years of schooling
- Ed-Wealth(Ineq) - states with wealthier population might have more mean years of schooling among adults
- So-NW - southern states migh have had more non-white population in 1960s
- Po1 / Po2 and wealth / ineq - people who have a job should be wealthier (hence higher equality for such individuals)

To see if the assumptions are true, let's build correlation plots and matrix.

```
corrplot(cor(data[,-16]), method='pie', type="upper", order="hclust")
```



The following variables have relatively high correlation, according to the plot:

- Po1-Po2 (correlation is almost 1)
- Wealth-Ineq
- U1-U2
- So-Ed, So-NW
- Po1(Po2)-Wealth, Po1(Po2)-Ineq
- NW-Ineq, NW-Wealth
- Ed-NW, Ed-Wealth, Ed-Ineq

To understand which correlations are significant, and which are not, create a matrix of p-values for the correlation of each pair of variables:

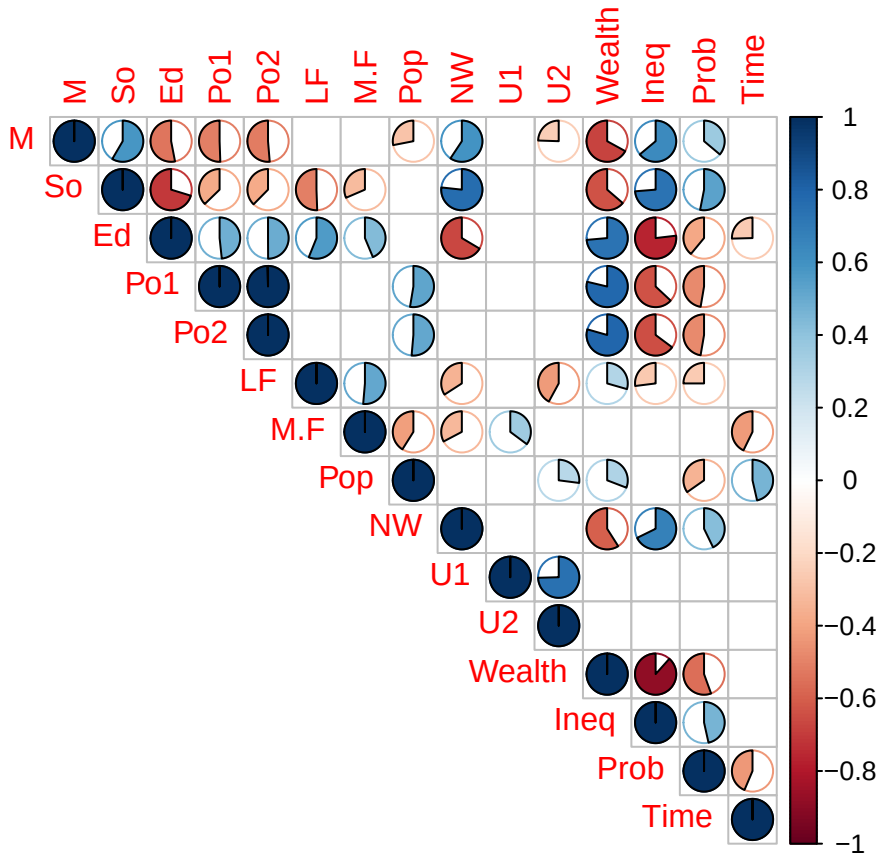
```
cor.mtest <- function(mat, ...) {
    mat <- as.matrix(data[,-16])
    n <- ncol(mat)
    p.mat<- matrix(NA, n, n)
    diag(p.mat) <- 0
    for (i in 1:(n - 1)) {
        for (j in (i + 1):n) {
            tmp <- cor.test(mat[, i], mat[, j], ...)
            p.mat[i, j] <- p.mat[j, i] <- tmp$p.value
```

```
        }
    }
  colnames(p.mat) <- rownames(p.mat) <- colnames(mat)
  p.mat
}
# matrix of the p-value of the correlation
p.mat <- cor.mtest(mtcars)
head(p.mat[, 1:5])
```

```
##                M           So           Ed          Po1          Po2
## M   0.000000e+00 1.613755e-05 1.263626e-04 2.875225e-04 2.255065e-04
## So  1.613755e-05 0.000000e+00 3.659957e-08 9.893859e-03 9.162861e-03
## Ed  1.263626e-04 3.659957e-08 0.000000e+00 5.852818e-04 3.520185e-04
## Po1 2.875225e-04 9.893859e-03 5.852818e-04 0.000000e+00 2.999512e-44
## Po2 2.255065e-04 9.162861e-03 3.520185e-04 2.999512e-44 0.000000e+00
## LF  2.798028e-01 2.900155e-04 4.071354e-05 4.159275e-01 4.767838e-01
```

Finally, combine the correlation plot with p-values for pairwise correlation and use the confidence interval of 90% to leave on the plot only correlation values that are significant:

```
corrplot(cor(data[,-16]),
         method='pie',
         type="upper",
         #order="hclust",
         p.mat = p.mat,
         sig.level = 0.1,
         insig = "blank")
```



This has narrowed down the amount of high correlation pairs, however we still have pairs with high correlations, such as P1-P2, So-Ed, Ed-NW, Wealth-Ineq, Ed-Ineq, U1-U2, and so on.

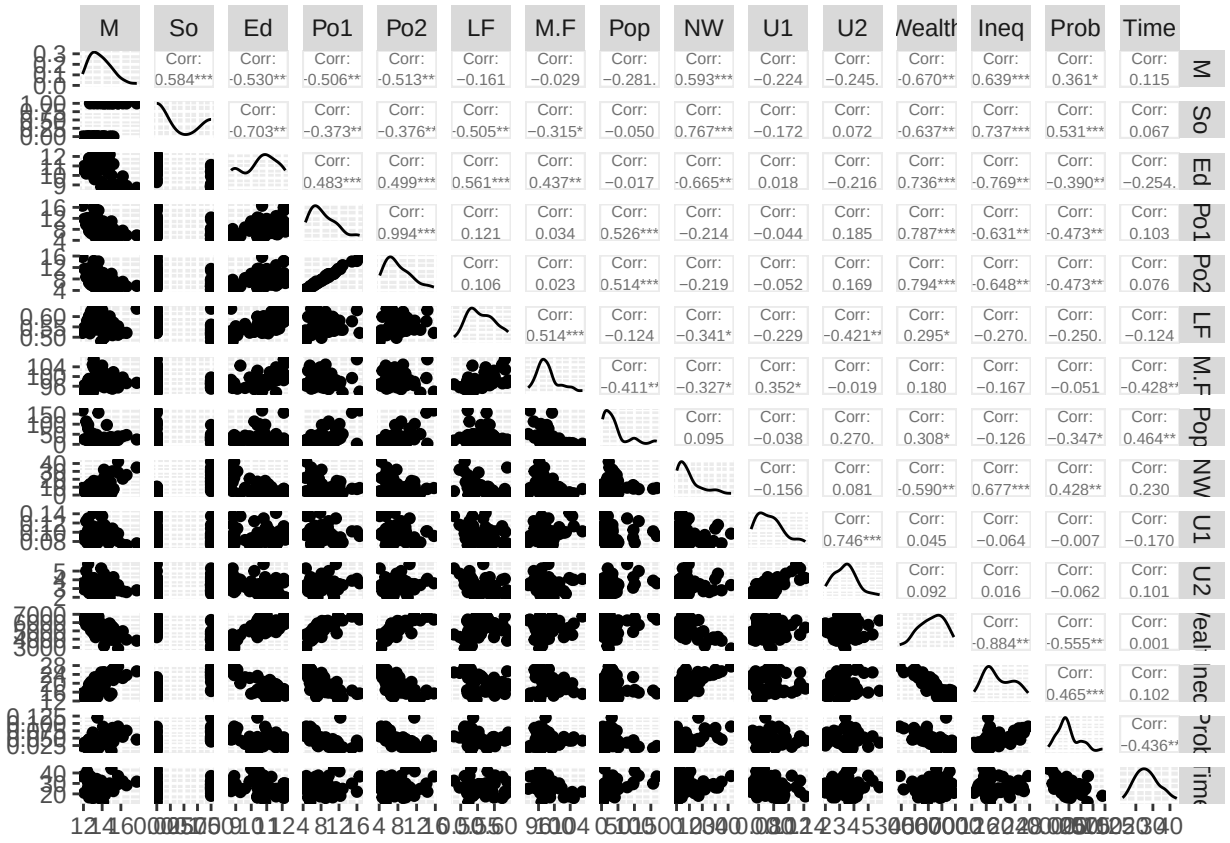
The exact coefficients for such variables can be found in the correlation matrix:

```
cm <- as.table(round(cor(data[,-16]),1))
cm
```

```
##            M   So   Ed  Po1  Po2   LF  M.F  Pop   NW   U1   U2 Wealth Ineq Prob
## M        1.0  0.6 -0.5 -0.5 -0.5 -0.2  0.0 -0.3  0.6 -0.2 -0.2   -0.7  0.6  0.4
## So       0.6  1.0 -0.7 -0.4 -0.4 -0.5 -0.3  0.0  0.8 -0.2  0.1   -0.6  0.7  0.5
## Ed      -0.5 -0.7  1.0  0.5  0.5  0.6  0.4  0.0 -0.7  0.0 -0.2    0.7 -0.8 -0.4
## Po1     -0.5 -0.4  0.5  1.0  1.0  0.1  0.0  0.5 -0.2  0.0  0.2    0.8 -0.6 -0.5
## Po2     -0.5 -0.4  0.5  1.0  1.0  0.1  0.0  0.5 -0.2 -0.1  0.2    0.8 -0.6 -0.5
## LF      -0.2 -0.5  0.6  0.1  0.1  1.0  0.5 -0.1 -0.3 -0.2 -0.4    0.3 -0.3 -0.3
## M.F      0.0 -0.3  0.4  0.0  0.0  0.5  1.0 -0.4 -0.3  0.4  0.0    0.2 -0.2 -0.1
## Pop     -0.3  0.0  0.0  0.5  0.5 -0.1 -0.4  1.0  0.1  0.0  0.3    0.3 -0.1 -0.3
## NW       0.6  0.8 -0.7 -0.2 -0.2 -0.3 -0.3  0.1  1.0 -0.2  0.1   -0.6  0.7  0.4
## U1      -0.2 -0.2  0.0  0.0 -0.1 -0.2  0.4  0.0 -0.2  1.0  0.7    0.0 -0.1  0.0
## U2      -0.2  0.1 -0.2  0.2  0.2 -0.4  0.0  0.3  0.1  0.7  1.0    0.1  0.0 -0.1
## Wealth  -0.7 -0.6  0.7  0.8  0.8  0.3  0.2  0.3 -0.6  0.0  0.1    1.0 -0.9 -0.6
## Ineq     0.6  0.7 -0.8 -0.6 -0.6 -0.3 -0.2 -0.1  0.7 -0.1  0.0   -0.9  1.0  0.5
## Prob     0.4  0.5 -0.4 -0.5 -0.5 -0.3 -0.1 -0.3  0.4  0.0 -0.1   -0.6  0.5  1.0
## Time     0.1  0.1 -0.3  0.1  0.1 -0.1 -0.4  0.5  0.2 -0.2  0.1    0.0  0.1 -0.4
##         Time
## M        0.1
## So       0.1
## Ed      -0.3
## Po1      0.1
## Po2      0.1
## LF      -0.1
## M.F     -0.4
## Pop      0.5
## NW       0.2
## U1      -0.2
## U2       0.1
## Wealth   0.0
## Ineq     0.1
## Prob    -0.4
## Time     1.0
```

Visualize correlation between pairs:

```
ggpairs(data[,-16], upper = list(continuous = wrap("cor", size = 2)))
```




```
#changed font size so that values would fit
```

High correlation is especially visible for such pairs as Po1-Po2, Wealth-Ineq, U1-U2, NW-Ineq, etc.

To summarize, we have variables with high correlation in the data set, which may result in multicollinearity in our LR model, if we use all variables. Moreover, our data set does not correspond to the 10 to 1 ratio of data points and varaibles - we have too many variables and too little data points, which makes it hard to build a model using all the variables with no overfit. Hence, we do need to take remedial measures here - and PCA can be helpful in this case, as it can help remove correlation within the data and helps us deal with the high amount of factors to decide which are more important.

Step 4: Principal Component Analysis

Let's run PCA on the predictors. We will exclude Crime, as it is a response variable that we are trying to predict, and will scale the predictors, since they have very different scales, as shown on the boxplot above. As for Crime, we will not scale it, since it is the response varaible, and scaling it won't bring any benefit to the analysis (we only need predictors to build the model).

PCA on the scale predictors:

```
pca <- prcomp(data[,-16],
              scale. = TRUE
              )
pca
```

```
## Standard deviations (1, .., p=15):
##  [1] 2.45335539 1.67387187 1.41596057 1.07805742 0.97892746 0.74377006
##  [7] 0.56729065 0.55443780 0.48492813 0.44708045 0.41914843 0.35803646
```

```
## [13] 0.26332811 0.24180109 0.06792764
##
## Rotation (n x k) = (15 x 15):
##                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
```

Get the summary of PCA:

```
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
```

As expected, the first component explains the highest proportion of variance among all 15 PCs - 40.13% and has a standard deviation of 2.45 (also the highest among other PCs).

The principal components that we are interested in for the model are stored in x object of pca. It is of the same dimension as the initial data set:

```
head(pca$x)
```

```
##            PC1        PC2         PC3         PC4         PC5        PC6
## [1,] -4.199284 -1.0938312 -1.11907395  0.67178115  0.05528338  0.3073383
## [2,]  1.172663  0.6770136 -0.05244634 -0.08350709 -1.17319982 -0.5832373
## [3,] -4.173725  0.2767750 -0.37107658  0.37793995  0.54134525  0.7187223
## [4,]  3.834962 -2.5769060  0.22793998  0.38262331 -1.64474650  0.7294884
## [5,]  1.839300  1.3309856  1.27882805  0.71814305  0.04159032 -0.3940902
## [6,]  2.907234 -0.3305421  0.53288181  1.22140635  1.37436096 -0.6922513
##              PC7          PC8         PC9        PC10        PC11        PC12
## [1,] -0.56640816 -0.007801727  0.22350995  0.45274365 -0.08474542  0.22096639
## [2,]  0.19561119  0.154566472  0.43677720  0.21208589 -0.03391661  0.35686524
## [3,]  0.10330693  0.351138883  0.06299232 -0.06719022 -0.48149156 -0.04701948
## [4,]  0.26699499 -1.547460841 -0.37954181  0.22922305  0.10984951  0.17727101
## [5,]  0.07050766 -0.543237437  0.22463245  0.47769084 -0.32958186  0.41807551
## [6,]  0.22648209  0.562323186  0.41772217  0.09100939  0.01022969 -0.70661980
##             PC13       PC14        PC15
## [1,]  0.11261680  0.3269649 -0.02338401
## [2,] -0.29751651  0.2523567  0.06076368
## [3,] -0.05216054 -0.4865511 -0.04211750
## [4,] -0.08838131  0.1496784 -0.02917497
## [5,]  0.72215224  0.1310272  0.07514940
## [6,]  0.13517271  0.1949257 -0.01558610
```

Eigenvectors of the covariance matrix is the rotation component:

```
head(pca$rotation)
```

```
##            PC1         PC2        PC3         PC4         PC5          PC6
## M   -0.3037119  0.06280357 0.17241999 -0.02035537 -0.35832737 -0.449132706
## So  -0.3308813 -0.15837219 0.01554331  0.29247181 -0.12061130 -0.100500743
## Ed   0.3396215  0.21461152 0.06773962  0.07974375 -0.02442839 -0.008571367
## Po1  0.3086341 -0.26981761 0.05064582  0.33325059 -0.23527680 -0.095776709
## Po2  0.3109929 -0.26396300 0.05306512  0.35192809 -0.20473383 -0.119524780
## LF   0.1761776  0.31943042 0.27153018 -0.14326529 -0.39407588  0.504234275
```

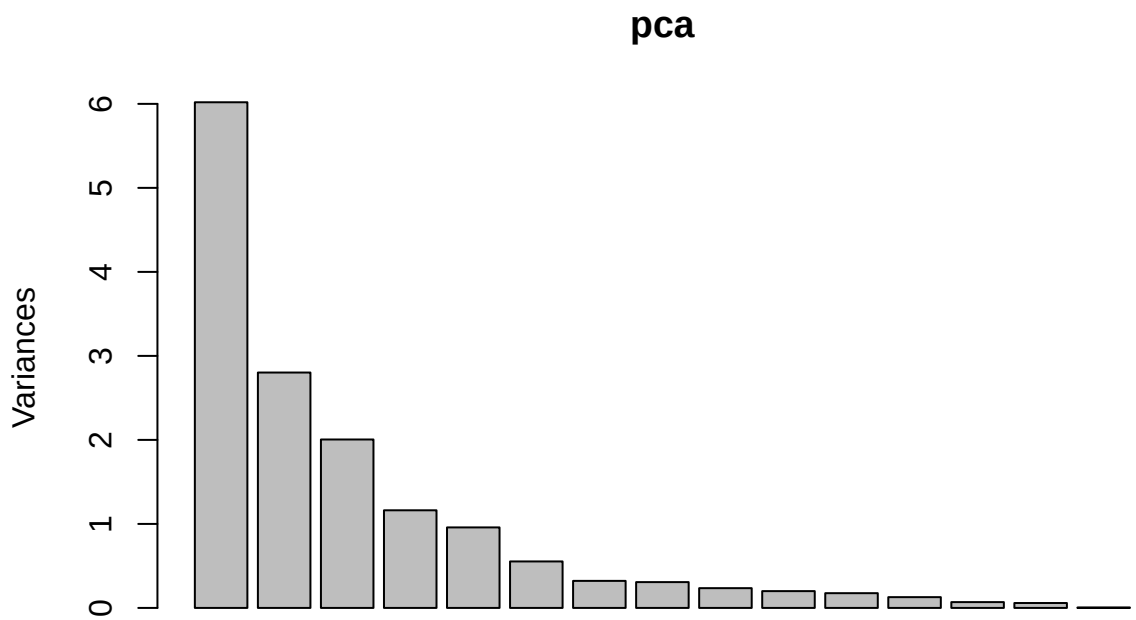
```
##             PC7         PC8        PC9        PC10       PC11        PC12
## M   -0.15707378 -0.55367691  0.1547479 -0.01443093  0.3944666  0.16580189
## So   0.19649727  0.22734157 -0.6559987  0.06141452  0.2339787 -0.05753357
## Ed  -0.23943629 -0.14644678 -0.4432698  0.51887452 -0.1182195  0.47786536
## Po1  0.08011735  0.04613156  0.1942547 -0.14320978 -0.1304200  0.22611207
## Po2  0.09518288  0.03168720  0.1951207 -0.05929780 -0.1388591  0.19088461
## LF  -0.15931612  0.25513777  0.1439350  0.03077073  0.3853283  0.02705134
##            PC13        PC14         PC15
## M    0.05142365  0.04901705 -0.005139801
## So   0.29368483 -0.29364512 -0.008436923
## Ed  -0.19441949  0.03964277  0.028005204
## Po1  0.18592255 -0.09490151  0.689415513
## Po2  0.13454940 -0.08259642 -0.720027010
## LF   0.27742957 -0.15385625 -0.033682319
```

Step 5: Visualizing PCA output

To understand how the number of PCA affects the shares of variance and cumulative variance explained, and to decide how many PCs we need to use for the model, let's create three principal plots.

First of all, the Screee plot. It shows the eigenvalues on the vertical axis and the number of factors on the horizontal one: proportion of variance within the data that is explained by each of the principal components:

```
screeplot(pca, npcs=15)
```




```
#display all PCs
```

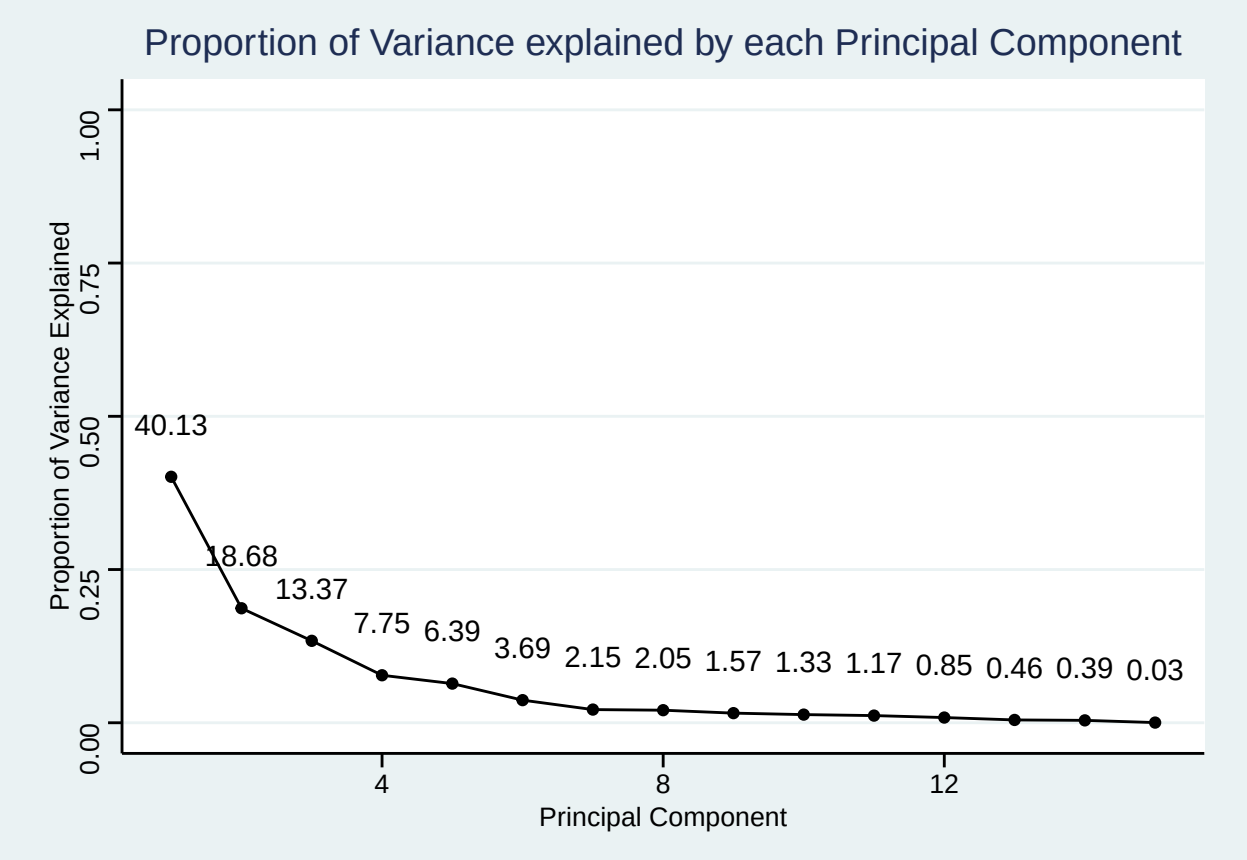
Ideally, at least 90% of the varaince should be explained - so we choose the number of PCs that have

eigenvalues higher than 1. Based on the Screeplot, **the first 4 principal components are suggested for use, as they all have a value above 1**, meaning that 4 principal components would explain at least 90% of the variance in the data. Considering that we have only 47 data points, it is a good proportion for the model. However, **I find it important to consider adding the 5th PC, as it is not very different in eigenvalue from PC4, so its explanation power may be useful for the model.

Second of all, compute variance explained by each PC using the standard deviation of PC from the PCA output.And create a plot of the share of variance explained by each PC. This is to **compare the explanation power of PC4 and PC5 and see whether the model can benefit from adding PC5**:

```
explained_var <- pca$sdev^2/sum(pca$sdev^2)
explained_var_round<-round(explained_var,4)
#plot explained variance by each PC:

qplot(c(1:15), explained_var_round,) +
  geom_line() +
  xlab("Principal Component") +
  ylab("Proportion of Variance Explained") +
  ggtitle("Proportion of Variance explained by each Principal Component") +
  ylim(0, 1) +
  geom_text(aes(label=explained_var_round*100),vjust=-2)+
  theme_stata()
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

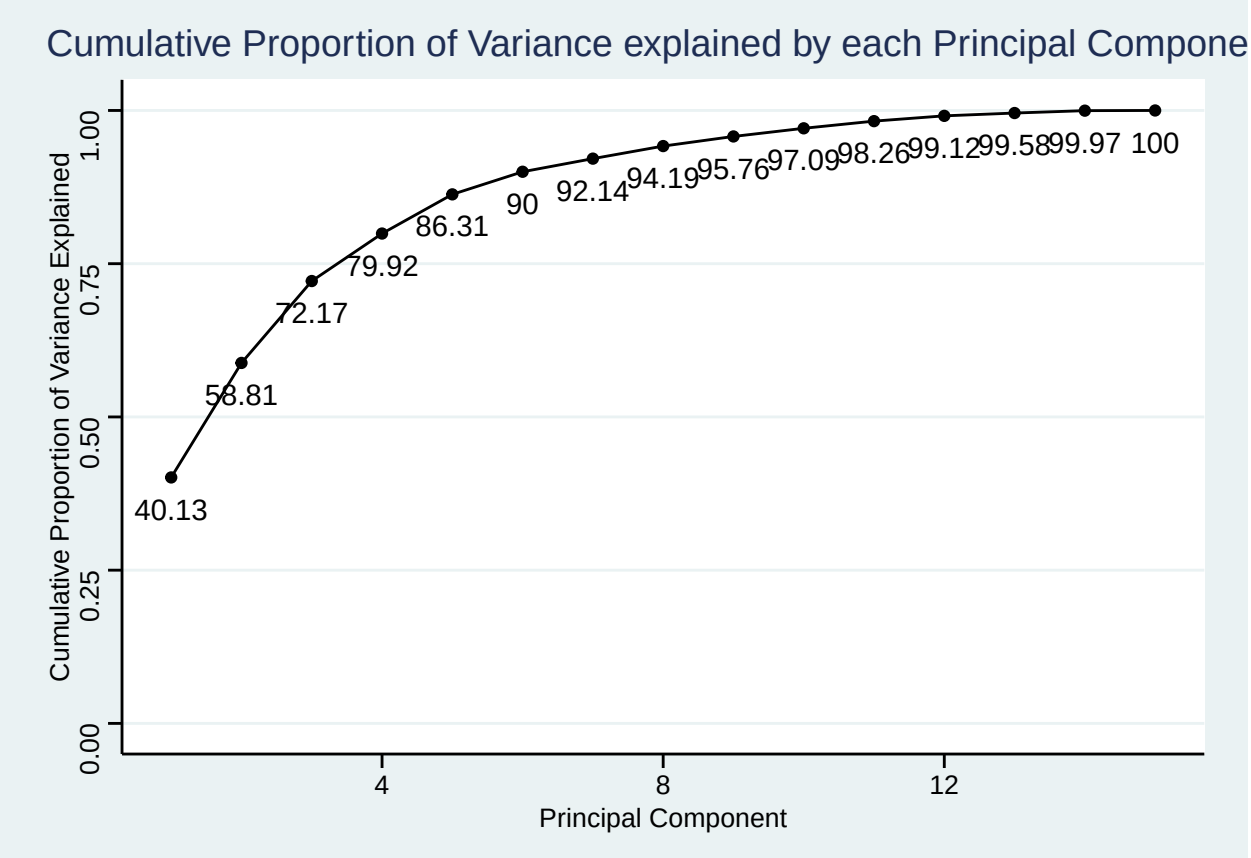


As we can see from the plot, the first principal component explains 40% of the variance in the data, while

the 15th PC - only 0.03%. However, **PC5 is not very different from PC4 in terms of the share of variance explained**. PC4 explains 7.75% of the variance, and PC5 explains another 6.39%. **After PC5 the variance explained by PCs starts to drop significantly and becomes closer to 1%**. I would suggest **including PC5 in the analysis as well**, since the sum of variance explained by PC1 through PC4 is ~79.93%, and with PC5 it increases to ~86.32%, which is more preferable for the model.

Finally, let's plot the cumulative sum of proportion of variance explained by PCs to support what we just discussed:

```
cumsum_var <- cumsum(explained_var)
cumsum_var_round <- round(cumsum_var,4)
#plot cumulative explained variance by each PC:

qplot(c(1:15), cumsum_var_round,) +
  geom_line() +
  xlab("Principal Component") +
  ylab("Cumulative Proportion of Variance Explained") +
  ggtitle("Cumulative Proportion of Variance explained by each Principal Component") +
  ylim(0, 1) +
  geom_text(aes(label=cumsum_var_round*100),vjust=2)+
  theme_stata()
```



As seen before, we should definitely consider using the **first five principal components** for the model, which altogether explain 86% of the variance in the data. 6 PCs explain 90%, but the values for the PC6 on the two previous plots are rather low, suggesting that the model would not benefit too much from its addition.

Nevertheless, I would suggest trying the three combinations - the first 4, 5 and 6 principal components, and

comparing the quality of prediction after cross-validation on each of them.

Step 6: Linear Regression on first 4 PCs

Step 6.1: Linear Regression (Cross-Validation)

In order to make sure that the first 5 PCs are the correct component set for the model, I will also run regression on the first 4 and 6 PCs. Let's start with PCs 1-4.

Firstly, we create a new data frame with the **x values of the first 4 principal components** and the response varaible, Crime:

```
#attach Crime values to the first 4 PCs
pc4 <- cbind(pca$x[,1:4], Crime=data[,16])
#check the result
head(as.data.frame(pc4))
```

```
##         PC1        PC2         PC3         PC4 Crime
## 1 -4.199284 -1.0938312 -1.11907395  0.67178115   791
## 2  1.172663  0.6770136 -0.05244634 -0.08350709  1635
## 3 -4.173725  0.2767750 -0.37107658  0.37793995   578
## 4  3.834962 -2.5769060  0.22793998  0.38262331  1969
## 5  1.839300  1.3309856  1.27882805  0.71814305  1234
## 6  2.907234 -0.3305421  0.53288181  1.22140635   682
```

Now, we create a linear regression model on the new data matrix with 4 PCs and Crime values. Crime is the response variable, while the 4 PCs are predictors. Cross Validation is the general practice for estimating quality of linear regression, as the risks of choosing a over-complicated or over-simplified model are unlikely, compared to AIC and BIC. AIC and BIC are both related to cross-validation, but the benefit of CV is that it does not produce the common issues related to them.

In order to **reduce overfitting** and get more accurate results, I will train a model using **repeated cross-validation** within the caret package. I chose repeated cross-validation, as it creates several models using the same number of folds, which gives us a more accurate result. We will have **10 repeats** withing cross-validation and **5 folds**. I chose 5 folds instead of 10 due to the size of our data set - we only have 47 data points, and having too many folds would lead to too much variability within them, which can be reflected in inaccurate results. 5 folds should be enough to have complete data representations in each fold.

```
set.seed(1)
data_ctrl <- trainControl(method = "repeatedcv", number = 5, repeats=10)
pc4_model <- train(Crime~.,
                   data=as.data.frame(pc4),
                   trControl=data_ctrl,
                   method="lm")
pc4_model
```

```
## Linear Regression
##
## 47 samples
##  4 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 10 times)
## Summary of sample sizes: 38, 38, 38, 37, 37, 38, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   344.7924  0.2822093  274.4788
```

```
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
install.packages('rpart.plot')
```

```
## Installing package into '/usr/local/lib/R/site-library'
## (as 'lib' is unspecified)
```

As a result, **linear regression using 4 Principal Components gives us after repeated cross-validation an R-squared of 28%**. This is not ideal, and perhaps we will have a better accuracy with five principal components. We can now use this model to make a prediction, extract model's parameters and unscale them.

Step 6.2: Prediction for new data (PC 1-4 model)

Now we can make a prediction for the new data point. To do that, I first need to **scale the test data** using the same scaling parameters which were applied to the rest of the data. We can do this by using our PCA in the predict function:

```
set.seed(1)
test_data_scaled <- data.frame(predict(pca,test_data))
test_data_scaled
```

```
##        PC1       PC2      PC3       PC4       PC5      PC6        PC7       PC8
## 1 1.224044 -2.767641 0.533605 -1.146837 -1.206098 2.333343 -0.1535916 -1.391625
##        PC9       PC10       PC11     PC12     PC13      PC14      PC15
## 1 1.460274 -0.4525158 -0.3466498 1.663782 1.811307 -2.174071 -1.288675
```

Now, using the scaled test data, we can make a prediction for Crime using our model:

```
set.seed(1)
pc4_prediction <- predict(pc4_model,test_data_scaled)
ceiling(pc4_prediction)
```

```
##    1
## 1113
```

The prediction of 1113 Crimes for new stat is not as bad as a linear regression with all parameters, which gave us 155 Crimes in the last project with this data. However, considering that the final function with 5 significant parameters gave us a ~1300 Crimes prediction last time, I would expect good predictions to be near that value.

Step 6.3: Transforming to original data (Unscaling)

Now, we need to transform the coefficients from PCA into original data coefficients. First, we extract the beta values of the model - the intercept and the coefficients:

```
#check all coefficients:
summary(pc4_model)$coefficients
```

```
##              Estimate Std. Error    t value     Pr(>|t|)
## (Intercept) 905.08511   49.07426 18.4431753 9.399579e-22
## PC1          65.21593   20.21917  3.2254508 2.437617e-03
## PC2         -70.08312   29.63477 -2.3648952 2.273084e-02
## PC3          25.19408   35.03262  0.7191606 4.760245e-01
## PC4          69.44603   46.01314  1.5092653 1.387178e-01
```

```
#extract the intercept:
intercept_scaled4 <- summary(pc4_model)$coefficients[1]
intercept_scaled4
```

```
## [1] 905.0851
```

```
#extract the coefficients
coefs_scaled4 <- summary(pc4_model)$coefficients[2:5]
coefs_scaled4
```

```
## [1]  65.21593 -70.08312  25.19408  69.44603
```

Then, using the rotation values from PCA for the first 4 components, we can transform the principal component regression coefficients into **coefficients for our initial variables:**

```
#rotation component of PCA:
pca$rotation[,1:4]
```

```
##                PC1         PC2           PC3         PC4
## M      -0.30371194  0.06280357  0.1724199946 -0.02035537
## So     -0.33088129 -0.15837219  0.0155433104  0.29247181
## Ed      0.33962148  0.21461152  0.0677396249  0.07974375
## Po1     0.30863412 -0.26981761  0.0506458161  0.33325059
## Po2     0.31099285 -0.26396300  0.0530651173  0.35192809
## LF      0.17617757  0.31943042  0.2715301768 -0.14326529
## M.F     0.11638221  0.39434428 -0.2031621598  0.01048029
## Pop     0.11307836 -0.46723456  0.0770210971 -0.03210513
## NW     -0.29358647 -0.22801119  0.0788156621  0.23925971
## U1      0.04050137  0.00807439 -0.6590290980 -0.18279096
## U2      0.01812228 -0.27971336 -0.5785006293 -0.06889312
## Wealth  0.37970331 -0.07718862  0.0100647664  0.11781752
## Ineq   -0.36579778 -0.02752240 -0.0002944563 -0.08066612
## Prob   -0.25888661  0.15831708 -0.1176726436  0.49303389
## Time   -0.02062867 -0.38014836  0.2235664632 -0.54059002
```

```
#alphas:
a4 <- pca$rotation[,1:4] %*% coefs_scaled4
#transpose:
t(a4)
```

```
##               M       So       Ed      Po1      Po2        LF       M.F      Pop
## [1,] -21.27796 10.22309 14.35261 63.45643 64.55797 -14.00535 -24.43757 39.83067
##             NW        U1       U2   Wealth      Ineq     Prob      Time
## [1,] 15.43455 -27.22228 1.425902 38.60786 -27.53635 3.295707 -6.612616
```

Finally, we need to **unscale the result**, as the current coefficients are based on the scaled data. To unscale, we divide the scaled alpha values by the scale component of the PCA:

```
a_unscaled4 <- a4/pca$scale
a_unscaled4
```

```
##                 [,1]
## M        -16.9307630
## So        21.3436771
## Ed        12.8297238
## Po1       21.3521593
## Po2       23.0883154
## LF      -346.5657125
## M.F       -8.2930969
## Pop        1.0462155
## NW         1.5009941
## U1     -1509.9345216
```

```
## U2          1.6883674
## Wealth      0.0400119
## Ineq       -6.9020218
## Prob      144.9492678
## Time       -0.9330765
```

Unscaled intercept (beta 0) (calculate using the center and scale components from PCA):

```
intercept_unscaled4 <- intercept_scaled4 - sum(a4*pca$center/pca$scale)
intercept_unscaled4
```

```
## [1] 1666.485
```

Using the original alpha and beta0 values, we can calculate the estimates that our model gives for the Crime variable:

```
estimated_vals4 <- as.matrix(data[,1:15]) %*% a_unscaled4 + intercept_unscaled4
estimated_vals4
```

```
##             [,1]
##  [1,]  726.3425
##  [2,]  926.9936
##  [3,]  630.3920
##  [4,] 1368.0977
##  [5,] 1013.8482
##  [6,] 1216.0958
##  [7,]  982.3622
##  [8,] 1106.9894
##  [9,]  788.3831
## [10,]  758.2196
## [11,] 1235.6183
## [12,]  913.7572
## [13,]  806.1162
## [14,]  823.7319
## [15,]  664.0043
## [16,]  845.0424
## [17,]  774.2024
## [18,] 1191.9501
## [19,] 1285.8694
## [20,] 1089.2713
## [21,]  825.1714
## [22,]  611.7868
## [23,]  926.1627
## [24,]  758.4594
## [25,]  788.3155
## [26,] 1182.6525
## [27,]  899.7572
## [28,]  780.7204
## [29,] 1534.7847
## [30,]  743.3047
## [31,]  579.9948
## [32,] 1045.5708
## [33,]  789.5315
## [34,] 1007.4079
## [35,] 1067.3629
## [36,]  982.4233
```

```
## [37,]  646.0266
## [38,]  610.7420
## [39,]  739.2180
## [40,]  921.9585
## [41,]  843.7503
## [42,]  756.8244
## [43,]  844.7808
## [44,]  965.2991
## [45,]  610.2137
## [46,] 1051.4106
## [47,]  878.0818
```

Step 6.4: Specifying the model in terms of original data

Now that we know the coefficients in terms of original variable scales, we can specify the model built on the **first four Principal Components**:

Crime ~ 1666.49 - 16.93M + 21.34So + 12.83Ed + 21.35Po1 + 23.09Po2 - 346.57LF - 8.29M.F. + 1.05Pop + 1.50NW - 1509.93U1 + 1.69U2 + 0.04Wealth - 6.90Ineq + 144.95Prob - 0.93Time

Let's check R-squared values and predictions for cross-validated models build using 5 and 6 principal components, and then decide which model is better.

Step 7: Linear Regression on first 5 PCs

Here, I will use the same approach in calculations as in the previous step, and will comment mostly on results.

Step 7.1: Linear Regression (Cross-Validation)

Create a new data frame with the **x values of the first 5 principal components** and the response varaible, Crime:

```
#attach Crime values to the first 5 PCs
pc5 <- cbind(pca$x[,1:5], Crime=data[,16])
#check the result
head(as.data.frame(pc5))
```

```
##         PC1        PC2         PC3         PC4         PC5 Crime
## 1 -4.199284 -1.0938312 -1.11907395  0.67178115  0.05528338   791
## 2  1.172663  0.6770136 -0.05244634 -0.08350709 -1.17319982  1635
## 3 -4.173725  0.2767750 -0.37107658  0.37793995  0.54134525   578
## 4  3.834962 -2.5769060  0.22793998  0.38262331 -1.64474650  1969
## 5  1.839300  1.3309856  1.27882805  0.71814305  0.04159032  1234
## 6  2.907234 -0.3305421  0.53288181  1.22140635  1.37436096   682
```

Linear regression model with repeated cross-validation (5 folds, 10 repetitions)

```
set.seed(1)
data_ctrl <- trainControl(method = "repeatedcv", number = 5, repeats=10)
pc5_model <- train(Crime~.,
                   data=as.data.frame(pc5),
                   trControl=data_ctrl,
                   method="lm")
pc5_model
```

```
## Linear Regression
##
## 47 samples
```

```
##  5 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 10 times)
## Summary of sample sizes: 38, 38, 38, 37, 37, 38, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   257.2567  0.5539977  211.3828
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Adding the fifth principal component has helped us improve R-squared by almost 200%! Instead of 28%, it is now at 55.4%. RMSE (Root Mean Square Error) and MAE (Mean Absolute Error) are also significantly lower. Let's check the prediction and specify the model's equation.

Step 7.2: Prediction for new data (PC 1-5 model)

We already have the test data scaled using PCA parameters, so we can make a prediction using the model built on first 5 principal components:

```
set.seed(1)
pc5_prediction <- predict(pc5_model,test_data_scaled)
#round
ceiling(pc5_prediction)
```

```
##    1
## 1389
```

The prediction appears to be a lot more accurate as well - 1389 Crimes. With our final model in the regression project on this data, we had a prediction of 1304 Crimes. Considering that here we reduce overfitting and multicollinearity by using PCA, predictions are supposed to be less random, so perhaps 1389 is a more correct forecast compared to 1304.

Step 7.3: Transforming to original data (Unscaling)

Transform the coefficients from PCA into original data coefficients. Extract the beta values of the model:

```
#check all coefficients:
summary(pc5_model)$coefficients
```

```
##               Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)  905.08511   35.59411 25.4279492 1.005883e-26
## PC1           65.21593   14.66519  4.4469890 6.506555e-05
## PC2          -70.08312   21.49443 -3.2605250 2.241780e-03
## PC3           25.19408   25.40955  0.9915201 3.272475e-01
## PC4           69.44603   33.37384  2.0808522 4.373510e-02
## PC5         -229.04282   36.75341 -6.2318803 2.024121e-07
```

```
#extract the intercept:
intercept_scaled5 <- summary(pc5_model)$coefficients[1]
intercept_scaled5
```

```
## [1] 905.0851
```

```
#extract the coefficients
coefs_scaled5 <- summary(pc5_model)$coefficients[2:6]
coefs_scaled5
```

```
## [1]   65.21593  -70.08312   25.19408   69.44603 -229.04282
```

Using the rotation values from PCA for the first 5 components, transform the principal component regression coefficients into **coefficients for the initial variables:**

```
#rotation component of PCA:
pca$rotation[,1:5]
```

```
##                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
```

```
#alphas:
a5 <- pca$rotation[,1:5] %*% coefs_scaled5
#transpose:
t(a5)
```

```
##             M       So       Ed      Po1      Po2      LF      M.F      Pop
## [1,] 60.79435 37.84824 19.94776 117.3449 111.4508 76.2549 108.1266 58.88024
##            NW       U1       U2   Wealth    Ineq      Prob     Time
## [1,] 98.07179 2.866783 32.34551 35.93336 22.1037 -34.64026 27.20502
```

Unscale the result:

```
#getting rid of 'e'values in output
options("scipen"=100, "digits"=4)
#unscaling
a_unscaled5 <- a5/pca$scale
a_unscaled5
```

```
##               [,1]
## M         48.37374
## So        79.01922
## Ed        17.83120
## Po1       39.48484
## Po2       39.85892
## LF      1886.94577
## M.F       36.69366
## Pop        1.54658
## NW         9.53738
## U1       159.01148
## U2        38.29933
## Wealth     0.03724
## Ineq       5.54032
```

```
## Prob   -1523.52142
## Time       3.83878
```

Unscaled intercept (beta 0):

```
intercept_unscaled5 <- intercept_scaled5 - sum(a5*pca$center/pca$scale)
intercept_unscaled5
```

```
## [1] -5934
```

Calculate the estimates of model Crime variable:

```
estimated_vals5 <- as.matrix(data[,1:15]) %*% a_unscaled5 + intercept_unscaled5
estimated_vals5
```

```
##          [,1]
##  [1,]  713.7
##  [2,] 1195.7
##  [3,]  506.4
##  [4,] 1744.8
##  [5,] 1004.3
##  [6,]  901.3
##  [7,]  817.8
##  [8,] 1158.0
##  [9,]  862.7
## [10,]  906.2
## [11,] 1309.8
## [12,]  831.7
## [13,]  668.7
## [14,]  653.8
## [15,]  663.3
## [16,]  933.8
## [17,]  467.8
## [18,] 1097.8
## [19,]  975.2
## [20,] 1238.8
## [21,]  805.8
## [22,]  769.7
## [23,]  768.1
## [24,]  929.0
## [25,]  604.2
## [26,] 1845.8
## [27,]  480.4
## [28,] 1015.1
## [29,] 1463.8
## [30,]  801.6
## [31,]  687.9
## [32,]  969.7
## [33,]  722.7
## [34,]  841.7
## [35,]  915.0
## [36,]  977.8
## [37,] 1211.7
## [38,]  604.3
## [39,]  627.6
## [40,] 1069.9
```

```
## [41,]  841.5
## [42,]  272.3
## [43,] 1043.5
## [44,] 1126.3
## [45,]  425.5
## [46,]  927.2
## [47,] 1139.4
```

Step 7.4: Specifying the model in terms of original data

Specify the model equation built on the **first five Principal Components**:

Crime ~ -5933.837 + 48.37 M + 79.02So + 17.83Ed + 39.48Po1 + 39.86Po2 + 1886.95LF + 36.69M.F. + 1.55Pop + 9.54NW + 159.01U1 + 38.30U2 + 0.04Wealth + 5.54Ineq - 1523.52Prob + 3.84Time

Using **the first five pricnipal components, we have built a more accuarate model with a higher R-squared and lower error metrics**. Finally, let's compare it to a model with six principal components.

Step 8: Linear Regression on first 6 PCs

Let's compare the previous two models with the one built using the first six principal components

Step 8.1: Linear Regression (Cross-Validation)

Create a new data frame with the **x values of the first 6 principal components** and the response varaible, Crime:

```
#attach Crime values to the first 5 PCs
pc6 <- cbind(pca$x[,1:6], Crime=data[,16])
#check the result
head(as.data.frame(pc6))
```

```
##      PC1     PC2      PC3      PC4      PC5     PC6 Crime
## 1 -4.199 -1.0938 -1.11907  0.67178  0.05528  0.3073   791
## 2  1.173  0.6770 -0.05245 -0.08351 -1.17320 -0.5832  1635
## 3 -4.174  0.2768 -0.37108  0.37794  0.54135  0.7187   578
## 4  3.835 -2.5769  0.22794  0.38262 -1.64475  0.7295  1969
## 5  1.839  1.3310  1.27883  0.71814  0.04159 -0.3941  1234
## 6  2.907 -0.3305  0.53288  1.22141  1.37436 -0.6923   682
```

Linear regression model with repeated cross-validation (5 folds, 10 repetitions)

```
set.seed(1)
data_ctrl <- trainControl(method = "repeatedcv", number = 5, repeats=10)
pc6_model <- train(Crime~.,
                   data=as.data.frame(pc6),
                   trControl=data_ctrl,
                   method="lm")
pc6_model
```

```
## Linear Regression
##
## 47 samples
##  6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 10 times)
```

```
## Summary of sample sizes: 38, 38, 38, 37, 37, 38, ...
## Resampling results:
##
##   RMSE  Rsquared  MAE
##   258   0.5498    211.4
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

For a model with six principal components, we have a slightly lower R-squared of 54.98% and a slightly larger RMSE. **Adding PC6 did not give a significant improvement/change to the model**.

Step 8.2: Prediction for new data (PC 1-6 model)

We already have the test data scaled using PCA parameters, so we can make a prediction using the model built on first 6 principal components:

```
set.seed(1)
pc6_prediction <- predict(pc6_model,test_data_scaled)
#round
ceiling(pc6_prediction)
```

```
##    1
## 1249
```

This time, the prediction is a bit lowerm 1249, compared to 1388 with 5 principal components.

Step 8.3: Transforming to original data (Unscaling)

Transform the coefficients from PCA into original data coefficients. Extract the beta values of the model:

```
#check all coefficients:
summary(pc6_model)$coefficients
```

```
##             Estimate Std. Error t value                          Pr(>|t|)
## (Intercept)   905.09      35.35 25.6044 0.00000000000000000000000002017
## PC1            65.22      14.56  4.4778 0.00006144260645328677127030570
## PC2           -70.08      21.35 -3.2832 0.00213695014513261630217932918
## PC3            25.19      25.23  0.9984 0.32408693622424999958298030833
## PC4            69.45      33.14  2.0953 0.04252152161573004990335888920
## PC5          -229.04      36.50 -6.2751 0.00000019397577273126576089214
## PC6           -60.21      48.04 -1.2534 0.21734082161307963221297256950
```

```
#extract the intercept:
intercept_scaled6 <- summary(pc6_model)$coefficients[1]
intercept_scaled6
```

```
## [1] 905.1
```

```
#extract the coefficients
coefs_scaled6 <- summary(pc6_model)$coefficients[2:7]
coefs_scaled6
```

```
## [1]   65.22  -70.08   25.19   69.45 -229.04  -60.21
```

Using the rotation values from PCA for the first 5 components, transform the principal component regression coefficients into **coefficients for the initial variables:**

```
#rotation component of PCA:
pca$rotation[,1:6]
```

```
##             PC1       PC2        PC3      PC4      PC5       PC6
## M      -0.30371  0.062804  0.1724200 -0.02036 -0.35833 -0.449133
## So     -0.33088 -0.158372  0.0155433  0.29247 -0.12061 -0.100501
## Ed      0.33962  0.214612  0.0677396  0.07974 -0.02443 -0.008571
## Po1     0.30863 -0.269818  0.0506458  0.33325 -0.23528 -0.095777
## Po2     0.31099 -0.263963  0.0530651  0.35193 -0.20473 -0.119525
## LF      0.17618  0.319430  0.2715302 -0.14327 -0.39408  0.504234
## M.F     0.11638  0.394344 -0.2031622  0.01048 -0.57877 -0.074502
## Pop     0.11308 -0.467235  0.0770211 -0.03211 -0.08317  0.547099
## NW     -0.29359 -0.228011  0.0788157  0.23926 -0.36079  0.051220
## U1      0.04050  0.008074 -0.6590291 -0.18279 -0.13137  0.017386
## U2      0.01812 -0.279713 -0.5785006 -0.06889 -0.13499  0.048155
## Wealth  0.37970 -0.077189  0.0100648  0.11782  0.01168 -0.154683
## Ineq   -0.36580 -0.027522 -0.0002945 -0.08067 -0.21673  0.272027
## Prob   -0.25889  0.158317 -0.1176726  0.49303  0.16563  0.283536
## Time   -0.02063 -0.380148  0.2235665 -0.54059 -0.14765 -0.148203
```

```r
#alphas:
a6 <- pca$rotation[,1:6] %*% coefs_scaled6
#transpose:
t(a6)
```

```
##          M   So    Ed   Po1   Po2    LF   M.F   Pop    NW   U1    U2 Wealth
## [1,] 87.84 43.9 20.46 123.1 118.6 45.89 112.6 25.94 94.99 1.82 29.45  45.25
##       Ineq   Prob  Time
## [1,] 5.724 -51.71 36.13
```

Unscale the result:

```r
#getting rid of 'e'values in output
options("scipen"=100, "digits"=4)
#unscaling
a_unscaled6 <- a6/pca$scale
a_unscaled6
```

```
##               [,1]
## M         69.89232
## So        91.65344
## Ed        18.29254
## Po1       41.42536
## Po2       42.43282
## LF      1135.64065
## M.F       38.21603
## Pop        0.68129
## NW         9.23746
## U1       100.94504
## U2        34.86602
## Wealth     0.04689
## Ineq       1.43474
## Prob   -2274.39686
## Time       5.09798
```

Unscaled intercept (beta 0):

```r
intercept_unscaled6 <- intercept_scaled6 - sum(a6*pca$center/pca$scale)
intercept_unscaled6
```

```
## [1] -5924
```

Calculate the estimates of model Crime variable:

```
estimated_vals6 <- as.matrix(data[,1:15]) %*% a_unscaled6 + intercept_unscaled6
estimated_vals6
```

```
##         [,1]
##  [1,]  695.2
##  [2,] 1230.8
##  [3,]  463.1
##  [4,] 1700.9
##  [5,] 1028.1
##  [6,]  943.0
##  [7,]  873.8
##  [8,] 1155.1
##  [9,]  855.0
## [10,]  886.2
## [11,] 1283.1
## [12,]  816.3
## [13,]  605.6
## [14,]  616.7
## [15,]  653.0
## [16,]  946.4
## [17,]  483.2
## [18,] 1092.2
## [19,] 1028.8
## [20,] 1197.5
## [21,]  778.1
## [22,]  761.6
## [23,]  735.6
## [24,]  928.1
## [25,]  512.5
## [26,] 1874.5
## [27,]  547.7
## [28,] 1059.7
## [29,] 1420.1
## [30,]  827.1
## [31,]  715.5
## [32,]  890.9
## [33,]  731.0
## [34,]  867.0
## [35,]  882.4
## [36,] 1124.4
## [37,] 1207.3
## [38,]  540.2
## [39,]  640.0
## [40,] 1031.8
## [41,]  903.8
## [42,]  273.7
## [43,] 1112.1
## [44,] 1165.6
## [45,]  454.5
## [46,]  884.0
## [47,] 1115.5
```

Step 8.4: Specifying the model in terms of original data

Specify the model equation built on the **first six Principal Components**:

Crime ~ -5924 + 68.89 M + 91.65So + 18.29Ed + 41.43Po1 + 42.43Po2 + 1135.64LF + 38.22M.F. + 0.68Pop + 9.24NW + 100.95U1 + 34.87U2 + 0.04Wealth + 1.43Ineq - 2274.4Prob + 5.1Time

Comparing this model to the one with 5 principal components, I do not see a significant improvement, so I would suggest to choose the model build using **the first 5 Principal Components** - it has the highest R-Squared and the lowest error values, provides a reasonable prediction, and the number of 5 PCs is supported by the screeplot and explained variance graphs.

[Extra] Step 9: Models for other PCs

Since PC5 and PC6 provide very similar results, I would like to check the output for cross-validated models for each of the PCs combinations.

Let's build and cross-validate 15 models for each number of components and predict Crime variable for each of the models:

```
set.seed(1)
data_ctrl <- trainControl(method = "repeatedcv", number = 5, repeats=10)

r2 <- numeric(15)
rmse <- numeric(15)
mae <- numeric(15)
prediction <- numeric(15)

for (i in 1:15) {
  pcomps <- pca$x[,1:i]  #extract the first i PCs
  pc_i <- cbind(Crime=data[,16],pcomps)  #add crime variable to the set
  model <- train(Crime~.,
                     data=as.data.frame(pc_i),
                     trControl=data_ctrl,
                     method="lm")
  r2[i] <- unlist(model$results[3]) #extract R squared
  rmse[i] <- unlist(model$results[2]) #extract RMSE
  mae[i] <- unlist(model$results[4]) #extract RMSE
  prediction[i] <- ceiling(predict(model,test_data_scaled))
}
```

```
## Warning: 'newdata' had 1 row but variables found have 47 rows
```

```
## Warning in prediction[i] <- ceiling(predict(model, test_data_scaled)): number
## of items to replace is not a multiple of replacement length
```

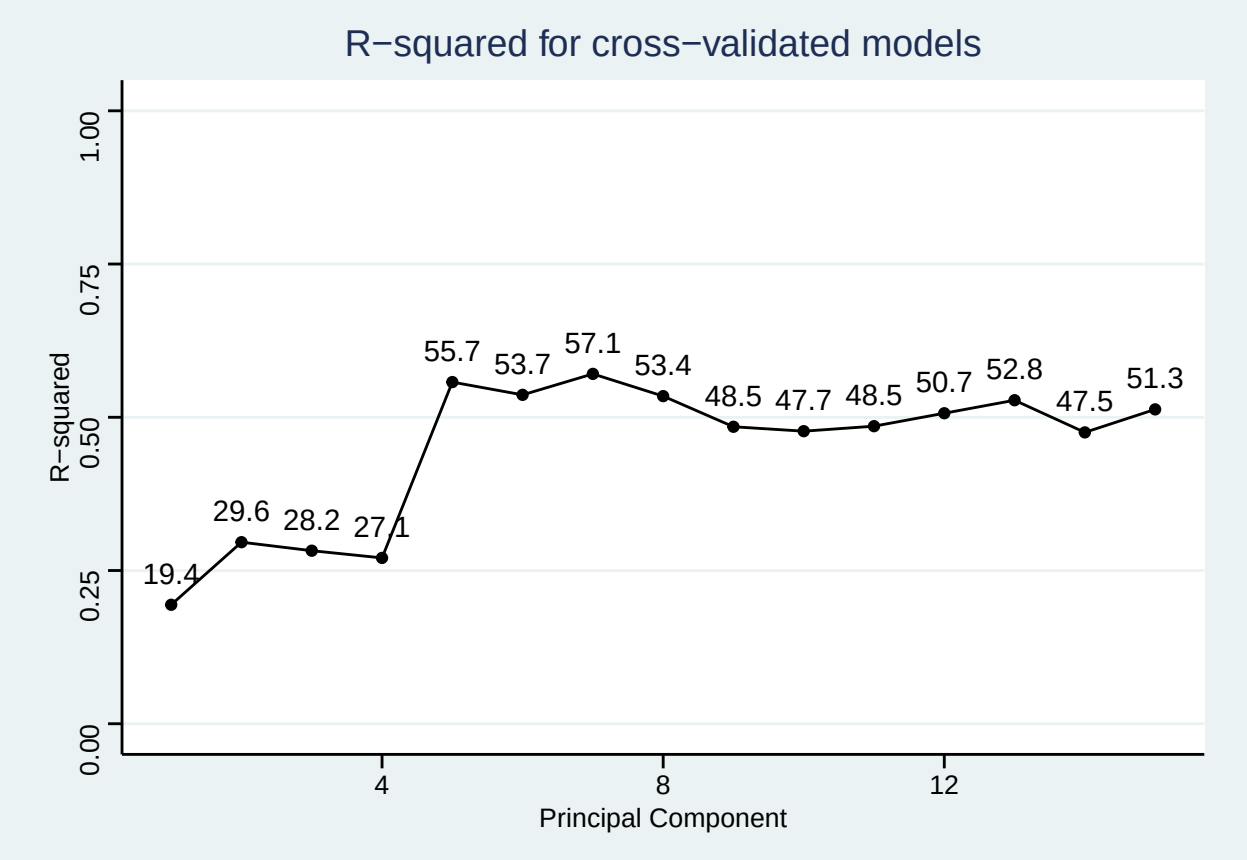
```
#compare results
models <- data.frame(PC=c(1:15),R2=r2, RMSE=rmse, MAE=mae, Prediction=prediction)
models
```

```
##    PC     R2  RMSE   MAE Prediction
## 1   1 0.1941 354.4 290.4        632
## 2   2 0.2962 343.1 268.6       1179
## 3   3 0.2823 350.4 274.3       1193
## 4   4 0.2706 352.7 279.0       1113
## 5   5 0.5574 257.9 212.7       1389
## 6   6 0.5366 269.9 224.7       1249
```

```
## 7   7 0.5708 256.0 204.8       1231
## 8   8 0.5345 269.6 214.8       1191
## 9   9 0.4845 288.4 229.5       1137
## 10 10 0.4773 289.3 229.6       1111
## 11 11 0.4855 308.9 245.7       1101
## 12 12 0.5066 285.3 217.3       1582
## 13 13 0.5278 283.7 224.2       1434
## 14 14 0.4754 298.9 239.8        958
## 15 15 0.5129 281.4 222.7        156
```

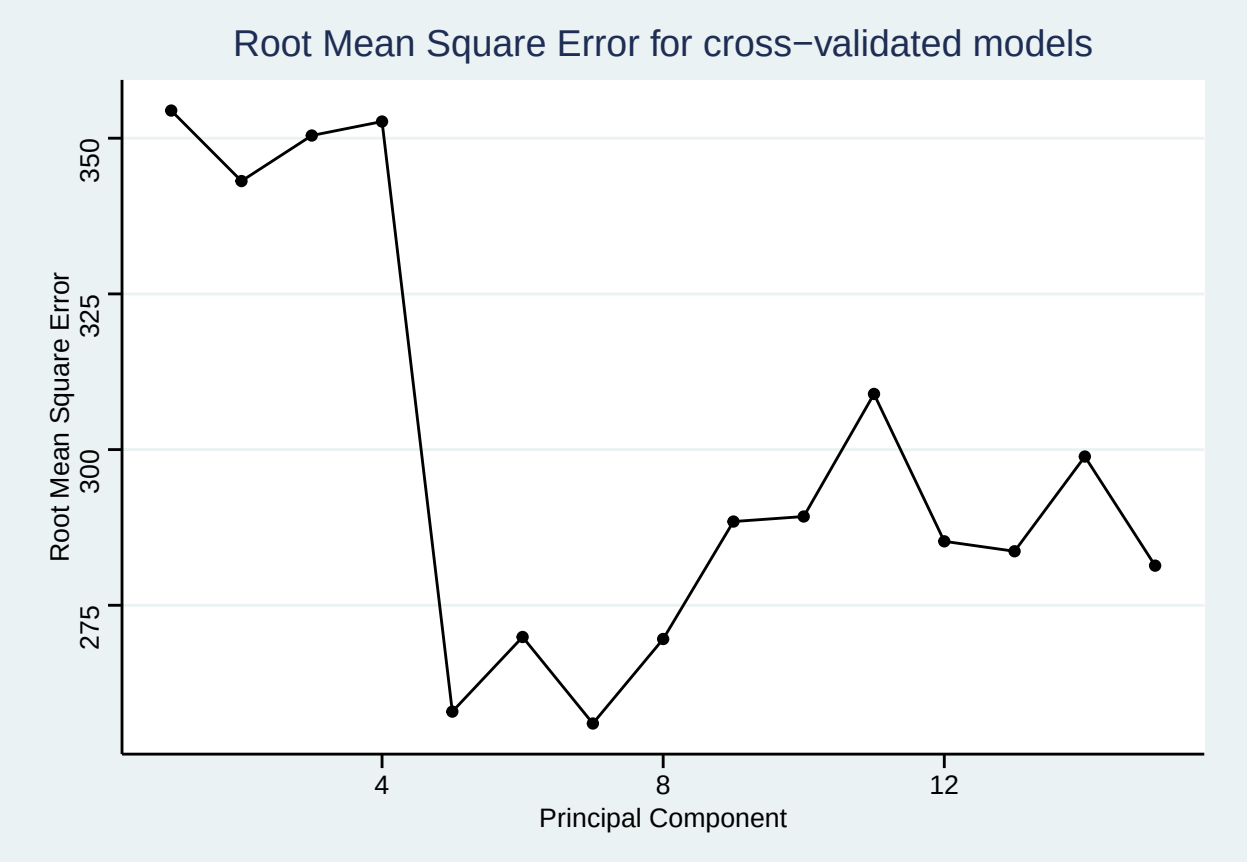
Visualize outputs of 15 models:

```
#r2
lab<-round(models$R2*100,1)
qplot(c(1:15), models$R2) +
  geom_line() +
  xlab("Principal Component") +
  ylab("R-squared") +
  ggtitle("R-squared for cross-validated models") +
  ylim(0, 1) +
  geom_text(aes(label=lab),vjust=-1)+
  theme_stata()
```

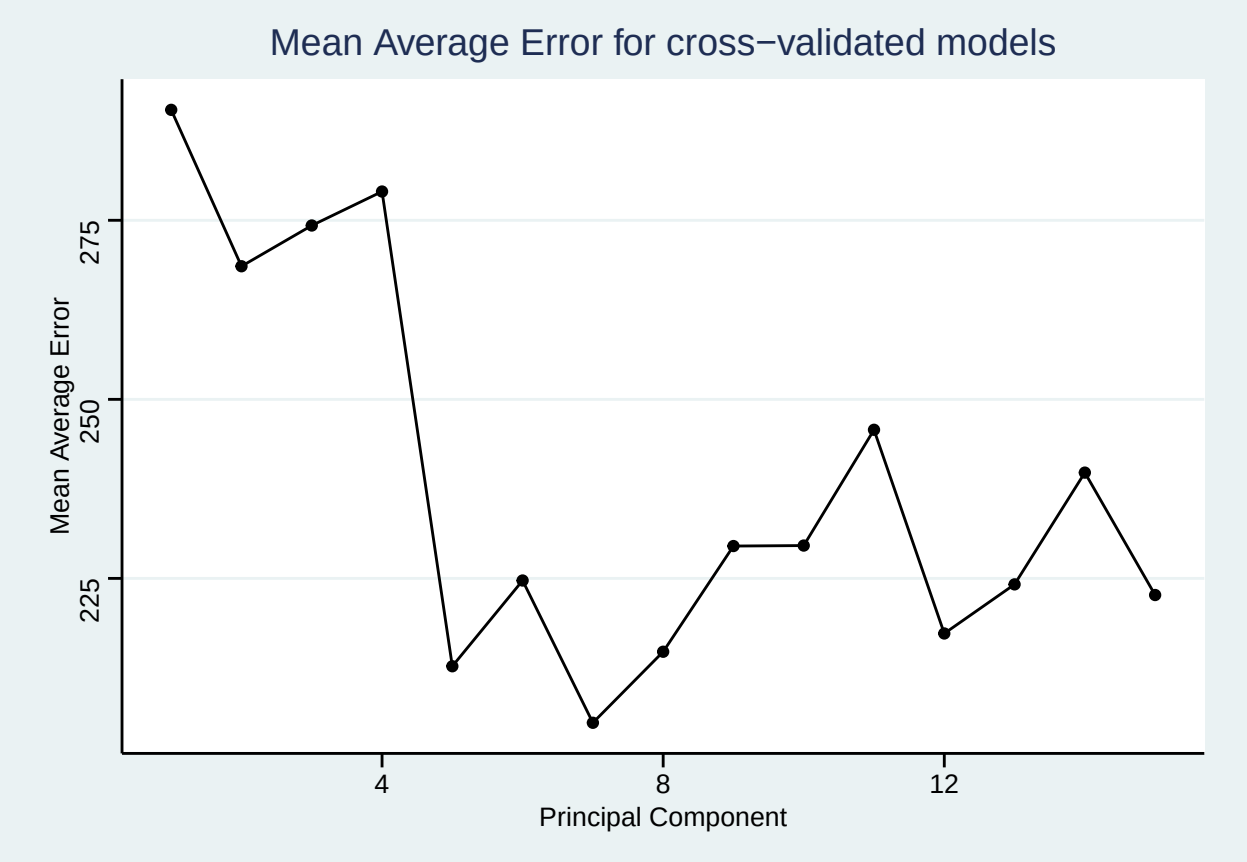



```
#RMSE
qplot(c(1:15), models$RMSE) +
  geom_line() +
  xlab("Principal Component") +
  ylab("Root Mean Square Error") +
```

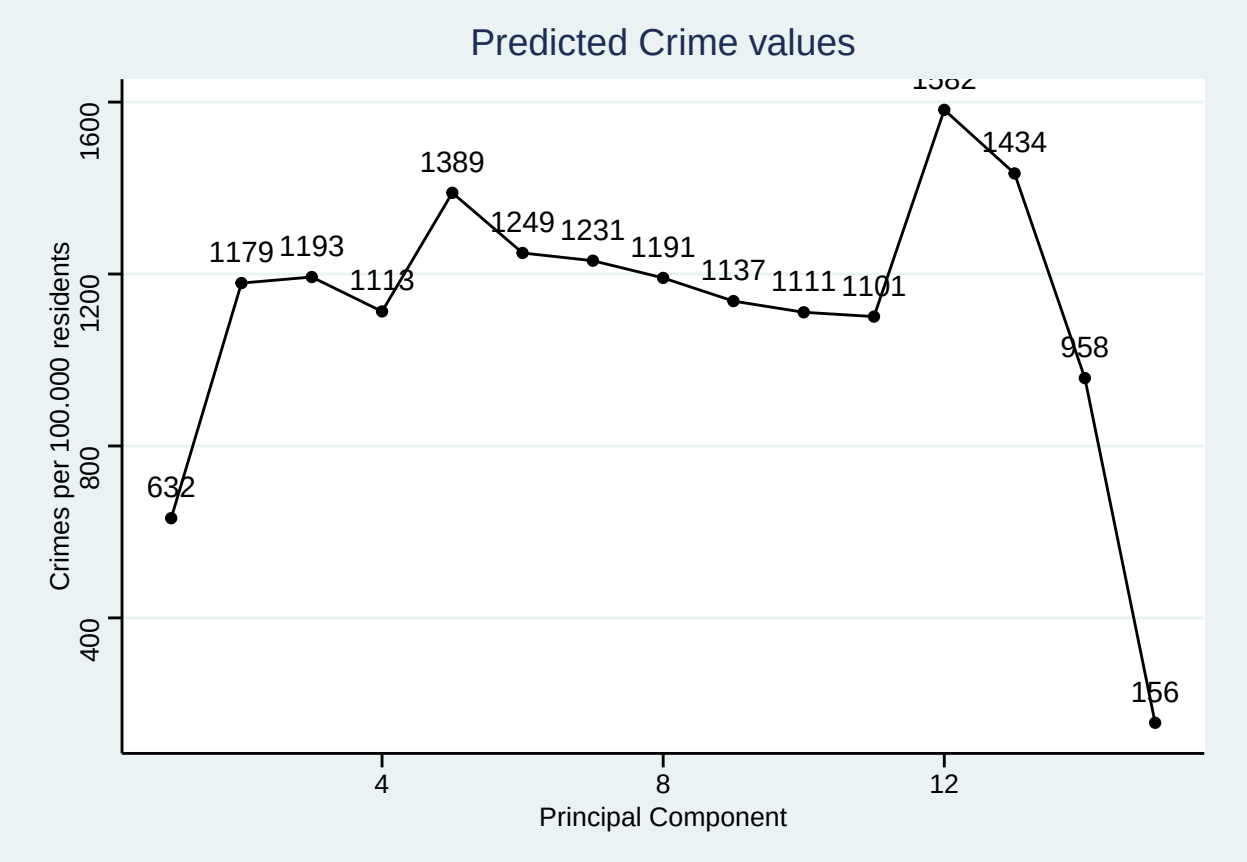
```
  ggtitle("Root Mean Square Error for cross-validated models") +
  theme_stata()
```




```
#MAE
qplot(c(1:15), models$MAE) +
  geom_line() +
  xlab("Principal Component") +
  ylab("Mean Average Error") +
  ggtitle("Mean Average Error for cross-validated models") +
  theme_stata()
```




```
#Prediction
qplot(c(1:15), models$Prediction) +
  geom_line() +
  xlab("Principal Component") +
  ylab("Crimes per 100.000 residents") +
  ggtitle("Predicted Crime values") +
  geom_text(aes(label=models$Prediction),vjust=-1)+
  theme_stata()
```



As we can see, **Adding the fifth Principal Component is a breakthough for the model** - it significantly reduces errors and doubles R-squared compared to model with PC4.

Step 10: Choosing the best model

Based on the results above, I would choose the model using **the first 5 principal components** as the best one. Analysis has shown that **adding the 5th component is very important** and gives significant improvements in model metrics. Although a model with 7 principal components gives a slightly better result (+1% to R-squared), I would not choose it for several reasons: Screeplot and plots with explained variance suggest that PC5 is the best choice; in terms of the predictor-data points ration, which is ideally 10 to 1, PC5 gives us 5 'predictors' for building a model, which is perfect for our 47 data points; finally, we are still likely to see overfitting with so little data, so racing for an extra 1% in R-squared is probably related to extra risks of increasing overfitting.

Therefore, our final model is:

Crime ~ -5933.837 + 48.37 M + 79.02So + 17.83Ed + 39.48Po1 + 39.86Po2 + 1886.95LF + 36.69M.F. + 1.55Pop + 9.54NW + 159.01U1 + 38.30U2 + 0.04Wealth + 5.54Ineq - 1523.52Prob + 3.84Time

Its key metrics are:

```
models[5,]
```

```
##   PC     R2  RMSE   MAE Prediction
## 5  5 0.5574 257.9 212.7       1389
```

Step 11: Comparing results

Here are the main metrics for the two models:

- The first model, using all the predictors, had only 6 significant factors and made very poor predictions. R-squared value, which was initially quite high at 80% (suggesting huge overfitting), lowered to 42% afterr 5-fold cross-validation:

Model 1: all predictors, 6 are significant, Residual standard error: 209.1 on 31 degrees of freedom, Multiple R-squared: 0.8031, Adjusted R-squared: 0.7078. Prediction: 155. R-squared after CV 41.9

- Then, after detecting multicollinearity, I tried building other models excluding components that caused multicollinearity, and ended up with a model with 6 predictors, which were all significant:

Model with final predictors set: M + Ed + Po1 + U2 + Ineq + Prob, Residual standard error: 200.7 on 40 degrees of freedom, Multiple R-squared: 0.7659, Adjusted R-squared: 0.7307. Prediction: 1304. R-squared after CV 63.4

This model's R-squared also reduced after cross-validation, to 63.4.

Comparing these results to the model that we built, although the prediction is very similar (1304 with the 6-factor model and 1389 with model based on the first 5 PCs), this time our R-squared is slightly lower, 55.8% instead of 63.4%.

However, using PCA gave us a lot of other benefits: we kept all the predictors and did not have to remove 60% of them for the sake of accuracy; PCA helped reduce noise in the data (which is important when we have so few data points) and gave us an improved performance of the model thanks to the independent and uncorrelated components created by it, without losing in model's accuracy. Therefore, if we need to choose one approach over another, I would go with PCA.

As a more in-depth analysis, there is one remedial measure that I would like to try in order to improve our PCA model. It is described in the next step.

[Extra] Step 12: Attempting to improve the model

One of the possible negative affects on the model may be the So variable, specifying whether the state is souther or not. The problem is that **this variable is binary**, and **PCA is designed for continuous variables**. The whole point of PCA is minimizing variance, which is impossible to do with binary variables.

I will try to improve our model by removing the So variable from the set - we will have 14 principal components then.

Just as in step 9, I will build a model for each number of components and compare the results afterwards.

Removing So from data:

```
data1 <- data[,-2]
head(data1)
```

```
##      M   Ed  Po1  Po2    LF   M.F Pop   NW    U1  U2 Wealth Ineq   Prob Time
## 1 15.1  9.1  5.8  5.6 0.510  95.0  33 30.1 0.108 4.1   3940 26.1 0.0846 26.2
## 2 14.3 11.3 10.3  9.5 0.583 101.2  13 10.2 0.096 3.6   5570 19.4 0.0296 25.3
## 3 14.2  8.9  4.5  4.4 0.533  96.9  18 21.9 0.094 3.3   3180 25.0 0.0834 24.3
## 4 13.6 12.1 14.9 14.1 0.577  99.4 157  8.0 0.102 3.9   6730 16.7 0.0158 29.9
## 5 14.1 12.1 10.9 10.1 0.591  98.5  18  3.0 0.091 2.0   5780 17.4 0.0414 21.3
## 6 12.1 11.0 11.8 11.5 0.547  96.4  25  4.4 0.084 2.9   6890 12.6 0.0342 21.0
##   Crime
## 1   791
## 2  1635
## 3   578
```

```
## 4  1969
## 5  1234
## 6   682
```

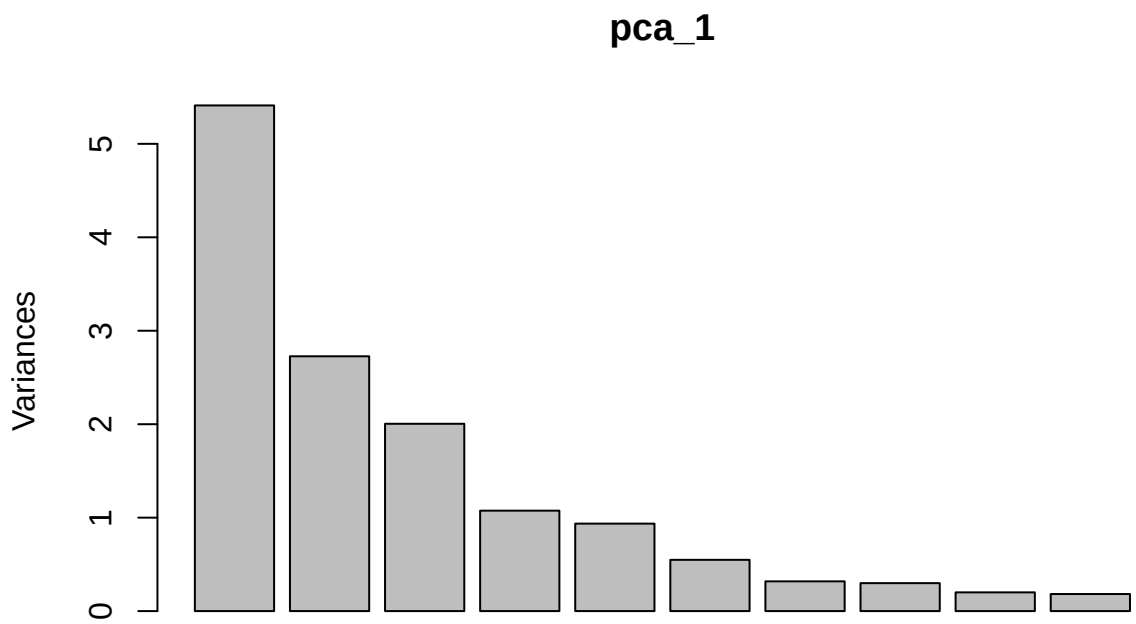
PCA for new data set:

```
pca_1 <- prcomp(data1[,-15],
                scale. = TRUE
                )
pca_1
```

```
## Standard deviations (1, .., p=14):
##  [1] 2.32616 1.65127 1.41578 1.03670 0.96745 0.74049 0.56415 0.54675 0.44751
## [10] 0.42747 0.35945 0.31852 0.25159 0.06802
##
## Rotation (n x k) = (14 x 14):
##             PC1        PC2       PC3       PC4       PC5       PC6       PC7
## M      -0.32074  0.0343672  0.174049  0.07532  0.368063  0.486976  0.06388
## Ed      0.34899  0.2577132  0.062231 -0.01365  0.002567 -0.009784 -0.22317
## Po1     0.34760 -0.2261150  0.048191 -0.26191  0.316249  0.118978  0.08789
## Po2     0.35004 -0.2195210  0.050461 -0.28699  0.290832  0.142733  0.10714
## LF      0.16641  0.3398003  0.267742  0.26712  0.328576 -0.513766 -0.24087
## M.F     0.10677  0.4170161 -0.208984  0.18561  0.529056  0.069977  0.14795
## Pop     0.14183 -0.4617924  0.080731  0.01652  0.101474 -0.538404  0.26020
## NW     -0.28869 -0.2498039  0.081408 -0.20542  0.462864 -0.005486 -0.32579
## U1      0.03517  0.0006216 -0.658286  0.21692  0.076314 -0.016795 -0.09763
## U2      0.03247 -0.2848299 -0.576046  0.09372  0.117379 -0.049419 -0.15126
## Wealth  0.40799 -0.0310539  0.006554 -0.07028 -0.016775  0.139190 -0.19609
## Ineq   -0.38224 -0.0662174  0.003019  0.09391  0.217976 -0.266691  0.33343
## Prob   -0.27282  0.1441670 -0.119344 -0.56783  0.010200 -0.242382 -0.48939
## Time   -0.01183 -0.4010023  0.230210  0.55034  0.004789  0.132277 -0.50162
##             PC8      PC9     PC10     PC11     PC12       PC13       PC14
## M      -0.49988 -0.03422 -0.44461  0.16770 -0.04653 -0.0650262 -0.0057315
## Ed     -0.33792 -0.45395  0.32920  0.52685  0.21152  0.0782934  0.0275178
## Po1     0.13925  0.12682  0.03646  0.17140 -0.30246 -0.0299259  0.6904141
## Po2     0.13104  0.04439  0.05399  0.14082 -0.26187 -0.0055491 -0.7193340
## LF      0.18168 -0.07516 -0.42990  0.01117 -0.23708 -0.0560991 -0.0303606
## M.F    -0.05048  0.39933  0.32737 -0.21232  0.25688  0.2225185 -0.0109759
## Pop    -0.57771  0.05207 -0.01711 -0.16069  0.12770  0.1143776 -0.0001785
## NW      0.17730 -0.49399  0.19082 -0.36129  0.19832  0.0463117  0.0338008
## U1     -0.24471 -0.23118  0.11527 -0.16599 -0.48133 -0.3417980 -0.0076386
## U2      0.20582  0.02203 -0.41241  0.35185  0.34019  0.2924988 -0.0092805
## Wealth -0.04089  0.12491 -0.19905 -0.22705  0.47922 -0.6567972 -0.0003812
## Ineq    0.19543  0.04013  0.23649  0.45683  0.14607 -0.5335031 -0.0219818
## Prob   -0.23726  0.42740  0.03906  0.13409 -0.08909 -0.0493604 -0.0302903
## Time   -0.01496  0.33022  0.28384  0.11818 -0.09414  0.0008812 -0.0369701
```

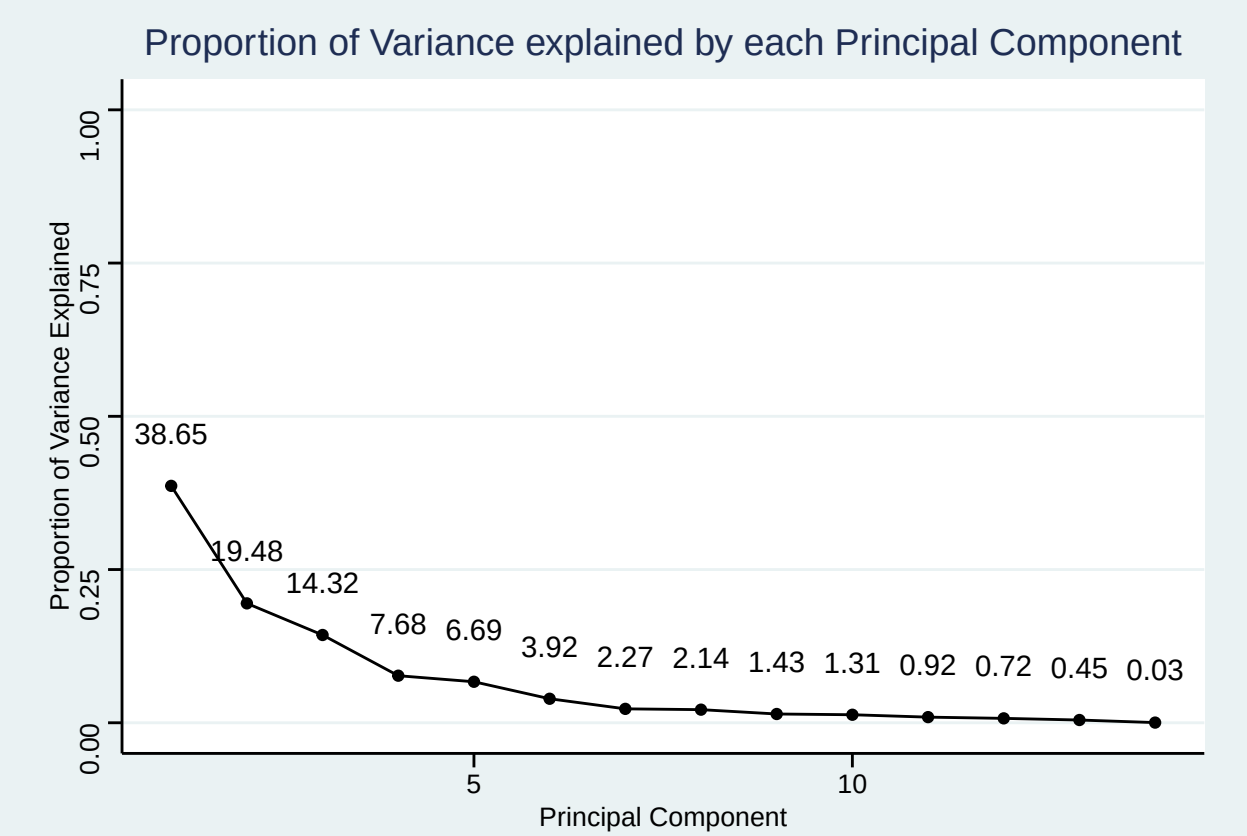
Screeplot:

```
screeplot(pca_1)
```



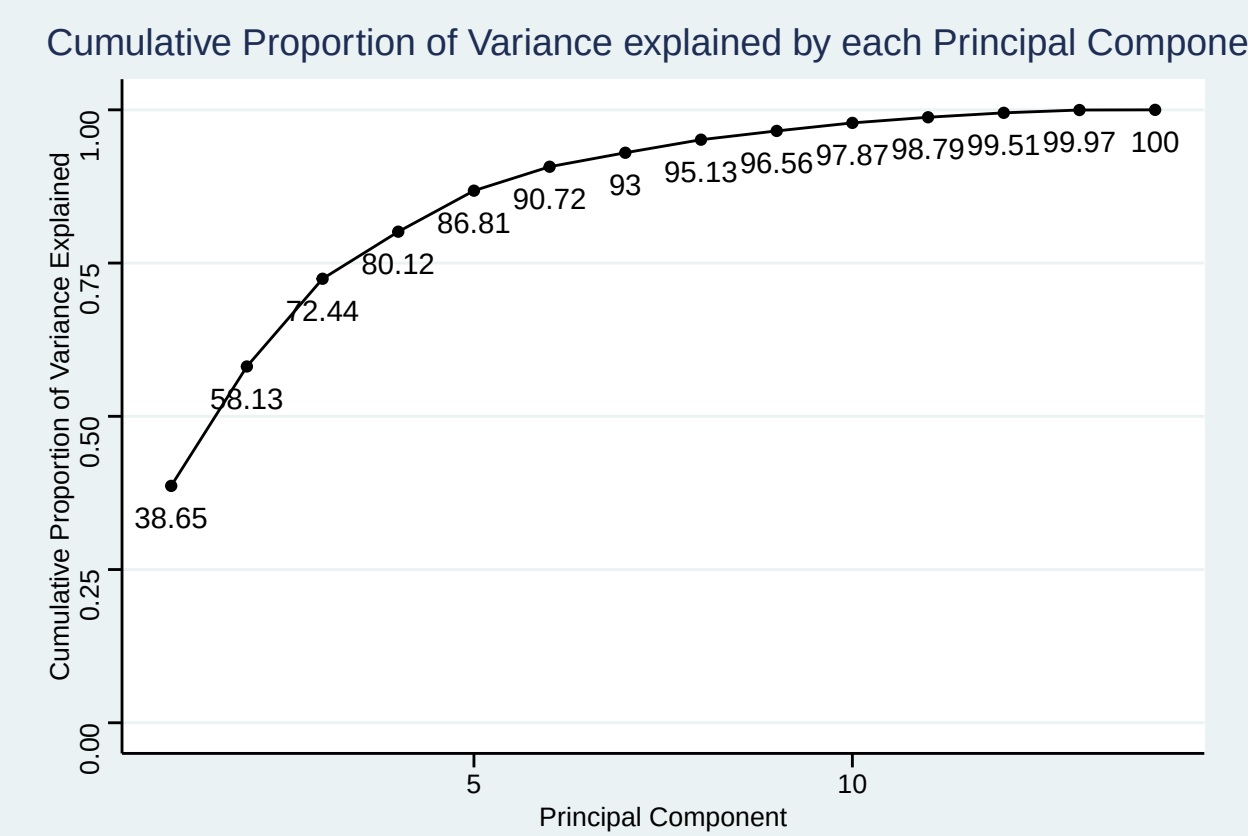
Screeplot suggests that we would need 5 Principal components for the model.

```
explained_var1 <- pca_1$sdev^2/sum(pca_1$sdev^2)
explained_var_round1<-round(explained_var1,4)
#plot explained variance by each PC:

qplot(c(1:14), explained_var_round1,) +
  geom_line() +
  xlab("Principal Component") +
  ylab("Proportion of Variance Explained") +
  ggtitle("Proportion of Variance explained by each Principal Component") +
  ylim(0, 1) +
  geom_text(aes(label=explained_var_round1*100),vjust=-2)+
  theme_stata()
```



No significant change in the Proportion of explained variance plot.

```
cumsum_var1 <- cumsum(explained_var1)
cumsum_var_round1 <- round(cumsum_var1,4)
#plot cumulative explained variance by each PC:

qplot(c(1:14), cumsum_var_round1,) +
  geom_line() +
  xlab("Principal Component") +
  ylab("Cumulative Proportion of Variance Explained") +
  ggtitle("Cumulative Proportion of Variance explained by each Principal Component") +
  ylim(0, 1) +
  geom_text(aes(label=cumsum_var_round1*100),vjust=2)+
  theme_stata()
```



As for Cumulative proportion of variance explained, 5 components now explain slightly more variance in data, 86.81% instead of 86.31% (with So).

Keeping that in mind, let's build and cross-validate model for each number of PC. Then make a prediction:

```
set.seed(1)
data_ctrl <- trainControl(method = "repeatedcv", number = 5, repeats=10)

r2_rem <- numeric(14)
rmse_rem <- numeric(14)
mae_rem <- numeric(14)
prediction_rem <- numeric(14)

for (i in 1:14) {
  pcomps <- pca_1$x[,1:i]  #extract the first i PCs
  pc_i <- cbind(Crime=data1[,15],pcomps)  #add crime variable to the set
  model <- train(Crime~.,
                     data=as.data.frame(pc_i),
                     trControl=data_ctrl,
                     method="lm")
  r2_rem[i] <- unlist(model$results[3]) #extract R squared
  rmse_rem[i] <- unlist(model$results[2]) #extract RMSE
  mae_rem[i] <- unlist(model$results[4]) #extract RMSE
  prediction_rem[i] <- ceiling(predict(model,test_data_scaled))
}
```

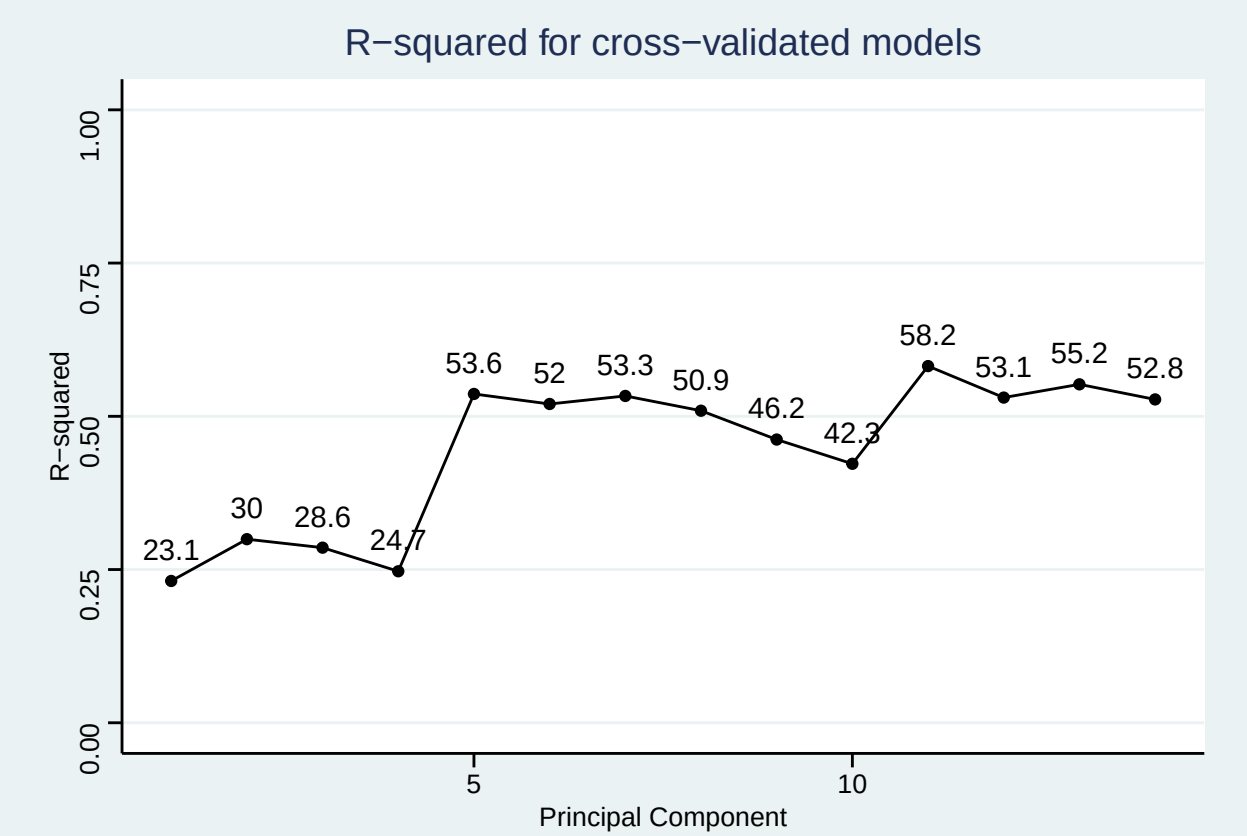
```
## Warning: 'newdata' had 1 row but variables found have 47 rows
```

```
## Warning in prediction_rem[i] <- ceiling(predict(model, test_data_scaled)):
## number of items to replace is not a multiple of replacement length
```

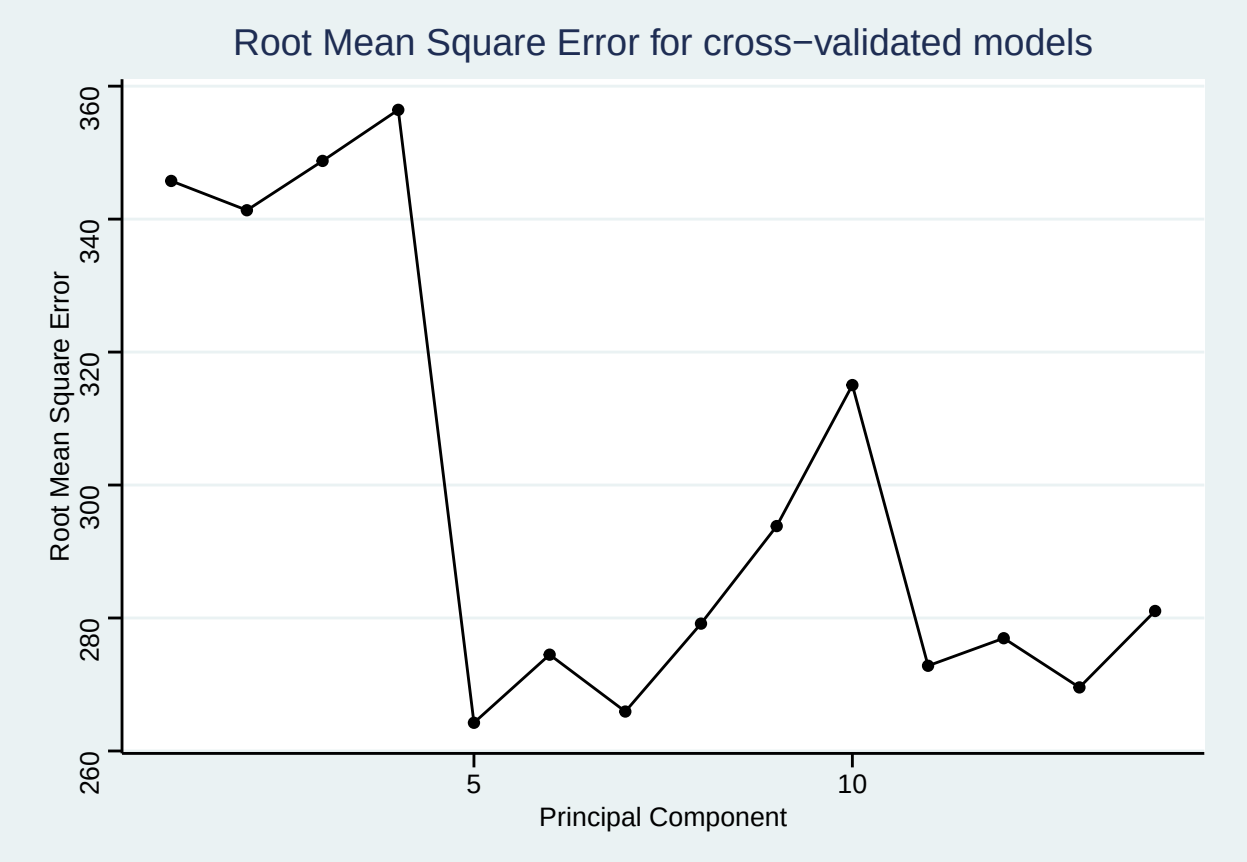
```
#compare results
models_rem <- data.frame(PC=c(1:14),R2=r2_rem, RMSE=rmse_rem, MAE=mae_rem, Prediction=prediction_rem)
models_rem
```

```
##    PC     R2  RMSE   MAE Prediction
## 1   1 0.2314 345.7 285.5        607
## 2   2 0.2995 341.3 269.8       1159
## 3   3 0.2857 348.8 275.1       1172
## 4   4 0.2472 356.4 282.7       1176
## 5   5 0.5364 264.2 218.0        892
## 6   6 0.5201 274.5 227.7       1042
## 7   7 0.5332 265.9 213.7       1026
## 8   8 0.5091 279.2 221.9        940
## 9   9 0.4621 293.8 233.4        873
## 10 10 0.4225 315.0 246.8        860
## 11 11 0.5819 272.8 208.8        750
## 12 12 0.5306 277.0 219.1       1026
## 13 13 0.5522 269.6 216.4        804
## 14 14 0.5275 281.1 219.7       -543
```

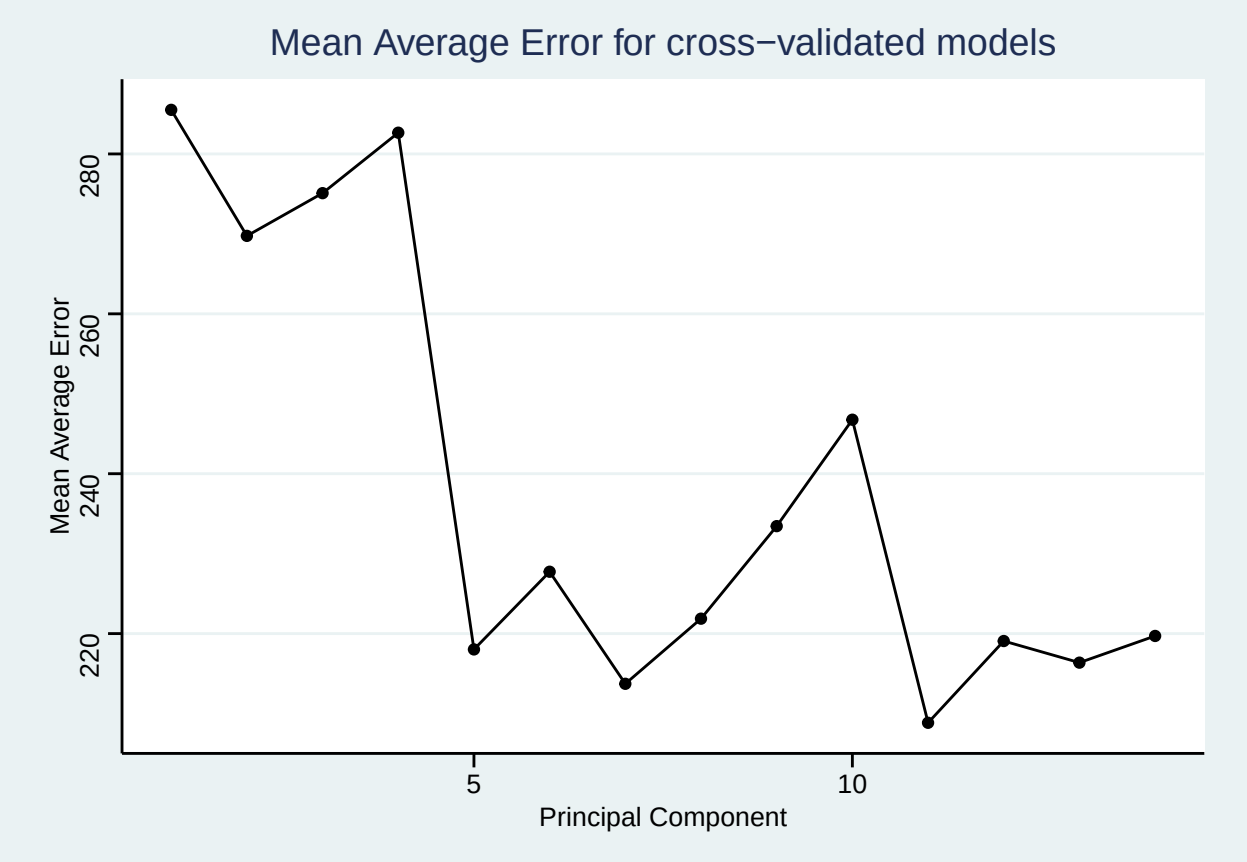
```
#r2
lab<-round(models_rem$R2*100,1)
qplot(c(1:14), models_rem$R2) +
  geom_line() +
  xlab("Principal Component") +
  ylab("R-squared") +
  ggtitle("R-squared for cross-validated models") +
  ylim(0, 1) +
  geom_text(aes(label=lab),vjust=-1)+
  theme_stata()
```



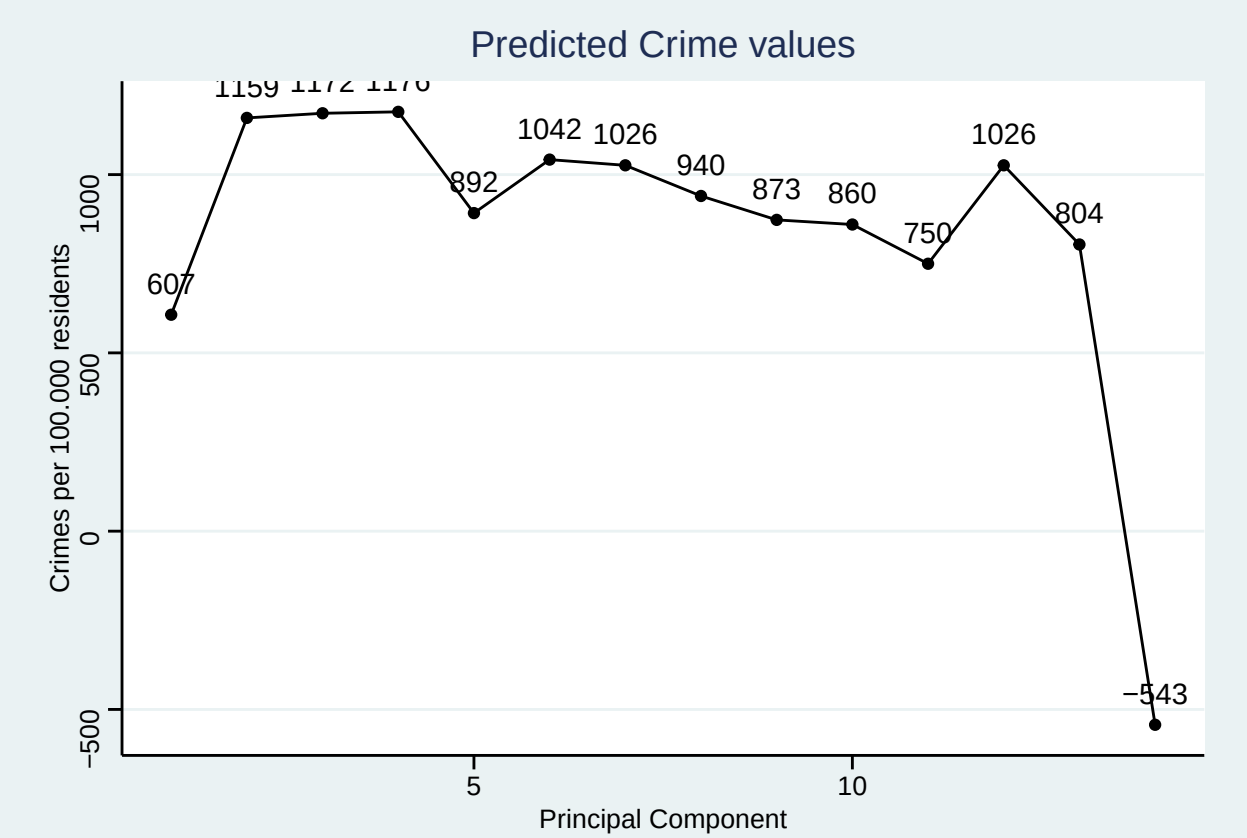

```
#RMSE
qplot(c(1:14), models_rem$RMSE) +
  geom_line() +
  xlab("Principal Component") +
  ylab("Root Mean Square Error") +
  ggtitle("Root Mean Square Error for cross-validated models") +
  theme_stata()
```




```
#MAE
qplot(c(1:14), models_rem$MAE) +
  geom_line() +
  xlab("Principal Component") +
  ylab("Mean Average Error") +
  ggtitle("Mean Average Error for cross-validated models") +
  theme_stata()
```




```
#Prediction
qplot(c(1:14), models_rem$Prediction) +
  geom_line() +
  xlab("Principal Component") +
  ylab("Crimes per 100.000 residents") +
  ggtitle("Predicted Crime values") +
  geom_text(aes(label=models_rem$Prediction),vjust=-1)+
  theme_stata()
```



Unfortunately, after removal of So the R-squared value has not improved and even lowered by 2% to 53.6%. However, knowing that PCA works better with no binary variables in the data, we can now see that the model with the first 5 PCs is clearly better that the one with 7PCS - for 5PCs, the R-squared is slightly higher, and the RMSE is a bit lower. The prediction has changed to 1452 (compared to 1389), which appears to be slightly high. Perhaps in the future modules I would be able to try other remedial measures on the So predictor to avoid its removal and make the model more accurate. For now, I would choose the previous model based on 5 principal components with R-square of 55% and the following equation:

Crime ~ -5933.837 + 48.37 M + 79.02So + 17.83Ed + 39.48Po1 + 39.86Po2 + 1886.95LF + 36.69M.F. + 1.55Pop + 9.54NW + 159.01U1 + 38.30U2 + 0.04Wealth + 5.54Ineq - 1523.52Prob + 3.84Time