Advanced Regression Models for Crime Data

Alena Fedash

2022-10-15

Contents

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

1. Stepwise Regression

Step 0: Load the libraries

```r
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)
library(rsample)
library(glmnet)
```

Step 1: Load the dataset

```r
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
```

Step 2: Basic Explorations

CHeck short summary for each variable to refer to later:

```r
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
```

As we can see, predictors have very different scales - so we should scale for each of our analysis steps to avoid uneven effect due to the range of values in some parameters.

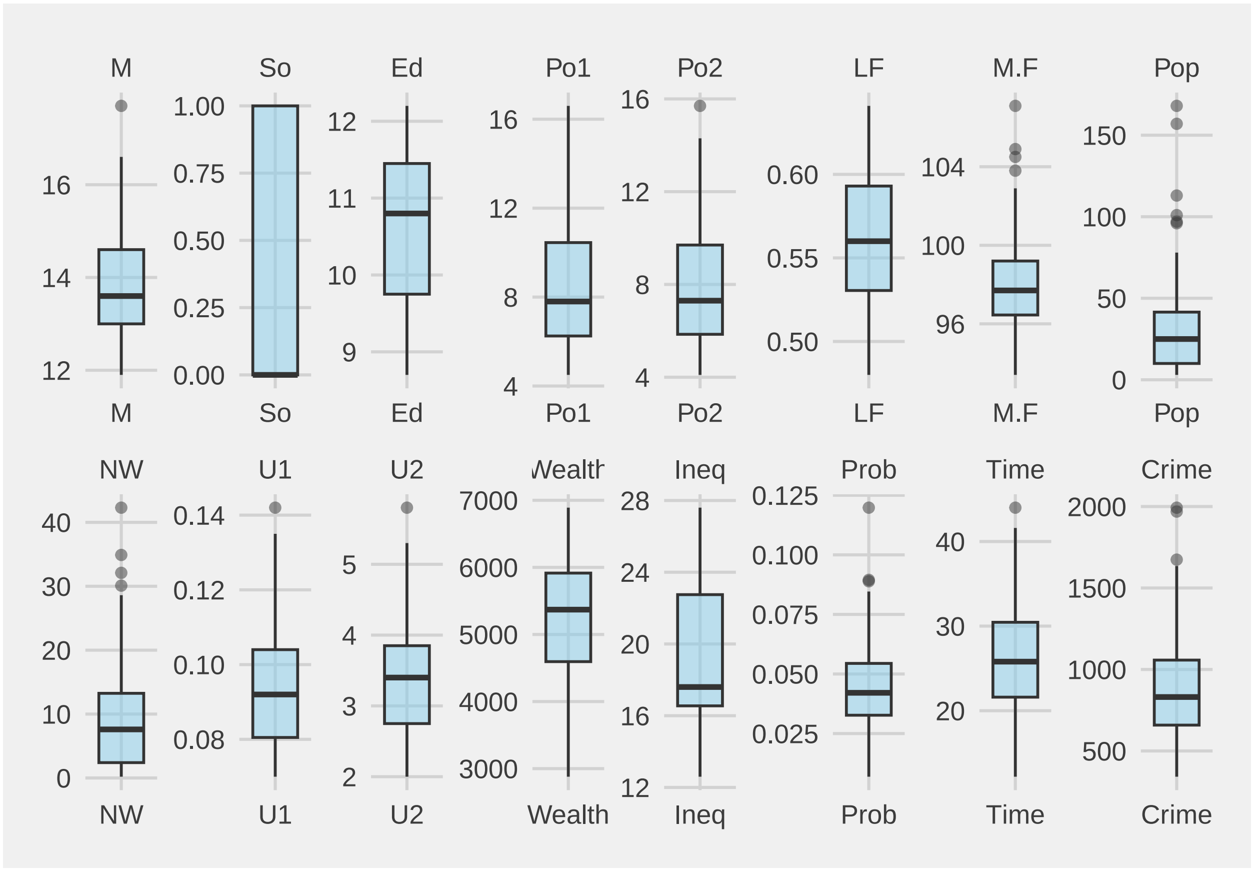
Visualize distribution of values in parameters:

```r
#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
```

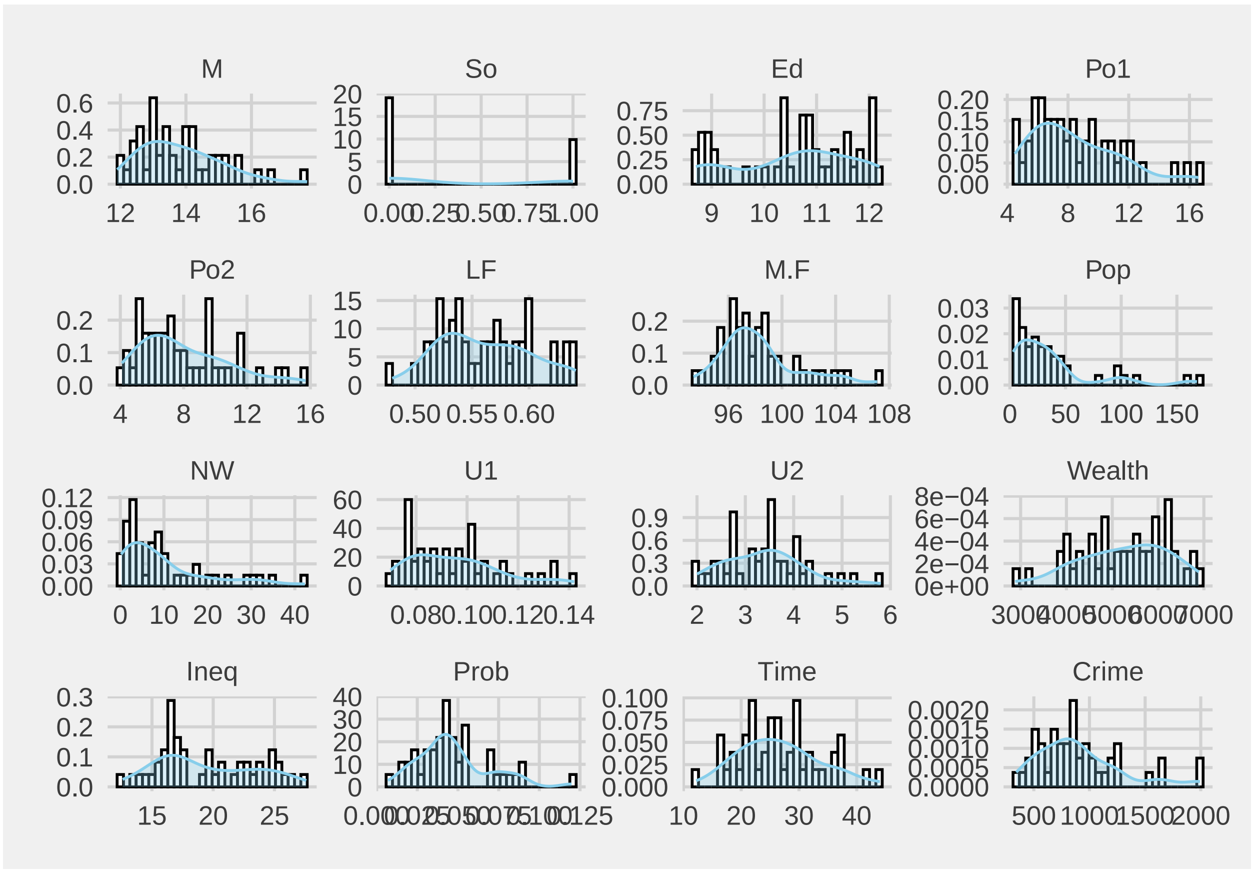
```r
#boxplots
box_plots <- ggplot(melted,
                    aes(x=factor(variable), y=value))+
              geom_boxplot(alpha=.5, fill="skyblue")+
              facet_wrap(~variable, ncol=8, scale="free")+
              theme_fivethirtyeight()
```

```
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
```

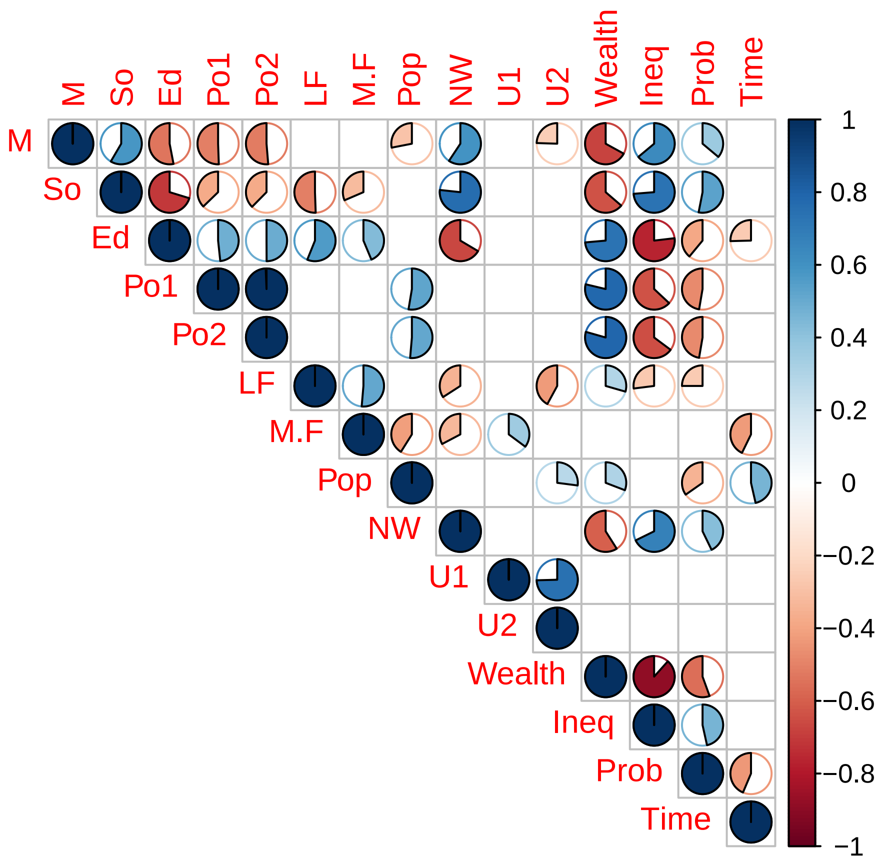
```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.

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



A lot of right-skewed parameters. 'So' variable is binary (southern state or not).

Check pairwise correlation - visualize correlation of variables in combination with p-values (to check for significant correlations):

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



As we see now and as we have seen in previous explorations, we do have some multicollinearity in the data. So, the first method that we're going to try, stepwise regression, coulf help us resolve it. However, multicollinearity could also cause some problems - for instance, if some correlated parameters have a similar contribution to the model fit, we might miss both of them in case they are not as significant individually. On the other hand, stepwise regression is a good technique for initial exploration of the variable effects on model's prediction.

Let's start with stepwise regression and see how it will deal with correlated parameters.

Step 3: Scaling the Data

Since the ranges of predictors are very different, we will scale them and add into a new data frame with unscaled response. Since 'So' is binary, we scale everything except it and the respone, and then join all columns together:

```
#scale except So (column 2) and Crime (response)
scaled <- as.data.frame(scale(data[,-c(2,16)]))
#add So and response back
scaled <- cbind(So=data[,2], scaled, Crime=data[,16])
head(scaled)
```

```
##   So          M         Ed        Po1        Po2         LF         M.F
## 1  1  0.9886930 -1.3085099 -0.9085105 -0.8666988 -1.2667456 -1.12060499
## 2  0  0.3521372  0.6580587  0.6056737  0.5280852  0.5396568  0.98341752
## 3  1  0.2725678 -1.4872888 -1.3459415 -1.2958632 -0.6976051 -0.47582390
## 4  0 -0.2048491  1.3731746  2.1535064  2.1732150  0.3911854  0.37257228
## 5  0  0.1929983  1.3731746  0.8075649  0.7426673  0.7376187  0.06714965
## 6  0 -1.3983912  0.3898903  1.1104017  1.2433590 -0.3511718 -0.64550313
##           Pop           NW          U1         U2     Wealth       Ineq
## 1 -0.09500679  1.943738564  0.69510600  0.8313680 -1.3616094  1.6793638
## 2 -0.62033844  0.008483424  0.02950365  0.2393332  0.3276683  0.0000000
## 3 -0.48900552  1.146296747 -0.08143007 -0.1158877 -2.1492481  1.4036474
## 4  3.16204944 -0.205464381  0.36230482  0.5945541  1.5298536 -0.6767585
## 5 -0.48900552 -0.691709391 -0.24783066 -1.6551781  0.5453053 -0.5013026
## 6 -0.30513945 -0.555560788 -0.63609870 -0.5895155  1.6956723 -1.7044289
##         Prob        Time Crime
## 1  1.6497631 -0.05599367   791
## 2 -0.7693365 -0.18315796  1635
## 3  1.5969416 -0.32416470   578
## 4 -1.3761895  0.46611085  1969
## 5 -0.2503580 -0.74759413  1234
## 6 -0.5669349 -0.78996812   682
```

Step 4: Stepwise Regression on scaled data

Let's start by performing stepwise regression using the scaled data and all variables. I will use the caret package to combine this method with repeated cross-validation (5 folds since we only have 47 data points and 10 repeats to minimize random effects).

4.1: Choosing between forward and backward stepwise regression

Should we use backward or forward stepwise regression?

I decided to go for backward stepwise regression for several reasons. First of all, we do not have an extreme case when the number of predictors is larger than the amount of data points (in such cases, forward regression is normally used). Also, backward has an advantage of starting with the full model instead of a null one - from the start, it simultaneously considers the effects of all variables, which is especially good for high collinearity cases like ours (as we saw on the correlation plot). With highly correlated variables, backward, unlike forward, is usually forced to keep both variables and not omitting both of them - so we are more likely to keep those important effects. Also, forward may cause suppressor effects when it forces some of predictors to be constant to make others significant. To sum up, we would most likely use forward only in a case when we have more predictors than data points.

Let's find the optimal variable set using cross-validated backward regression:

4.2: Running backward stepwise regression

```r
set.seed(1)
data_ctrl <- trainControl(method = "repeatedcv", number = 5, repeats=10)
stepwise <- train(Crime~.,
                    data=scaled,
                    trControl=data_ctrl,
                    method="lmStepAIC",
                    direction = "backward", trace=F)
summary(stepwise)
```

```
##
## Call:
## lm(formula = .outcome ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq +
##     Prob, data = dat)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -444.70 -111.07    3.03  122.15  483.30
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      28.52  31.731  < 2e-16 ***
## M             117.28      42.10   2.786  0.00828 **
## Ed            201.50      59.02   3.414  0.00153 **
## Po1           305.07      46.14   6.613 8.26e-08 ***
## M.F            65.83      40.08   1.642  0.10874
## U1           -109.73      60.20  -1.823  0.07622 .
## U2            158.22      61.22   2.585  0.01371 *
## Ineq          244.70      55.69   4.394 8.63e-05 ***
## Prob          -86.31      33.89  -2.547  0.01505 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 195.5 on 38 degrees of freedom
## Multiple R-squared:  0.7888, Adjusted R-squared:  0.7444
## F-statistic: 17.74 on 8 and 38 DF,  p-value: 1.159e-10
```

As we can see, backward stepwise regression run on full variable set suggests us to use a model with **8 variables: M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob**. R2 **on the training set** is **78.88%**, and **Adjusted R2 is 74.44%**.

The p-value for one of the predictors, M.F., is 0.10. We can keep it for now and use it in a model, since it is lower than 0.15: it might turn out that this factors is just dragging the quality of the model down, or it might turn out to be important. Let's keep it for now.

4.3: Linear Regression using the variables from stepwise regression

Now we can use those variables only to build a linear regression model and **cross-validate** it (cross-validation above was used to choose the best predictor set only, not to validate the final chosen set). I will use caret with **leave-one-out CV**, since we have so few data points.

```
set.seed(1)
#leave-one-out CV
data_ctrl <- trainControl(method = "LOOCV")

m1 <- train(Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob,
                     data=scaled,
                     trControl=data_ctrl,
                     method="lm")
m1$results
```

```
##   intercept     RMSE  Rsquared      MAE
## 1      TRUE 220.5928 0.6737516 174.1015
```

```
summary(m1)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -444.70 -111.07    3.03  122.15  483.30
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      28.52  31.731  < 2e-16 ***
## M             117.28      42.10   2.786  0.00828 **
## Ed            201.50      59.02   3.414  0.00153 **
## Po1           305.07      46.14   6.613 8.26e-08 ***
## M.F            65.83      40.08   1.642  0.10874
## U1           -109.73      60.20  -1.823  0.07622 .
## U2            158.22      61.22   2.585  0.01371 *
## Ineq          244.70      55.69   4.394 8.63e-05 ***
## Prob          -86.31      33.89  -2.547  0.01505 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 195.5 on 38 degrees of freedom
## Multiple R-squared:  0.7888, Adjusted R-squared:  0.7444
## F-statistic: 17.74 on 8 and 38 DF,  p-value: 1.159e-10
```

As we can see, after cross-validation, the **R2 is 67.4%**. Will it improve if we remove the M.F. predictor with p-value of 0.1? Let's check:

4.4: Removing insignificant predictors from the model

```r
set.seed(1)
#leave-one-out CV
data_ctrl <- trainControl(method = "LOOCV")

m2 <- train(Crime ~ M + Ed + Po1 + U1 + U2 + Ineq + Prob,
                     data=scaled,
                     trControl=data_ctrl,
                     method="lm")
m2$results
```

```
##   intercept   RMSE  Rsquared     MAE
## 1      TRUE 222.38 0.6680698 167.584
```

```r
summary(m2)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -520.76 -105.67    9.53  136.28  519.37
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      29.14  31.062  < 2e-16 ***
## M             134.20      41.70   3.218   0.0026 **
## Ed            244.38      54.07   4.520 5.62e-05 ***
## Po1           314.89      46.73   6.738 4.91e-08 ***
## U1            -63.86      54.48  -1.172   0.2482
## U2            134.13      60.71   2.209   0.0331 *
## Ineq          264.65      55.52   4.767 2.61e-05 ***
## Prob          -84.83      34.61  -2.451   0.0188 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 199.8 on 39 degrees of freedom
## Multiple R-squared:  0.7738, Adjusted R-squared:  0.7332
## F-statistic: 19.06 on 7 and 39 DF,  p-value: 8.805e-11
```

Now, **on training data we have a slightly lower R2 of 77.4% (compared to 77.88 before), and after cross-validation it reduces to 66.8%** - this is not a significant reduction, so removing M.F. seems to have been the right choice.

However, **now U1 looks insignificant with a p-value of 0.24 - way higher than 0.15**. Let's remove that as well and cross-validate the new model to see the change:

```r
set.seed(1)
#leave-one-out CV
data_ctrl <- trainControl(method = "LOOCV")

m3 <- train(Crime ~ M + Ed + Po1 + U2 + Ineq + Prob,
                     data=scaled,
                     trControl=data_ctrl,
                     method="lm")
m3$results
```

```
##   intercept     RMSE  Rsquared      MAE
## 1      TRUE 221.0758 0.6706867 165.5131
```

```r
summary(m3)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -470.68  -78.41  -19.68  133.12  556.23
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      29.27  30.918  < 2e-16 ***
## M             131.98      41.85   3.154  0.00305 **
## Ed            219.79      50.07   4.390 8.07e-05 ***
## Po1           341.84      40.87   8.363 2.56e-10 ***
## U2             75.47      34.55   2.185  0.03483 *
## Ineq          269.91      55.60   4.855 1.88e-05 ***
## Prob          -86.44      34.74  -2.488  0.01711 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 200.7 on 40 degrees of freedom
## Multiple R-squared:  0.7659, Adjusted R-squared:  0.7307
## F-statistic: 21.81 on 6 and 40 DF,  p-value: 3.418e-11
```

This run, all predictors are significant, so the model looks good. Moreover, **R2 on training data**

is 76.6%, and 67.1% after cross-validation.

4.5: Final model evaluation

With the final model we get a better result compared to the previous model, and a very similar one to the model with 8 parameters. **Another improvement is the reduction of mean absolute error MAE to 165.5, compared to 174 in the model with 8 variables and 167 in the model with 7**. It means that the average error for a prediction made with this model would be lower compared to the others.

Considering that **the model with 6 predictors is simpler than the others, hence, easier to explain**, has **lower MAE** and **similar performance based on R2**, we should choose it as the final one.

```
m3$finalModel
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Coefficients:
## (Intercept)            M           Ed          Po1           U2         Ineq
##      905.09       131.98       219.79       341.84        75.47       269.91
##        Prob
##      -86.44
```

If we take a look at coefficients, we can see that there are 6 significant factors affecting the Crime rate in a state. 5 of them have a positive effect on the crime number (increase it) - percentage of males aged 14-24 in total population (M), mean years of schooling for people aged 25+ (Ed), police protection expenditure in 1960 (Po1), unemployment rate of 35-39y.o. males (U2) and Income inequality (Ineq). Probability of imprisonment has a negative effect (reduces) on Crime. The largest coefficient (and effect) is once again on Po1 (police protection expenditure 1960). It is followed by Inequality and Education.

Step 5: Observing benefits and drawbacks Stepwise Regression

There are several **benefits** that I observed while performing stepwise regression:

- It was easy to implement: we just needed to run 1 function to get the narrowed down set of variables. We did scale the data beforehand, but it was a logical step considering the different scales of parameters.
- It helped generalize our model: at the beginning we had 15 predictors for 47 data points, some predictors were describing similar factors (like Wealth and Inequality, U1 and U2, etc), some were highly correlated. Stepwise regression helped us quickly reduce the number of predictors to 8, which also gave a good quality of fit
- It made the model easier to interpret: explaining a model with fewer variables is always better.

As a **drawback** of implementation, I can point out the fact that not all of the variables we got were significant - we still had to remove two of them to come to a model with significant parameters only. Also, we still had to build new models and cross-validate them separately, so it is not a 1-function solution (which is not a problem).

Overall, this method sets a good baseline for further analysis and gives a quick, simple, easy-to-explain solution. We can now use these results to compare stepwise to other methods.

(Additional) Step 6: Stepwise Regression on PCA

Although the results we got with backward stepwise regression seem to be better than what we had before, and the model is simple and easy to explain, it is still worth to try improving the quality of the model by running it on PCA.

In general, PCA can be helpful for cases with few data points, as it gets more variability within its first components, which we can use to find variables with higher effect on the respones.

6.1: Principal Component Analysis

Start by running PCA on the data (excluding the response):

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

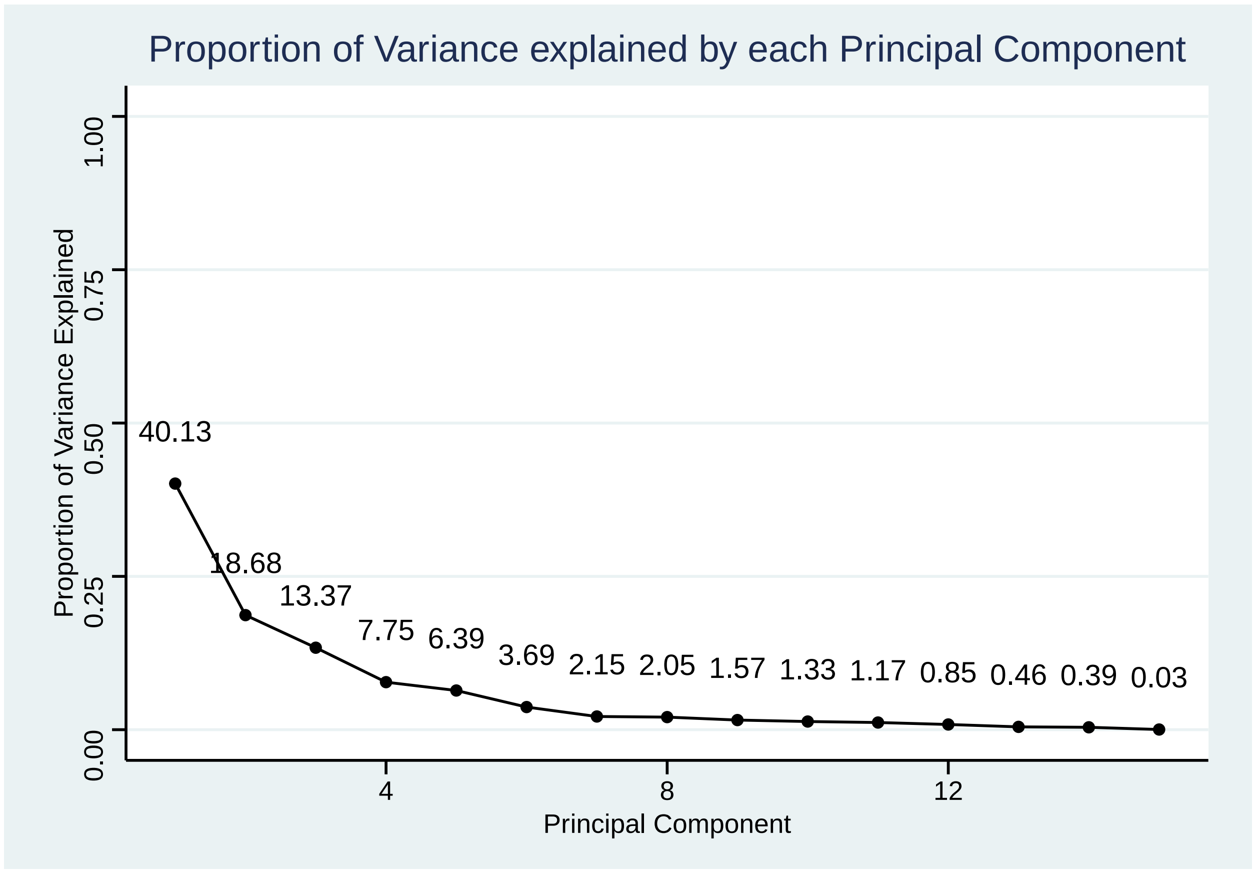
```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.4534 1.6739 1.4160 1.07806 0.97893 0.74377 0.56729
## Proportion of Variance 0.4013 0.1868 0.1337 0.07748 0.06389 0.03688 0.02145
## Cumulative Proportion  0.4013 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
```

The first component explains the highest proportion of variance - 40.13% and has a standard deviation of 2.45.

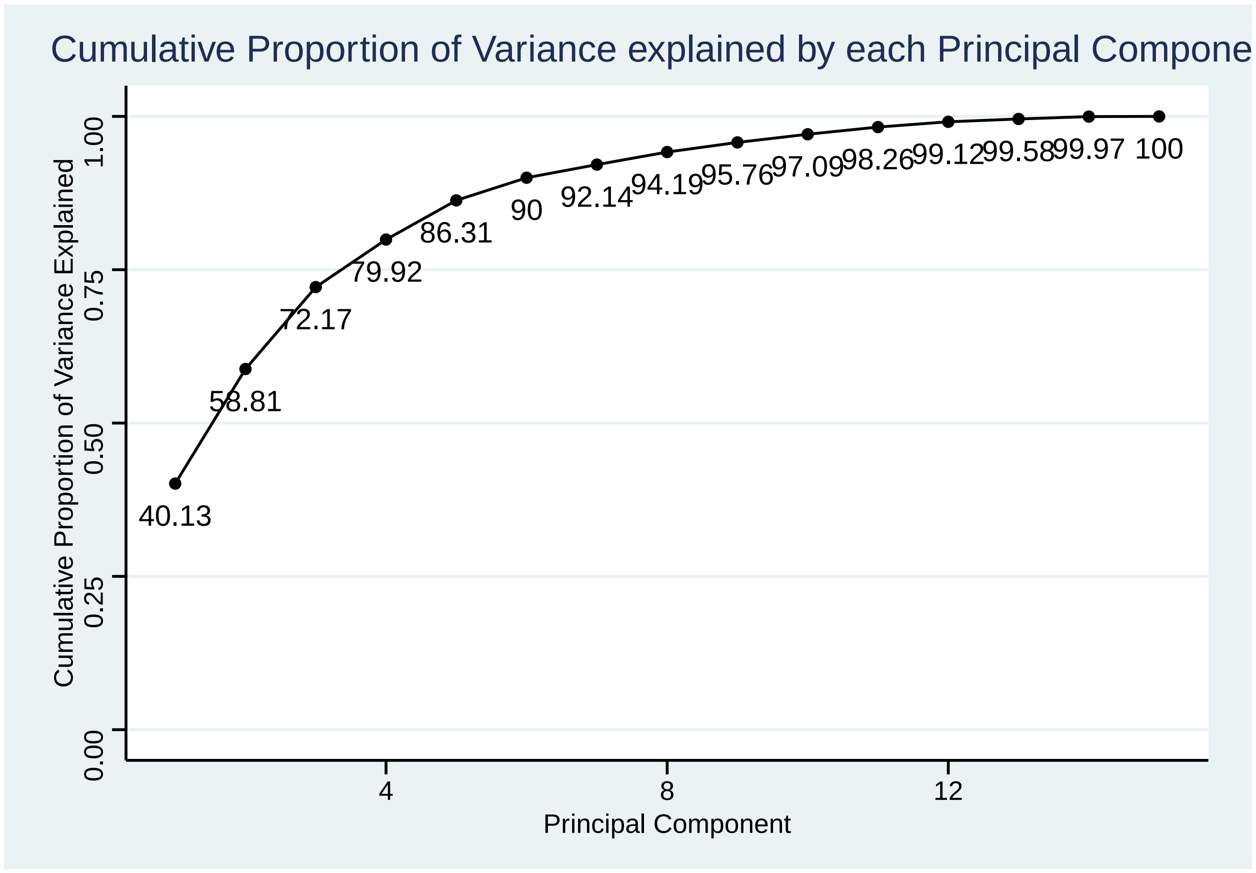
Check how much variance explained reduces with each PC:

```r
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()
```

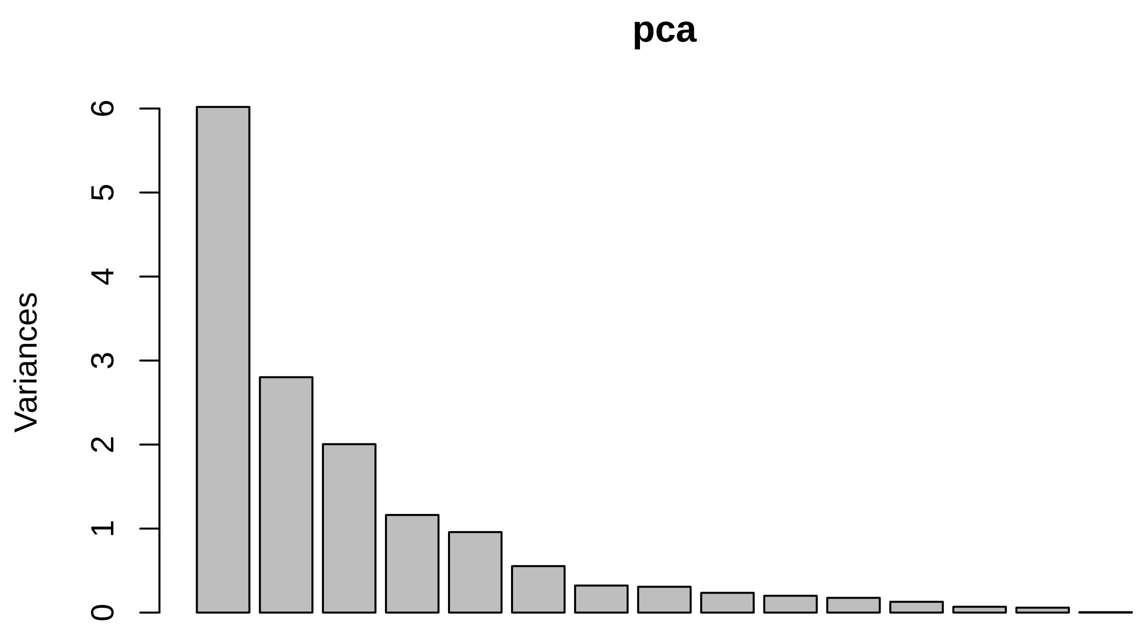
```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```



```r
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()
```



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



Based on the screeplot, we could choose the first 5 PCs and run stepwise on them. However, since stepwise regression is supposed to be reducing the number of variables,and those on the very right might have least variance, but not necessarily the least predictive power, let's run stepwise on all the components and see which ones would be left.

Add response variable to PCs:

```
pca_data <- as.data.frame(cbind(pca$x, Crime=data[,16]))
head(pca_data)
```

```
##         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 Crime
## 1  0.11261680  0.3269649 -0.02338401   791
## 2 -0.29751651  0.2523567  0.06076368  1635
## 3 -0.05216054 -0.4865511 -0.04211750   578
## 4 -0.08838131  0.1496784 -0.02917497  1969
## 5  0.72215224  0.1310272  0.07514940  1234
## 6  0.13517271  0.1949257 -0.01558610   682
```

6.2: Stepwise Regression on all PCs

Let's use the same parameters of repeated cross-validation and run backward stepwise regression on all components:

```
set.seed(1)
data_ctrl <- trainControl(method = "repeatedcv", number = 5, repeats=10)
stepwise_pca <- train(Crime~.,
                      data=pca_data,
                      trControl=data_ctrl,
                      method="lmStepAIC",
                      direction = "backward", trace=F)
summary(stepwise_pca)
```

```
##
## Call:
## lm(formula = .outcome ~ PC1 + PC2 + PC4 + PC5 + PC6 + PC7 + PC12 +
##     PC14 + PC15, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -351.54  -93.58  -20.42  121.74  553.91
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      29.35  30.836  < 2e-16 ***
## PC1            65.22      12.09   5.393 4.17e-06 ***
## PC2           -70.08      17.72  -3.954 0.000334 ***
## PC4            69.45      27.52   2.523 0.016048 *
## PC5          -229.04      30.31  -7.557 5.20e-09 ***
## PC6           -60.21      39.89  -1.509 0.139665
## PC7           117.26      52.30   2.242 0.031040 *
## PC12          289.61      82.87   3.495 0.001249 **
## PC14          219.19     122.70   1.786 0.082235 .
## PC15          622.21     436.77   1.425 0.162660
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 201.2 on 37 degrees of freedom
## Multiple R-squared:  0.7823, Adjusted R-squared:  0.7293
## F-statistic: 14.77 on 9 and 37 DF,  p-value: 8.755e-10
```

As we can see, this time we have **9 predictors suggested for use by stepwise method**. Even the last two components were used. On training data, we have a similar R2 value of 78.2 and adjusted R2 of 72.9.

6.3: Linear regression on predictors from stepwise regression

Let's use the variables suggested by stepwise regression to build a new model and cross-validate it. Although some predictors do not seem to be significant, let's still include them in the first model:

```
set.seed(1)
#leave-one-out CV
data_ctrl <- trainControl(method = "LOOCV")

m1_pca <- train(Crime ~ PC1 + PC2 + PC4 + PC5 + PC6 + PC7 + PC12 + PC14 + PC15,
                     data=pca_data,
                     trControl=data_ctrl,
                     method="lm")
summary(m1_pca)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -351.54  -93.58  -20.42  121.74  553.91
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      29.35  30.836  < 2e-16 ***
## PC1            65.22      12.09   5.393 4.17e-06 ***
## PC2           -70.08      17.72  -3.954 0.000334 ***
## PC4            69.45      27.52   2.523 0.016048 *
## PC5          -229.04      30.31  -7.557 5.20e-09 ***
## PC6           -60.21      39.89  -1.509 0.139665
## PC7           117.26      52.30   2.242 0.031040 *
## PC12          289.61      82.87   3.495 0.001249 **
## PC14          219.19     122.70   1.786 0.082235 .
## PC15          622.21     436.77   1.425 0.162660
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 201.2 on 37 degrees of freedom
## Multiple R-squared:  0.7823, Adjusted R-squared:  0.7293
## F-statistic: 14.77 on 9 and 37 DF,  p-value: 8.755e-10
```

```
m1_pca$results
```

```
##   intercept    RMSE Rsquared      MAE
## 1      TRUE 232.367 0.639978 175.6689
```

Compared to the models run on initial variables, not PCs, we have a worse result - **R2 of 64% after cross-validation**, and higher errors. Perhaps it is caused by the two insignificant variables in the model? Let's try removing them.

6.4: Final set of predictors

```
set.seed(1)
#leave-one-out CV
data_ctrl <- trainControl(method = "LOOCV")
#removing PC6 and PC15
m2_pca <- train(Crime ~ PC1 + PC2 + PC4 + PC5 + PC7 + PC12 + PC14 ,
                     data=pca_data,
                     trControl=data_ctrl,
                    method="lm")
summary(m2_pca)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -389.89 -112.34  -20.12  134.00  465.46
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      30.21  29.962  < 2e-16 ***
## PC1            65.22      12.45   5.240 5.86e-06 ***
## PC2           -70.08      18.24  -3.842 0.000438 ***
## PC4            69.45      28.32   2.452 0.018791 *
## PC5          -229.04      31.19  -7.343 7.28e-09 ***
## PC7           117.26      53.82   2.178 0.035477 *
## PC12          289.61      85.28   3.396 0.001585 **
## PC14          219.19     126.28   1.736 0.090503 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 207.1 on 39 degrees of freedom
## Multiple R-squared:  0.7569, Adjusted R-squared:  0.7133
## F-statistic: 17.35 on 7 and 39 DF,  p-value: 3.412e-10
```

```
m2_pca$results
```

```
##   intercept     RMSE  Rsquared      MAE
## 1      TRUE 233.6217 0.6330384 190.5296
```

No significant improvements - R2 of 63% after cross-validation, and a larger MAE compared to the previous model. However, this model is **simpler, so it is easier to explain**.

Comparing this model run on PCs to the one with initial variables, I would choose the latter, since the model with initial variables showed better performance, higher R2 and a lower error.

Step 7: Stepwise Regression: Summary of Results

Model	Predictors	R2 (training)	Adj. R2 (training)	R2 (cross-validated)
Stepwise Regression, original data, all variables	M, Ed, Po1, M.F., U1, U2, Ineq, Prob	78.88	74.44	67.38
Stepwise Regression, original data, significant variables	M, Ed, Po1, U2, Ineq, Prob	76.59	73.07	67.07
Stepwise Regression, PCA data, all variables	PC1, PC2, PC4, PC5, PC6, PC7, PC12, PC14, PC15	78.23	72.93	63.99

Model	Predictors	R2 (training)	Adj. R2 (training)	R2 (cross-validated)
Stepwise Regression, PCA data, significant variables	PC1, PC2, PC4, PC5, PC7, PC12, PC14	75.69	71.33	63.30

Generally, all models seem to provide a better result compared to previous models, suggesting that this approach is better for our data. The best model among the ones with stepwise regression appears to be the one with significant variables only, as it has lower errors and good R2 values after cross-validation. It is also simpler to explain with fewer variables. The use of PCA was not helpful for this data set, but perhaps it would have been benefitial on different data.

2. Lasso Regression

Step 1: Preparing the data

Moving onto the LASSO Regression, we know that we need scaled data, so we will be using our **scaled data** set from previous section. Since LASSO is a good method for data with multicollinearity (which we suspected during data exploration), it might give us a better model than stepwise regression.

For LASSO, we need matrices of the response and predictor variables:

```r
y <- data.matrix(scaled$Crime)
head(y)
```

```
##      [,1]
## [1,]  791
## [2,] 1635
## [3,]  578
## [4,] 1969
## [5,] 1234
## [6,]  682
```

```r
x <- data.matrix(scaled[,-16])
head(x)
```

```
##      So          M         Ed        Po1        Po2         LF         M.F
## [1,]  1  0.9886930 -1.3085099 -0.9085105 -0.8666988 -1.2667456 -1.12060499
## [2,]  0  0.3521372  0.6580587  0.6056737  0.5280852  0.5396568  0.98341752
## [3,]  1  0.2725678 -1.4872888 -1.3459415 -1.2958632 -0.6976051 -0.47582390
## [4,]  0 -0.2048491  1.3731746  2.1535064  2.1732150  0.3911854  0.37257228
## [5,]  0  0.1929983  1.3731746  0.8075649  0.7426673  0.7376187  0.06714965
## [6,]  0 -1.3983912  0.3898903  1.1104017  1.2433590 -0.3511718 -0.64550313
##              Pop           NW          U1         U2     Wealth       Ineq
## [1,] -0.09500679  1.943738564  0.69510600  0.8313680 -1.3616094  1.6793638
## [2,] -0.62033844  0.008483424  0.02950365  0.2393332  0.3276683  0.0000000
## [3,] -0.48900552  1.146296747 -0.08143007 -0.1158877 -2.1492481  1.4036474
## [4,]  3.16204944 -0.205464381  0.36230482  0.5945541  1.5298536 -0.6767585
## [5,] -0.48900552 -0.691709391 -0.24783066 -1.6551781  0.5453053 -0.5013026
## [6,] -0.30513945 -0.555560788 -0.63609870 -0.5895155  1.6956723 -1.7044289
##            Prob        Time
## [1,]  1.6497631 -0.05599367
## [2,] -0.7693365 -0.18315796
## [3,]  1.5969416 -0.32416470
## [4,] -1.3761895  0.46611085
## [5,] -0.2503580 -0.74759413
## [6,] -0.5669349 -0.78996812
```

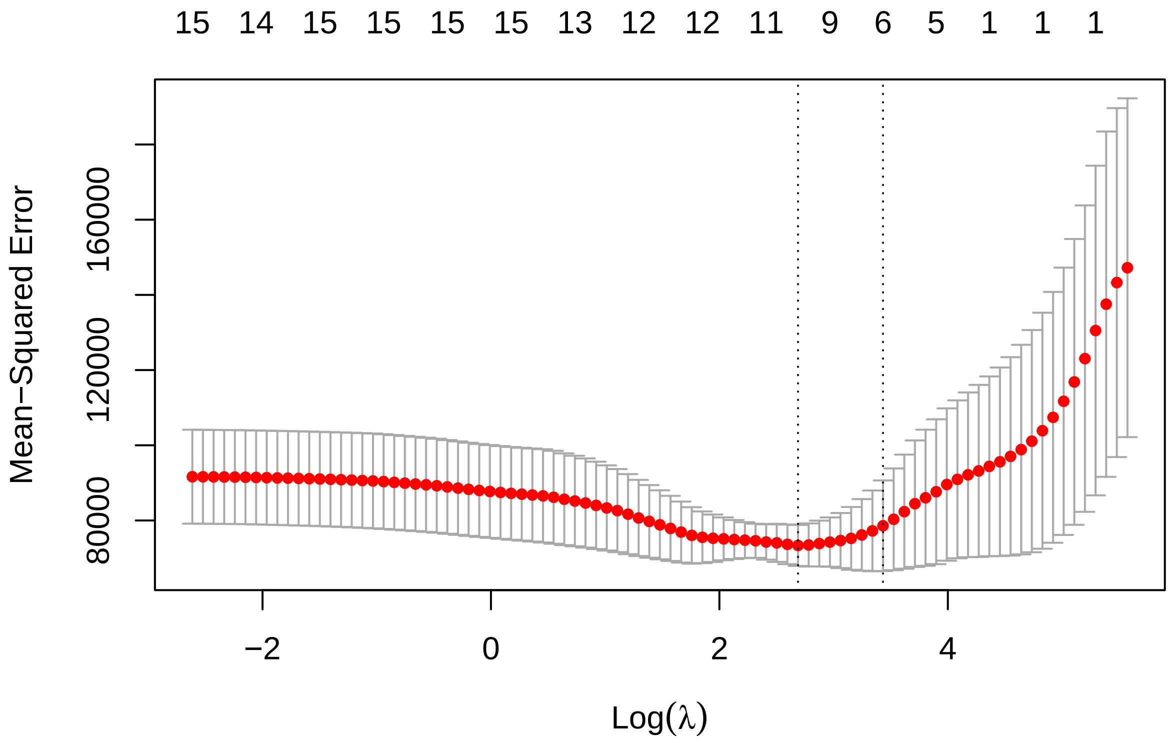
Step 2: Cross-validated LASSO

Next, we perform 5-fold cross-validation to find the **optimal lambda value that minimizes test MSE**:

```
set.seed(1)
lasso <- cv.glmnet(x, y, #matrices
                   alpha=1,
                   nfolds=5, #5-fold CV
                   type.measure='mse', #use MSE to find best lambda
                   family='gaussian') #in r refers to normal distribution

best.lambda <- lasso$lambda.min
best.lambda
```

```
## [1] 14.70916
```

```
plot(lasso)
```



As we can see, the best lambda found is 14.71. On the plot, the dashed lines are the log(lambda) values, and the left of those lines is the min lambda, for which the model has the lowest mean squared error after cross-validation. The right dashed line is the lambda with 1 standard error that is supposed to be less prone to overfitting. Top numbers are the number of the non-zero coefficients in our model (so min lambda corresponds to something between 9-11 predictors, and lambda with 1 standard error corresponds to 5-9), and we can see how **with increasing lambda from left to right on the plot the penalty for the inclusion of new features is increasing, and the number of variables in the model is becoming lower**.

Let's use the min lambda as the best one and see, which variables are selected with it by lasso method:

```
coefs_minlambda <- coef(lasso, s=best.lambda)
coefs_minlambda
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                     s1
## (Intercept) 889.648600
## So           45.344736
## M            77.635868
## Ed           98.182711
## Po1         311.196426
## Po2           .
## LF            2.887732
## M.F          46.955007
## Pop           .
## NW            2.654767
## U1            .
## U2           29.302470
## Wealth        .
## Ineq        164.140537
## Prob        -77.051224
## Time          .
```

As expected, we have 12 variables that lasso suggests. The other predictors have zero coefficients. With min lambda, Lasso suggests the following predictors: **So + M + Ed + Po1 + LF + M.F + NW + U2 + Ineq + Prob**.

Compared to stepwise regression, LASSO provided more predictors that should be used and we need to compare both cross-validated model to see if that would be good for our data, considering that we do not have so many data points, and the predictor-data set ration is not well balanced when 10 predictors are used.

During further analysis I will also keep in mind the **lambda+1st.error**, which, according to the plot, suggests to use less variables. This line on the plot shows the lambda that is supposed to reduce overfitting in the model. Let's check what lambda value this is and which coefficients it suggests to use:

```
lambda.1se <-lasso$lambda.1se
lambda.1se
```

```
## [1] 30.96138
```

```
coefs_1selambda <- coef(lasso, s=lambda.1se)
coefs_1selambda
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) 905.08510638
## So            .
## M            44.78226134
## Ed            0.07286805
## Po1         282.78158603
## Po2           .
## LF            .
## M.F          54.26395274
## Pop           .
## NW            .
## U1            .
## U2            .
## Wealth        .
## Ineq        82.41002096
## Prob       -51.91991843
## Time          .
```

So for lambda+1se we would use **M+Ed+Po1+M.F.+Ineq+Prob**. Let's try both lambdas in our analysis.

We could use glmnet to build another lasso model with the best lambda, but I want to see how significant each of those coefficients would be if we use them in a linear regression. I also want to be able to adjust this set of predictors from lasso in case not all variables are significant to create a better model. So, I will go back to the caret package to build a model with those variables and cross-validate it.

Step 3: Linear Regression using variables suggested by LASSO (min lambda)

Let's start by running a linear regression on all variables with non-zero coefficients from Lasso. I will also **cross-validate the model using leave-one-out CV** so that we get a more real R2 value:

```
set.seed(1)
#leave-one-out CV
data_ctrl <- trainControl(method = "LOOCV")

lasso1 <- train(Crime ~ So + M + Ed + Po1 + LF + M.F + NW + U2 + Ineq + Prob,
                     data=scaled,
                     trControl=data_ctrl,
                     method="lm")
summary(lasso1)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -410.55 -121.42    5.76  110.54  550.24
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  863.284     53.510  16.133  < 2e-16 ***
## So           122.792    129.896   0.945  0.35080
## M            113.614     49.962   2.274  0.02903 *
## Ed           188.755     64.750   2.915  0.00608 **
## Po1          333.470     49.353   6.757 6.86e-08 ***
## LF            25.162     49.588   0.507  0.61496
## M.F           31.513     43.179   0.730  0.47021
## NW            -2.883     59.595  -0.048  0.96169
## U2            72.881     41.795   1.744  0.08973 .
## Ineq         229.121     68.845   3.328  0.00203 **
## Prob         -99.145     40.243  -2.464  0.01867 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 206.6 on 36 degrees of freedom
## Multiple R-squared:  0.7767, Adjusted R-squared:  0.7147
## F-statistic: 12.53 on 10 and 36 DF,  p-value: 5.374e-09
```

```
lasso1$results
```

```
##   intercept    RMSE  Rsquared      MAE
## 1      TRUE 246.772 0.6010411 186.3704
```

The model on training set is **overfit**, as it has an **R2 of 77.7% and Adjusted R2 of 71.5**, and only an **R2 of 60.1% after cross-validation**.

Moreover, **not all variables are significant**. Those that are not have**p-values of 0.15, so we would not consider them for our model - they are very unlikely to contain important effects**.

Step 4: Linear Regression using only significant variables (min lambda)

The significant variables are: **M, Ed, Po1, U2, Ineq, Prob**, and those are **the same variables that compose our final model after stepwise regression -** we came to that after excluding insignificant ones. So **both LASSO with min lambda and stepwise regression lead us to the same final model**. There is no need to rebuild the same model again:

```
summary(m3)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -470.68  -78.41  -19.68  133.12  556.23
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      29.27  30.918  < 2e-16 ***
## M             131.98      41.85   3.154  0.00305 **
## Ed            219.79      50.07   4.390 8.07e-05 ***
## Po1           341.84      40.87   8.363 2.56e-10 ***
## U2             75.47      34.55   2.185  0.03483 *
## Ineq          269.91      55.60   4.855 1.88e-05 ***
## Prob          -86.44      34.74  -2.488  0.01711 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 200.7 on 40 degrees of freedom
## Multiple R-squared:  0.7659, Adjusted R-squared:  0.7307
## F-statistic: 21.81 on 6 and 40 DF,  p-value: 3.418e-11
```

```
m3$results
```

```
##   intercept     RMSE  Rsquared      MAE
## 1      TRUE 221.0758 0.6706867 165.5131
```

As we know, this model has **6 variables, R2=76.6% and adjusted R2=73.1% on training data, and R2 of 67.1% after cross-validation**, and we have already described the coefficients of this model in the past section.

Step 4: Linear Regression using variables suggested by LASSO (lambda+1se)

Having seen that the final model we come to after narrowing down significant coefficients from Lasso with min lambda, let's also check what happens if we go for the predictor set provided by lambda + 1 standard error value.

Why should we use this value?

Lambda1se can provide us with a simpler model, which has comparable accuracy with the best model. Min lambda uses more variable, hence the higher quality and smaller error of a model built on its suggested variables can be due to **overfit**. Lambda+1se, on the other hand, leads to a model that does not sacrifice too much quality, but provides a less overfit model with a simpler set of variables.

Using lambda1se would be logical for our set, since **we have only 47 data points, and the ideal number of parameters according to the 10:1 ratio is 4-5, not 15 as we have.** A min lambda can give us a better fit on the training set, but as we saw, after cross validation this quality fades once we reduce the overfit.

Let's see if using lambda1se would lead us to a significantly worse quality with all variables, and if we would end up with a better model after cross-validation. Use 6 variables from lambda1se to fit and cross-validate a model:

```r
set.seed(1)
#leave-one-out CV
data_ctrl <- trainControl(method = "LOOCV")

lasso2 <- train(Crime ~ M + Ed + Po1 + M.F + Ineq + Prob,
                     data=scaled,
                     trControl=data_ctrl,
                     method="lm")
summary(lasso2)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -484.68 -106.59   -2.44  135.89  505.69
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      30.24  29.934  < 2e-16 ***
## M              91.75      40.96   2.240   0.0307 *
## Ed            140.01      55.54   2.521   0.0158 *
## Po1           364.24      41.41   8.797 6.80e-11 ***
## M.F            50.71      36.16   1.402   0.1685
## Ineq          259.59      58.15   4.464 6.41e-05 ***
## Prob          -89.69      35.90  -2.498   0.0167 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 207.3 on 40 degrees of freedom
## Multiple R-squared:  0.7502, Adjusted R-squared:  0.7127
## F-statistic: 20.02 on 6 and 40 DF,  p-value: 1.2e-10
```

```
lasso2$results
```

```
##   intercept     RMSE  Rsquared      MAE
## 1      TRUE 230.0676 0.6432119 179.7269
```

Although we have a lower R2=75% and adjusted R2=71.3% on training data, after cross validation R2 is 64.3%, which is better compared to the cross-validate model using the variables suggested by min lambda. However, one of the variables, M.F, is insignificant and has a p-value of 0.168. Let's remove it to see if the model improves:

Step 5: Linear Regression on significant variables, LASSO (lambda+1se)

```
set.seed(1)
#leave-one-out CV
data_ctrl <- trainControl(method = "LOOCV")

lasso3 <- train(Crime ~ M + Ed + Po1 + Ineq + Prob,
                     data=scaled,
                     trControl=data_ctrl,
                     method="lm")
summary(lasso3)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -528.2  -74.0   -7.0  139.8  503.3
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      30.59  29.587  < 2e-16 ***
## M             100.15      41.00   2.443 0.018964 *
## Ed            179.16      48.58   3.688 0.000656 ***
## Po1           360.28      41.79   8.621 9.47e-11 ***
## Ineq          272.53      58.09   4.692 3.00e-05 ***
## Prob          -87.93      36.30  -2.422 0.019930 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 209.7 on 41 degrees of freedom
## Multiple R-squared:  0.7379, Adjusted R-squared:  0.706
## F-statistic: 23.09 on 5 and 41 DF,  p-value: 5.926e-11
```

```
lasso3$results
```

```
##   intercept     RMSE  Rsquared      MAE
## 1      TRUE 229.1873 0.6456266 172.1402
```

The model has improved! Although on training R2=73.8% and Adjusted R=70.1%, after cross-validation the R2 has increased to 64.6%, and all the coefficients are now significant. This is still not as high as R2=67% after cross-validation of the final model with 6 variables for stepwise and lasso with min lambda, but this model has **5 variables instead of 6, so it is even more easy to explain, and the predictor:data points ration is close to 1:10 now, which is perfect**. If we needed to simplify our previous final model, this is the one we would choose.

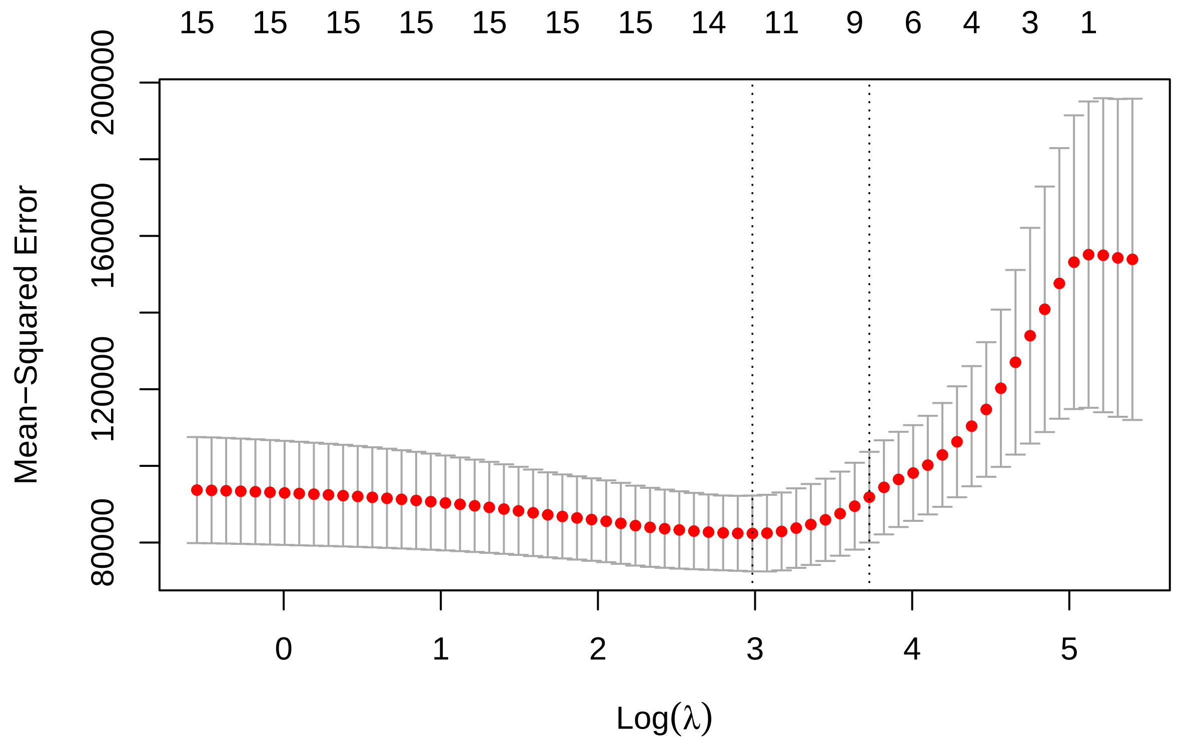
(Additional) Step 6: LASSO Regression on PCA

Let's repeat the same process on Principal components and see if this time the model would benefit from it. As the last time with stepwise, I will use all PCs to let the lasso method decide which variables we should use:

```
set.seed(1)
#matrices
y.pca <- data.matrix(pca_data$Crime)
x.pca <- data.matrix(pca_data[,-16])

#LASSO
lasso.pca <- cv.glmnet(x.pca, y.pca, #matrices
                       alpha=1,
                       nfolds=5, #5-fold CV
                       type.measure='mse', #use MSE to find best lambda
                       family='gaussian') #in r refers to normal distribution

best.lambda.pca <- lasso.pca$lambda.min
best.lambda.pca
```

```
## [1] 19.74659
```

```
plot(lasso.pca)
```



With LASSO run on PCA data, our min lambda is 19 and it suggests to use 12 predictors (left dashed line). And lambda1se suggests something between 8-9 predictors. Let's see coefficients suggested by both.

For min lambda the coefficients are:

```r
coef(lasso.pca, s=best.lambda.pca)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)  905.085106
## PC1           57.080104
## PC2          -58.158625
## PC3           11.097588
## PC4           50.931178
## PC5         -208.653084
## PC6          -33.376936
## PC7           82.070982
## PC8            .
## PC9            .
## PC10          11.672337
## PC11           .
## PC12         233.864614
## PC13          -5.987125
## PC14         136.639295
## PC15         328.368912
```

The best lambda that minimizes MSE and as many variables as possible to have that best fit, suggests to use 12 coefficients in the model : PC1 + PC2 + PC3 + PC4 + PC5 + PC6 + PC7 + PC10 + PC12 + PC13 + PC14 + PC15.

Check what lambda1se suggests:

```r
lambda.1se.pca <-lasso.pca$lambda.1se
lambda.1se.pca
```

```
## [1] 41.56468
```

```r
coef(lasso.pca, s=lambda.1se.pca)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)  905.085106
## PC1           48.090795
## PC2          -44.983203
## PC3            .
## PC4           30.474041
## PC5         -186.124379
## PC6           -3.725341
## PC7           43.195023
## PC8            .
## PC9            .
## PC10           .
## PC11           .
## PC12         172.267623
## PC13           .
## PC14          45.432225
## PC15           3.700378
```

Lambda 1se, which allows a bit more error to reduce overfit but get similar results to the best fit, suggests to use 9 parameters: PC1 + PC2 + PC4 + PC5 + PC6 + PC7 + PC12 + PC14 + PC15.

Let’s build models using both suggestsions. Start with min lambda set - 12 predictors:

```
set.seed(1)
#leave-one-out CV
data_ctrl <- trainControl(method = "LOOCV")

lassopca1 <- train(Crime ~ PC1 + PC2 + PC3 + PC4 + PC5 + PC6 + PC7 + PC10 + PC12 + PC13 + PC14 + PC15,
                     data=pca_data,
                     trControl=data_ctrl,
                     method="lm")
summary(lassopca1)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -404.62  -85.20   -6.35  111.78  526.89
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      29.48  30.698  < 2e-16 ***
## PC1            65.22      12.15   5.369 5.70e-06 ***
## PC2           -70.08      17.80  -3.936 0.000389 ***
## PC3            25.19      21.05   1.197 0.239582
## PC4            69.45      27.64   2.512 0.016913 *
## PC5          -229.04      30.44  -7.523 9.82e-09 ***
## PC6           -60.21      40.07  -1.503 0.142142
## PC7           117.26      52.53   2.232 0.032313 *
## PC10           56.32      66.66   0.845 0.404101
## PC12          289.61      83.24   3.479 0.001398 **
## PC13          -81.79     113.18  -0.723 0.474838
## PC14          219.19     123.25   1.778 0.084288 .
## PC15          622.21     438.74   1.418 0.165237
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 202.1 on 34 degrees of freedom
## Multiple R-squared:  0.7981, Adjusted R-squared:  0.7269
## F-statistic:  11.2 on 12 and 34 DF,  p-value: 1.408e-08
```

```
lassopca1$results
```

```
##   intercept    RMSE  Rsquared      MAE
## 1      TRUE 246.256 0.6025509 188.5251
```

Just as with initial data, **we get a high R2=79.8% and Adj R2=72.7%, and a significantly lower R2= 60.3% after cross-validation.** Also, **not all variables are significant** - so let's redo the model and leave only those that are important:

```
set.seed(1)
#leave-one-out CV
data_ctrl <- trainControl(method = "LOOCV")

lassopca2 <- train(Crime ~ PC1 + PC2 + PC4 + PC5 + PC7 + PC12 + PC14,
                     data=pca_data,
                     trControl=data_ctrl,
                     method="lm")
summary(lassopca2)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -389.89 -112.34  -20.12  134.00  465.46
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      30.21  29.962  < 2e-16 ***
## PC1            65.22      12.45   5.240 5.86e-06 ***
## PC2           -70.08      18.24  -3.842 0.000438 ***
## PC4            69.45      28.32   2.452 0.018791 *
## PC5          -229.04      31.19  -7.343 7.28e-09 ***
## PC7           117.26      53.82   2.178 0.035477 *
## PC12          289.61      85.28   3.396 0.001585 **
## PC14          219.19     126.28   1.736 0.090503 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 207.1 on 39 degrees of freedom
## Multiple R-squared:  0.7569, Adjusted R-squared:  0.7133
## F-statistic: 17.35 on 7 and 39 DF,  p-value: 3.412e-10
```

```
lassopca2$results
```

```
##   intercept     RMSE  Rsquared      MAE
## 1      TRUE 233.6217 0.6330384 190.5296
```

With significant variables only, the result is better: **R2=75.7% and Asj R2=71.3% on the training set, and R2=63.3% after cross-validation**. But still not as good as the final model on initial data with R2 of 67%.

Let's see if we would end up with a different model using the lambda 1se set of parameters. Use the 9 variables

```
set.seed(1)
#leave-one-out CV
data_ctrl <- trainControl(method = "LOOCV")

lassopca3 <- train(Crime ~ PC1 + PC2 + PC4 + PC5 + PC6 + PC7 + PC12 + PC14 + PC15,
                     data=pca_data,
                     trControl=data_ctrl,
                     method="lm")
summary(lassopca3)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -351.54  -93.58  -20.42  121.74  553.91
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      29.35  30.836  < 2e-16 ***
## PC1            65.22      12.09   5.393 4.17e-06 ***
## PC2           -70.08      17.72  -3.954 0.000334 ***
## PC4            69.45      27.52   2.523 0.016048 *
## PC5          -229.04      30.31  -7.557 5.20e-09 ***
## PC6           -60.21      39.89  -1.509 0.139665
## PC7           117.26      52.30   2.242 0.031040 *
## PC12          289.61      82.87   3.495 0.001249 **
## PC14          219.19     122.70   1.786 0.082235 .
## PC15          622.21     436.77   1.425 0.162660
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 201.2 on 37 degrees of freedom
## Multiple R-squared:  0.7823, Adjusted R-squared:  0.7293
## F-statistic: 14.77 on 9 and 37 DF,  p-value: 8.755e-10
```

```
lassopca3$results
```

```
##   intercept    RMSE Rsquared      MAE
## 1      TRUE 232.367 0.639978 175.6689
```

Better than the first model on pca with min lambda predictors, but still 2 varaiables are insignificant. **On training data we have R2=78.2% and Adj R2=72.9, and R2=63.99 after cross-validation**.

Notice that the same variables are significant as in the previous model - so we would end up with the **same final model** on pca using lasso after leaving significant coefficients only.

Step 7: LASSO: Summary of Results

As we have seen, LASSO is also relatively simple to implement, and it also gives some freedom with the opportunity to opt for different lambdas and therefore different sets of variables.

On our data, LASSO led to initial models with more variables compared to stepwise regression, but we ended up with the same final model and set of predictors as stepwise regression.

PCA, once again, did not help to get a better model - perhaps for our small data set Lasso method works better on the initial variables. However, with Lasso we got the same set of final model parameters (PCs), as with stepwise regression - so both approaches confirm that those are the best solutions. Different lambda options from lasso on PCA data also led us to the same final models.

Model	**Predictors**	**R2 (training)**	**Adj. R2 (training)**	**R2 (cross-validated)**
Lasso Regression, original data, all variables (min lambda)	So, M, Ed, Po1, LF, M.F, NW, U2, Ineq, Prob	77.67	71.47	60.1

Model	Predictors	R2 (training)	Adj. R2 (training)	R2 (cross-validated)
Lasso Regression, original data, significant variables (min lambda)	M, Ed, Po1, U2, Ineq, Prob	76.59	73.07	67.07
Lasso Regression, original data, all variables (lambda+1se)	M, Ed, Po1, M.F, Ineq, Prob	75.02	71.27	64.32
Lasso Regression, original data, significant variables (lambda+1se)	M, Ed, Po1, Ineq, Prob	73.79	70.5	64.56
Lasso Regression, PCA data, all variables (min lambda)	PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC10, PC12, PC13, PC14, PC15	79.81	72.69	60.25
Lasso Regression, PCA data, significant variables (min lambda)	PC1, PC2, PC4, PC5, PC7, PC12, PC14	75.69	71.33	63.30
Lasso Regression, PCA data, all variables (lambda+1se)	PC1, PC2, PC4, PC5, PC6, PC7, PC12, PC14, PC15	78.23	72.93	63.99
Lasso Regression, PCA data, significant variables (lambda+1se)	PC1, PC2, PC4, PC5, PC7, PC12, PC14	75.69	71.33	63.30

3. Elastic Net

Step 1: Choosing the value of alpha

To perform elastic net, we need to input an alpha value. Let's test several alphas and see which R2 we get for each cross-validated elastic net.

We use alpha values from 0 to 1 with a 0.05 step and for each do elastic net with 5-fold cross validation. Then, we compare R2 for each alpha: within each alpha value we have different values of lambda used in elastic net that lead to different amount of deviance explained. Just as with Lasso, we can choose the **min lambda, which gives the best fit and lowest error**, and get the explained deviance share from each alpha model's fit - since it is a regression model, this is the same as R2.

```
set.seed(1)
r2list<-c()
alphalist <- seq(0,1,by=0.05)
for (i in alphalist){
  elasticnetcv = cv.glmnet(x, y, alpha=i,nfolds = 5,type.measure="mse",family="gaussian")

  r2list = cbind(r2list,
               elasticnetcv$glmnet.fit$dev.ratio[which(elasticnetcv$glmnet.fit$lambda == elasticnetcv$l

#merge results in a data frame
alpha_r2 <- data.table(alphalist, t(r2list))
colnames(alpha_r2) <- c('alpha', 'R2')
alpha_r2
```

```
##     alpha        R2
##  1:  0.00 0.7493364
##  2:  0.05 0.7525162
##  3:  0.10 0.7490443
##  4:  0.15 0.7392534
##  5:  0.20 0.7632796
##  6:  0.25 0.7858073
##  7:  0.30 0.7956576
##  8:  0.35 0.7728516
##  9:  0.40 0.7648523
## 10:  0.45 0.7932145
## 11:  0.50 0.7552074
## 12:  0.55 0.7944774
## 13:  0.60 0.7769852
## 14:  0.65 0.7714496
## 15:  0.70 0.7819289
## 16:  0.75 0.7737543
## 17:  0.80 0.7552396
## 18:  0.85 0.7749567
## 19:  0.90 0.7754385
## 20:  0.95 0.7845227
## 21:  1.00 0.7113245
##     alpha        R2
```

Now that we have R2 values for min lambda for each alpha, let's find the highest R2 to figure out which alpha we should use:

```
alpha_r2[which.max(alpha_r2$R2)]
```

```
##    alpha        R2
## 1:   0.3 0.7956576
```

Looks like the best R2 is achieved with min lambda for alpha=0.3.

Step 2: Selecting variables with Elastic Net

Now that we know the best alpha, let's get the coefficients of this model: we rebuild the elastic net model using alpha=0.3 and get the coefficients:

```
set.seed(1)
#elastic net
elasticnetcv <- cv.glmnet(x, y, alpha=0.3, nfolds = 5,type.measure="mse",family="gaussian")
#get coefs
coef(elasticnetcv, s=elasticnetcv$lambda.min)
```

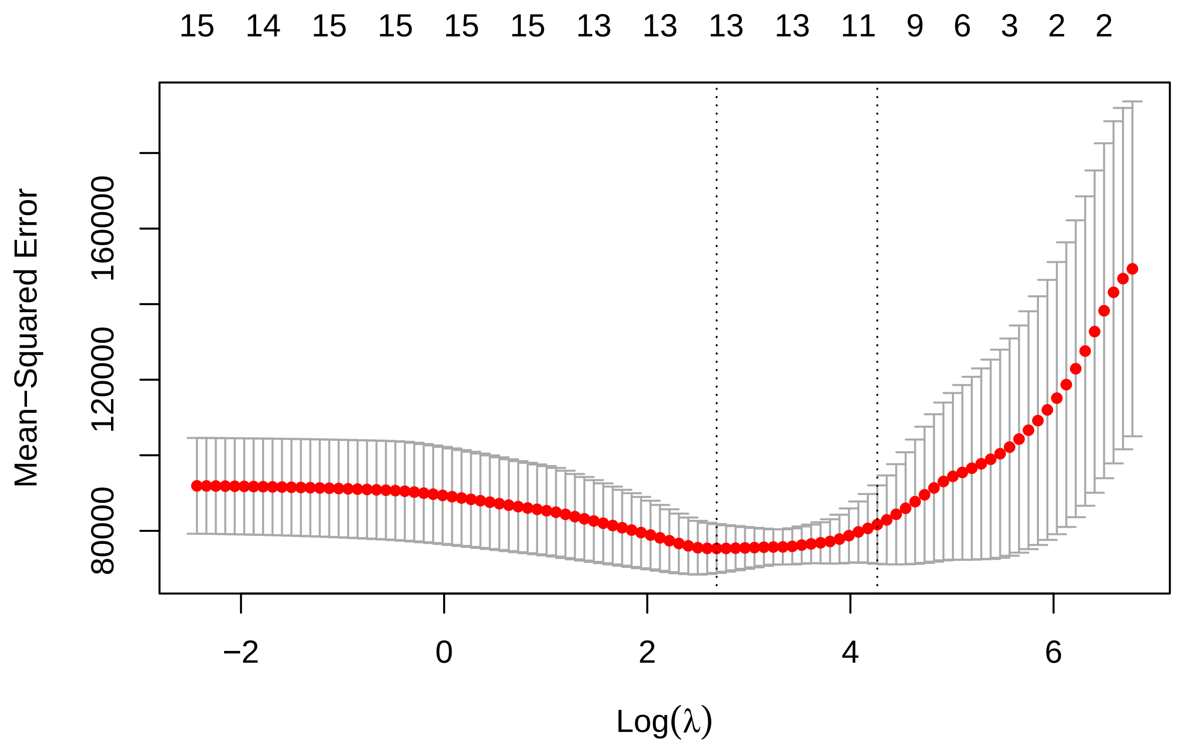
```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept) 886.204378
## So           55.462140
## M            91.803278
## Ed          139.495392
## Po1         207.461699
## Po2          73.321836
## LF            5.738319
## M.F          64.182378
## Pop          -1.727255
## NW           19.394608
## U1          -56.956645
## U2           94.842331
## Wealth       34.071995
## Ineq        193.465584
## Prob        -88.395645
## Time          .
```

```
coef(elasticnetcv, s=elasticnetcv$lambda.1se)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept) 893.405223
## So           34.309659
## M            44.626951
## Ed           28.574486
## Po1         147.867584
## Po2         104.519201
## LF            9.921127
## M.F          51.869728
## Pop           .
## NW           23.177000
## U1            .
## U2           14.188031
## Wealth        .
## Ineq         68.413953
## Prob        -65.285478
## Time          .
```

As we can see, with Elastic net and min lambda we get 14 variables selected out of 15, and with lambda+1se we get 11.

```
plot(elasticnetcv)
```



As we can see, elastic net suggests that the error would be very high for fewer than 11 variables. The left vertical line shows that best fit with lowest error (and more overfitting) is ar 14 variables, and the right line shows that we can go down to 11 variables to keep a similar quality of fit in order to reduce overfitting.

Let's see how the 2 models would perform on training data and after cross-validation.

Step 3: Linear regression on all variables from Elastic net (min lambda)

Let's start with the min lambda which is supposed to provide the best fit. Let's see the R2 after cross-validation and if many variables are significant (since 14 seems like to many for 47 data points):

```
set.seed(1)
#leave-one-out CV
data_ctrl <- trainControl(method = "LOOCV")

net1 <- train(Crime ~ .-Time,
                     data=scaled,
                     trControl=data_ctrl,
                     method="lm")
summary(net1)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -385.20  -98.21    6.29  108.37  488.17
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  903.146     58.031  15.563  < 2e-16 ***
## So             5.695    145.692   0.039  0.96906
## M            106.360     51.144   2.080  0.04566 *
## Ed           211.846     68.583   3.089  0.00413 **
## Po1          526.791    297.035   1.773  0.08566 .
## Po2         -249.764    303.726  -0.822  0.41697
## LF           -24.617     58.508  -0.421  0.67675
## M.F           56.367     58.351   0.966  0.34129
## Pop          -33.627     47.088  -0.714  0.48032
## NW            33.675     62.903   0.535  0.59611
## U1          -100.068     74.305  -1.347  0.18753
## U2           138.191     68.326   2.023  0.05155 .
## Wealth        87.243     98.179   0.889  0.38085
## Ineq         282.919     89.520   3.160  0.00343 **
## Prob         -96.225     42.125  -2.284  0.02913 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 206.6 on 32 degrees of freedom
## Multiple R-squared:  0.8016, Adjusted R-squared:  0.7148
## F-statistic: 9.234 on 14 and 32 DF,  p-value: 1.249e-07
```

```
net1$results
```

```
##   intercept     RMSE Rsquared      MAE
## 1      TRUE 260.5987 0.565233 200.9782
```

This is not surprising: **R2 on training is 80.16%, Adjusted R2=71.5%, but after cross-validation R2 is only 56.5 - worst we've seen so far**. Moreover, only 6 predictors are significant - **the same set of variables as in our final models after stepwise and lasso regressions**! So we end up with the same model which has the best R2 of 67% after cross-validation:

```
summary(m3)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -470.68  -78.41  -19.68  133.12  556.23
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      29.27  30.918  < 2e-16 ***
## M             131.98      41.85   3.154  0.00305 **
## Ed            219.79      50.07   4.390 8.07e-05 ***
## Po1           341.84      40.87   8.363 2.56e-10 ***
## U2             75.47      34.55   2.185  0.03483 *
## Ineq          269.91      55.60   4.855 1.88e-05 ***
## Prob          -86.44      34.74  -2.488  0.01711 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 200.7 on 40 degrees of freedom
## Multiple R-squared:  0.7659, Adjusted R-squared:  0.7307
## F-statistic: 21.81 on 6 and 40 DF,  p-value: 3.418e-11
```

```
m3$results
```

```
##   intercept     RMSE  Rsquared      MAE
## 1      TRUE 221.0758 0.6706867 165.5131
```

Let's not describe this model another time and move on to linear regression on variables that we got with lambda+1se.

Step 4: Linear regression on all variables from Elastic net (lambda+1se)

Let's check how bad or good the results are after cross-validation of the model with 11 variables which we got using elastic net with alpha 0.3 and lambda+1se

```
set.seed(1)
#leave-one-out CV
data_ctrl <- trainControl(method = "LOOCV")

net2 <- train(Crime ~ .- Time - Pop - U1 - Wealth ,
                     data=scaled,
                     trControl=data_ctrl,
                     method="lm")
summary(net2)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -387.31 -101.53   -3.73   99.06  569.06
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  863.009     53.928  16.003  < 2e-16 ***
## So           123.599    130.914   0.944  0.35158
## M            109.839     50.669   2.168  0.03706 *
## Ed           198.221     66.778   2.968  0.00537 **
## Po1          527.689    295.241   1.787  0.08255 .
## Po2         -203.547    304.998  -0.667  0.50891
## LF            15.464     52.045   0.297  0.76813
## M.F           31.758     43.517   0.730  0.47037
## NW             5.298     61.297   0.086  0.93161
## U2            67.681     42.835   1.580  0.12309
## Ineq         222.193     70.154   3.167  0.00319 **
## Prob        -101.677     40.734  -2.496  0.01742 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 208.2 on 35 degrees of freedom
## Multiple R-squared:  0.7796, Adjusted R-squared:  0.7103
## F-statistic: 11.25 on 11 and 35 DF,  p-value: 1.613e-08
```

```
net2$results
```

```
##   intercept     RMSE  Rsquared      MAE
## 1      TRUE 251.9427 0.5875694 188.2989
```

The results are not good for this model either: **R2=77.96% , Adj R2=71.03% on training set, but R2=58.76% is achieved after cross-validation. 5 variables are significant - the same ones we had in the final model for lasso with lambda+1se** - so we once again have come to the same 2 final models for our data, confirmed by all the methods - it's a great sign that those two models are really the best on initial data.

(Additional) Step 5: Elastic Net on PCA

Let's repeat the process for PCA data to see if we get something better:

```
set.seed(1)
r2list<-c()
alphalist <- seq(0,1,by=0.05)
for (i in alphalist){

  elasticnetcvpca = cv.glmnet(x.pca, y.pca, alpha=i,nfolds = 5,type.measure="mse",family="gaussian")

  r2list = cbind(r2list,
                 elasticnetcvpca$glmnet.fit$dev.ratio[which(elasticnetcvpca$glmnet.fit$lambda == elastic

#merge results in a data frame
alpha_r2_pca <- data.table(alphalist, t(r2list))
colnames(alpha_r2_pca) <- c('alpha', 'R2')
alpha_r2_pca
```

```
##     alpha        R2
##  1:  0.00 0.7818306
##  2:  0.05 0.7889210
##  3:  0.10 0.7654027
##  4:  0.15 0.7795872
##  5:  0.20 0.7558492
##  6:  0.25 0.7932600
##  7:  0.30 0.7860755
##  8:  0.35 0.7531235
##  9:  0.40 0.7833446
## 10:  0.45 0.7941640
## 11:  0.50 0.7737693
## 12:  0.55 0.8030496
## 13:  0.60 0.7761970
## 14:  0.65 0.7903800
## 15:  0.70 0.7853068
## 16:  0.75 0.8030487
## 17:  0.80 0.7790476
## 18:  0.85 0.8028076
## 19:  0.90 0.6910333
## 20:  0.95 0.7656279
## 21:  1.00 0.6370799
##     alpha        R2
```

Find best alpha:

```
alpha_r2_pca[which.max(alpha_r2_pca$R2)]
```

```
##    alpha        R2
## 1:  0.55 0.8030496
```

This run the best alpha is 0.55. Let's get the coefficients for both lambdas:

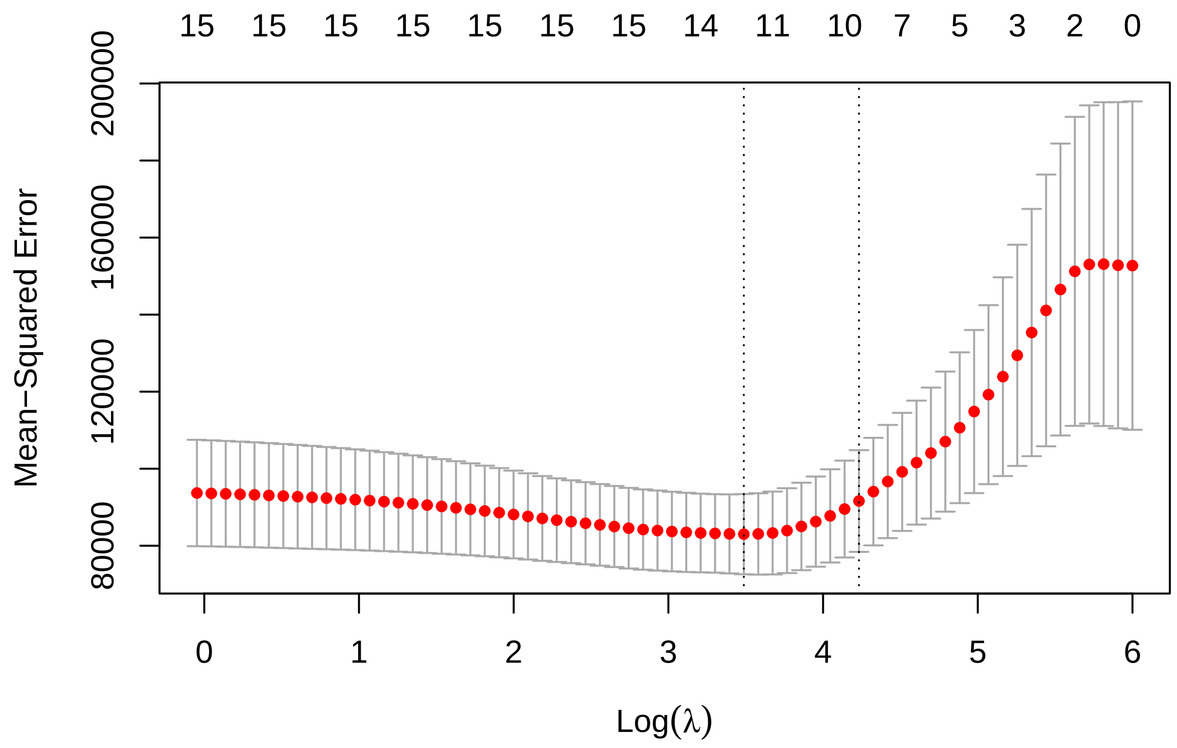
```
set.seed(1)
#elastic net
elasticnetcvpca <- cv.glmnet(x.pca, y.pca, alpha=0.55, nfolds = 5,type.measure="mse",family="gaussian")
#get coefs
coef(elasticnetcvpca, s=elasticnetcvpca$lambda.min)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept)  905.08511
## PC1           55.66137
## PC2          -57.02404
## PC3           11.89234
## PC4           50.62814
## PC5         -202.66711
## PC6          -34.43612
## PC7           82.04032
## PC8            .
## PC9            .
## PC10          15.05913
## PC11           .
## PC12         229.96941
## PC13         -12.24963
## PC14         138.63865
## PC15         341.34051
```

```
coef(elasticnetcvpca, s=elasticnetcvpca$lambda.1se)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)  905.085106
## PC1           45.895383
## PC2          -43.675983
## PC3            .
## PC4           31.393828
## PC5         -175.707736
## PC6           -8.088537
## PC7           46.045471
## PC8            .
## PC9            .
## PC10           .
## PC11           .
## PC12         169.005651
## PC13           .
## PC14          56.308085
## PC15          54.253614
```

```
plot(elasticnetcvpca)
```



Once again, many variables are suggested by elastic net. Min lambda recommends to use 12 variables, and lambda+1se - 9.

With min lambda, we get the same set of variables as with PCA in Lasso with min lambda:

```
summary(lassopca1)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -404.62  -85.20   -6.35  111.78  526.89
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      29.48  30.698  < 2e-16 ***
## PC1            65.22      12.15   5.369 5.70e-06 ***
## PC2           -70.08      17.80  -3.936 0.000389 ***
## PC3            25.19      21.05   1.197 0.239582
## PC4            69.45      27.64   2.512 0.016913 *
## PC5          -229.04      30.44  -7.523 9.82e-09 ***
## PC6           -60.21      40.07  -1.503 0.142142
## PC7           117.26      52.53   2.232 0.032313 *
## PC10           56.32      66.66   0.845 0.404101
## PC12          289.61      83.24   3.479 0.001398 **
## PC13          -81.79     113.18  -0.723 0.474838
## PC14          219.19     123.25   1.778 0.084288 .
## PC15          622.21     438.74   1.418 0.165237
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 202.1 on 34 degrees of freedom
## Multiple R-squared:  0.7981, Adjusted R-squared:  0.7269
## F-statistic:  11.2 on 12 and 34 DF,  p-value: 1.408e-08
```

```
lassopca1$results
```

```
##   intercept    RMSE  Rsquared      MAE
## 1      TRUE 246.256 0.6025509 188.5251
```

And this result narrows down to the same model with all 7 significant PCs:

```
summary(lassopca2)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -389.89 -112.34  -20.12  134.00  465.46
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      30.21  29.962  < 2e-16 ***
## PC1            65.22      12.45   5.240 5.86e-06 ***
## PC2           -70.08      18.24  -3.842 0.000438 ***
## PC4            69.45      28.32   2.452 0.018791 *
## PC5          -229.04      31.19  -7.343 7.28e-09 ***
## PC7           117.26      53.82   2.178 0.035477 *
## PC12          289.61      85.28   3.396 0.001585 **
## PC14          219.19     126.28   1.736 0.090503 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 207.1 on 39 degrees of freedom
## Multiple R-squared:  0.7569, Adjusted R-squared:  0.7133
## F-statistic: 17.35 on 7 and 39 DF,  p-value: 3.412e-10
```

```
lassopca2$results
```

```
##   intercept     RMSE  Rsquared      MAE
## 1      TRUE 233.6217 0.6330384 190.5296
```

The results, as we remember, still do not exceed the R2 after cross-validation that we had for the final model on the initial data.

The same goes for lambda+1se - it is the exact same set of PC components as in lasso's PCA model with lambda+1se:

```
summary(lassopca3)
```

```
##
```

```
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -351.54  -93.58  -20.42  121.74  553.91
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      29.35  30.836  < 2e-16 ***
## PC1            65.22      12.09   5.393 4.17e-06 ***
## PC2           -70.08      17.72  -3.954 0.000334 ***
## PC4            69.45      27.52   2.523 0.016048 *
## PC5          -229.04      30.31  -7.557 5.20e-09 ***
## PC6           -60.21      39.89  -1.509 0.139665
## PC7           117.26      52.30   2.242 0.031040 *
## PC12          289.61      82.87   3.495 0.001249 **
## PC14          219.19     122.70   1.786 0.082235 .
## PC15          622.21     436.77   1.425 0.162660
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 201.2 on 37 degrees of freedom
## Multiple R-squared:  0.7823, Adjusted R-squared:  0.7293
## F-statistic: 14.77 on 9 and 37 DF,  p-value: 8.755e-10
```

```
lassopca3$results
```

```
##   intercept    RMSE Rsquared      MAE
## 1      TRUE 232.367 0.639978 175.6689
```

And with all significant coefficients, we get the same final model as with min lambda (the second model above called lassopca2).

Step 6: Elastic Net: Summary of Results

As we have seen, with Elastic Net we don't get any new final models - with initial data it comes down to the best model we've seen so far on this set - the one with R2 of 67% after CV. With PCA, all the models we get with Elastic Net are exactly the same as the ones we got with Lasso.

Elastic Net was the most complex to implement

LASSO is also relatively simple to implement, and it also gives some freedom with the opportunity to opt for different lambdas and therefore different sets of variables.

On our data, LASSO led to initial models with more variables compared to stepwise regression, but we ended up with the same final model and set of predictors as stepwise regression.

PCA, once again, did not help to get a better model - perhaps for our small data set Lasso method works better on the initial variables. However, with Lasso we got the same set of final model parameters (PCs), as with stepwise regression - so both approaches confirm that those are the best solutions. Different lambda options from lasso on PCA data also led us to the same final models.

Model	**Predictors**	**R2 (training)**	**Adj. R2 (training)**	**R2 (cross-validated)**
Elastic Net, original data, all variables (alpha=0.3, min lambda)	So, M, Ed, Po1, Po2, LF, M.F, Pop, NW, U1, U2, Wealth, Ineq, Prob	80.16	71.48	56.52
Elastic Net, original data, significant variables (alpha=0.3, min lambda)	So, M, Ed, Po1,Po2, LF, M.F, NW, U2, Ineq, Prob	77.96	71.03	58.76
Elastic Net, original data, all variables (alpha=0.3, lambda+1se)	M, Ed, Po1, M.F, Ineq, Prob	75.02	71.27	64.32
Elastic Net, original data, significant variables (alpha=0.3, lambda+1se)	M, Ed, Po1, U2, Ineq, Prob	76.59	73.07	67.07
Elastic Net, PCA data, all variables (alpha=0.55, min lambda)	PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC10, PC12, PC13, PC14, PC15	79.81	72.69	60.25
Elastic Net, PCA data, significant variables (alpha=0.55, min lambda)	PC1, PC2, PC4, PC5, PC7, PC12, PC14	75.69	71.33	63.30
Elastic Net, PCA data, all variables (alpha=0.55, lambda+1se)	PC1, PC2, PC4, PC5, PC6, PC7, PC12, PC14, PC15	78.23	72.93	63.99
Elastic Net, PCA data, significant variables (alpha=0.55, lambda+1se)	PC1, PC2, PC4, PC5, PC7, PC12, PC14	75.69	71.33	63.30

Final Thoughts

As we have seen through our analysis, all three methods are different from each other, and tend to suggest different variables for use, but after removal of insignificant variables from linear regression we are always left with the same set of variables and the same model for both, initial data and PCA data.

PCA did not give any benefit to any of the methods - although we were able to achieve good models with all significant variables, the R-squared after cross-validation was still smaller compared to the final model based on initial data. Of course, this does not suggest that PCA is useless in such methods - it just was not the best approach for our data set in particular.

As for the methods we have tried, the stepwise regression seems like a good initial technique that provides a quick solution and insight into the data patterns. Elastic Net was the most complex one to apply, however it had more tuning parameters that we could experiment with, which is great for complex data sets.

Overall, for our data the stepwise approach appeared to be enough - the other models used more variables, but did not perform as well after cross-validation. However, it is clear that it is important to apply different methods to make sure that there is no better model - and for different data sets, different approaches could be found best. Perhaps sometimes it is also beneficial to try more parameters than we end up with in the final model - we can always remove the insignificant ones.

Here is the final table of all the models we have built - although not all of them performed as great as the model with 6 predictors and R2 of 67% after cross-validation, the results in general are much better than what we have seen for this data set in the previous projects, meaning that variable selection techniques can be very helpful:

Model	**Predictors**	**R2 (training)**	**Adj. R2 (training)**	**R2 (cross-validated)**
Stepwise Regression, original data, all variables	M, Ed, Po1, M.F., U1, U2, Ineq, Prob	78.88	74.44	67.38
Stepwise Regression, original data, significant variables	M, Ed, Po1, U2, Ineq, Prob	76.59	73.07	67.07
Stepwise Regression, PCA data, all variables	PC1, PC2, PC4, PC5, PC6, PC7, PC12, PC14, PC15	78.23	72.93	63.99
Stepwise Regression, PCA data, significant variables	PC1, PC2, PC4, PC5, PC7, PC12, PC14	75.69	71.33	63.30
Lasso Regression, original data, all variables (min lambda)	So, M, Ed, Po1, LF, M.F, NW, U2, Ineq, Prob	77.67	71.47	60.1
Lasso Regression, original data, significant variables (min lambda)	M, Ed, Po1, U2, Ineq, Prob	76.59	73.07	67.07
Lasso Regression, original data, all variables (lambda+1se)	M, Ed, Po1, M.F, Ineq, Prob	75.02	71.27	64.32
Lasso Regression, original data, significant variables (lambda+1se)	M, Ed, Po1, Ineq, Prob	73.79	70.5	64.56
Lasso Regression, PCA data, all variables (min lambda)	PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC10, PC12, PC13, PC14, PC15	79.81	72.69	60.25

Model	**Predictors**	**R2 (training)**	**Adj. R2 (training)**	**R2 (cross-validated)**
Lasso Regression, PCA data, significant variables (min lambda)	PC1, PC2, PC4, PC5, PC7, PC12, PC14	75.69	71.33	63.30
Lasso Regression, PCA data, all variables (lambda+1se)	PC1, PC2, PC4, PC5, PC6, PC7, PC12, PC14, PC15	78.23	72.93	63.99
Lasso Regression, PCA data, significant variables (lambda+1se)	PC1, PC2, PC4, PC5, PC7, PC12, PC14	75.69	71.33	63.30
Elastic Net, original data, all variables (alpha=0.3, min lambda)	So, M, Ed, Po1, Po2, LF, M.F, Pop, NW, U1, U2, Wealth, Ineq, Prob	80.16	71.48	56.52
Elastic Net, original data, significant variables (alpha=0.3, min lambda)	So, M, Ed, Po1,Po2, LF, M.F, NW, U2, Ineq, Prob	77.96	71.03	58.76
Elastic Net, original data, all variables (alpha=0.3, lambda+1se)	M, Ed, Po1, M.F, Ineq, Prob	75.02	71.27	64.32
Elastic Net, original data, significant variables (alpha=0.3, lambda+1se)	M, Ed, Po1, U2, Ineq, Prob	76.59	73.07	67.07
Elastic Net, PCA data, all variables (alpha=0.55, min lambda)	PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC10, PC12, PC13, PC14, PC15	79.81	72.69	60.25
Elastic Net, PCA data, significant variables (alpha=0.55, min lambda)	PC1, PC2, PC4, PC5, PC7, PC12, PC14	75.69	71.33	63.30
Elastic Net, PCA data, all variables (alpha=0.55, lambda+1se)	PC1, PC2, PC4, PC5, PC6, PC7, PC12, PC14, PC15	78.23	72.93	63.99

Elastic Net, PCA data, significant variables (alpha=0.55, lambda+1se)	PC1, PC2, PC4, PC5, PC7, PC12, PC14	75.69	71.33	63.30