Choose Your Own Project

Osman Yardimci

2023-03-17

Introduction:

The relationship between crime rates and punishment regimes has long been of interest to criminologists. In this project, we will be using a data set that contains information on various factors, such as unemployment rates, education levels, and police expenditures, that may influence crime rates. The data set includes information on 47 states in the USA for the year 1960.

The project is divided into four parts. In the first part, we will use the R package 'outliers' to test for any outliers in the crime rate data. In the second part, we will use regression analysis to predict the crime rate in a hypothetical city based on the given data. We will also assess the quality of fit of the model.

In the third part, we will apply principal component analysis (PCA) to the data set and then create a regression model using the first few principal components. We will compare the quality of this model with the model created in part two. Finally, in the fourth part, we will build regression models using stepwise regression, lasso, and elastic net methods, and compare their results.

Overall, this project aims to explore the relationship between various factors and crime rates and to test different methods of regression analysis to build models that can predict crime rates.

Importing the required packages.

```r
library(outliers)
library(ggplot2)
library(DAAG)
library(boot)
library(GGally)
library(pls)
library(caret)
library(glmnet)
```

Uploading data

```r
# read the table from the web link
url <- "http://www.statsci.org/data/general/uscrime.txt"
data <- read.table(url, header = TRUE, sep = "\t")
```

Part 1

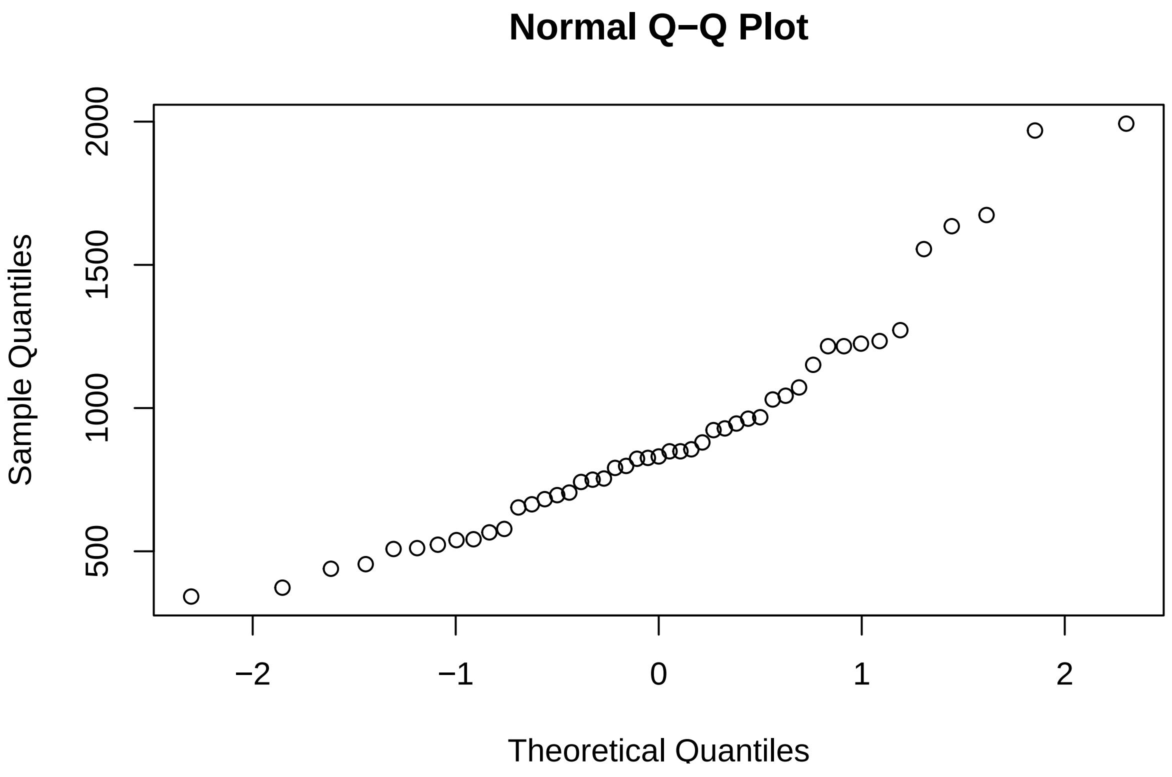
This section will involve the implementation of the 'outliers' R package to examine the presence of any outliers in the crime rate dataset.

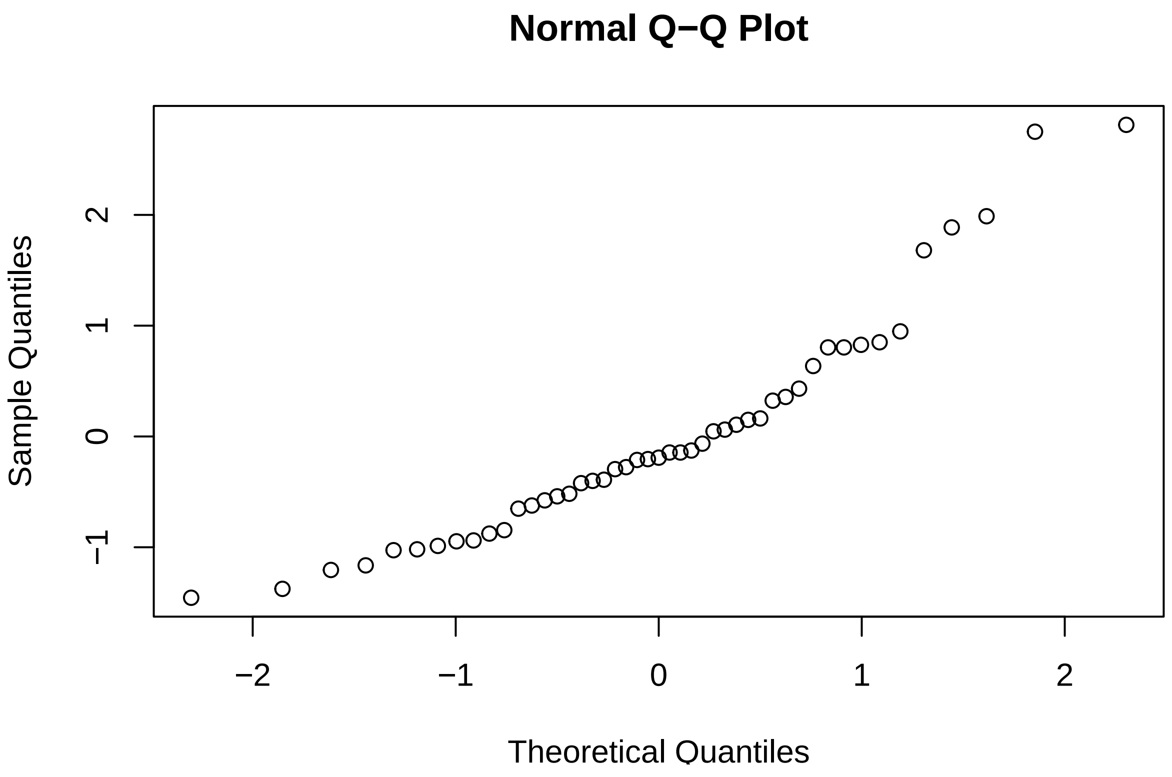
```r
# Crime is the variable of interest
crime <- data[,"Crime"]

# Run the Shapiro-Wilk test to test the normality of the crime data
```

```
shapiro.test(crime)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  crime
## W = 0.91273, p-value = 0.001882
```

Initially, the Grubbs test is based on the assumption of normality. Hence, we perform a normality test, the Shapiro-Wilk test, which is commonly known from elementary statistics. However, the test indicates non-normality of the data (p=0.001882). Despite this, upon inspecting the Q-Q plot, we observe that the non-normality is primarily due to the tails, which suggests that the test may be influenced by possible outliers. As the central part of the distribution appears to be normal, we proceed with the Grubbs test.





Upon examining the Q-Q plot, it is evident that the central portion of the data adheres to normal distribution. Therefore, we can make an assumption that the data is approximately normally distributed and proceed with conducting the Grubbs' test.

```
# Run the Grubbs' test for two outliers on opposite tails

test <- grubbs.test(crime, type = 11)

# Print results of grubbs test

test
```

```
##
##  Grubbs test for two opposite outliers
##
## data:  crime
## G = 4.26877, U = 0.78103, p-value = 1
## alternative hypothesis: 342 and 1993 are outliers
```

The obtained p-value implies the rejection of the null hypothesis that the data follows a normal distribution. However, it is essential to note that normality tests have a tendency to overlook the bigger picture and may provide inaccurate outcomes by concentrating excessively on specific data points. This is particularly true in the presence of outliers, which is precisely what we aim to detect.

It is worth noting that the decision to proceed with the Grubbs test is subjective. On one hand, it is possible that the Shapiro-Wilk test is detecting non-normal tails, particularly on the upper end, such that the extreme values are not outliers but are rather part of the distribution. On the other hand, it is possible that the distribution is reasonably close to normal, and the test's failure is due to the presence of outliers. The reliability of the Grubbs test relies on which of these situations is more accurate.

In this scenario, we will proceed with the Grubbs test. At the worst, the test will either reveal no outliers, or it will highlight potential outliers that we would investigate more closely. We would thoroughly investigate these data points to determine whether they are genuine outliers or whether they are a genuine part of the distribution.

Now, let us examine each one separately.

```r
test <- grubbs.test(crime, type = 10)
test
```

```
##
##  Grubbs test for one outlier
##
## data:  crime
## G = 2.81287, U = 0.82426, p-value = 0.07887
## alternative hypothesis: highest value 1993 is an outlier
```

The statistical analysis indicates that it is improbable for the city with the lowest crime rate to be an outlier. The p-value is so close to 1 that it is rounded up to 1.

The classification of the data point as an outlier depends on the selected threshold p-value. Different individuals may choose to use different threshold values such as p=0.05 or p=0.10. For this analysis, we will designate the data point as an outlier. Moving forward, we will now examine the second highest point to determine whether it is also an outlier.

To proceed with the analysis, we will create a new dataset that excludes the largest value.

```r
crime2 <- crime[-which.max(crime)]

# Now test it

test <- grubbs.test(crime2, type = 10)
test
```

```
##
##  Grubbs test for one outlier
##
## data:  crime2
## G = 3.06343, U = 0.78682, p-value = 0.02848
## alternative hypothesis: highest value 1969 is an outlier
```

The p-value for the second-highest crime city in the initial dataset is notably low, indicating that it is also an outlier. Therefore, we will remove this city from the analysis and proceed to examine the next city.

```r
# So, let's remove it, and test the next one.

crime3 <- crime2[-which.max(crime2)]
test <- grubbs.test(crime3, type = 10)
test
```

```
##
##  Grubbs test for one outlier
##
## data:  crime3
## G = 2.56457, U = 0.84712, p-value = 0.1781
## alternative hypothesis: highest value 1674 is an outlier
```

The p-value for the next city is sufficiently high, which makes it unclear whether or not it should be considered an outlier. Therefore, we will stop here and remove the two highest points as outliers. However, it is also necessary to check the lowest point in the data set. The grubbs.test function selected the most outlying data points, which were both high points. To identify the most outlying low point, we will use the opposite=TRUE parameter.

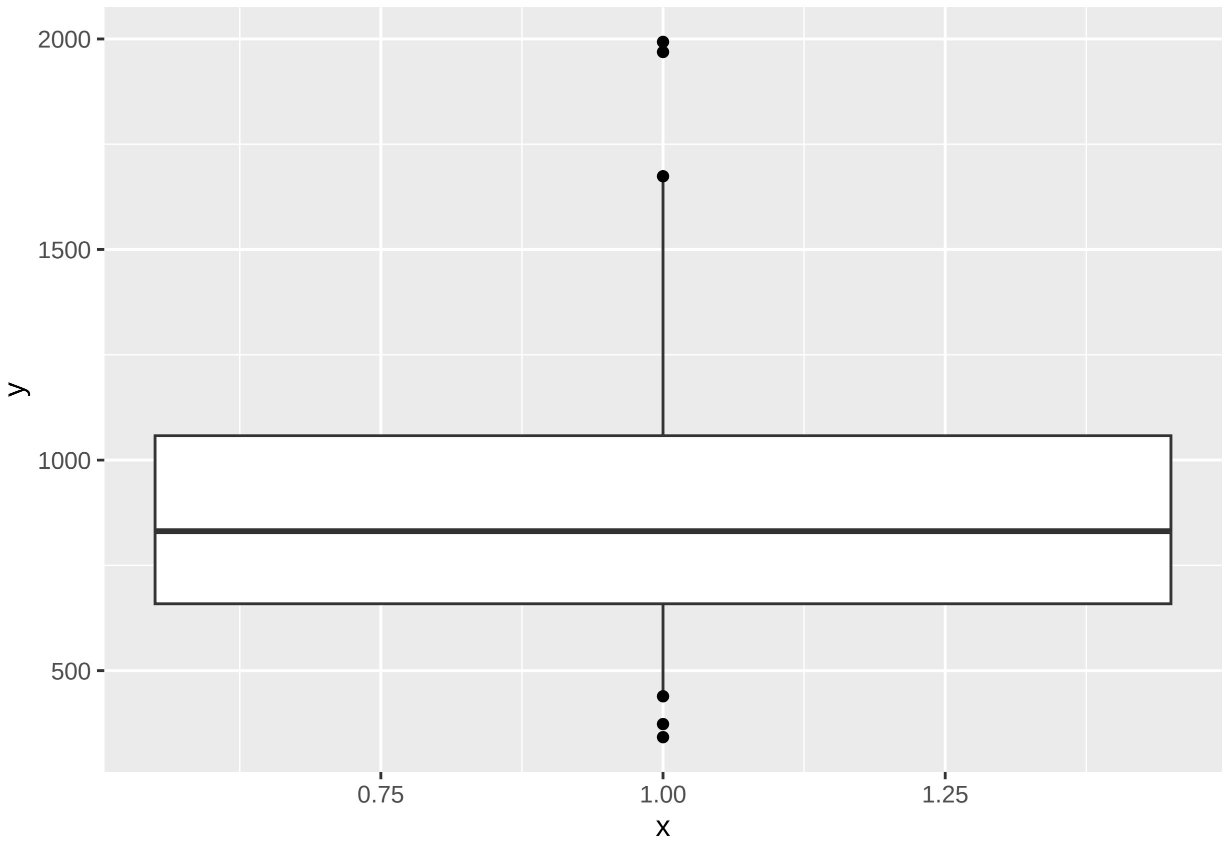
```r
test <- grubbs.test(crime3,type=10,opposite=TRUE)
test
```

```
##
##  Grubbs test for one outlier
##
## data:  crime3
## G = 1.61796, U = 0.93915, p-value = 1
## alternative hypothesis: lowest value 342 is an outlier
```

The p-value obtained after testing the lowest-crime city for being an outlier is very high and rounds to 1. Hence, it seems that this city is not an outlier. This is consistent with the result obtained from the previous test, which checked both the extreme values and also returned a p-value of 1. It is worth noting that removing the two outliers before testing the lowest value did not affect the outcome.

Conversely, the city with the highest crime rate may be considered an outlier as the p-value is 0.079. Furthermore, upon removal of this city, the second-highest crime city is also suspected to be an outlier, with a p-value of 0.028. These outliers are illustrated more explicitly in the box-and-whisker plot presented below.

```
## Warning: The `fun.y` argument of `stat_summary()` is deprecated as of ggplot2 3.3.0.
## i Please use the `fun` argument instead.
```



Results of Part 1

From the visualization, it appears that the two cities with the highest crime rates are likely outliers, while the two cities with the lowest crime rates are not too far from the norm and may not be considered outliers.

Part 2

In this part of our analysis involves utilizing regression analysis to forecast the crime rate in an imaginary city using the provided data. We will also evaluate the model's level of accuracy.

Initially, the code reads in the data and fits a linear regression model using all 15 factors. The predicted crime rate for a new data point based on this model is approximately 155, which is problematic because the lowest crime rate in the dataset is 342. This occurs because the model includes factors that are not significant, as evidenced by their high p-values. Although removing factors with high p-values is not always a good idea, the code proceeds to do so for this particular case, considering only factors with initial p-values of 0.10 or lower. This results in a reduced number of factors and all remaining factors have p-values less than 0.05. With this revised model, the predicted crime rate for the new data point is a more reasonable 1304.

The dependent variable in this analysis is Crime, while the other variables are independent predictors. The model is built using the lm() function with the entire dataset, and is used for prediction without choosing between different models or estimating model quality. Therefore, there is no need for validation or test data.

```r
model <- lm( Crime ~ ., data = data)

#Summary of the model

summary(model)
```

```
##
## Call:
## lm(formula = Crime ~ ., data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -395.74  -98.09   -6.69  112.99  512.67
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.984e+03  1.628e+03  -3.675 0.000893 ***
## M            8.783e+01  4.171e+01   2.106 0.043443 *
## So          -3.803e+00  1.488e+02  -0.026 0.979765
## Ed           1.883e+02  6.209e+01   3.033 0.004861 **
## Po1          1.928e+02  1.061e+02   1.817 0.078892 .
## Po2         -1.094e+02  1.175e+02  -0.931 0.358830
## LF          -6.638e+02  1.470e+03  -0.452 0.654654
## M.F          1.741e+01  2.035e+01   0.855 0.398995
## Pop         -7.330e-01  1.290e+00  -0.568 0.573845
## NW           4.204e+00  6.481e+00   0.649 0.521279
## U1          -5.827e+03  4.210e+03  -1.384 0.176238
## U2           1.678e+02  8.234e+01   2.038 0.050161 .
## Wealth       9.617e-02  1.037e-01   0.928 0.360754
## Ineq         7.067e+01  2.272e+01   3.111 0.003983 **
## Prob        -4.855e+03  2.272e+03  -2.137 0.040627 *
## Time        -3.479e+00  7.165e+00  -0.486 0.630708
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 209.1 on 31 degrees of freedom
## Multiple R-squared:  0.8031, Adjusted R-squared:  0.7078
## F-statistic: 8.429 on 15 and 31 DF,  p-value: 3.539e-07
```

Create a new data point for testing purposes using a dataframe.

```
test <-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)


#Predict the crime rate for test data point

pred_model <- predict(model, test)
pred_model
```

```
##        1
## 155.4349
```

The result is surprising! The predicted crime rate is significantly lower than the next lowest city's crime rate. Even though the test data point's factor values are all within the range of other data points, the prediction is not accurate. I intentionally chose this data point to demonstrate that the full model used earlier includes several insignificant factors. It's natural to wonder why we can't use the entire model, even if some factors are insignificant. This is a perfect example of why that's not a good idea. We need to go back and use only the significant factors to get a reliable estimate. Let's try including only the factors with p-values less than or equal to 0.1. (In Part 4, we'll explore better ways to do this.)

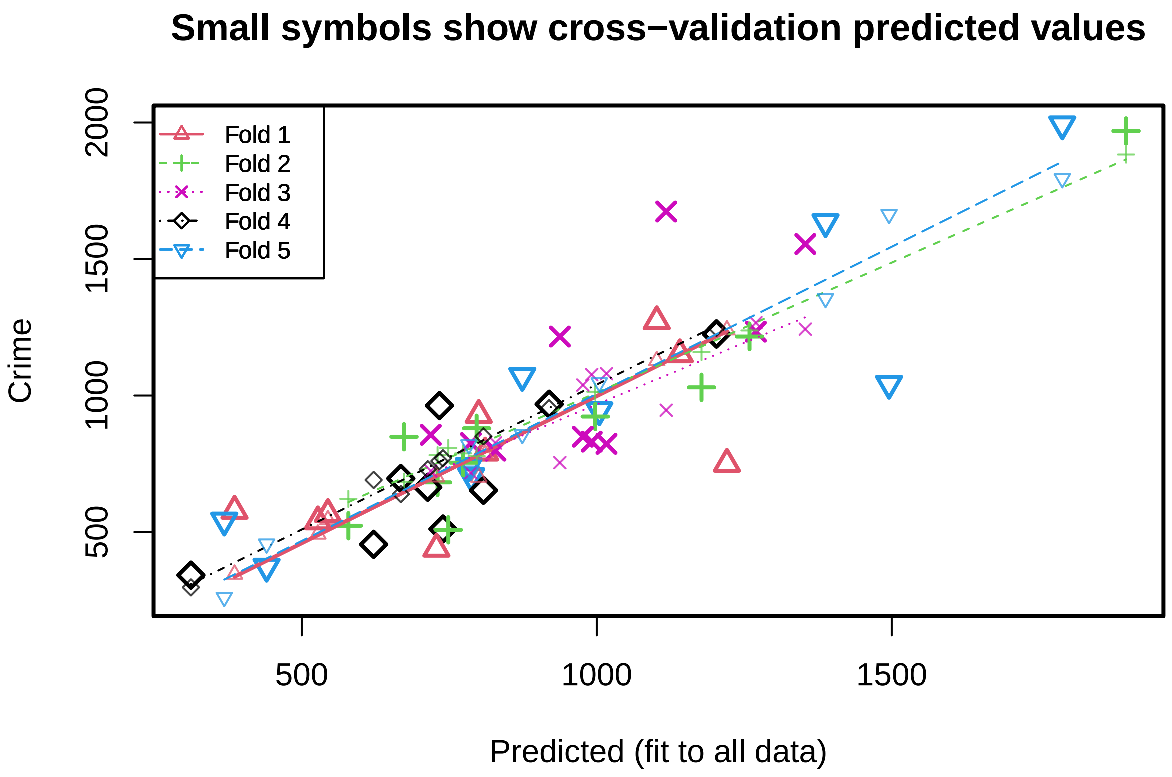
```
model2 <- lm( Crime ~  M + Ed + Po1 + U2 + Ineq + Prob, data = data)

#Summary of the model

summary(model2)
```

```
##
## Call:
## lm(formula = Crime ~ M + Ed + Po1 + U2 + Ineq + Prob, data = data)
##
## 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) -5040.50     899.84  -5.602 1.72e-06 ***
## M             105.02      33.30   3.154  0.00305 **
## Ed            196.47      44.75   4.390 8.07e-05 ***
## Po1           115.02      13.75   8.363 2.56e-10 ***
## U2             89.37      40.91   2.185  0.03483 *
## Ineq           67.65      13.94   4.855 1.88e-05 ***
## Prob        -3801.84    1528.10  -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
```

Lets predict on our test observation

```
# do 5-fold cross-validation
```

```r
c <- cv.lm(data,model2,m=5)
```



```
##
## fold 1
## Observations in test set: 9
##                       1        3       17       18        19        22        36
## Predicted    810.825487 386.1368 527.3659 800.0046 1220.6767  728.3110 1101.7167
## cvpred       785.364736 345.3417 492.2016 700.5751 1240.2916  701.5126 1127.3318
## Crime        791.000000 578.0000 539.0000 929.0000  750.0000  439.0000 1272.0000
## CV residual    5.635264 232.6583  46.7984 228.4249 -490.2916 -262.5126  144.6682
##                     38         40
## Predicted    544.37325 1140.79061
## cvpred       544.69903 1168.21107
## Crime        566.00000 1151.00000
## CV residual   21.30097  -17.21107
##
## Sum of squares = 439507.2    Mean square = 48834.14    n = 9
##
## fold 2
## Observations in test set: 10
##                       4         6       12        25         28        32
## Predicted    1897.18657 730.26589 673.3766 579.06379 1259.00338 773.68402
## cvpred       1882.73805 781.75573 684.3525 621.37453 1238.31917 788.03429
## Crime        1969.00000 682.00000 849.0000 523.00000 1216.00000 754.00000
## CV residual    86.26195 -99.75573 164.6475 -98.37453  -22.31917 -34.03429
##                      34       41        44        46
## Predicted     997.54981 796.4198 1177.5973  748.4256
## cvpred       1013.92532 778.0437 1159.3155  807.6968
## Crime         923.00000 880.0000 1030.0000  508.0000
## CV residual  -90.92532 101.9563 -129.3155 -299.6968
##
## Sum of squares = 181038.4    Mean square = 18103.83    n = 10
##
## fold 3
## Observations in test set: 10
##                     5         8        9        11        15        23
## Predicted   1269.84196 1353.5532 718.7568 1117.7702 828.34178  937.5703
## cvpred      1266.79544 1243.1763 723.5331  946.1309 826.28548  754.2511
## Crime       1234.00000 1555.0000 856.0000 1674.0000 798.00000 1216.0000
## CV residual  -32.79544  311.8237 132.4669  727.8691 -28.28548  461.7489
##                    37       39        43        47
## Predicted    991.5623 786.6949 1016.5503  976.4397
## cvpred      1076.5799 717.0989 1079.7748 1038.3321
## Crime        831.0000 826.0000  823.0000  849.0000
## CV residual -245.5799 108.9011 -256.7748 -189.3321
##
## Sum of squares = 1033612    Mean square = 103361.1    n = 10
##
## fold 4
## Observations in test set: 9
##                    7        13        14        20        24        27
## Predicted   733.3799  739.3727 713.56395 1202.9607 919.39117 312.20470
## cvpred      759.9655  770.2015 730.05546 1247.8616 953.72478 297.19321
## Crime       963.0000  511.0000 664.00000 1225.0000 968.00000 342.00000
## CV residual 203.0345 -259.2015 -66.05546  -22.8616  14.27522  44.80679
##                    30        35        45
## Predicted   668.01610  808.0296  621.8592
## cvpred      638.87118  850.6961  690.6802
## Crime       696.00000  653.0000  455.0000
## CV residual  57.12882 -197.6961 -235.6802
##
## Sum of squares = 213398.5    Mean square = 23710.94    n = 9
##
## fold 5
## Observations in test set: 9
##                     2        10        16        21        26        29
## Predicted   1387.8082 787.27124 1004.3984 783.27334 1789.1406 1495.4856
## cvpred      1355.7097 723.66781 1046.8197 819.71145 1794.6456 1663.6272
## Crime       1635.0000 705.00000  946.0000 742.00000 1993.0000 1043.0000
## CV residual  279.2903 -18.66781 -100.8197 -77.71145  198.3544 -620.6272
##                   31        33       42
## Predicted   440.4394  873.8469 368.7031
## cvpred      456.5736  857.7052 260.9211
## Crime       373.0000 1072.0000 542.0000
## CV residual -83.5736  214.2948 281.0789
##
## Sum of squares = 650990    Mean square = 72332.23    n = 9
##
## Overall (Sum over all 9 folds)
##       ms
## 53586.08
```

```
# note that here, "m" is used for the number of folds, rather than the usual "k"
#c
```

The cross-validation technique shows the average squared prediction error denoted as "ms." However, it is important to note that there is an error in the cv.lm function, which incorrectly states that "n" represents the sum of all folds when in reality, "n" indicates the number of data points in the last fold. To calculate the R-squared values, we can directly use the formula:

$$\text{R-squared} = 1 - \frac{\text{SSEresiduals}}{\text{SSEtotal}}$$

```
# total sum of squared differences between data and its mean
SStot <- sum((data$Crime - mean(data$Crime))^2)
# for model, model2, and cross-validation, calculated SEres
SSres_model <- sum(model$residuals^2)
SSres_model2 <- sum(model2$residuals^2)
SSres_c <- attr(c,"ms")*nrow(data)
# mean squared error, times number of data points, gives sum of squared errors
# Calculate R-squareds for model, model2, cross-validation
1 - SSres_model/SStot # initial model with insignificant factors
```

```
## [1] 0.8030868
```

```
1 - SSres_model2/SStot # model2 without insignificant factors
```

```
## [1] 0.7658663
```

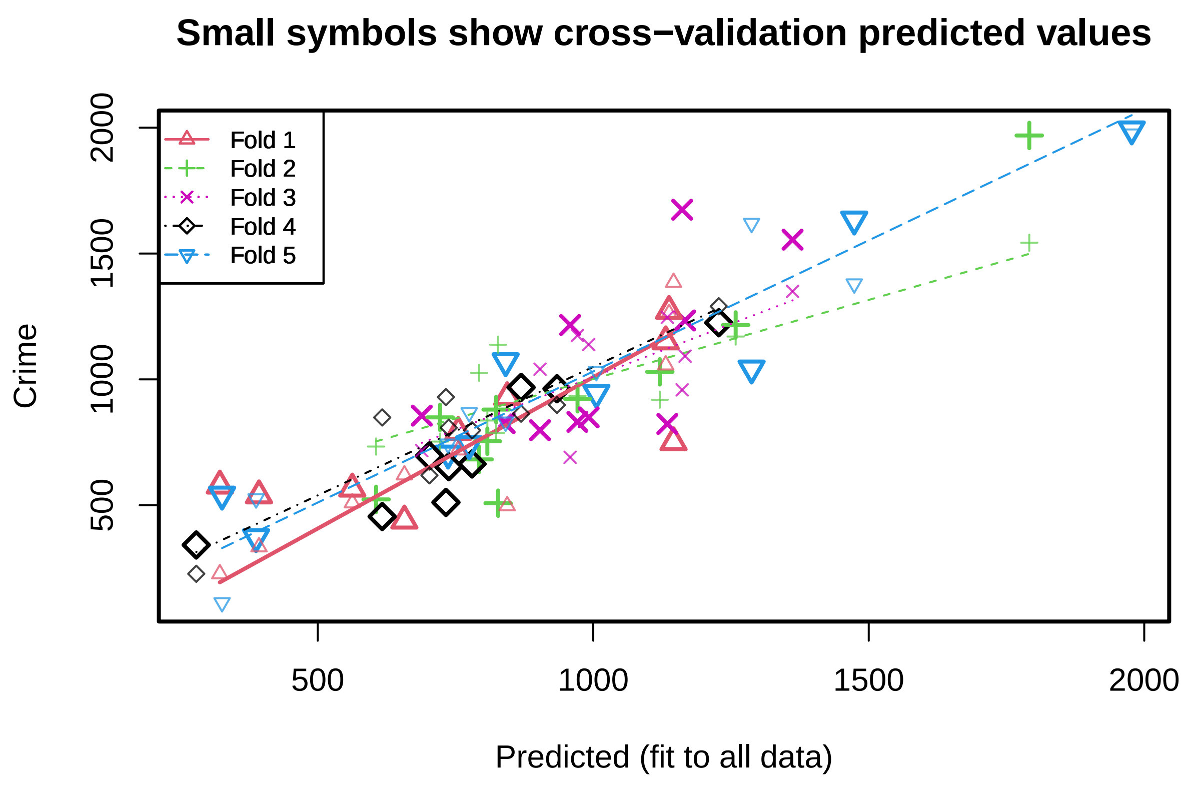
```
1 - SSres_c/SStot # cross-validated
```

```
## [1] 0.6339817
```

The inclusion of insignificant factors leads to overfitting compared to excluding them, and it's possible that the fitted model is also overfitted. This outcome is not unexpected since we had a limited number of data points (only 47) to predict 15 different factors. The ratio of data points to factors was roughly 3:1, which is typically inadequate, as a ratio of 10:1 or greater is considered ideal. In Part 4, we will explore possible solutions to this issue.

We could also attempt to perform cross-validation using the initial 15-factor model.

```
cfirst <- cv.lm(data,model,m=5)
```



```
##
## fold 1
## Observations in test set: 9
##                     1         3        17        18        19         22          36
## Predicted   755.03222 322.2615 393.3633 843.8072 1145.7379  657.2092 1137.61711
## cvpred      719.48189 227.3811 334.2928 497.4904 1384.9349  620.1834 1261.61602
## Crime       791.00000 578.0000 539.0000 929.0000  750.0000  439.0000 1272.00000
## CV residual  71.51811 350.6189 204.7072 431.5096 -634.9349 -181.1834   10.38398
##                    38         40
## Predicted   562.6934 1131.45326
## cvpred      509.0826 1057.08701
## Crime       566.0000 1151.00000
## CV residual  56.9174   93.91299
##
## Sum of squares = 804290.7    Mean square = 89365.64    n = 9
##
## fold 2
## Observations in test set: 10
##                     4         6        12        25         28        32
## Predicted   1791.3619  792.9301 722.04080  605.8824 1258.48423 807.81667
## cvpred      1542.8663 1025.6864 752.84607  733.1797 1170.10415 836.60938
## Crime       1969.0000  682.0000 849.00000  523.0000 1216.00000 754.00000
## CV residual  426.1337 -343.6864  96.15393 -210.1797   45.89585 -82.60938
##                    34        41        44        46
## Predicted   971.45581 823.74192 1120.8227  827.3543
## cvpred      934.62797 786.74042  919.1066 1137.6778
## Crime       923.00000 880.00000 1030.0000  508.0000
## CV residual -11.62797  93.25958  110.8934 -629.6778
##
```

```
## Sum of squares = 779686.2    Mean square = 77968.62    n = 10
##
## fold 3
## Observations in test set: 10
##                     5         8        9        11        15        23
## Predicted   1166.6840 1361.7468 688.8682 1161.3291  903.3541  957.9918
## cvpred      1092.1924 1349.7715 717.0401  958.3058 1040.2775  690.2073
## Crime       1234.0000 1555.0000 856.0000 1674.0000  798.0000 1216.0000
## CV residual  141.8076  205.2285 138.9599  715.6942 -242.2775  525.7927
##                    37       39        43        47
## Predicted    971.1513 839.2864 1134.4172  991.7629
## cvpred      1174.2195 838.1895 1246.7022 1138.2873
## Crime        831.0000 826.0000  823.0000  849.0000
## CV residual -343.2195 -12.1895 -423.7022 -289.2873
##
## Sum of squares = 1310071    Mean square = 131007.1    n = 10
##
## fold 4
## Observations in test set: 9
##                     7        13        14         20       24       27
## Predicted   934.16366  732.6412  780.0401 1227.83873 868.9805 279.4772
## cvpred      898.53488  929.2776  797.4106 1290.40739 863.7702 227.4408
## Crime       963.00000  511.0000  664.0000 1225.00000 968.0000 342.0000
## CV residual  64.46512 -418.2776 -133.4106  -65.40739 104.2298 114.5592
##                    30        35        45
## Predicted   702.69454  737.7888  616.8983
## cvpred      618.72406  808.0845  848.6350
## Crime       696.00000  653.0000  455.0000
## CV residual  77.27594 -155.0845 -393.6350
##
## Sum of squares = 410147.4    Mean square = 45571.93    n = 9
##
## fold 5
## Observations in test set: 9
##                     2        10         16        21         26        29
## Predicted   1473.6764 736.50802 1005.65694  774.8506 1977.37067 1287.3917
## cvpred      1379.5108 743.27567 1031.35676  867.6315 1975.12567 1619.8299
## Crime       1635.0000 705.00000  946.00000  742.0000 1993.00000 1043.0000
## CV residual  255.4892 -38.27567  -85.35676 -125.6315   17.87433 -576.8299
##                    31        33       42
## Predicted    388.0334  840.9992 326.3324
## cvpred       525.4791  830.6871 112.9800
## Crime        373.0000 1072.0000 542.0000
## CV residual -152.4791  241.3129 429.0200
##
## Sum of squares = 688401.1    Mean square = 76489.01    n = 9
##
## Overall (Sum over all 9 folds)
##       ms
## 84948.87
```

```
SSres_cfirst <- attr(cfirst,"ms")*nrow(data)
# mean squared error, times number of data points, gives sum of squared errors
1 - SSres_cfirst/SStot # cross-validated
```

```
## [1] 0.419759
```

The significant deviation from the lm() reported R-squared value of 0.803 on the training dataset highlights the importance of validation.

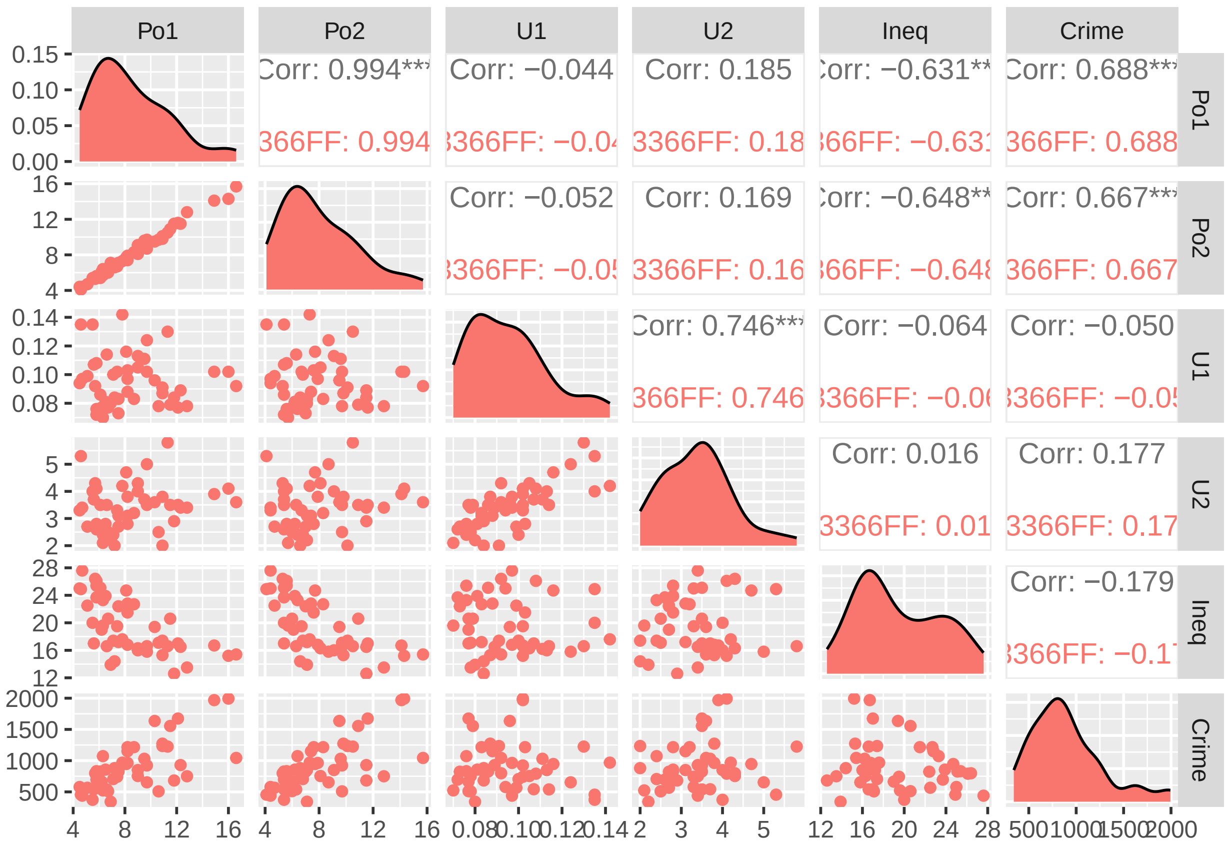
Results of Part 2

Now let's look at quality of the model. The first model's R2 on training data was 0.803, and the second's was 0.766 – including the factors with high p-values leads to some overfitting. But measuring on training data isn't a good estimate, also because of the possibility of overfitting. We can use cross-validation to estimate the quality of this model. In the R-code, I used 5-fold cross-validation, and found an R2 of about 0.638. This is lower than the performance on the training data, indicating that there was still some overfitting going on. And that's not surprising. We have just 47 data points, and 15 factors, a ratio of about 3:1, and it's usually good to have a ratio of 10:1 or more. Even the smaller model's ratio was below 10:1. (We'll see in Module 11 ways we can try to get around this problem.)

But note that the 15-factor model (including the factors with high p-values) was much worse – its cross-validation R2 was just 0.413 (compared to its reported performance on training data of 0.803). That's almost a factor of 2 difference; it suggests a lot of overfitting, and demonstrates why we can't just use a fitted model's reported R2 without doing some kind of validation!

Part 3

In this part, we will utilize principal component analysis (PCA) on the dataset and use the first few principal components to develop a regression model. We will then evaluate the performance of this model against the one we created in part two.



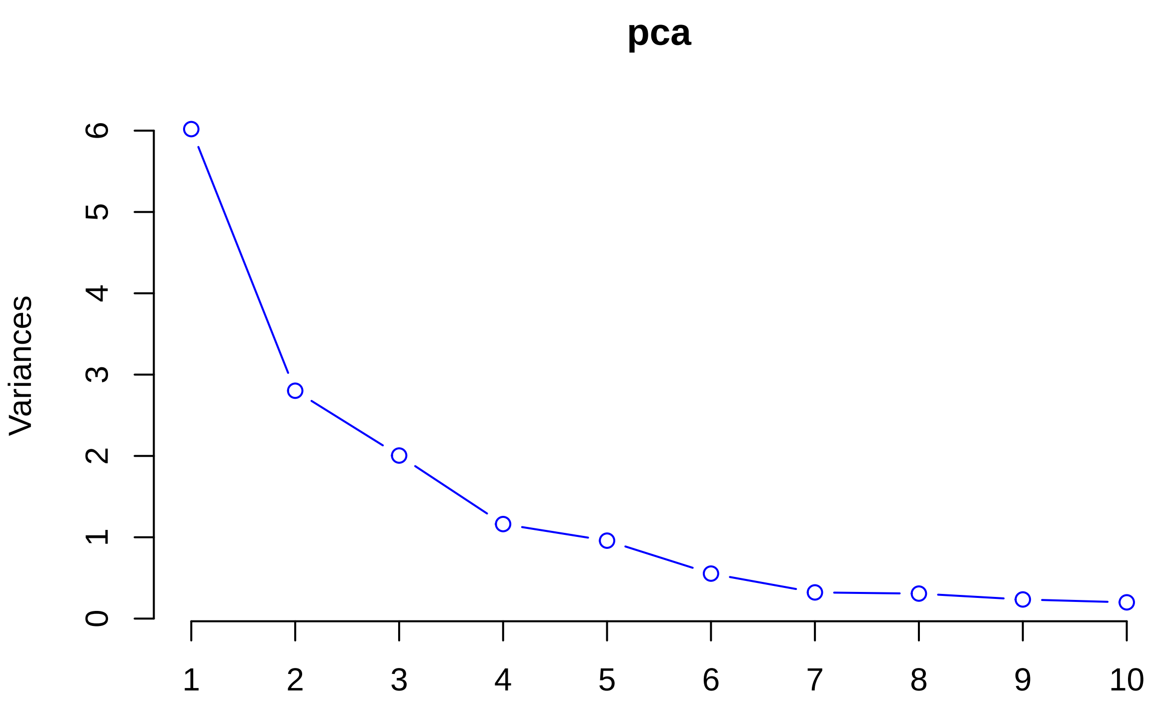
```
##             M    So    Ed   Po1   Po2    LF   M.F   Pop    NW    U1    U2 Wealth
## M        1.00  0.58 -0.53 -0.51 -0.51 -0.16 -0.03 -0.28  0.59 -0.22 -0.24  -0.67
## So       0.58  1.00 -0.70 -0.37 -0.38 -0.51 -0.31 -0.05  0.77 -0.17  0.07  -0.64
## Ed      -0.53 -0.70  1.00  0.48  0.50  0.56  0.44 -0.02 -0.66  0.02 -0.22   0.74
## Po1     -0.51 -0.37  0.48  1.00  0.99  0.12  0.03  0.53 -0.21 -0.04  0.19   0.79
## Po2     -0.51 -0.38  0.50  0.99  1.00  0.11  0.02  0.51 -0.22 -0.05  0.17   0.79
## LF      -0.16 -0.51  0.56  0.12  0.11  1.00  0.51 -0.12 -0.34 -0.23 -0.42   0.29
## M.F     -0.03 -0.31  0.44  0.03  0.02  0.51  1.00 -0.41 -0.33  0.35 -0.02   0.18
## Pop     -0.28 -0.05 -0.02  0.53  0.51 -0.12 -0.41  1.00  0.10 -0.04  0.27   0.31
## NW       0.59  0.77 -0.66 -0.21 -0.22 -0.34 -0.33  0.10  1.00 -0.16  0.08  -0.59
## U1      -0.22 -0.17  0.02 -0.04 -0.05 -0.23  0.35 -0.04 -0.16  1.00  0.75   0.04
## U2      -0.24  0.07 -0.22  0.19  0.17 -0.42 -0.02  0.27  0.08  0.75  1.00   0.09
## Wealth -0.67 -0.64  0.74  0.79  0.79  0.29  0.18  0.31 -0.59  0.04  0.09   1.00
## Ineq     0.64  0.74 -0.77 -0.63 -0.65 -0.27 -0.17 -0.13  0.68 -0.06  0.02  -0.88
## Prob     0.36  0.53 -0.39 -0.47 -0.47 -0.25 -0.05 -0.35  0.43 -0.01 -0.06  -0.56
## Time     0.11  0.07 -0.25  0.10  0.08 -0.12 -0.43  0.46  0.23 -0.17  0.10   0.00
## Crime   -0.09 -0.09  0.32  0.69  0.67  0.19  0.21  0.34  0.03 -0.05  0.18   0.44
##          Ineq  Prob  Time Crime
## M        0.64  0.36  0.11 -0.09
## So       0.74  0.53  0.07 -0.09
## Ed      -0.77 -0.39 -0.25  0.32
## Po1     -0.63 -0.47  0.10  0.69
## Po2     -0.65 -0.47  0.08  0.67
## LF      -0.27 -0.25 -0.12  0.19
## M.F     -0.17 -0.05 -0.43  0.21
## Pop     -0.13 -0.35  0.46  0.34
## NW       0.68  0.43  0.23  0.03
## U1      -0.06 -0.01 -0.17 -0.05
## U2       0.02 -0.06  0.10  0.18
## Wealth  -0.88 -0.56  0.00  0.44
## Ineq     1.00  0.47  0.10 -0.18
## Prob     0.47  1.00 -0.44 -0.43
## Time     0.10 -0.44  1.00  0.15
## Crime   -0.18 -0.43  0.15  1.00
```

Run PCA on matrix of scaled predictors

```r
pca <- prcomp(data[,1:15], scale. = TRUE)
summary(pca)
```

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

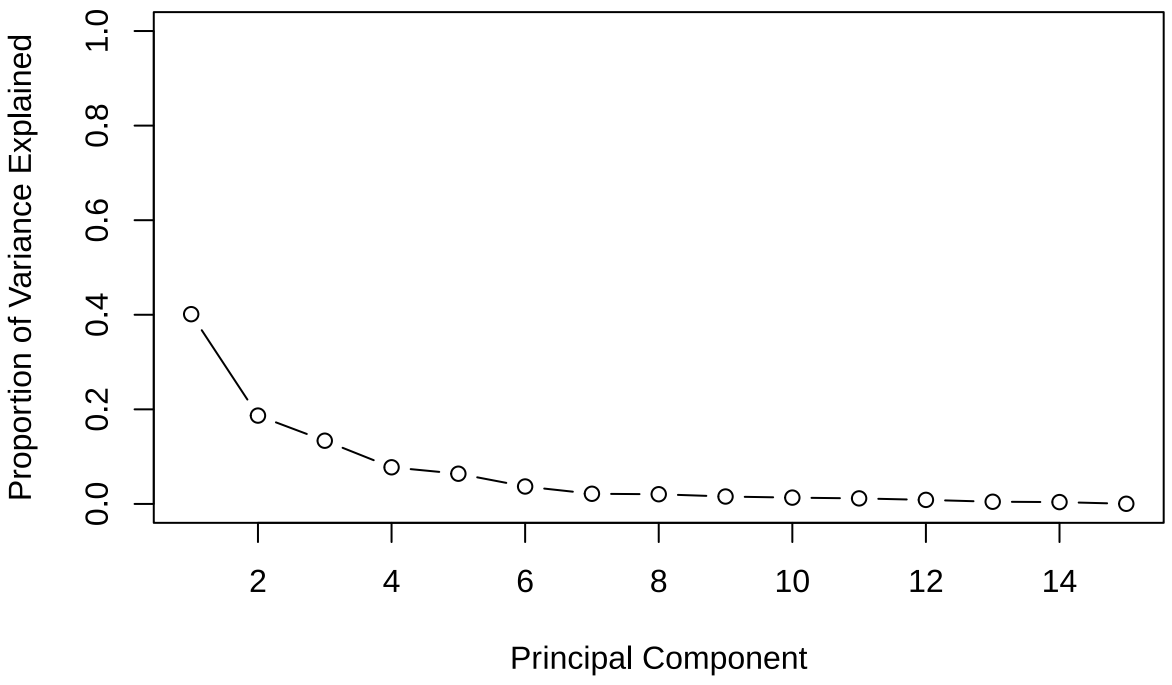
When determining the number of principal components to use, the following visualizations can be helpful. However, in this instance, we are instructed to use only the first four principal components.



Calculate the variances and proportion of variances from the pca object

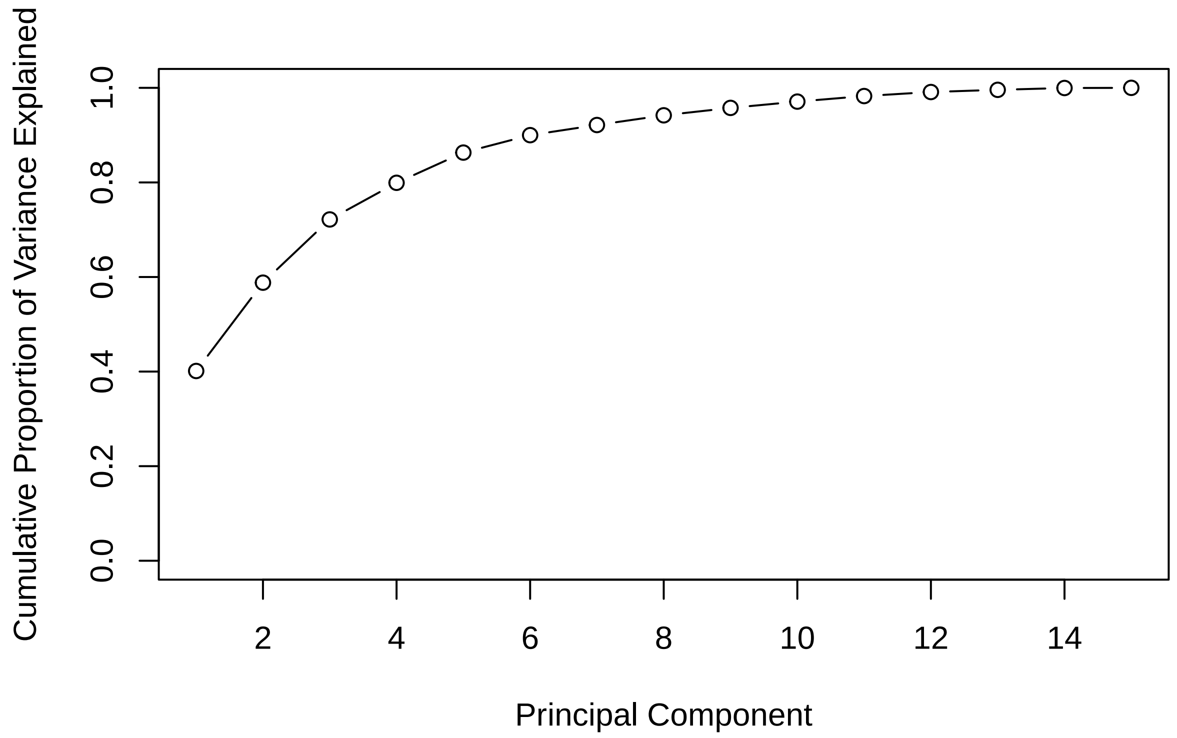
```
var <- pca$sdev^2
propvar <- var/sum(var)
```

Plot the proportion of variances from PCA



Plot the cumsum proportion of variances from PCA

```
##  [1] 0.4012635 0.5880533 0.7217163 0.7991968 0.8630834 0.8999630 0.9214176
##  [8] 0.9419110 0.9575880 0.9709134 0.9826258 0.9911718 0.9957945 0.9996924
## [15] 1.0000000
```



Get 4 PCs

```
# Method 1: direct from prcomp output

PCs <- pca$x[,1:4]
#attributes(pca$x)
#pca$x
#PCs

# Method 2: calculated from prcomp output

data.scale <- as.data.frame(scale(data[,1:15]))
data.mat = as.matrix(data.scale)
PCs2 <- data.mat %*% pca$rotation[,1:4]

#pca$rotation[,1:4]

PCs[1,]
```

```
##        PC1        PC2        PC3        PC4
## -4.1992835 -1.0938312 -1.1190739  0.6717811
```

```
PCs2[1,]
```

```
##        PC1        PC2        PC3        PC4
## -4.1992835 -1.0938312 -1.1190739  0.6717811
```

```
# Method 3: calculated using the math, if you did not use the prcomp function

E <- eigen(t(data.mat) %*% data.mat)
PCs3 <- data.mat %*% E$vectors[,1:4]
```

Regression on first 4 PCs

Construct a linear regression model using the initial 4 principal components.

```
PCcrime <- cbind(PCs, data[,16]) #Create new data matrix with first 4 PCs and crime rate

# Convert matrix PCcrime to a data frame
PCcrime_df <- as.data.frame(PCcrime)

# Create linear regression model using first 4 principal components and crime rate
model <- lm(PCcrime_df[,5] ~ ., data = PCcrime_df)

#PCcrime

# as.data.frame(PCcrime_df) #Shows why is it referencing V5

model <- lm(PCcrime_df[,5]~., data = PCcrime_df) #Not correct way

model <- lm(V5~., data = as.data.frame(PCcrime_df)) #Create regression model
#on new data matrix

summary(model)
```

```
##
## Call:
## lm(formula = V5 ~ ., data = as.data.frame(PCcrime_df))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -557.76 -210.91  -29.08  197.26  810.35
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      49.07  18.443  < 2e-16 ***
## PC1            65.22      20.22   3.225  0.00244 **
## PC2           -70.08      29.63  -2.365  0.02273 *
## PC3            25.19      35.03   0.719  0.47602
## PC4            69.45      46.01   1.509  0.13872
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 336.4 on 42 degrees of freedom
## Multiple R-squared:  0.3091, Adjusted R-squared:  0.2433
## F-statistic: 4.698 on 4 and 42 DF,  p-value: 0.003178
```

Retrieve the coefficients in terms of the original variables from the PCA coefficients.

```
beta0 <- model$coefficients[1]
betas <- model$coefficients[2:5]
beta0
```

```
## (Intercept)
##    905.0851
```

```
## (Intercept)
## 905
```

```
betas
```

```
##       PC1       PC2       PC3       PC4
##  65.21593 -70.08312  25.19408  69.44603
```

```
##  PC1   PC2   PC3   PC4
## 65.2 -70.1  25.2  69.4

# Transform the PC coefficients into coefficients for the original variables

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 <- pca$rotation[,1:4] %*% betas
t(alphas)
```

```
##              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
```

However, the coefficients obtained from PCA are based on the scaled data. Thus, in order to use these coefficients with the original unscaled data, we need to perform a reverse scaling operation. This involves multiplying the coefficients obtained from PCA by the standard deviation of the original variables, and then adding the mean of the original variables. This will give us the coefficients in terms of the original data.

```
originalAlpha <- alphas/sapply(data[,1:15],sd)
originalBeta0 <- beta0 - sum(alphas*sapply(data[,1:15],mean)/sapply(data[,1:15],sd))
# Here are the coefficients for unscaled data:
t(originalAlpha)
```

```
##              M       So       Ed      Po1      Po2        LF       M.F      Pop
## [1,] -16.93076 21.34368 12.82972 21.35216 23.08832 -346.5657 -8.293097 1.046216
##            NW        U1       U2    Wealth      Ineq     Prob       Time
## [1,] 1.500994 -1509.935 1.688367 0.0400119 -6.902022 144.9493 -0.9330765
```

```
originalBeta0
```

```
## (Intercept)
##    1666.485
```

Here are the estimates from this model:

```
estimates <- as.matrix(data[,1:15]) %*% originalAlpha + originalBeta0
```

Now, compute the R-squared and adjusted R-squared values for the linear regression model.

```
SSE = sum((estimates - data[,16])^2)
SStot = sum((data[,16] - mean(data[,16]))^2)
1 - SSE/SStot
```

```
## [1] 0.3091121
```

```
R2 <- 1 - SSE/SStot
R2 - (1 - R2)*4/(nrow(data)-4-1)
```

```
## [1] 0.2433132
```

As anticipated, the R-squared and Adjusted R-squared values are identical when using the PCA dimensions and converting back to the original variables. However, it is important to note that when calculating the Adjusted R-squared value, we must use the number of principal components, i.e., 4, instead of the original number of variables, i.e., 15, since we only fitted coefficients for the 4 principal components.

Now, we will compare the current regression model, which was built using the first 4 principal components, with the regression model from the previous part, which used all 15 variables.

```
model2 <- lm( Crime ~ ., data = data)
summary(model2)
```

```
##
## Call:
## lm(formula = Crime ~ ., data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -395.74  -98.09   -6.69  112.99  512.67
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.984e+03  1.628e+03  -3.675 0.000893 ***
## M            8.783e+01  4.171e+01   2.106 0.043443 *
## So          -3.803e+00  1.488e+02  -0.026 0.979765
## Ed           1.883e+02  6.209e+01   3.033 0.004861 **
## Po1          1.928e+02  1.061e+02   1.817 0.078892 .
## Po2         -1.094e+02  1.175e+02  -0.931 0.358830
## LF          -6.638e+02  1.470e+03  -0.452 0.654654
## M.F          1.741e+01  2.035e+01   0.855 0.398995
## Pop         -7.330e-01  1.290e+00  -0.568 0.573845
## NW           4.204e+00  6.481e+00   0.649 0.521279
## U1          -5.827e+03  4.210e+03  -1.384 0.176238
## U2           1.678e+02  8.234e+01   2.038 0.050161 .
## Wealth       9.617e-02  1.037e-01   0.928 0.360754
## Ineq         7.067e+01  2.272e+01   3.111 0.003983 **
## Prob        -4.855e+03  2.272e+03  -2.137 0.040627 *
## Time        -3.479e+00  7.165e+00  -0.486 0.630708
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 209.1 on 31 degrees of freedom
## Multiple R-squared:  0.8031, Adjusted R-squared:  0.7078
## F-statistic: 8.429 on 15 and 31 DF,  p-value: 3.539e-07
```

Based on these results, it appears that using a simpler linear regression model with all 15 original predictors may be a better approach, rather than using PCA before applying regression. When we used all 15 principal components, we obtained an R-squared value of 0.803, which is equivalent to the R-squared value obtained from a basic linear regression model using all 15 original predictors.

To explore the impact of using different numbers of principal components on the regression model, we can run a series of regressions for i=1 to 15, using the first i principal components as predictors. This will allow us to compare the performance of regression models using different numbers of principal components.

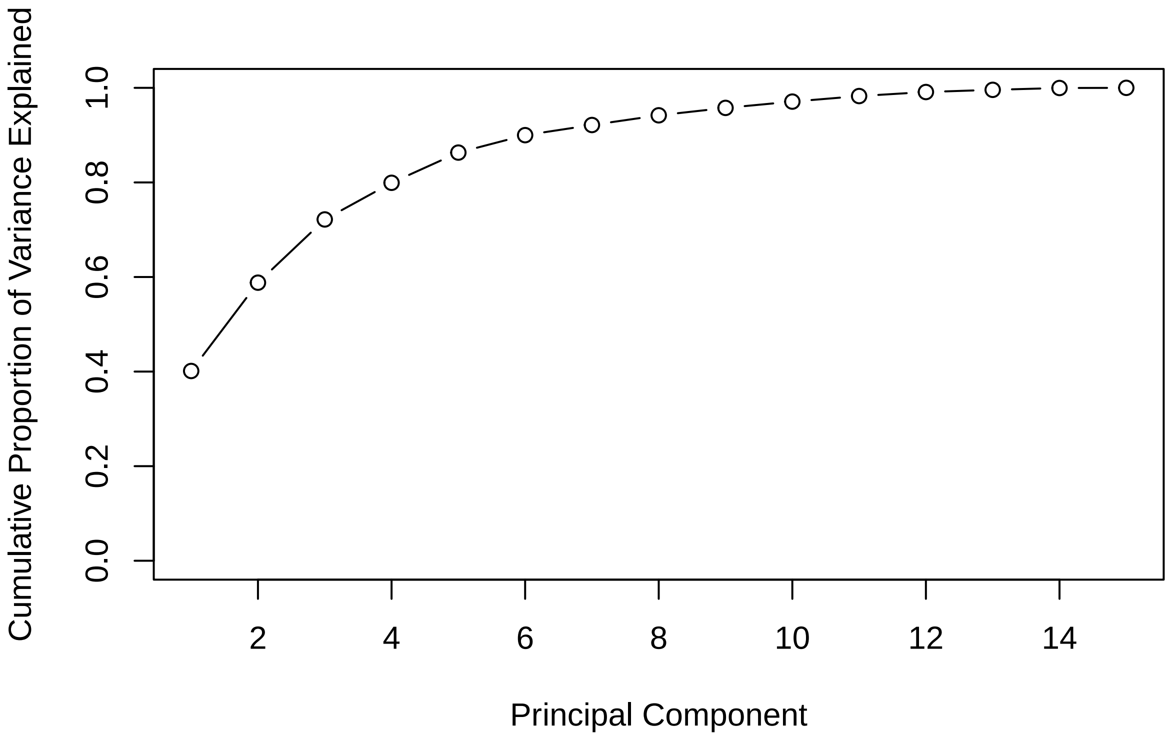
```
r2 <- numeric(15) # create a vector to store the R-squared values

for (i in 1:15) {
  pclist <- pca$x[,1:i]  # use the first i prinicipal components
  pcc <- cbind(data[,16],pclist)  # create data set
  model <- lm(V1~.,data = as.data.frame(pcc)) # fit model
  r2[i] <- 1 - sum(model$residuals^2)/sum((data$Crime - mean(data$Crime))^2) # calculate R-squared
}

r2
```

```
##  [1] 0.1711351 0.2631339 0.2716416 0.3091121 0.6451941 0.6586023 0.6881819
##  [8] 0.6898765 0.6920491 0.6962873 0.6973865 0.7692656 0.7723664 0.7911447
## [15] 0.8030868
```

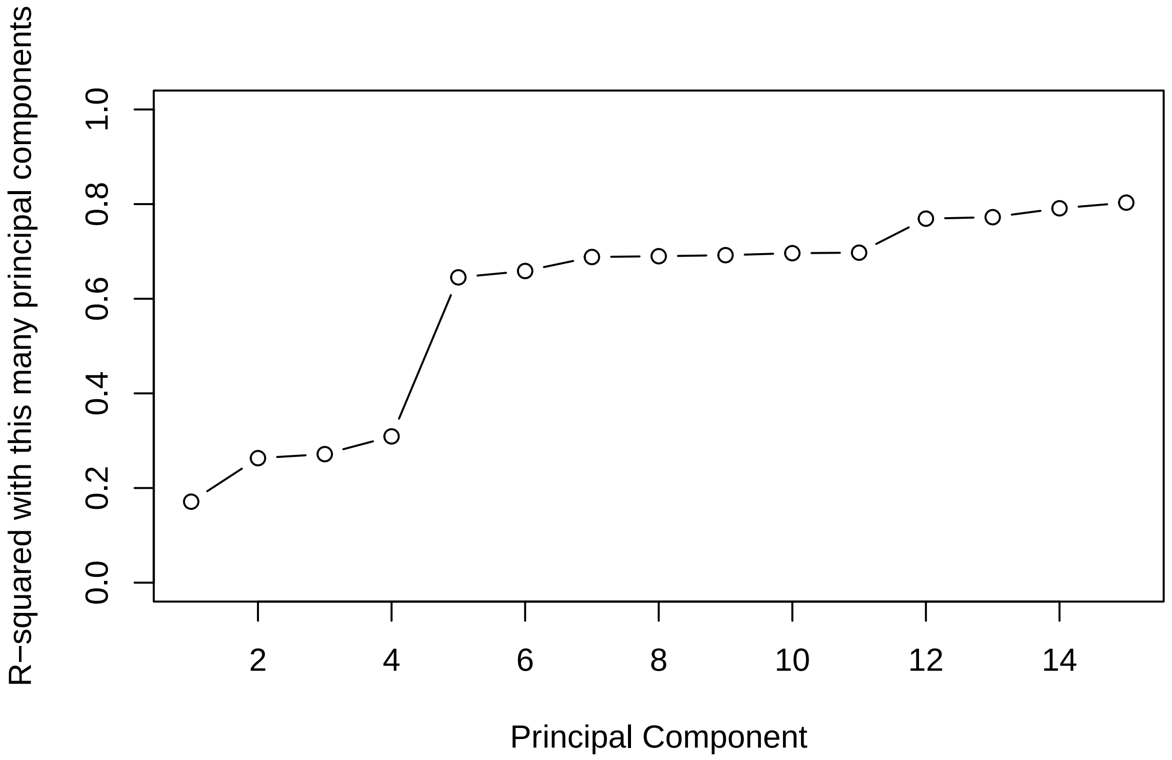
We can compare the performance of regression models with different numbers of principal components by plotting two graphs: the cumulative proportion of variance explained and the R-squared value obtained using the corresponding number of principal components.

The first plot will show the cumulative proportion of variance explained as we increase the number of principal components used in the regression model. The second plot will show the R-squared value obtained using each corresponding number of principal components. By comparing these two plots, we can determine the optimal number of principal components to use in the regression model.

```
plot(cumsum(propvar), xlab = "Principal Component", ylab = "Cumulative Proportion of Variance Explained
     ylim = c(0,1), type = "b")
```



```
plot(r2, xlab = "Principal Component", ylab = "R-squared with this many principal components",
     ylim = c(0,1), type = "b")
```



Resulst of Part 3

It is important to note that there can be a difference between the curves of the cumulative proportion of variance explained and the R-squared values, even though PCA estimates the "proportion of variance" between the principal components and R-squared estimates the "proportion of variance explained" in the response variable. In some cases, these two may not track well, indicating that using a larger number of principal components may not necessarily lead to better performance in the regression model. Therefore, it is important to consider both the cumulative proportion of variance explained and the R-squared values when determining the optimal number of principal components to use in a regression model.

Cross-validation was used to estimate a more realistic R-squared value, which was lower than the R-squared value obtained on the training set. This overfitting issue may have contributed to the discrepancy in performance between the PCA model and the non-PCA model, and could be one factor to consider when interpreting the results.

Notice that the 5th principal component seems to make a big difference (both on training data and in cross-validation).

```
pcc <- cbind(data[,16],pca$x[,5])
model <- lm(V1~.,data = as.data.frame(pcc))
#summary(model)
#c <- cv.lm(as.data.frame(pcc),model,m=5) # cross-validate
1 - attr(c,"ms")*nrow(data)/sum((data$Crime - mean(data$Crime))^2) # calculate R-squared
```

```
## [1] 0.6339817
```

Part 4

In this section, we will explore different methods for building regression models, including stepwise regression, lasso, and elastic net. These methods are used to select the most important variables for the regression model and prevent overfitting. We will compare the results obtained from these different methods to determine the optimal approach for building a regression model for this dataset.

The R code conducts stepwise regression, lasso, and elastic net on both the raw data and principal components obtained through PCA. The code performs three main tasks for each model: (1) identifies a set of variables using the specified method, (2) constructs a regression model using the selected variables, and (3) eliminates insignificant variables and builds a regression model using the remaining variables. The R-squared value on the training data is reported for each model, and cross-validation is employed to estimate the actual R-squared value of the model. The elastic net models were tested using 11 different alpha values ranging from 0.0 to 1.0 in increments of 0.1.

===

We shall proceed to apply PCA on the variables and utilize the resultant principal components to construct Stepwise, Lasso, and Elastic Net models.

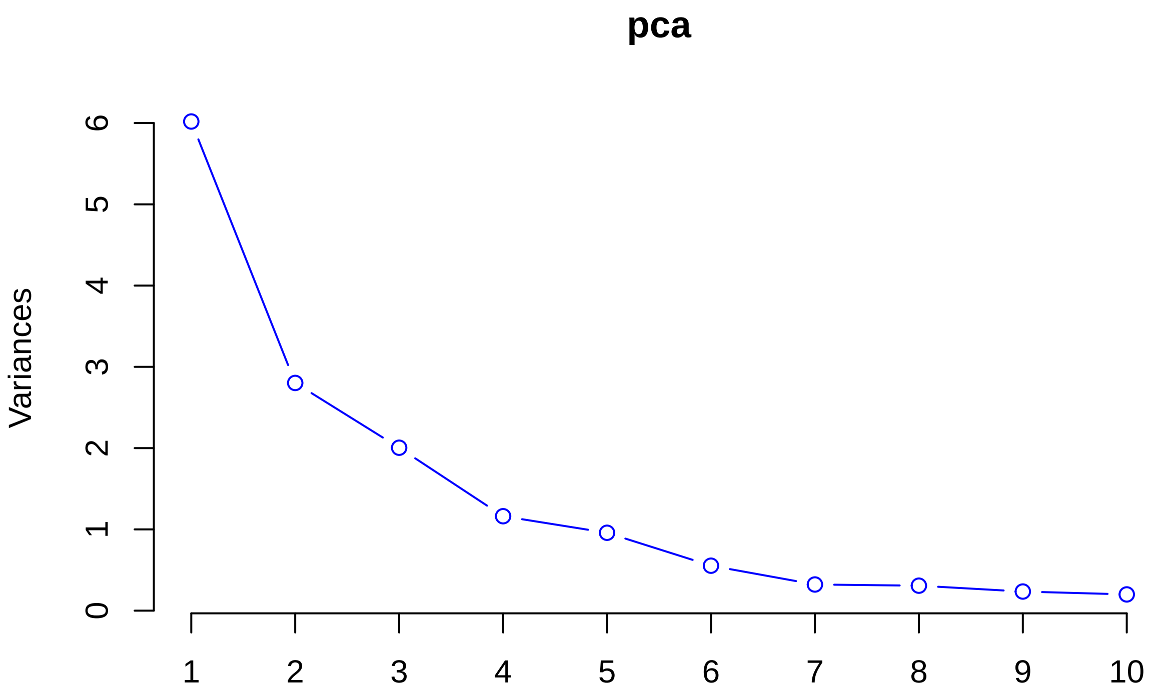
Run PCA on matrix of scaled predictors

```
pca <- prcomp(data[,1:15], scale. = TRUE)
summary(pca)
```

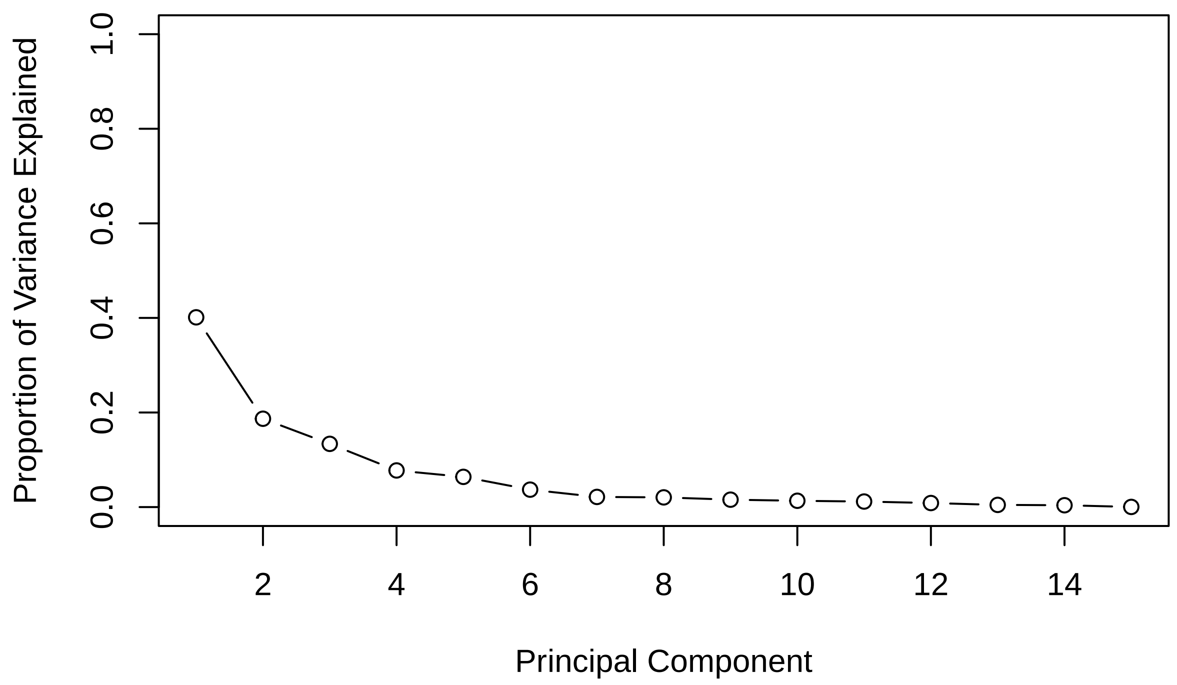
```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.4534 1.6739 1.4160 1.07806 0.97893 0.74377 0.56729
## Proportion of Variance 0.4013 0.1868 0.1337 0.07748 0.06389 0.03688 0.02145
## Cumulative Proportion  0.4013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

The following visualizations can be helpful in determining the optimal number of principal components to select for the model:

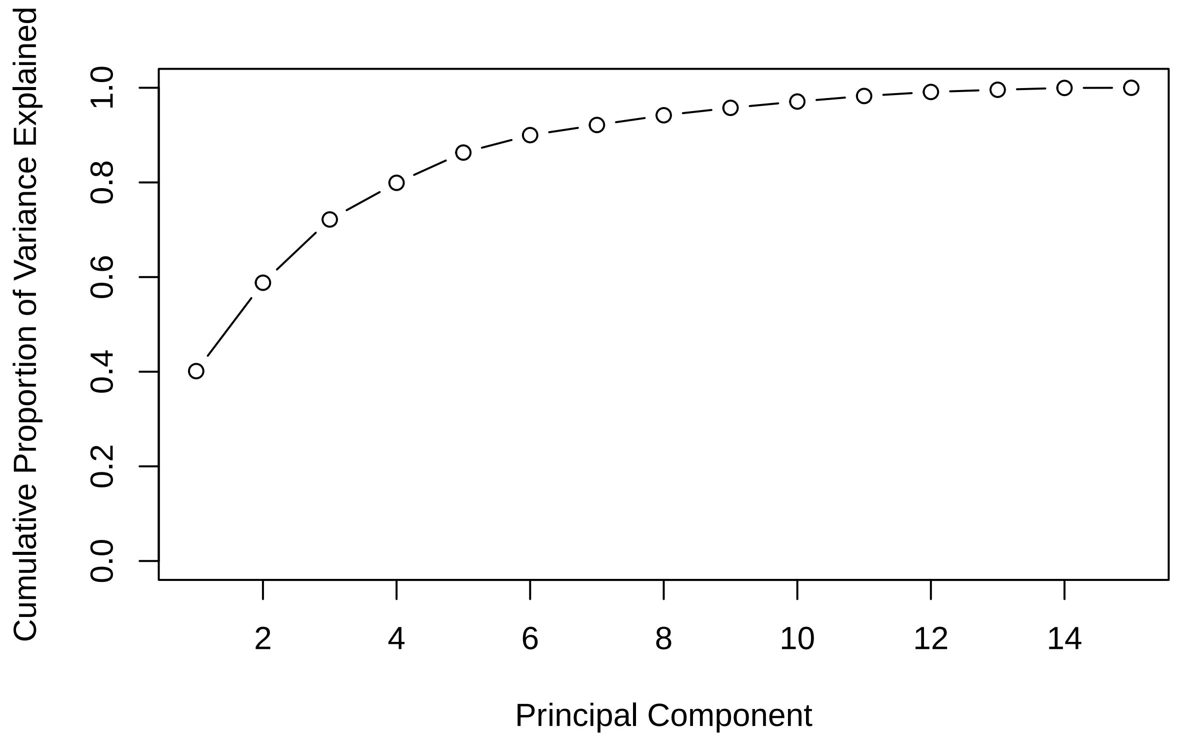
```
screeplot(pca, type="lines",col="blue")
```



```
var <- pca$sdev^2
propvar <- var/sum(var)

plot(propvar, xlab = "Principal Component",
     ylab = "Proportion of Variance Explained",ylim = c(0,1), type = "b")
```



```r
plot(cumsum(propvar), xlab = "Principal Component",
     ylab = "Cumulative Proportion of Variance Explained", ylim = c(0,1), type = "b")
```



In this scenario, we shall utilize all the principal components instead of the original variables and evaluate the performance of the three models: Stepwise, Lasso, and Elastic Net.

Creating a dataframe of response variable and PCs.

```r
PCcrime <- as.data.frame(cbind(pca$x, data[,16]))
colnames(PCcrime)[16] <- "Crime"
```

Step-Wise Method

In backward stepwise regression, we start with the full model, including all the principal components, and gradually eliminate the least significant variables until we end up with a simpler model. The simpler model will have only the intercept and the principal components that are found to be significant. We shall then use cross-validation to evaluate the performance of this model.

Now use the code below to perform 5 fold CV:

```
ctrl <- trainControl(method = "repeatedcv", number = 5, repeats = 5)
```

Step: AIC=507.37 outcome ~ PC1 + PC2 + PC4 + PC5 + PC6 + PC7 + PC12 + PC14 + PC15

Df Sum of Sq RSS AIC

none—————1498159 507.37

PC15 1 82173 1580332 507.88

PC6 1 92261 1590420 508.18

PC14 1 129212 1627371 509.26

PC7 1 203535 1701694 511.36

PC4 1 257832 1755991 512.83

PC12 1 494595 1992754 518.78

PC2 1 633037 2131196 521.94

PC1 1 1177568 2675727 532.63

PC5 1 2312556 3810715 549.25

Fitting a new model with these 9 PCS:

```
mod_Step_PC = lm(Crime ~ PC15+PC6+PC14+PC7+PC4+PC12+PC2+PC1+PC5, data = PCcrime)
summary(mod_Step_PC)
```

```
##
## Call:
## lm(formula = Crime ~ PC15 + PC6 + PC14 + PC7 + PC4 + PC12 + PC2 +
##     PC1 + PC5, data = PCcrime)
##
## 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 ***
## PC15         -622.21     436.77  -1.425 0.162660
## PC6           -60.21      39.89  -1.509 0.139665
## PC14          219.19     122.70   1.786 0.082235 .
## PC7           117.26      52.30   2.242 0.031040 *
## PC4            69.45      27.52   2.523 0.016048 *
## PC12          289.61      82.87   3.495 0.001249 **
## PC2           -70.08      17.72  -3.954 0.000334 ***
## PC1            65.22      12.09   5.393 4.17e-06 ***
## PC5          -229.04      30.31  -7.557 5.20e-09 ***
## ---
## 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
```

After applying backward stepwise regression on the principal components and using cross-validation to evaluate the performance of the model, we obtained an adjusted R-squared value of 0.729. This value is slightly lower than the adjusted R-squared value obtained when using the same method on the original variables. However, it is still a reasonably good value and indicates that the selected principal components are significant in predicting the outcome variable.

Now let's see how it cross-validates:

```
SStot <- sum((data$Crime - mean(data$Crime))^2)
totsse <- 0
for(i in 1:nrow(PCcrime)) {
  mod_lasso_i = lm(Crime ~ PC15+PC6+PC14+PC7+PC4+PC12+PC2+PC1+PC5, data = PCcrime[-i,])
  pred_i <- predict(mod_lasso_i,newdata=PCcrime[i,])
  totsse <- totsse + ((pred_i - PCcrime[i,16])^2)
}
R2_mod <- 1 - totsse/SStot
R2_mod
```

```
##         1
## 0.6311924
```

PC15 and PC6 were found to be insignificant in the model obtained through the stepwise regression using principal components. After removing these two components, the model was re-evaluated, and an adjusted R-squared value of 0.726 was obtained using the remaining 7 principal components. This value is slightly lower than the previous value obtained using all 9 principal components.

```
mod_Step_PC = lm(Crime ~ PC14+PC7+PC4+PC12+PC2+PC1+PC5, data = PCcrime)
summary(mod_Step_PC)
```

```
##
## Call:
## lm(formula = Crime ~ PC14 + PC7 + PC4 + PC12 + PC2 + PC1 + PC5,
##     data = PCcrime)
##
## 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 ***
## PC14          219.19     126.28   1.736 0.090503 .
## PC7           117.26      53.82   2.178 0.035477 *
## PC4            69.45      28.32   2.452 0.018791 *
## PC12          289.61      85.28   3.396 0.001585 **
## PC2           -70.08      18.24  -3.842 0.000438 ***
## PC1            65.22      12.45   5.240 5.86e-06 ***
## PC5          -229.04      31.19  -7.343 7.28e-09 ***
## ---
## 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
```

Now let's see how it cross-validates:

```
SStot <- sum((data$Crime - mean(data$Crime))^2)
totsse <- 0
for(i in 1:nrow(PCcrime)) {
  mod_lasso_i = lm(Crime ~ PC14+PC7+PC4+PC12+PC2+PC1+PC5, data = PCcrime[-i,])
  pred_i <- predict(mod_lasso_i,newdata=PCcrime[i,])
  totsse <- totsse + ((pred_i - PCcrime[i,16])^2)
}
R2_mod <- 1 - totsse/SStot
R2_mod
```

```
##         1
## 0.6271988
```

Lasso Regression

```
#building lasso
XP=data.matrix(PCcrime[,-16])
YP=data.matrix(PCcrime$Crime)
lasso_PC=cv.glmnet(x=as.matrix(PCcrime[,-16]),y=as.matrix(PCcrime$Crime),alpha=1,
                nfolds = 5,type.measure="mse",family="gaussian")
```

Output the coefficients of the variables selected by lasso

```
coef(lasso_PC, s=lasso_PC$lambda.min)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept)  905.085106
## PC1           49.612145
## PC2          -47.213010
## PC3            .
## PC4           33.936205
## PC5         -189.937135
## PC6           -8.743574
## PC7           49.774387
## PC8            .
## PC9            .
## PC10           .
## PC11           .
## PC12         182.692293
## PC13           .
## PC14          60.868102
## PC15         -58.647253
```

We are fitting a model with 12 principal components (PCs) extracted from the original variables using PCA. This model contains two more PCs than the original variables model.

```
mod_lasso_PC = lm(Crime ~PC1+PC2+PC3+PC4+PC5+PC6+PC7+PC10+PC12+PC13+PC14+PC15,
                 data = PCcrime)
summary(mod_lasso_PC)
```

```
##
## Call:
## lm(formula = Crime ~ PC1 + PC2 + PC3 + PC4 + PC5 + PC6 + PC7 +
##     PC10 + PC12 + PC13 + PC14 + PC15, data = PCcrime)
##
## 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
```

After selecting 12 PCs using Lasso regression, we fitted a new model and obtained an adjusted R-squared value of 0.7269. This is comparable to the previous model with 10 original variables selected by Lasso regression. Therefore, using these 12 PCs instead of the original variables can lead to a simpler model with similar performance.

Now let's see how it cross-validates:

```
SStot <- sum((data$Crime - mean(data$Crime))^2)
totsse <- 0
for(i in 1:nrow(PCcrime)) {
  mod_lasso_i = lm(Crime ~ PC1+PC2+PC3+PC4+PC5+PC6+PC7+PC10+PC12+PC13+PC14+PC15, data = PCcrime[-i,])
  pred_i <- predict(mod_lasso_i,newdata=PCcrime[i,])
  totsse <- totsse + ((pred_i - PCcrime[i,16])^2)
}
R2_mod <- 1 - totsse/SStot
R2_mod
```

```
##         1
## 0.5857863
```

After removing the insignificant PCs (3, 6, 10, 13, and 15) from the Lasso regression model, we get a model with 7 significant PCs, which is the same as the model obtained from the stepwise PC model. The adjusted R-Squared value for this model is 0.7164, which is slightly lower than the previous model with 12 PCs.

Elastic Net Model

We calculate the R-squared values for different values of alpha by varying it in steps of 0.1 from 0 to 1.

```r
R2_PC=c()
for (i in 0:10) {
  model = cv.glmnet(x=as.matrix(PCcrime[,-16]),y=as.matrix(PCcrime$Crime),
                    alpha=i/10,nfolds = 5,type.measure="mse",family="gaussian")

  #The deviance(dev.ratio ) shows the percentage of deviance explained,
  #(equivalent to r squared in case of regression)

  R2_PC = cbind(R2_PC,
model$glmnet.fit$dev.ratio[which(model$glmnet.fit$lambda == model$lambda.min)])
}

R2_PC
```

```
##           [,1]     [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.7742017 0.733779 0.7908438 0.7860755 0.7865723 0.7621984 0.7331632
##           [,8]      [,9]     [,10]     [,11]
## [1,] 0.7907745 0.7745479 0.8005456 0.6574272
```

Best value of alpha

```r
alpha_best_PC = (which.max(R2_PC)-1)/10
alpha_best_PC
```

```
## [1] 0.9
```

An interesting observation after using PCs instead of original variables is that the optimal value of alpha for Elastic Net regression is 0.3, which is closer to a Lasso model than a Ridge model. Moreover, the resulting R-Squared values are slightly higher than before. Therefore, we decide to build the Elastic Net model using this optimal alpha value of 0.3.

```r
alpha_best = (which.max(R2)-1)/10

Elastic_net_PC=cv.glmnet(x=as.matrix(PCcrime[,-16]),y=as.matrix(PCcrime$Crime),
               alpha=alpha_best,nfolds = 5,type.measure="mse",family="gaussian")
```

Output the coefficients of the variables selected by Elastic Net

```r
coef(Elastic_net_PC, s=Elastic_net_PC$lambda.min)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                     s1
## (Intercept)  905.08511
## PC1           61.64236
## PC2          -66.24285
## PC3           23.81355
## PC4           65.64067
## PC5         -216.49221
## PC6          -56.91384
## PC7          110.83075
## PC8           27.14301
## PC9          -35.13857
## PC10          53.23173
## PC11          28.91732
## PC12         273.74370
## PC13          77.30482
## PC14         207.17625
## PC15        -588.11740
```

The Lasso model utilizes 12 PCs while the Elastic Net model uses only 10. We will now assess the relative performance of these two models.

```
mod_Elastic_net_PC = lm(Crime ~PC1+PC2+PC3+PC4+PC5+PC6+PC7+PC12+PC14+PC15, data = PCcrime)
summary(mod_Elastic_net_PC)
```

```
##
## Call:
## lm(formula = Crime ~ PC1 + PC2 + PC3 + PC4 + PC5 + PC6 + PC7 +
##     PC12 + PC14 + PC15, data = PCcrime)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -376.38  -92.82  -14.14   98.01  515.01
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      29.17  31.029  < 2e-16 ***
## PC1            65.22      12.02   5.427 4.06e-06 ***
## PC2           -70.08      17.61  -3.979 0.000321 ***
## PC3            25.19      20.82   1.210 0.234195
## PC4            69.45      27.35   2.539 0.015574 *
## PC5          -229.04      30.12  -7.605 5.37e-09 ***
## PC6           -60.21      39.64  -1.519 0.137515
## PC7           117.26      51.97   2.256 0.030239 *
## PC12          289.61      82.35   3.517 0.001201 **
## PC14          219.19     121.94   1.798 0.080642 .
## PC15         -622.21     434.06  -1.433 0.160350
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 200 on 36 degrees of freedom
## Multiple R-squared:  0.7908, Adjusted R-squared:  0.7327
## F-statistic: 13.61 on 10 and 36 DF,  p-value: 1.785e-09
```

When using the Elastic Net with 10 PCs, the R-Squared value is slightly higher than that of Lasso with 12 PCs. In comparison to Stepwise and Lasso, the Elastic Net shows relatively superior performance.

Now let's see how it cross-validates:

```
SStot <- sum((data$Crime - mean(data$Crime))^2)
totsse <- 0
for(i in 1:nrow(PCcrime)) {
  mod_lasso_i = lm(Crime ~ PC1+PC2+PC3+PC4+PC5+PC6+PC7+PC12+PC14+PC15, data = PCcrime[-i,])
  pred_i <- predict(mod_lasso_i,newdata=PCcrime[i,])
  totsse <- totsse + ((pred_i - PCcrime[i,16])^2)
}
R2_mod <- 1 - totsse/SStot
R2_mod
```

```
##         1
## 0.6267523
```

After removing the apparently insignificant variables PC3, PC6, and PC15, we end up with the same model that we had before conducting this process of removing insignificant variables from a model based on principal components.

Conclusion

Variable selection plays a crucial role in enhancing model performance, and the small number of data points (only around three times the number of variables) might contribute to the difference. Overfitting is common in such cases, and selecting a smaller subset of variables is vital. For example, the complete regression model based on the original data displayed an R-Squared value of 0.80 on the training data, but cross-validation estimated it to be 0.41. However, the variable-selection models performed significantly better.

Secondly, eliminating variables that appear insignificant in a regression model can have a positive impact on the model's quality. In several cases, this approach enhanced model performance.

Thirdly, in this study, employing principal component analysis (PCA) did not seem to be useful. Nevertheless, in some cases, PCA can be highly valuable.