Variable Selection

Using the crime data set from statsci I ran Stepwise Regression, LASSO, and Elastic Net before making Linear Regression models. The code below explores each of the different variable selection models and reports the R^2 value for each.

```
rm(list = ls())

library(tidyverse)
library(glmnet)
library(DAAG)

#setwd("/Users/Ryan/Desktop/DS/")

crime_data = read.table("uscrime.txt.",
                        sep="",
                        fill=FALSE,
                        strip.white=TRUE,
                        header = TRUE)

s_crime_data = cbind(as.data.frame(scale(crime_data[,1])),
                     as.data.frame(crime_data[,2]), #leave the 2nd column
                     as.data.frame(scale(crime_data[,c(3,4,5,6,7,8,9,10,11,12,13,14,15)])),
                     as.data.frame(crime_data[,16]))

colnames(s_crime_data) = colnames(crime_data) #fix the names of columns
```

First, we clear the environment and load in the necessary libraries. Then, we set the working directory in order to load the crime data from that folder. We load in the crime data and after that we make a scaled copy of the data. We scale all columns except for the second one because it is a binary variable which shouldn't be scaled. However, the first and last column are screwed up with the scaling, so we reset the names of the columns.

Stepwise Regression

```
model_both <- lm(Crime~., data = crime_data)
step(model_both,
     scope = list(lower = formula(lm(Crime~1, data = crime_data)), #forward
                  upper = formula(lm(Crime~., data = crime_data))), #backward
     direction = 'both')
```

```
## Start:  AIC=514.65
## Crime ~ M + So + Ed + Po1 + Po2 + LF + M.F + Pop + NW + U1 +
##     U2 + Wealth + Ineq + Prob + Time
##
```

```
##          Df Sum of Sq     RSS    AIC
## - So      1        29 1354974 512.65
## - LF      1      8917 1363862 512.96
## - Time    1     10304 1365250 513.00
## - Pop     1     14122 1369068 513.14
## - NW      1     18395 1373341 513.28
## - M.F     1     31967 1386913 513.74
## - Wealth  1     37613 1392558 513.94
## - Po2     1     37919 1392865 513.95
## <none>                1354946 514.65
## - U1      1     83722 1438668 515.47
## - Po1     1    144306 1499252 517.41
## - U2      1    181536 1536482 518.56
## - M       1    193770 1548716 518.93
## - Prob    1    199538 1554484 519.11
## - Ed      1    402117 1757063 524.86
## - Ineq    1    423031 1777977 525.42
##
## Step:  AIC=512.65
## Crime ~ M + Ed + Po1 + Po2 + LF + M.F + Pop + NW + U1 + U2 +
##     Wealth + Ineq + Prob + Time
##
##          Df Sum of Sq     RSS    AIC
## - Time    1     10341 1365315 511.01
## - LF      1     10878 1365852 511.03
## - Pop     1     14127 1369101 511.14
## - NW      1     21626 1376600 511.39
## - M.F     1     32449 1387423 511.76
## - Po2     1     37954 1392929 511.95
## - Wealth  1     39223 1394197 511.99
## <none>                1354974 512.65
## - U1      1     96420 1451395 513.88
## + So      1        29 1354946 514.65
## - Po1     1    144302 1499277 515.41
## - U2      1    189859 1544834 516.81
## - M       1    195084 1550059 516.97
## - Prob    1    204463 1559437 517.26
## - Ed      1    403140 1758114 522.89
## - Ineq    1    488834 1843808 525.13
##
## Step:  AIC=511.01
## Crime ~ M + Ed + Po1 + Po2 + LF + M.F + Pop + NW + U1 + U2 +
##     Wealth + Ineq + Prob
##
##          Df Sum of Sq     RSS    AIC
## - LF      1     10533 1375848 509.37
## - NW      1     15482 1380797 509.54
## - Pop     1     21846 1387161 509.75
## - Po2     1     28932 1394247 509.99
## - Wealth  1     36070 1401385 510.23
## - M.F     1     41784 1407099 510.42
## <none>                1365315 511.01
## - U1      1     91420 1456735 512.05
## + Time    1     10341 1354974 512.65
```

```
## + So      1        65 1365250 513.00
## - Po1     1    134137 1499452 513.41
## - U2      1    184143 1549458 514.95
## - M       1    186110 1551425 515.01
## - Prob    1    237493 1602808 516.54
## - Ed      1    409448 1774763 521.33
## - Ineq    1    502909 1868224 523.75
##
## Step:  AIC=509.37
## Crime ~ M + Ed + Po1 + Po2 + M.F + Pop + NW + U1 + U2 + Wealth +
##     Ineq + Prob
##
##          Df Sum of Sq     RSS    AIC
## - NW      1     11675 1387523 507.77
## - Po2     1     21418 1397266 508.09
## - Pop     1     27803 1403651 508.31
## - M.F     1     31252 1407100 508.42
## - Wealth  1     35035 1410883 508.55
## <none>                1375848 509.37
## - U1      1     80954 1456802 510.06
## + LF      1     10533 1365315 511.01
## + Time    1      9996 1365852 511.03
## + So      1      3046 1372802 511.26
## - Po1     1    123896 1499744 511.42
## - U2      1    190746 1566594 513.47
## - M       1    217716 1593564 514.27
## - Prob    1    226971 1602819 514.54
## - Ed      1    413254 1789103 519.71
## - Ineq    1    500944 1876792 521.96
##
## Step:  AIC=507.77
## Crime ~ M + Ed + Po1 + Po2 + M.F + Pop + U1 + U2 + Wealth + Ineq +
##     Prob
##
##          Df Sum of Sq     RSS    AIC
## - Po2     1     16706 1404229 506.33
## - Pop     1     25793 1413315 506.63
## - M.F     1     26785 1414308 506.66
## - Wealth  1     31551 1419073 506.82
## <none>                1387523 507.77
## - U1      1     83881 1471404 508.52
## + NW      1     11675 1375848 509.37
## + So      1      7207 1380316 509.52
## + LF      1      6726 1380797 509.54
## + Time    1      4534 1382989 509.61
## - Po1     1    118348 1505871 509.61
## - U2      1    201453 1588976 512.14
## - Prob    1    216760 1604282 512.59
## - M       1    309214 1696737 515.22
## - Ed      1    402754 1790276 517.74
## - Ineq    1    589736 1977259 522.41
##
## Step:  AIC=506.33
## Crime ~ M + Ed + Po1 + M.F + Pop + U1 + U2 + Wealth + Ineq +
```

```
##     Prob
##
##          Df Sum of Sq     RSS    AIC
## - Pop     1     22345 1426575 505.07
## - Wealth  1     32142 1436371 505.39
## - M.F     1     36808 1441037 505.54
## <none>                1404229 506.33
## - U1      1     86373 1490602 507.13
## + Po2     1     16706 1387523 507.77
## + NW      1      6963 1397266 508.09
## + So      1      3807 1400422 508.20
## + LF      1      1986 1402243 508.26
## + Time    1       575 1403654 508.31
## - U2      1    205814 1610043 510.76
## - Prob    1    218607 1622836 511.13
## - M       1    307001 1711230 513.62
## - Ed      1    389502 1793731 515.83
## - Ineq    1    608627 2012856 521.25
## - Po1     1   1050202 2454432 530.57
##
## Step:  AIC=505.07
## Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Wealth + Ineq + Prob
##
##          Df Sum of Sq     RSS    AIC
## - Wealth  1     26493 1453068 503.93
## <none>                1426575 505.07
## - M.F     1     84491 1511065 505.77
## - U1      1     99463 1526037 506.24
## + Pop     1     22345 1404229 506.33
## + Po2     1     13259 1413315 506.63
## + NW      1      5927 1420648 506.87
## + So      1      5724 1420851 506.88
## + LF      1      5176 1421398 506.90
## + Time    1      3913 1422661 506.94
## - Prob    1    198571 1625145 509.20
## - U2      1    208880 1635455 509.49
## - M       1    320926 1747501 512.61
## - Ed      1    386773 1813348 514.35
## - Ineq    1    594779 2021354 519.45
## - Po1     1   1127277 2553852 530.44
##
## Step:  AIC=503.93
## Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob
##
##          Df Sum of Sq     RSS    AIC
## <none>                1453068 503.93
## + Wealth  1     26493 1426575 505.07
## - M.F     1    103159 1556227 505.16
## + Pop     1     16697 1436371 505.39
## + Po2     1     14148 1438919 505.47
## + So      1      9329 1443739 505.63
## + LF      1      4374 1448694 505.79
## + NW      1      3799 1449269 505.81
## + Time    1      2293 1450775 505.86
```

```
## - U1      1    127044 1580112 505.87
## - Prob    1    247978 1701046 509.34
## - U2      1    255443 1708511 509.55
## - M       1    296790 1749858 510.67
## - Ed      1    445788 1898855 514.51
## - Ineq    1    738244 2191312 521.24
## - Po1     1   1672038 3125105 537.93
```

```
##
## Call:
## lm(formula = Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob,
##     data = crime_data)
##
## Coefficients:
## (Intercept)            M           Ed          Po1          M.F           U1
##    -6426.10        93.32       180.12       102.65        22.34     -6086.63
##          U2         Ineq         Prob
##      187.35        61.33     -3796.03
```

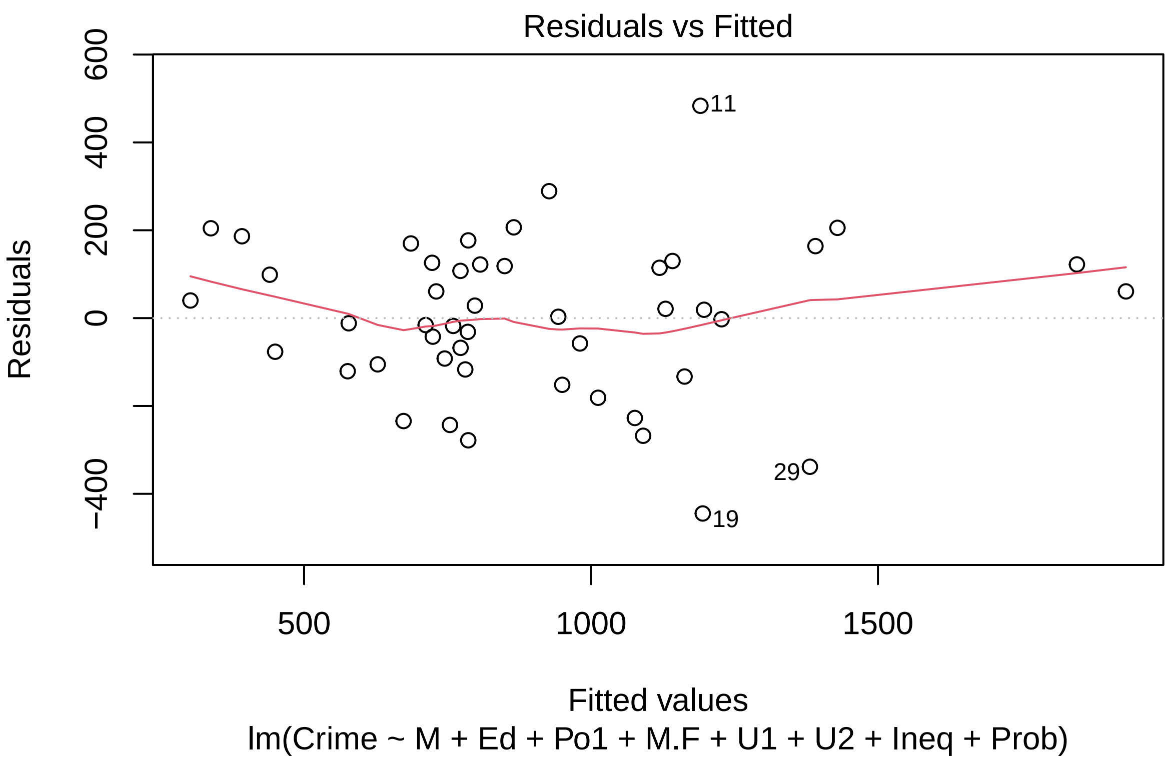
```
#took the last predictors from the last stepwise regression model
#this will hopefully tune the model more
final_model <- lm(Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob,
                  data = crime_data)
summary(final_model)
```

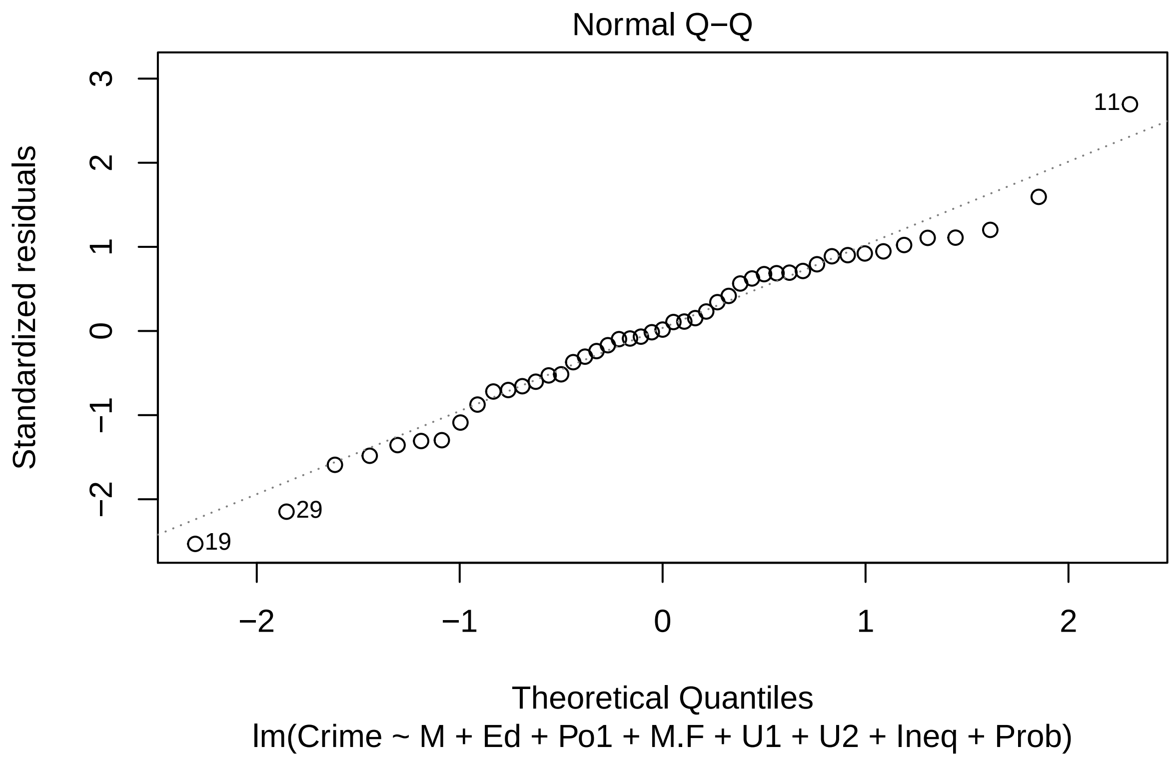
```
##
## Call:
## lm(formula = Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob,
##     data = crime_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -444.70 -111.07    3.03  122.15  483.30
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6426.10    1194.61  -5.379 4.04e-06 ***
## M              93.32      33.50   2.786  0.00828 **
## Ed            180.12      52.75   3.414  0.00153 **
## Po1           102.65      15.52   6.613 8.26e-08 ***
## M.F            22.34      13.60   1.642  0.10874
## U1          -6086.63    3339.27  -1.823  0.07622 .
## U2            187.35      72.48   2.585  0.01371 *
## Ineq           61.33      13.96   4.394 8.63e-05 ***
## Prob        -3796.03    1490.65  -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
```

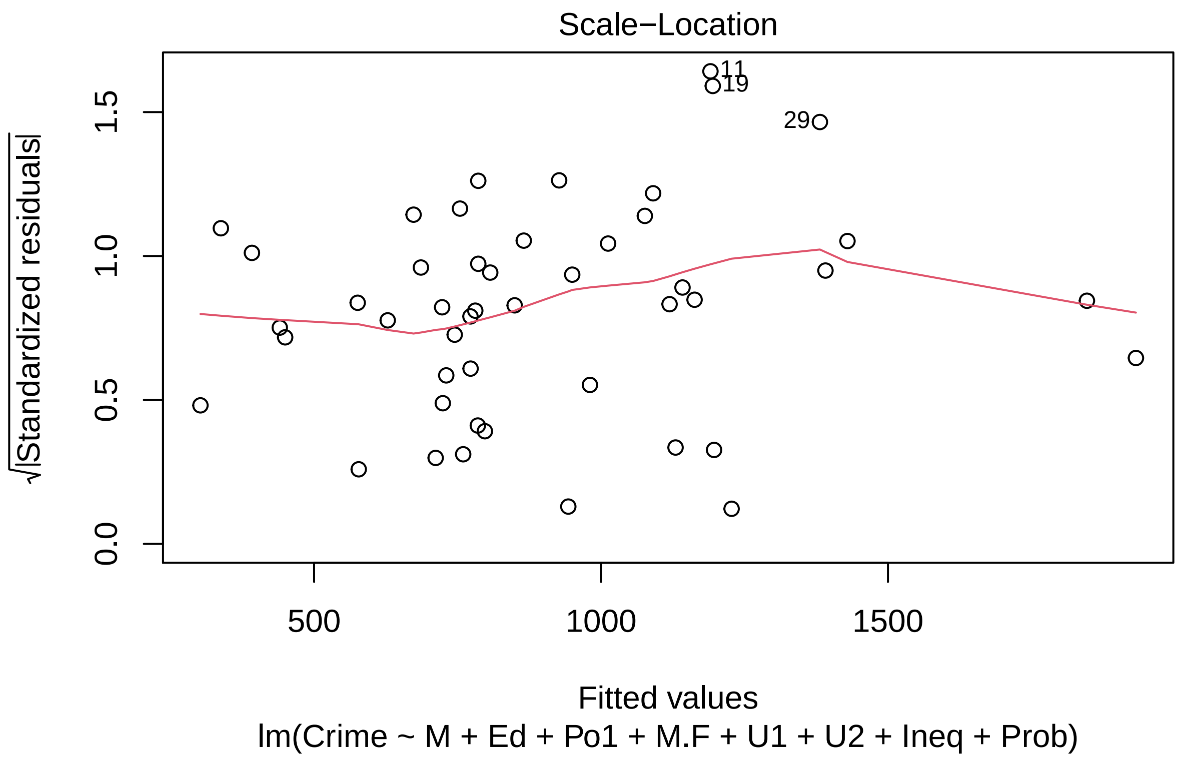
```r
AIC(final_model)
```

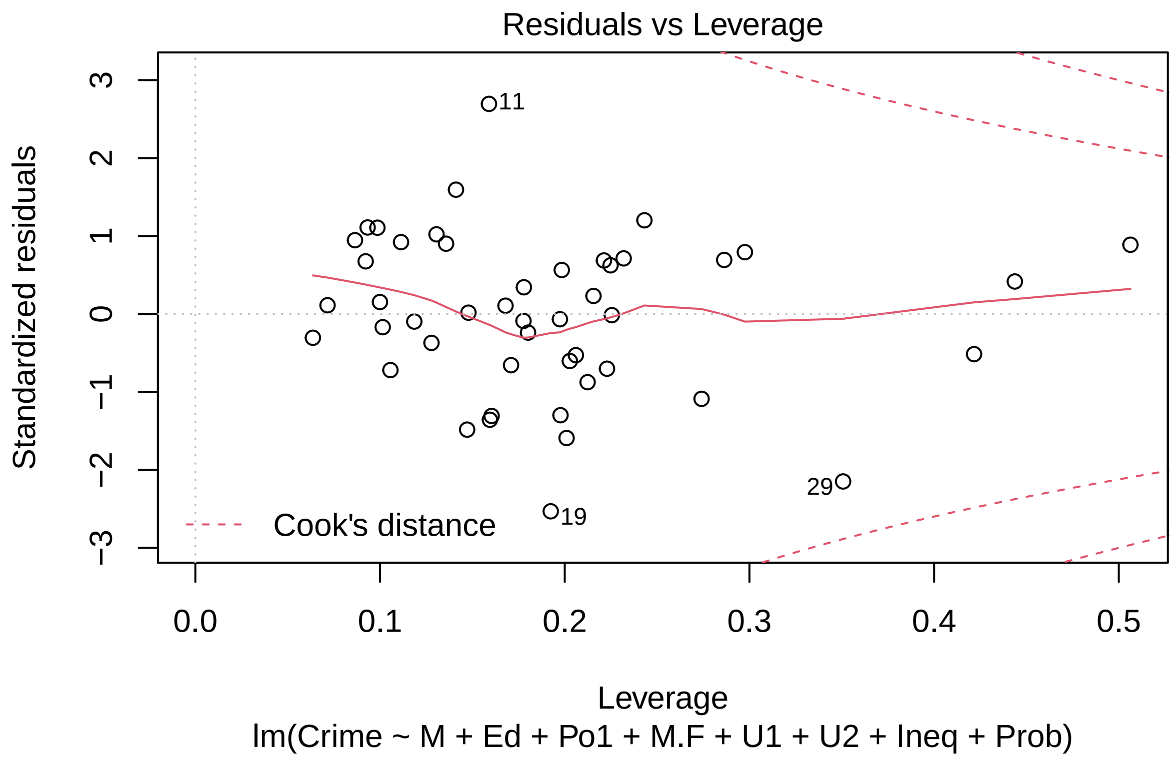
```
## [1] 639.3151
```

```r
plot(final_model)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob)
11
29
19
−2
−1
0
1
2
3


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
500
1000
1500
Fitted values
lm(Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob)
11
19
29



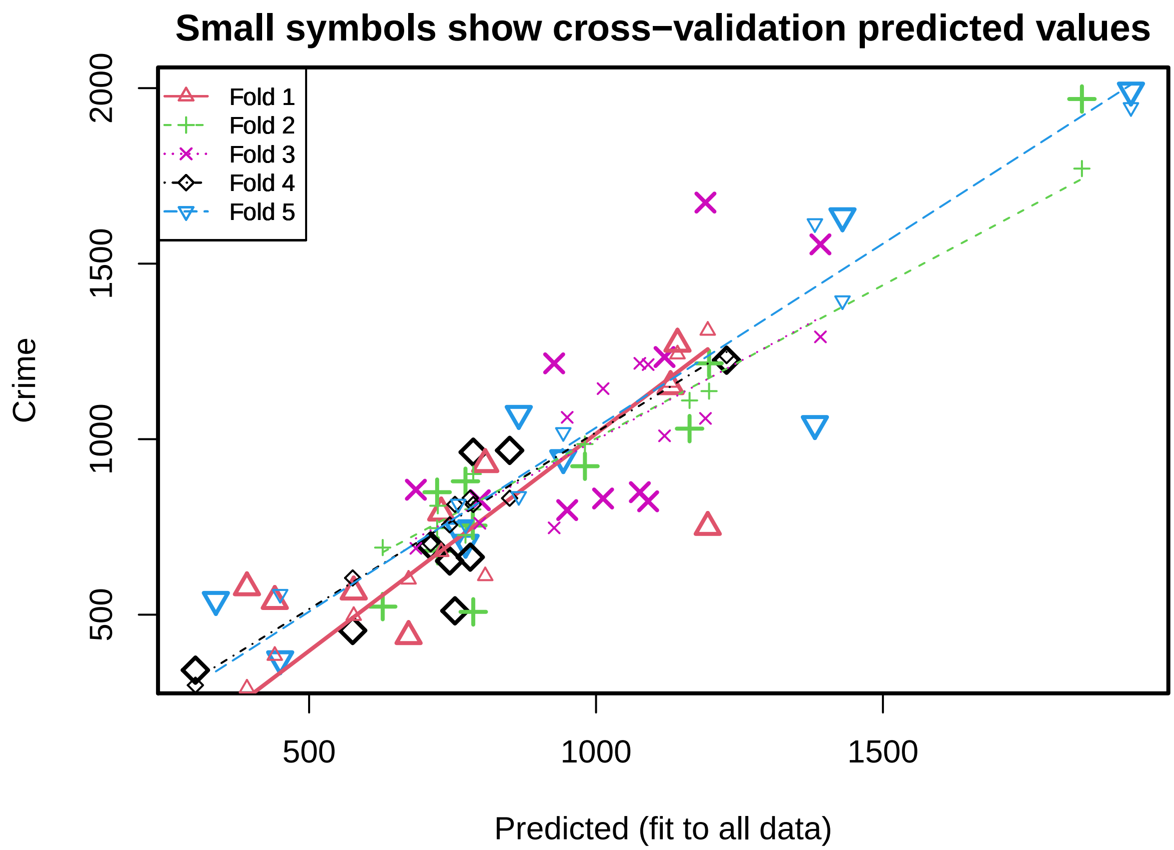
Stepwise regression is a combination of forward and backward elimination, so the scope needs to include both formulas. The direction is set to both since we are moving in either of the directions. I made a model using the most important predictors from the stepwise results. You can see from the plots that the assumption of normality is upheld because we meet all the conditions. The data follows the qq plot fairly well and the residuals are within boundry, so I'd say normally distrubted.

```
cv_model <- cv.lm(crime_data, final_model, m=5)
```

```
## Analysis of Variance Table
##
## Response: Crime
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## M          1   55084   55084    1.44 0.23748
## Ed         1  725967  725967   18.99 9.7e-05 ***
## Po1        1 3173852 3173852   83.00 4.3e-11 ***
## M.F        1  177521  177521    4.64 0.03759 *
## U1         1       4       4    0.00 0.99191
## U2         1  395014  395014   10.33 0.00267 **
## Ineq       1  652440  652440   17.06 0.00019 ***
## Prob       1  247978  247978    6.49 0.01505 *
## Residuals 38 1453068   38239
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## Warning in cv.lm(crime_data, final_model, m = 5):
##
```

```
##  As there is >1 explanatory variable, cross-validation
##  predicted values for a fold are not a linear function
##  of corresponding overall predicted values.  Lines that
##  are shown for the different folds are approximate
```




```
##
## fold 1
## Observations in test set: 9
##                1   3  17  18   19   22     36  38     40
## Predicted    730 392 440 807 1195  673 1142.0 578 1129.9
## cvpred       679 290 383 610 1309  601 1242.1 497 1161.3
## Crime        791 578 539 929  750  439 1272.0 566 1151.0
## CV residual 112 288 156 319 -559 -162   29.9  69  -10.3
##
## Sum of squares = 565964    Mean square = 62885    n = 9
##
## fold 2
## Observations in test set: 10
##                 4    6  12   25     28  32    34  41     44   46
## Predicted    1847  724 723  628 1197.0 785 980.7 772 1163.0  786
## cvpred       1771  810 747  691 1136.9 800 986.1 727 1110.3  901
## Crime        1969  682 849  523 1216.0 754 923.0 880 1030.0  508
## CV residual  198 -128 102 -168   79.1 -46 -63.1 153  -80.3 -393
##
## Sum of squares = 291189    Mean square = 29119    n = 10
```

```
##
## fold 3
## Observations in test set: 10
##                 5    8   9   11   15   23   37    39   43   47
## Predicted   1119 1391 686 1191  950  927 1012 797.6 1091 1076
## cvpred      1010 1291 689 1059 1062  747 1144 760.4 1212 1216
## Crime       1234 1555 856 1674  798 1216  831 826.0  823  849
## CV residual  224  264 167  615 -264  469 -313  65.6 -389 -367
##
## Sum of squares = 1203811    Mean square = 120381    n = 10
##
## fold 4
## Observations in test set: 9
##                7   13   14     20  24    27     30   35   45
## Predicted    786  754  781 1227.6 850 301.9 711.82  745  576
## cvpred       814  814  832 1238.1 832 299.2 702.06  756  605
## Crime        963  511  664 1225.0 968 342.0 696.00  653  455
## CV residual 149 -303 -168  -13.1 136  42.8  -6.06 -103 -150
##
## Sum of squares = 196039    Mean square = 21782    n = 9
##
## fold 5
## Observations in test set: 9
##                 2  10     16    21     26   29   31   33  42
## Predicted   1430 773  943.0 759.8 1932.2 1381  450  865 338
## cvpred      1395 757 1019.5 815.5 1945.3 1615  558  837 218
## Crime       1635 705  946.0 742.0 1993.0 1043  373 1072 542
## CV residual  240 -52  -73.5 -73.5   47.7 -572 -185  235 324
##
## Sum of squares = 594802    Mean square = 66089    n = 9
##
## Overall (Sum over all 9 folds)
##    ms
## 60677
```

```r
sse <- 60677 * nrow(crime_data)
## total sum of squares
sst <- sum((crime_data$Crime - mean(crime_data$Crime))^2)
# mean squared error
rsq <- 1 - sse / sst
rsq
```

```
## [1] 0.586
```

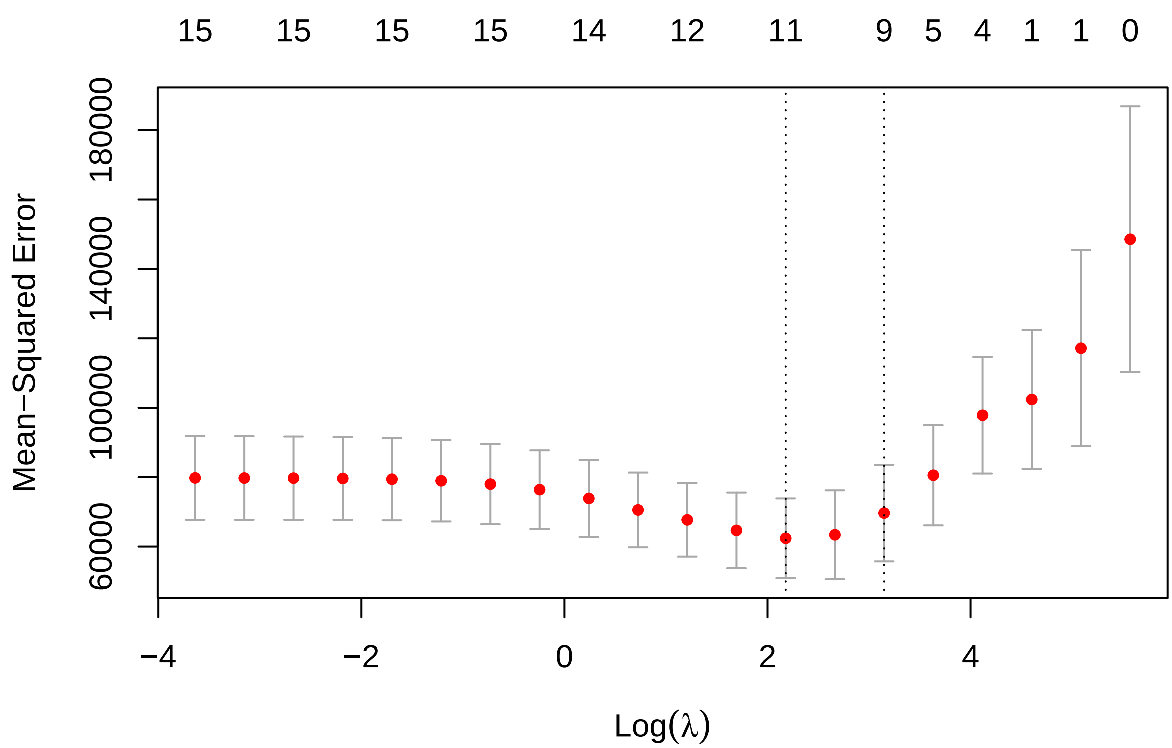
I cross validated the model to help with overfitting and then computed the R sqaured. The results of the R^2 was 0.586 which was lower than I expected and previous homeworks. Maybe I filtered the model too much, but it also could be the method we are using.

LASSO

```r
set.seed(42)

model_lasso <- cv.glmnet(x = as.matrix(s_crime_data[,-16]),
                         y = as.matrix(s_crime_data[,16]),
                         alpha=1, # 1 = Lasso method
                         nfolds = 8, #how many cv folds
                         nlambda = 20, #try different lambdas to tune model
                         type.measure = 'mse',
                         family = 'gaussian')
                         #standardize = TRUE) #scales the data

plot(model_lasso)
```




```r
model_lasso$lambda.min #8.839
```

```
## [1] 8.84
```

```r
cbind(model_lasso$lambda, model_lasso$cvm, model_lasso$nzero) #s7 has the smallest error and uses 11 pr
```

```
##          [,1]   [,2] [,3]
## s0   263.0954 148554    0
## s1   162.0268 117145    1
## s2    99.7839 102375    1
```

```
## s3   61.4518  97833    4
## s4   37.8450  80549    5
## s5   23.3067  69651    9
## s6   14.3534  63388   10
## s7    8.8395  62393   11
## s8    5.4438  64659   12
## s9    3.3526  67694   12
## s10   2.0647  70554   13
## s11   1.2715  73867   14
## s12   0.7831  76397   15
## s13   0.4822  77984   15
## s14   0.2970  78952   15
## s15   0.1829  79408   15
## s16   0.1126  79630   14
## s17   0.0694  79710   15
## s18   0.0427  79746   15
## s19   0.0263  79782   15
```

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

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                  1
## (Intercept) 889.88
## M            90.29
## So           44.66
## Ed          140.27
## Po1         304.14
## Po2            .
## LF             .
## M.F          55.64
## Pop            .
## NW            6.49
## U1          -38.65
## U2           74.62
## Wealth        7.44
## Ineq        194.79
## Prob        -83.86
## Time           .
```

We need to set the seed because the cv.glmnet function uses some randomization, so it lets us replicate the results. In the function alpha needs to set to 1 because that's the value for running LASSO method. If alpha was set to 0 then the function would run as Ridge Regression. The cbind function combines the output of our Lasso function, giving us the lambda values with MSE and predictors. The coef function shows the predictors with their minmum value for Lambda, so any zero values are not important.

```
sse2 <- 62392.77 * nrow(crime_data)
# mean squared error
rsq2 <- 1 - sse2 / sst
rsq2 #.57
```

```
## [1] 0.574
```

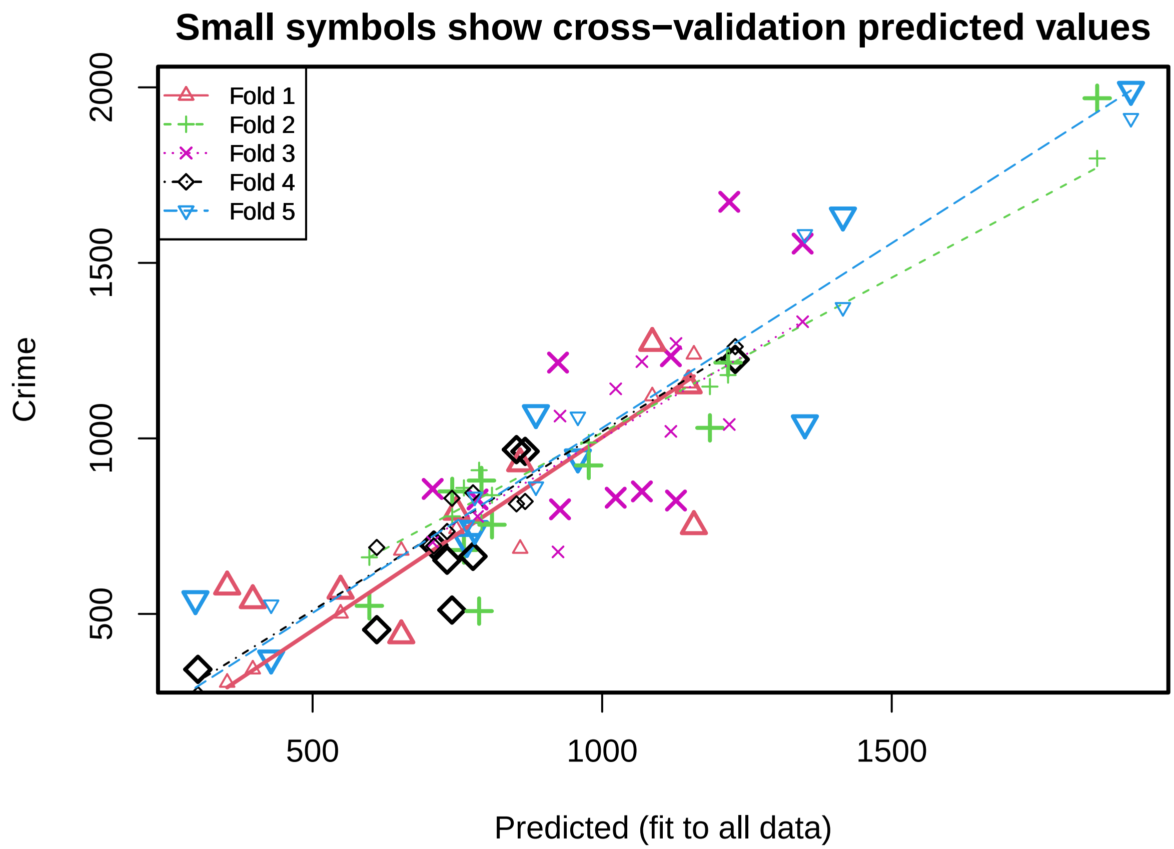
Next, we compute the R squared value for the Lasso model, which came out to be 0.57. About the same value as the Stepwise model so maybe our previous homeworks were overfitting since they had higher R^2 values. This is just a guess about why my values this week are lower than previous.

```
final_lasso <- lm(Crime ~ M + So+ Ed + Po1 + M.F + NW + U1 + U2 + Wealth + Ineq + Prob,
                  data = s_crime_data)

cv_model <- cv.lm(s_crime_data, final_lasso, m=5)
```

```
## Analysis of Variance Table
##
## Response: Crime
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## M          1   55084   55084    1.36 0.25144
## So         1   15370   15370    0.38 0.54188
## Ed         1  905668  905668   22.36 3.6e-05 ***
## Po1        1 3076033 3076033   75.94 2.7e-10 ***
## M.F        1  209271  209271    5.17 0.02927 *
## NW         1   23590   23590    0.58 0.45050
## U1         1     852     852    0.02 0.88551
## U2         1  314234  314234    7.76 0.00857 **
## Wealth     1   44977   44977    1.11 0.29922
## Ineq       1  626094  626094   15.46 0.00038 ***
## Prob       1  192052  192052    4.74 0.03627 *
## Residuals 35 1417704   40506
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## Warning in cv.lm(s_crime_data, final_lasso, m = 5):
##
##  As there is >1 explanatory variable, cross-validation
##  predicted values for a fold are not a linear function
##  of corresponding overall predicted values.  Lines that
##  are shown for the different folds are approximate
```




```
##
## fold 1
## Observations in test set: 9
##                  1   3  17  18   19   22   36    38      40
## Predicted    748.3 353 397 858 1158  653 1087 548.4 1149.38
## cvpred       742.2 304 342 686 1239  681 1120 501.5 1156.98
## Crime        791.0 578 539 929  750  439 1272 566.0 1151.00
## CV residual   48.8 274 197 243 -489 -242  152  64.5   -5.98
##
## Sum of squares = 5e+05    Mean square = 55568    n = 9
##
## fold 2
## Observations in test set: 10
##                 4    6  12   25     28    32    34  41   44   46
## Predicted    1855  761 741  598 1217.4 809.7 976.8 792 1186  788
## cvpred       1797  859 778  661 1180.4 838.2 988.2 760 1148  909
## Crime        1969  682 849  523 1216.0 754.0 923.0 880 1030  508
## CV residual   172 -177  71 -138   35.6 -84.2 -65.2 120 -118 -401
##
## Sum of squares = 286660    Mean square = 28666    n = 10
##
## fold 3
## Observations in test set: 10
##                 5    8   9   11   15   23   37    39   43   47
## Predicted    1119 1346 707 1219  927  924 1023 784.8 1128 1069
## cvpred       1020 1332 704 1040 1064  677 1141 777.7 1270 1219
```

```
## Crime       1234 1555 856 1674  798 1216  831 826.0  823  849
## CV residual  214  223 152  634 -266  539 -310  48.3 -447 -370
##
## Sum of squares = 1317792    Mean square = 131779    n = 10
##
## fold 4
## Observations in test set: 9
##               7   13   14     20  24  27     30    35   45
## Predicted   867  741  777 1229.8 852 302 709.30 732.2  611
## cvpred      820  829  845 1261.3 814 270 693.57 734.8  689
## Crime       963  511  664 1225.0 968 342 696.00 653.0  455
## CV residual 143 -318 -181  -36.3 154  72   2.43 -81.8 -234
##
## Sum of squares = 246428    Mean square = 27381    n = 9
##
## fold 5
## Observations in test set: 9
##                2    10   16    21     26   29   31   33  42
## Predicted   1416 766.5  958 780.1 1913.0 1350  428  886 298
## cvpred      1374 753.3 1062 833.6 1912.3 1581  526  863 145
## Crime       1635 705.0  946 742.0 1993.0 1043  373 1072 542
## CV residual  261 -48.3 -116 -91.6   80.7 -538 -153  209 397
##
## Sum of squares = 614063    Mean square = 68229    n = 9
##
## Overall (Sum over all 9 folds)
##    ms
## 63086
```

```r
sse2 <- 63086 * nrow(crime_data)
# mean squared error
rsq2 <- 1 - sse2 / sst
rsq2
```

```
## [1] 0.569
```

I decided to run a model where I took out the least important factors according to LASSO and run the linear model again. This time I got a lower value by about 0.02, which means I could have taken out a predicting factor or more than likely it's random error. The more I tried to optimize the model the worse the R squared value becomes so this value will suffice.

Elastic Net

```r
r2=c()
for (i in 0:100) {
  model.enet = cv.glmnet(x=as.matrix(s_crime_data[,-16]),
                         y=as.matrix(s_crime_data[,16]),
                         alpha=i/100,
                         nfolds = 5,
                         nlambda = 20,
                         type.measure="mse",
```

```
                            family="gaussian")
                            #standardize = TRUE)

  #dev.ratio is the percentage of deviance explained
  x = which(model.enet$glmnet.fit$lambda == model.enet$lambda.min)
  r2 = cbind(r2, model.enet$glmnet.fit$dev.ratio[x])
}
r2
```

```
##       [,1] [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12]
## [1,] 0.738 0.73 0.709 0.758 0.798 0.773 0.741 0.673 0.771 0.758 0.761 0.741
##      [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## [1,]  0.78 0.748  0.79 0.772 0.755 0.757 0.731 0.761 0.778 0.779  0.78 0.697
##      [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36]
## [1,] 0.781 0.743 0.769 0.746  0.77 0.771 0.772 0.785 0.751 0.774 0.753 0.775
##      [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48]
## [1,] 0.717 0.756 0.719 0.757 0.777 0.758 0.759 0.794 0.794 0.779 0.761 0.726
##      [,49] [,50] [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60]
## [1,] 0.727 0.781 0.763 0.764 0.782 0.765 0.765 0.792 0.766 0.784 0.784 0.767
##      [,61] [,62] [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72]
## [1,] 0.767 0.795 0.768 0.785 0.736 0.792  0.77 0.793 0.786 0.786 0.786 0.786
##      [,73] [,74] [,75] [,76] [,77] [,78] [,79] [,80] [,81] [,82] [,83] [,84]
## [1,]  0.74 0.771 0.796 0.772 0.772 0.772  0.69 0.793 0.743 0.743 0.773 0.793
##      [,85] [,86] [,87] [,88] [,89] [,90] [,91] [,92] [,93] [,94] [,95] [,96]
## [1,] 0.773 0.773 0.773 0.744 0.787 0.695 0.788 0.788 0.774 0.745 0.774 0.796
##      [,97] [,98] [,99] [,100] [,101]
## [1,] 0.788 0.794 0.746  0.774  0.699
```

```
best_alpha = (which.max(r2)-1)/100
best_alpha
```

```
## [1] 0.04
```

The last model that we needed to run was the Elastic Net model which is shown above. This model format is very close to LASSO except that we are concerned with the Alpha value. The for loop goes through 101 different values for alpha and computes the R sqaured value. I found the R squared computation online because I didn't undertand how to calculate it from the results. However, dev.ratio shows the % of deviance explained, which in the context of regression is equal to R squared. To get the best Alpha we want the one with the lowest R squared which we'll use to run the model again.

```
model_enet <- cv.glmnet(x = as.matrix(crime_data[,-16]),
                        y = as.matrix(crime_data[,16]),
                        alpha = best_alpha,
                        nfolds = 8,
                        type.measure = 'mse',
                        family = 'gaussian',
                        standardize = TRUE) #scales the data

model_enet$lambda.min #6.895
```

```
## [1] 35.9
```

```r
cbind(model_enet$lambda, model_enet$cvm, model_enet$nzero) #s45 has smallest error
```

```
##         [,1]   [,2] [,3]
## s0  6577.385 147334    0
## s1  5993.068 146889    2
## s2  5460.661 146039    2
## s3  4975.550 144690    2
## s4  4533.536 143164    2
## s5  4130.789 141582    2
## s6  3763.821 139947    4
## s7  3429.454 138105    4
## s8  3124.791 136084    4
## s9  2847.193 133995    4
## s10 2594.256 131849    5
## s11 2363.790 129718    6
## s12 2153.797 127649    6
## s13 1962.460 125602    7
## s14 1788.120 123571    7
## s15 1629.268 121617    7
## s16 1484.529 119691    7
## s17 1352.647 117834    7
## s18 1232.482 116036    9
## s19 1122.992 114261   10
## s20 1023.228 112532   11
## s21  932.327 110850   11
## s22  849.502 109111   12
## s23  774.035 107251   12
## s24  705.271 105325   13
## s25  642.617 103318   14
## s26  585.529 101214   14
## s27  533.512  98936   14
## s28  486.116  96636   15
## s29  442.931  94355   15
## s30  403.582  92115   15
## s31  367.729  89864   15
## s32  335.061  87682   15
## s33  305.295  85642   15
## s34  278.174  83721   15
## s35  253.461  81904   15
## s36  230.945  80231   15
## s37  210.428  78755   15
## s38  191.734  77386   15
## s39  174.701  76131   15
## s40  159.181  74982   15
## s41  145.040  73925   15
## s42  132.155  72941   15
## s43  120.415  72059   15
## s44  109.717  71265   15
## s45   99.970  70560   15
## s46   91.089  69921   15
## s47   82.997  69366   15
## s48   75.624  68930   15
## s49   68.906  68562   15
```

```
## s50   62.784  68280   15
## s51   57.207  68070   15
## s52   52.125  67920   15
## s53   47.494  67778   14
## s54   43.275  67641   14
## s55   39.430  67562   14
## s56   35.927  67529   14
## s57   32.736  67643   15
## s58   29.828  67804   15
## s59   27.178  67972   15
## s60   24.763  68058   15
## s61   22.564  68079   15
## s62   20.559  68215   14
## s63   18.733  68412   14
## s64   17.068  68636   14
## s65   15.552  68910   14
## s66   14.171  69197   14
## s67   12.912  69529   14
## s68   11.765  69869   15
## s69   10.719  70208   15
## s70    9.767  70548   15
## s71    8.900  70883   15
## s72    8.109  71189   15
## s73    7.389  71476   15
## s74    6.732  71798   15
## s75    6.134  72111   14
## s76    5.589  72396   15
## s77    5.093  72693   15
## s78    4.640  72959   15
## s79    4.228  73243   15
## s80    3.852  73511   15
## s81    3.510  73783   15
## s82    3.198  74049   15
## s83    2.914  74298   15
## s84    2.655  74536   15
## s85    2.419  74763   15
## s86    2.204  74979   15
## s87    2.009  75187   15
## s88    1.830  75390   15
## s89    1.668  75584   15
## s90    1.519  75777   15
## s91    1.384  76013   15
## s92    1.261  76204   15
## s93    1.149  76366   15
## s94    1.047  76524   15
## s95    0.954  76680   15
## s96    0.869  76826   15
## s97    0.792  76965   15
## s98    0.722  77091   15
## s99    0.658  77107   15
```

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

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
```

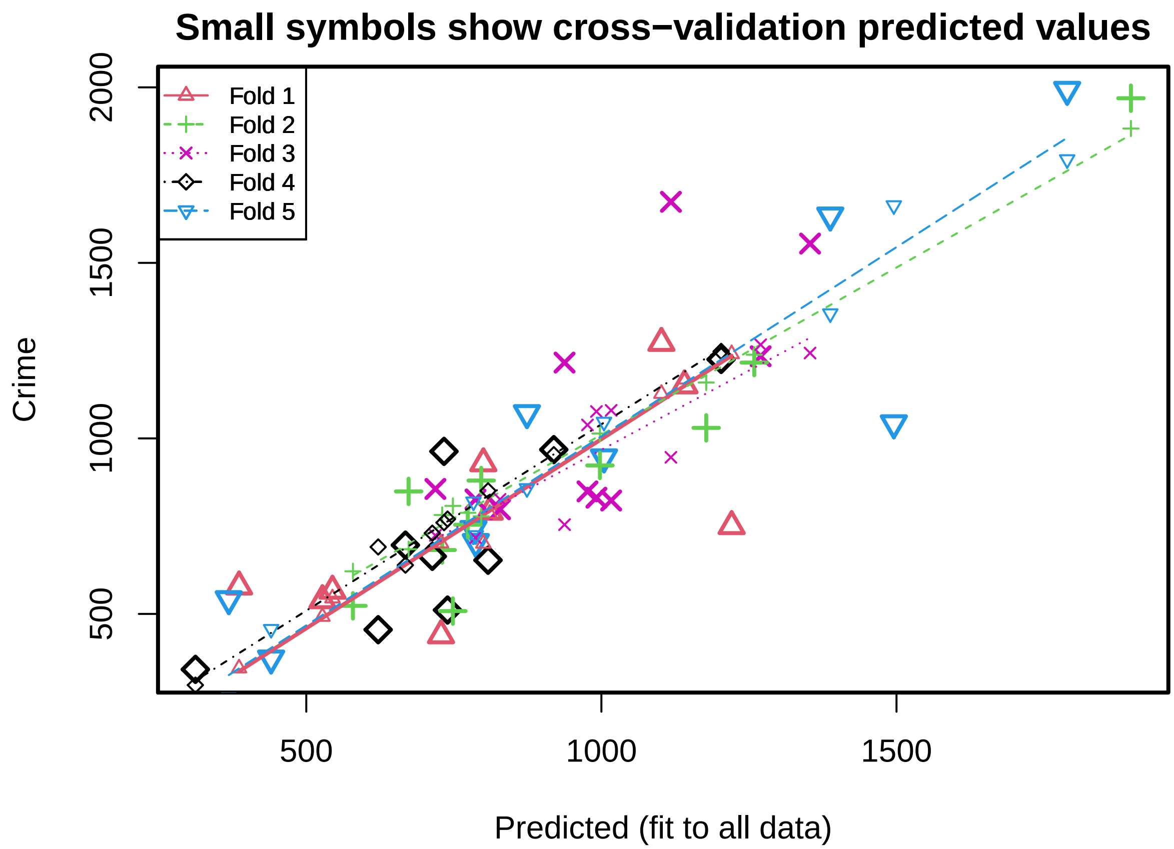
```
##                     1
## (Intercept) -5.22e+03
## M            6.55e+01
## So           8.37e+01
## Ed           9.92e+01
## Po1          5.20e+01
## Po2          3.87e+01
## LF           4.82e+02
## M.F          2.30e+01
## Pop          .
## NW           2.90e+00
## U1          -2.72e+03
## U2           1.02e+02
## Wealth       3.69e-02
## Ineq         3.78e+01
## Prob        -3.89e+03
## Time         2.42e-01
```

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

cv_model <- cv.lm(s_crime_data, final_enet, m=5)
```

```
## Analysis of Variance Table
##
## Response: Crime
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## M          1   55084   55084    1.37 0.24914
## Ed         1  725967  725967   18.02 0.00013 ***
## Po1        1 3173852 3173852   78.80 5.3e-11 ***
## U2         1  217386  217386    5.40 0.02534 *
## Ineq       1  848273  848273   21.06 4.3e-05 ***
## Prob       1  249308  249308    6.19 0.01711 *
## Residuals 40 1611057   40276
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## Warning in cv.lm(s_crime_data, final_enet, m = 5):
##
##  As there is >1 explanatory variable, cross-validation
##  predicted values for a fold are not a linear function
##  of corresponding overall predicted values.  Lines that
##  are shown for the different folds are approximate
```




```
##
## fold 1
## Observations in test set: 9
##                   1    3     17   18    19    22    36    38     40
## Predicted    810.83 386 527.4 800 1221   728 1102 544.4 1140.8
## cvpred       785.36 345 492.2 701 1240   702 1127 544.7 1168.2
## Crime        791.00 578 539.0 929  750   439 1272 566.0 1151.0
## CV residual    5.64 233  46.8 228 -490 -263  145  21.3  -17.2
##
## Sum of squares = 439507    Mean square = 48834    n = 9
##
## fold 2
## Observations in test set: 10
##                   4     6   12     25     28   32     34   41    44   46
## Predicted    1897.2 730.3 673 579.1 1259.0 774  997.5 796 1178  748
## cvpred       1882.7 781.8 684 621.4 1238.3 788 1013.9 778 1159  808
## Crime        1969.0 682.0 849 523.0 1216.0 754  923.0 880 1030  508
## CV residual    86.3 -99.8 165 -98.4  -22.3 -34  -90.9 102 -129 -300
##
## Sum of squares = 181038    Mean square = 18104    n = 10
##
## fold 3
## Observations in test set: 10
##                   5    8   9   11    15    23   37   39   43   47
## Predicted    1269.8 1354 719 1118 828.3  938  992 787 1017  976
## cvpred       1266.8 1243 724  946 826.3  754 1077 717 1080 1038
```

```
## Crime       1234.0 1555 856 1674 798.0 1216  831 826  823  849
## CV residual  -32.8  312 132  728 -28.3  462 -246 109 -257 -189
##
## Sum of squares = 1033612    Mean square = 103361    n = 10
##
## fold 4
## Observations in test set: 9
##                7   13    14     20    24    27    30   35   45
## Predicted   733  739 713.6 1203.0 919.4 312.2 668.0  808  622
## cvpred      760  770 730.1 1247.9 953.7 297.2 638.9  851  691
## Crime       963  511 664.0 1225.0 968.0 342.0 696.0  653  455
## CV residual 203 -259 -66.1  -22.9  14.3  44.8  57.1 -198 -236
##
## Sum of squares = 213398    Mean square = 23711    n = 9
##
## fold 5
## Observations in test set: 9
##                2    10   16    21   26   29    31   33  42
## Predicted   1388 787.3 1004 783.3 1789 1495 440.4  874 369
## cvpred      1356 723.7 1047 819.7 1795 1664 456.6  858 261
## Crime       1635 705.0  946 742.0 1993 1043 373.0 1072 542
## CV residual  279 -18.7 -101 -77.7  198 -621 -83.6  214 281
##
## Sum of squares = 650990    Mean square = 72332    n = 9
##
## Overall (Sum over all 9 folds)
##    ms
## 53586
```

```r
sse3 <- 53586 * nrow(crime_data)
# mean squared error
rsq3 <- 1 - sse3 / sst
rsq3
```

```
## [1] 0.634
```

We plug in the Alpha with the lowest R squared back into the equation and run it one last time. Then we are just repeating the steps again and computing the R sqaured to see how it compares. Elastic Net gives us the best R squared of the three models at 0.634, which is a big improvement compared to the other two. In my limited experience I have noticed that Elastic Net in general performs consistently better than most models. It would be interesting to combine this method with PCA.