In [1]:

```
#reading the data
data <- read.table('./uscrime.txt', stringsAsFactors = F, header = T)
```

In [2]:

```
#printing the head of the data
head(data)
```

M	So	Ed	Po1	Po2	LF	M.F	Pop	NW	U1	U2	Wealth	Ineq	Prob
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
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
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
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
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
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

In [3]:

```
#scaling the data

#keeping the So variable which is a factor out of scaling. Also, keeping the dep
endent variable out.

drops <- c("So","Crime")
df <- data[ , !(names(data) %in% drops)]
scaled_data <- scale(df)

#binding the data altogether.
scaled_data <- cbind(scaled_data, data[,drops])
```

In [4]:

```
head(scaled_data)
```

M	Ed	Po1	Po2	LF	M.F	Pop
0.9886930	-1.3085099	-0.9085105	-0.8666988	-1.2667456	-1.12060499	-0.09500679
0.3521372	0.6580587	0.6056737	0.5280852	0.5396568	0.98341752	-0.62033844
0.2725678	-1.4872888	-1.3459415	-1.2958632	-0.6976051	-0.47582390	-0.48900552
-0.2048491	1.3731746	2.1535064	2.1732150	0.3911854	0.37257228	3.16204944
0.1929983	1.3731746	0.8075649	0.7426673	0.7376187	0.06714965	-0.48900552
-1.3983912	0.3898903	1.1104017	1.2433590	-0.3511718	-0.64550313	-0.30513945

In [5]:

```
options(warn=-1)
library(caret)
library(leaps)
library(MASS)
library(glmnet)
```

```
Loading required package: lattice
Loading required package: ggplot2
Loading required package: Matrix
Loading required package: foreach
Loaded glmnet 2.0-13
```

In [6]:

```
set.seed(1)
```

References:

http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/154-stepwise-regression-essentials-in-r/ (http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/154-stepwise-regression-essentials-in-r/)

In [7]:

```
#defining a control method to do repeated cv

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

#here we use the train function from caret package. method = "leapSeq" is for st
epwise regression.
#tuneGrid = data.frame(nvmax = 1:15) means that it will try all the models with
1 features upto 15 features.
#the best 1-variable model, the best 2-variables model, …, the best 15-variables
model

lm_step <- train(Crime ~., data = scaled_data, method = "leapSeq",
                     tuneGrid = data.frame(nvmax = 1:15), trControl = control)
```

In [8]:

```
#here we print the stats for all nvmax best models

lm_step$results
```

nvmax	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
1	282.9300	0.4999792	226.2151	70.67819	0.2125575	53.74191
2	310.7000	0.4334649	248.3198	86.46990	0.2741488	72.16826
3	273.1027	0.5310749	214.0348	74.52462	0.2073685	61.93288
4	298.4097	0.4655277	236.8706	53.48436	0.1921420	42.46384
5	289.2306	0.4869785	233.2909	62.55661	0.2145011	56.34413
6	279.3289	0.5446330	221.5333	80.05025	0.2078361	68.68278
7	286.6147	0.5283167	224.7424	71.09796	0.1956827	64.51009
8	286.9273	0.4869795	227.3073	60.09801	0.2087288	54.08324
9	299.6539	0.4934172	231.2610	63.50853	0.1798948	59.63639
10	292.4513	0.5150836	228.5901	66.06529	0.1939786	55.64956
11	297.3778	0.4640345	231.5788	60.65984	0.1975678	57.45597
12	285.4662	0.5157689	224.6591	60.03197	0.2114072	57.48195
13	282.2549	0.5349635	222.1849	61.77533	0.2103352	53.76748
14	286.1901	0.5362130	224.7453	64.94866	0.1970353	57.80696
15	292.0083	0.5197076	229.6161	61.32444	0.2019892	56.76443

In [9]:

```
#it can be seen that the best model is one with 3 features. Also, good point to
note
# different seed values give different nvmax. When I set seed = 42, I got nvmax
= 6
lm_step$bestTune
```

	nvmax
3	3

In [10]:

```
#printing the summary to know the 3 features

summary(lm_step$finalModel)
```

```
Subset selection object
15 Variables  (and intercept)
       Forced in Forced out
M          FALSE      FALSE
Ed         FALSE      FALSE
Po1        FALSE      FALSE
Po2        FALSE      FALSE
LF         FALSE      FALSE
M.F        FALSE      FALSE
Pop        FALSE      FALSE
NW         FALSE      FALSE
U1         FALSE      FALSE
U2         FALSE      FALSE
Wealth     FALSE      FALSE
Ineq       FALSE      FALSE
Prob       FALSE      FALSE
Time       FALSE      FALSE
So         FALSE      FALSE
1 subsets of each size up to 3
Selection Algorithm: 'sequential replacement'
         M   Ed  Po1 Po2 LF  M.F Pop NW  U1  U2  Wealth Ineq Prob Ti
me So
1  ( 1 ) " " " " "*" " " " " " " " " " " " " " " " "    " "  " "  "
"  " "
2  ( 1 ) " " " " "*" " " " " " " " " " " " " " " " "    "*"  " "  "
"  " "
3  ( 1 ) " " "*" "*" " " " " " " " " " " " " " " " "    "*"  " "  "
"  " "
```

In [11]:

```
# * indicates that the features was included in the model.
# so the features selected are Ed, Po1, Ineq
#Using those features to fit the linear regression model
```

In [12]:

```
#fitting simple linear regression model on those 3 features

mod <- lm(Crime ~ Ed + Po1 + Ineq, data = scaled_data)
summary(mod)
```

```
Call:
lm(formula = Crime ~ Ed + Po1 + Ineq, data = scaled_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-590.30 -102.06   -1.73  129.16  511.60 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   905.09      33.74  26.825  < 2e-16 ***
Ed            176.61      53.32   3.312  0.00188 ** 
Po1           369.45      43.94   8.408 1.26e-10 ***
Ineq          299.45      60.15   4.978 1.09e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 231.3 on 43 degrees of freedom
Multiple R-squared:  0.6656,	Adjusted R-squared:  0.6423 
F-statistic: 28.53 on 3 and 43 DF,  p-value: 2.59e-10
```

In [13]:

```
#all the coefficients are significant as per the p-value.
```

In [14]:

```
#now lets perform leave out one cross validation and get the R squared.
```

In [15]:

```
test <- numeric()
for (i in 1:nrow(scaled_data)){
    model <- lm(Crime ~ Ed + Po1 +Ineq, data = scaled_data[-i,])
    test <- cbind(test, predict(model, newdata = scaled_data[i,]))
}
```

In [16]:

```
#function to calculate r-squared

compute_rsquared <- function(y_hat, y){
SSR <- sum((y_hat - y)^2)
SST <- sum((y - mean(y))^2)
rsquared <- 1 - SSR/SST
return (rsquared)
}
```

In [17]:

```
compute_rsquared(test, scaled_data$Crime)
```

0.574751286403875

In [18]:

```
#Using just three features we were able to get Rsquared of 0.57. Please note --
adding more features will
#definitely increase Rsquared. But we need to strike a balance between simplicit
y and generalization.
#i was able to get a pretty simple model using just three features.
```

In [19]:

```
full.model <- lm(Crime ~., data = scaled_data)
# Stepwise regression model
step.model <- stepAIC(full.model, direction = "both",
                      trace = FALSE)
summary(step.model)
```

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

In [20]:

```
#So the above model selected 8 features and was able to get a rsquared of 0.7888
#and adjusted r-squared of 0.7444.
```

In [21]:

```
#using the above model... lets cross validate
test1 <- numeric()
for (i in 1:nrow(scaled_data)){
    model <- lm(Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob, data =
scaled_data[-i,])
    test1 <- cbind(test1, predict(model, newdata = scaled_data[i,]))
}
```

In [22]:

```
compute_rsquared(test1, scaled_data$Crime)
```

0.667620969502124

In [23]:

```
#But it can be seen that M.F and U1 are not that siginificant
#as per p-value so lets take that out.

#using the above model... lets cross validate
test2 <- numeric()
for (i in 1:nrow(scaled_data)){
    model <- lm(Crime ~ M + Ed + Po1 + U2 + Ineq + Prob, data = scaled_data[-i,]
)
    test2 <- cbind(test2, predict(model, newdata = scaled_data[i,]))
}
```

In [24]:

```
compute_rsquared(test2, scaled_data$Crime)
```

0.666163842867471

In [25]:

```
#So it can be seen that, we were able to get about the same r-squared by omittin
g M.F and U1.
#Occam's razor -- the simpler the better.
#Hence we would go with the latter one with just 6 features.
```

Now it is quite a judgement call here. We have two models one with 3 features and other with 6 features. The final selection totally depends upon whether we want simplicity or generalization.

Lasso Regression

In [32]:

```
lasso_model = cv.glmnet(x = as.matrix(scaled_data[,-16]),
                        y = as.matrix(scaled_data$Crime),
                        alpha = 1, nfolds = 5,
                        type.measure = "mse", family = "gaussian", standardize =
F)
```

In [33]:

```
lasso_model
```

```
$lambda
 [1] 260.2814612 237.1587736 216.0902418 196.8933803 179.4019150 163
.4643433
 [7] 148.9426216 135.7109696 123.6547811 112.6696312 102.6603717  93
.5403072
[13]  85.2304441  77.6588063  70.7598120  64.4737054  58.7460391  53
```

```
.5272029
[19]  48.7719937  44.4392242  40.4913660  36.8942246  33.6166434  30
.6302335
[25]  27.9091279  25.4297579  23.1706483  21.1122318  19.2366793  17
.5277457
[31]  15.9706291  14.5518424  13.2590969  12.0811952  11.0079352  10
.0300205
[37]   9.1389812   8.3270993   7.5873427   6.9133041   6.2991452   5
.7395465
[43]   5.2296610   4.7650723   4.3417565   3.9560468   3.6046025   3
.2843795
[49]   2.9926043   2.7267496   2.4845127   2.2637954   2.0626861   1
.8794427
[55]   1.7124782   1.5603464   1.4217295   1.2954270   1.1803448   1
.0754862
[61]   0.9799430   0.8928876   0.8135659   0.7412909   0.6754367   0
.6154328
[67]   0.5607594   0.5109431   0.4655523   0.4241939   0.3865097   0
.3521733
[73]   0.3208872   0.2923804   0.2664061

$cvm
 [1] 145826.80 142162.52 136720.36 128986.49 122615.74 117372.22 113
030.77
 [8] 109190.33 105894.32 103045.85 100968.76  99444.38  98316.42  97
599.12
[15]  96893.45  95717.79  93560.54  90268.30  87180.21  84856.90  82
851.73
[22]  81247.48  79741.87  78108.00  75845.12  73630.84  71379.19  68
963.85
[29]  67177.64  65867.71  64735.74  63927.27  63312.68  62864.51  62
936.45
[36]  63170.13  63362.86  63288.38  63349.75  63406.67  63468.32  63
593.58
[43]  63757.60  63946.58  64161.48  64412.40  64726.41  65256.90  65
993.48
[50]  66685.82  67208.62  67754.40  68262.80  68796.78  69325.67  69
780.42
[57]  70227.05  70660.63  71069.55  71459.47  71815.03  72174.90  72
513.98
[64]  72842.40  73135.52  73405.11  73658.93  73885.81  74099.19  74
296.06
[71]  74478.23  74629.45  74772.41  74904.31  75025.78

$cvsd
 [1] 30833.26 32189.43 31448.90 28670.62 26213.00 24057.37 22171.84
20419.36
 [9] 18899.04 17610.39 16619.38 15841.90 15263.54 14922.87 14824.25
14835.13
[17] 14712.28 14635.46 14689.81 14856.27 15011.28 15155.07 15400.04
15622.36
[25] 15237.15 14875.81 14449.95 14082.18 13791.42 13556.77 13392.32
13228.64
```

```
[33] 13080.19 12988.30 13105.46 13257.97 13303.73 13298.18 13271.08
13253.50
[41] 13247.52 13236.88 13226.06 13213.93 13199.09 13177.26 13133.24
13091.20
[49] 13113.98 13142.55 13054.33 13032.83 12990.46 12968.60 12939.69
12939.30
[57] 12945.52 12942.86 12949.56 12954.77 12961.84 12981.55 13016.57
13065.06
[65] 13109.91 13151.54 13193.91 13239.68 13279.49 13315.02 13351.15
13378.89
[73] 13403.56 13429.20 13451.80

$cvup
 [1] 176660.07 174351.95 168169.26 157657.11 148828.74 141429.58 135
202.61
 [8] 129609.69 124793.36 120656.24 117588.13 115286.28 113579.96 112
522.00
[15] 111717.70 110552.92 108272.82 104903.75 101870.02  99713.16  97
863.01
[22]  96402.55  95141.91  93730.37  91082.27  88506.66  85829.14  83
046.04
[29]  80969.06  79424.48  78128.06  77155.91  76392.87  75852.80  76
041.91
[36]  76428.10  76666.59  76586.56  76620.83  76660.17  76715.85  76
830.46
[43]  76983.66  77160.50  77360.57  77589.66  77859.64  78348.10  79
107.46
[50]  79828.37  80262.95  80787.23  81253.26  81765.39  82265.35  82
719.72
[57]  83172.57  83603.48  84019.12  84414.24  84776.86  85156.45  85
530.54
[64]  85907.46  86245.43  86556.66  86852.84  87125.49  87378.68  87
611.07
[71]  87829.37  88008.34  88175.97  88333.50  88477.58

$cvlo
 [1] 114993.54 109973.09 105271.47 100315.88  96402.73  93314.85  90
858.93
 [8]  88770.97  86995.29  85435.46  84349.38  83602.48  83052.89  82
676.25
[15]  82069.20  80882.65  78848.26  75632.84  72490.40  70000.63  67
840.44
[22]  66092.41  64341.83  62485.64  60607.98  58755.03  56929.25  54
881.67
[29]  53386.23  52310.94  51343.41  50698.63  50232.49  49876.21  49
830.99
[36]  49912.16  50059.13  49990.20  50078.68  50153.18  50220.80  50
356.70
[43]  50531.55  50732.65  50962.39  51235.14  51593.17  52165.69  52
879.50
[50]  53543.27  54154.29  54721.57  55272.34  55828.18  56385.98  56
841.13
[57]  57281.53  57717.77  58119.99  58504.70  58853.19  59193.35  59
```

```
497.41
[64] 59777.33 60025.60 60253.57 60465.01 60646.12 60819.70 60
981.04
[71] 61127.08 61250.57 61368.85 61475.11 61573.99

$nzero
 s0  s1  s2  s3  s4  s5  s6  s7  s8  s9 s10 s11 s12 s13 s14 s15 s16
s17 s18 s19
  0   1   1   1   1   1   1   1   1   1   1   1   1   1   3   4   4
5   5   5
s20 s21 s22 s23 s24 s25 s26 s27 s28 s29 s30 s31 s32 s33 s34 s35 s36
s37 s38 s39
  5   5   5   6   7   8   8   8   8   8   8   9   9   9   9  10  10
10  11  11
s40 s41 s42 s43 s44 s45 s46 s47 s48 s49 s50 s51 s52 s53 s54 s55 s56
s57 s58 s59
 11  11  11  11  11  11  11  12  12  12  13  13  13  14  14  14  14
14  15  15
s60 s61 s62 s63 s64 s65 s66 s67 s68 s69 s70 s71 s72 s73 s74
 15  15  15  15  15  15  15  15  15  15  15  15  14  14  14

$name
                 mse
"Mean-Squared Error"

$glmnet.fit

Call:  glmnet(x = as.matrix(scaled_data[, -16]), y = as.matrix(scale
d_data$Crime),      alpha = 1, family = "gaussian", standardize = F)

      Df    %Dev    Lambda
 [1,]  0 0.00000 260.30000
 [2,]  1 0.08027 237.20000
 [3,]  1 0.14690 216.10000
 [4,]  1 0.20220 196.90000
 [5,]  1 0.24820 179.40000
 [6,]  1 0.28630 163.50000
 [7,]  1 0.31800 148.90000
 [8,]  1 0.34430 135.70000
 [9,]  1 0.36610 123.70000
[10,]  1 0.38420 112.70000
[11,]  1 0.39920 102.70000
[12,]  1 0.41170  93.54000
[13,]  1 0.42210  85.23000
[14,]  1 0.43070  77.66000
[15,]  3 0.44240  70.76000
[16,]  4 0.45870  64.47000
[17,]  4 0.48700  58.75000
[18,]  5 0.52490  53.53000
[19,]  5 0.55650  48.77000
[20,]  5 0.58260  44.44000
[21,]  5 0.60430  40.49000
[22,]  5 0.62240  36.89000
```

```
[23,]  5 0.63730  33.62000
[24,]  6 0.64980  30.63000
[25,]  7 0.66700  27.91000
[26,]  8 0.68230  25.43000
[27,]  8 0.69750  23.17000
[28,]  8 0.71000  21.11000
[29,]  8 0.72040  19.24000
[30,]  8 0.72900  17.53000
[31,]  8 0.73610  15.97000
[32,]  9 0.74290  14.55000
[33,]  9 0.75080  13.26000
[34,]  9 0.75730  12.08000
[35,]  9 0.76270  11.01000
[36,] 10 0.76790  10.03000
[37,] 10 0.77220   9.13900
[38,] 10 0.77580   8.32700
[39,] 11 0.77930   7.58700
[40,] 11 0.78230   6.91300
[41,] 11 0.78480   6.29900
[42,] 11 0.78680   5.74000
[43,] 11 0.78850   5.23000
[44,] 11 0.79000   4.76500
[45,] 11 0.79110   4.34200
[46,] 11 0.79210   3.95600
[47,] 11 0.79290   3.60500
[48,] 12 0.79360   3.28400
[49,] 12 0.79420   2.99300
[50,] 12 0.79470   2.72700
[51,] 13 0.79510   2.48500
[52,] 13 0.79550   2.26400
[53,] 13 0.79580   2.06300
[54,] 14 0.79610   1.87900
[55,] 14 0.79630   1.71200
[56,] 14 0.79650   1.56000
[57,] 14 0.79670   1.42200
[58,] 14 0.79690   1.29500
[59,] 15 0.79710   1.18000
[60,] 15 0.79800   1.07500
[61,] 15 0.79880   0.97990
[62,] 15 0.79950   0.89290
[63,] 15 0.80010   0.81360
[64,] 15 0.80060   0.74130
[65,] 15 0.80100   0.67540
[66,] 15 0.80130   0.61540
[67,] 15 0.80160   0.56080
[68,] 15 0.80180   0.51090
[69,] 15 0.80200   0.46560
[70,] 15 0.80220   0.42420
[71,] 15 0.80230   0.38650
[72,] 15 0.80240   0.35220
[73,] 14 0.80250   0.32090
[74,] 14 0.80260   0.29240
[75,] 14 0.80270   0.26640
```

```
[76,] 14 0.80270   0.24270
[77,] 14 0.80280   0.22120
[78,] 14 0.80280   0.20150
[79,] 14 0.80290   0.18360
[80,] 14 0.80290   0.16730
[81,] 14 0.80290   0.15240
[82,] 14 0.80290   0.13890
[83,] 14 0.80300   0.12660
[84,] 14 0.80300   0.11530
[85,] 14 0.80300   0.10510
[86,] 14 0.80300   0.09574
[87,] 14 0.80300   0.08724
[88,] 14 0.80300   0.07949

$lambda.min
[1] 12.0812

$lambda.1se
[1] 27.90913

attr(,"class")
[1] "cv.glmnet"
```

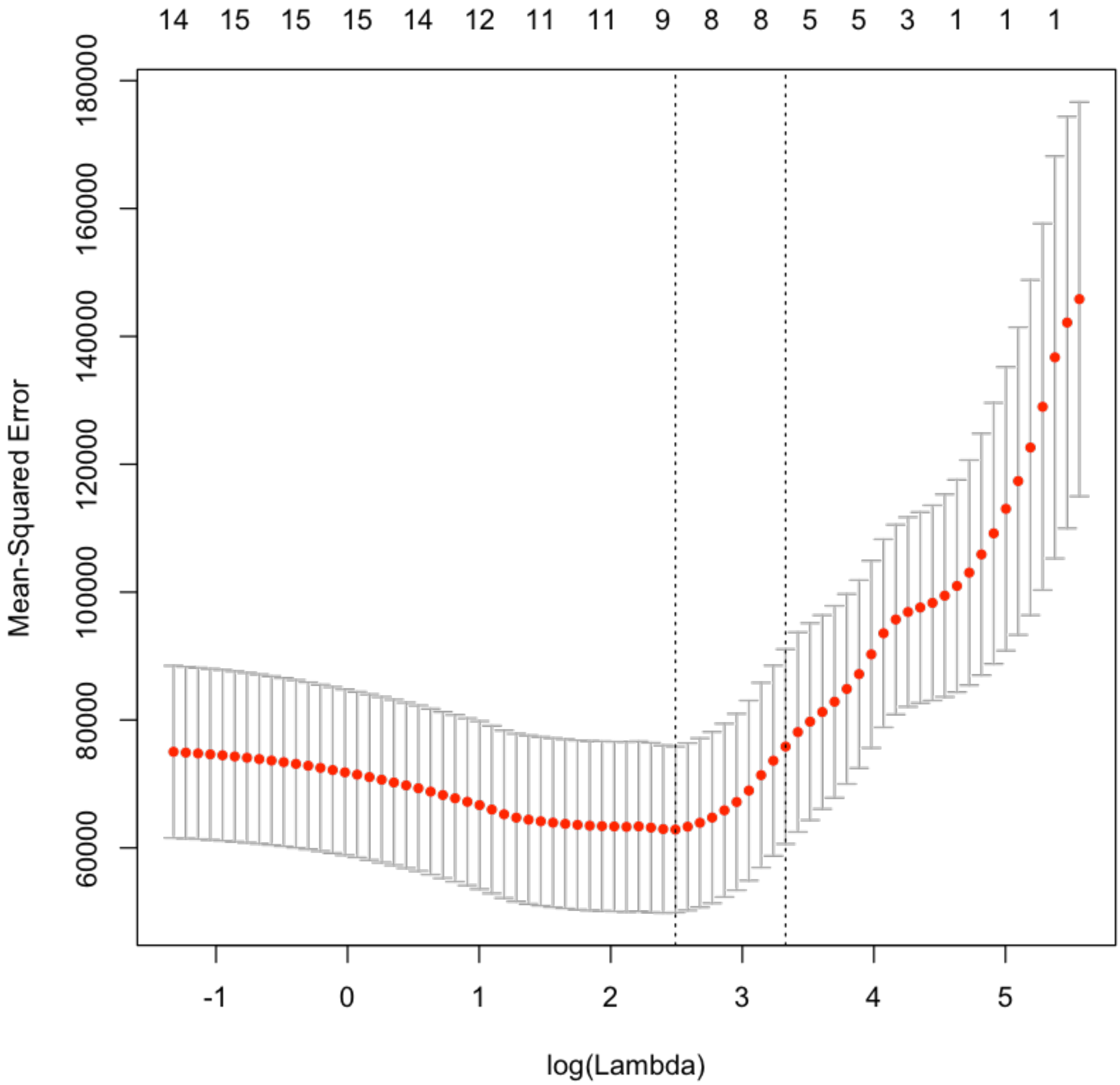
In [34]:

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

```
16 x 1 sparse Matrix of class "dgCMatrix"
                    1
(Intercept) 905.08511
M            85.09006
Ed          115.31161
Po1         309.89409
Po2           .
LF            .
M.F          50.46511
Pop           .
NW           10.57842
U1          -21.05244
U2           52.84583
Wealth        .
Ineq        182.41508
Prob        -75.82115
Time          .
So            .
```

In [35]:

```
plot(lasso_model)
```



In [36]:

```
#fitting a model with the above 9 variables. These variables are same to stepAIC

lm_lasso <- lm(Crime ~ M + Ed + Po1 + M.F + NW + U1 + U2 + Ineq + Prob, data =
scaled_data)
```

In [37]:

```
summary(lm_lasso)
```

```
Call:
lm(formula = Crime ~ M + Ed + Po1 + M.F + NW + U1 + U2 + Ineq +
    Prob, data = scaled_data)

Residuals:
   Min     1Q Median     3Q    Max
-439.2 -102.2   -6.3  124.1  476.6

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   905.09      28.87  31.352  < 2e-16 ***
M             111.23      46.83   2.375 0.022820 *
Ed            203.63      60.12   3.387 0.001687 **
Po1           297.89      52.08   5.719 1.51e-06 ***
M.F            68.74      41.63   1.651 0.107134
NW             16.55      53.15   0.311 0.757222
U1           -109.46      60.94  -1.796 0.080609 .
U2            156.94      62.09   2.528 0.015889 *
Ineq          236.70      61.95   3.821 0.000492 ***
Prob          -89.99      36.28  -2.481 0.017791 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 197.9 on 37 degrees of freedom
Multiple R-squared:  0.7894,    Adjusted R-squared:  0.7381
F-statistic: 15.41 on 9 and 37 DF,  p-value: 4.881e-10
```

Point to note -- different value of seeds gives different results. One more point to note, the significant variables from the above model are same as the variables selected from step AIC. So I will not rerun the lm model on those 6 variables.

Elastic Net

In [102]:

```
#we loop over different values of alpha. In each loop we select, minimum cross v
alidation error and lambda.min

mse_list <- numeric()
find_alpha <- function(num, scaled_data){
    alpha <- num
    elastic_net <- cv.glmnet(x=as.matrix(scaled_data[,-16]),
                         y=as.matrix(scaled_data[,16]),
                         alpha = alpha,
                         nfolds=5,
                         type.measure="mse",
                         family="gaussian",
                         standardize=FALSE)

        mse_list <<- cbind(mse_list, c(alpha, min(elastic_net$cvm),elastic_net$l
ambda.min))
}
```

In [103]:

```
#looping over different values of alpha

for (i in seq(.01,1,by = .01)){find_alpha(i,scaled_data)}
```

In [106]:

```
minIndex <- which.min(mse_list[2,])
```

In [107]:

```
mse_list[2,minIndex]
```

50367.424492338

In [108]:

```
mse_list[1, minIndex]
```

0.89

In [109]:

```
#Using alpha = 0.89 and running the elastic net

elastic_net_final <- cv.glmnet(x=as.matrix(scaled_data[,-16]),
                        y=as.matrix(scaled_data[,16]),
                        alpha = 0.89,
                        nfolds=5,
                        type.measure="mse",
                        family="gaussian",
                        standardize=FALSE)
```

In [110]:

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

```
16 x 1 sparse Matrix of class "dgCMatrix"
                      1
(Intercept) 901.772626
M           110.002785
Ed          186.022954
Po1         292.041940
Po2           .
LF           -2.821441
M.F          51.181343
Pop         -25.993778
NW           21.045777
U1          -85.025620
U2          131.958964
Wealth       72.742142
Ineq        272.634500
Prob        -91.904515
Time         -3.396470
So            9.730411
```

In [111]:

```
#So elastic_net selects 14 variables. Let's rerun lm using those features.
```

In [113]:

```
lm_elastic <- lm(Crime ~ ., data = scaled_data[, -4])
```

In [114]:

```
summary(lm_elastic)
```

```
Call:
lm(formula = Crime ~ ., data = scaled_data[, -4])

Residuals:
    Min      1Q  Median      3Q     Max 
-442.55 -116.46    8.86  118.26  473.49 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  903.155     58.889  15.336 2.66e-16 ***
M            112.934     52.244   2.162 0.038232 *  
Ed           198.350     68.044   2.915 0.006445 ** 
Po1          286.864     71.091   4.035 0.000317 ***
LF           -11.321     56.896  -0.199 0.843538    
M.F           53.684     59.798   0.898 0.376026    
Pop          -29.833     48.950  -0.609 0.546523    
NW            25.149     63.619   0.395 0.695239    
U1           -97.649     75.332  -1.296 0.204164    
U2           143.034     69.378   2.062 0.047441 *  
Wealth        87.540     99.662   0.878 0.386292    
Ineq         290.076     90.023   3.222 0.002921 ** 
Prob         -97.432     49.655  -1.962 0.058484 .  
Time          -7.991     47.425  -0.168 0.867251    
So             5.669    148.100   0.038 0.969705    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 208.6 on 32 degrees of freedom
Multiple R-squared:  0.7976,	Adjusted R-squared:  0.709 
F-statistic: 9.006 on 14 and 32 DF,  p-value: 1.673e-07
```

In [115]:

```
#Again it can be seen that the siginificant variables are M, Ed, Po1, U2, Ineq, Prob. However lets perform
#cv using those 14 variables.
```

In [116]:

```
test4 <- numeric()
for (i in 1:nrow(scaled_data)){
    model <- lm(Crime ~ ., data = scaled_data[-i,-4])
    test4 <- cbind(test4, predict(model, newdata = scaled_data[i,-4]))
}
```

In [117]:

```
compute_rsquared(test4, scaled_data$Crime)
```

0.493682047286474

In [118]:

```
#it can be seen that using those 14 variables we got a huge drop in r_square = 0
.49
```

We saw that all models agreed on the final 6 variables - M, Ed, Po1, U2, Ineq, Prob. Also, I was able to get a very simple model with just 3 features which had decent performance considering that we have 3 features. I'm amazed that all other methods were selecting more features (more than 3) with very little improvement in performance. I would prefer as simple models as possible because it makes it easy to explain it to someone. Having worked as a data scientist, you always need to explain what the model is doing to the senior management.