Homework 8

Richard Albright
ISYE6501
Spring 2018

3/2/2019

Question 11.1

Using the crime data set uscrime.txt from Questions 8.2, 9.1, and 10.1, build a regression model using: ## 1. Stepwise regression

Read in the CSV

```r
data <-
  read.table(
    "/Users/ralbright/Dropbox/ISYE6501/week3/homework/uscrime.txt",
    header=TRUE,
    sep="\t"
  )
```

Head:

```r
table <- xtable(head(data))
print(table, type='latex', comment=FALSE, scalebox='0.75')
```

	M	So	Ed	Po1	Po2	LF	M.F	Pop	NW	U1	U2	Wealth	Ineq	Prob	Time	Crime
1	15.10	1	9.10	5.80	5.60	0.51	95.00	33	30.10	0.11	4.10	3940	26.10	0.08	26.20	791
2	14.30	0	11.30	10.30	9.50	0.58	101.20	13	10.20	0.10	3.60	5570	19.40	0.03	25.30	1635
3	14.20	1	8.90	4.50	4.40	0.53	96.90	18	21.90	0.09	3.30	3180	25.00	0.08	24.30	578
4	13.60	0	12.10	14.90	14.10	0.58	99.40	157	8.00	0.10	3.90	6730	16.70	0.02	29.90	1969
5	14.10	0	12.10	10.90	10.10	0.59	98.50	18	3.00	0.09	2.00	5780	17.40	0.04	21.30	1234
6	12.10	0	11.00	11.80	11.50	0.55	96.40	25	4.40	0.08	2.90	6890	12.60	0.03	21.00	682

Tail:

```r
table <- xtable(tail(data))
print(table, type='latex', comment=FALSE, scalebox='0.75')
```

	M	So	Ed	Po1	Po2	LF	M.F	Pop	NW	U1	U2	Wealth	Ineq	Prob	Time	Crime
42	14.10	0	10.90	5.60	5.40	0.52	96.80	4	0.20	0.11	3.70	4890	17.00	0.09	12.20	542
43	16.20	1	9.90	7.50	7.00	0.52	99.60	40	20.80	0.07	2.70	4960	22.40	0.05	32.00	823
44	13.60	0	12.10	9.50	9.60	0.57	101.20	29	3.60	0.11	3.70	6220	16.20	0.03	30.00	1030
45	13.90	1	8.80	4.60	4.10	0.48	96.80	19	4.90	0.14	5.30	4570	24.90	0.06	32.60	455
46	12.60	0	10.40	10.60	9.70	0.60	98.90	40	2.40	0.08	2.50	5930	17.10	0.05	16.70	508
47	13.00	0	12.10	9.00	9.10	0.62	104.90	3	2.20	0.11	4.00	5880	16.00	0.05	16.10	849

Summary:

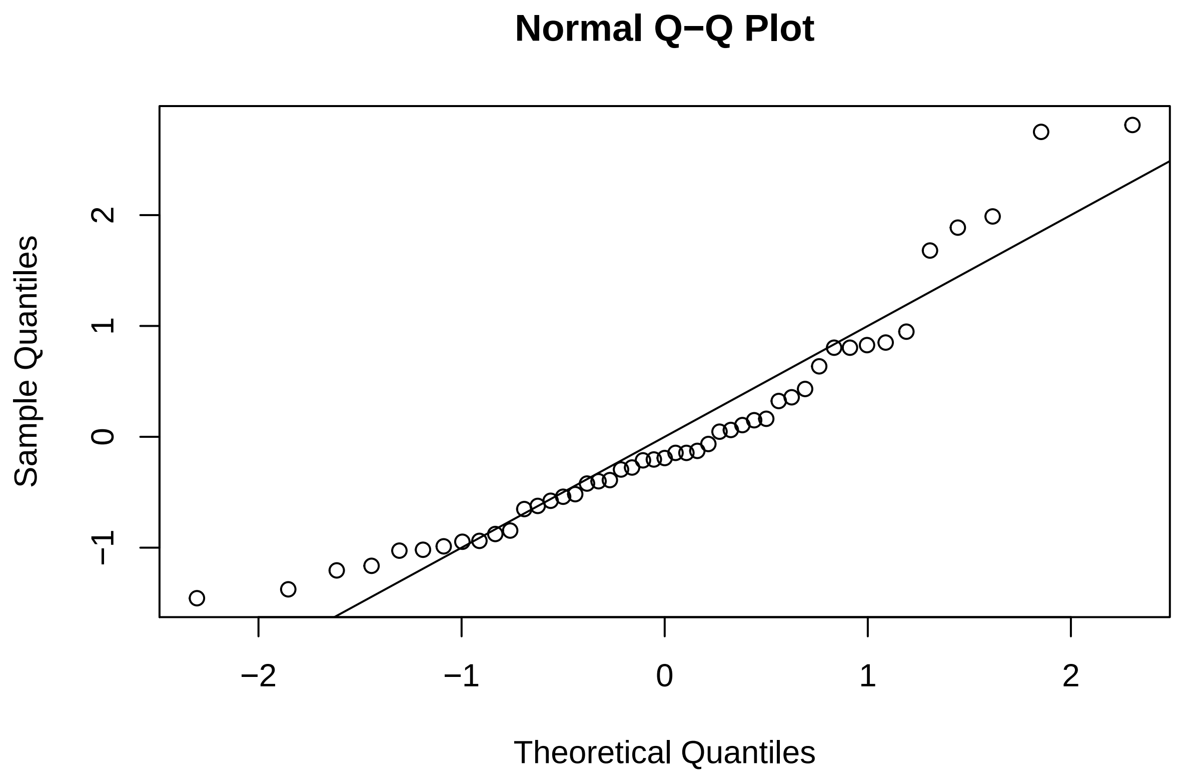
```r
table <- xtable(summary(data))
print(table, type='latex', comment=FALSE, scalebox='0.4')
```

	M	So	Ed	Po1	Po2	LF	M.F	Pop
X	Min. :11.90	Min. :0.0000	Min. : 8.70	Min. : 4.50	Min. : 4.100	Min. :0.4800	Min. : 93.40	Min. : 3.00
X.1	1st Qu.:13.00	1st Qu.:0.0000	1st Qu.: 9.75	1st Qu.: 6.25	1st Qu.: 5.850	1st Qu.:0.5305	1st Qu.: 96.45	1st Qu.: 10.00
X.2	Median :13.60	Median :0.0000	Median :10.80	Median : 7.80	Median : 7.300	Median :0.5600	Median : 97.70	Median : 25.00
X.3	Mean :13.86	Mean :0.3404	Mean :10.56	Mean : 8.50	Mean : 8.023	Mean :0.5612	Mean : 98.30	Mean : 36.62
X.4	3rd Qu.:14.60	3rd Qu.:1.0000	3rd Qu.:11.45	3rd Qu.:10.45	3rd Qu.: 9.700	3rd Qu.:0.5930	3rd Qu.: 99.20	3rd Qu.: 41.50
X.5	Max. :17.70	Max. :1.0000	Max. :12.20	Max. :16.60	Max. :15.700	Max. :0.6410	Max. :107.10	Max. :168.00

	NW	U1	U2	Wealth	Ineq	Prob	Time	Crime
X	Min. : 0.20	Min. :0.07000	Min. :2.000	Min. :2880	Min. :12.60	Min. :0.00690	Min. :12.20	Min. : 342.0
X.1	1st Qu.: 2.40	1st Qu.:0.08050	1st Qu.:2.750	1st Qu.:4595	1st Qu.:16.55	1st Qu.:0.03270	1st Qu.:21.60	1st Qu.: 658.5
X.2	Median : 7.60	Median :0.09200	Median :3.400	Median :5370	Median :17.60	Median :0.04210	Median :25.80	Median : 831.0
X.3	Mean :10.11	Mean :0.09547	Mean :3.398	Mean :5254	Mean :19.40	Mean :0.04709	Mean :26.60	Mean : 905.1
X.4	3rd Qu.:13.25	3rd Qu.:0.10400	3rd Qu.:3.850	3rd Qu.:5915	3rd Qu.:22.75	3rd Qu.:0.05445	3rd Qu.:30.45	3rd Qu.:1057.5
X.5	Max. :42.30	Max. :0.14200	Max. :5.800	Max. :6890	Max. :27.60	Max. :0.11980	Max. :44.00	Max. :1993.0

The plot of the scaled Crime Response Variable using qqnorm also looks like.

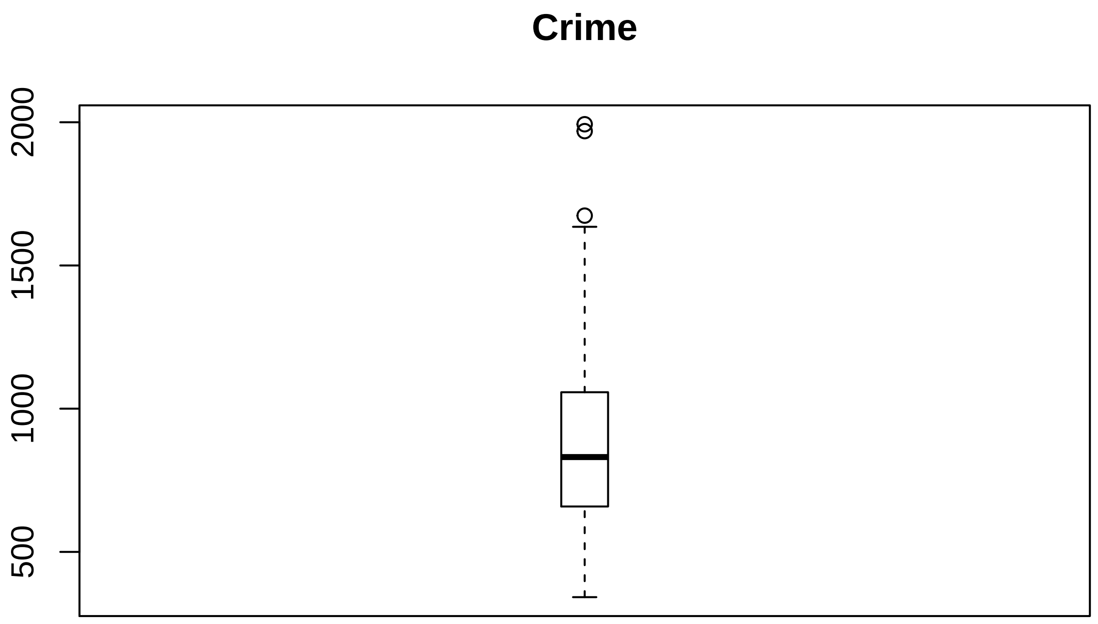
```
scaled_crime = scale(data$Crime)
qqnorm(scaled_crime)
abline(0,1)
```



Which seems to indicate that there may outliers in both tails.

Lets take a look at a box plot of our Crime response variable as well.

```
boxplot(data$Crime, main="Crime", boxwex=0.1)
```



```r
possible_outliers <- boxplot.stats(data$Crime)$out
possible_outliers
```

```
## [1] 1969 1674 1993
```

The boxplot points to possible outliers in the upper tail. Output from boxplot.stats indicates that the 3 possible outliers are 1969, 1674, & 1993. We will now use the grubbs.test function to test for the outliers from the data set.

We will use the 1st 2 tests of the The grubbs.test function below (taken directly from the R Documentation).

First test (10) is used to detect if the sample dataset contains one outlier, statistically different than the other values. Test is based by calculating score of this outlier G (outlier minus mean and divided by sd) and comparing it to appropriate critical values. Alternative method is calculating ratio of variances of two datasets - full dataset and dataset without outlier. The obtained value called U is bound with G by simple formula.

Second test (11) is used to check if lowest and highest value are two outliers on opposite tails of sample. It is based on calculation of ratio of range to standard deviation of the sample.

We will loop through the 1st two test types on the Crime column.

```r
tests <- c(10, 11)
for(test in tests) {
  for(truth in c(TRUE,FALSE)) {
    gtest <- grubbs.test(as.vector(data$Crime), type=test, opposite=truth)
    print(paste('Grubbs Test Type:', test, collapse=' '))
    print(gtest)
  }
}
```

```
## [1] "Grubbs Test Type: 10"
##
##  Grubbs test for one outlier
##
## data:  as.vector(data$Crime)
## G = 1.45590, U = 0.95292, p-value = 1
## alternative hypothesis: lowest value 342 is an outlier
##
## [1] "Grubbs Test Type: 10"
##
##  Grubbs test for one outlier
##
## data:  as.vector(data$Crime)
## G = 2.81290, U = 0.82426, p-value = 0.07887
## alternative hypothesis: highest value 1993 is an outlier
##
## [1] "Grubbs Test Type: 11"
##
##  Grubbs test for two opposite outliers
##
## data:  as.vector(data$Crime)
## G = 4.26880, U = 0.78103, p-value = 1
## alternative hypothesis: 342 and 1993 are outliers
##
## [1] "Grubbs Test Type: 11"
##
```

```
##  Grubbs test for two opposite outliers
##
## data:  as.vector(data$Crime)
## G = 4.26880, U = 0.78103, p-value = 1
## alternative hypothesis: 342 and 1993 are outliers
```

Using a 95% confidence interval, We accept the null hypothesis that there are not any outliers in our Crime reponse variable.

Before we start performing stepwise regressions. The data needs divided into training and test sets.

```
# Split data into train and test sets
r = nrow(data)
train_set = sample(1:r, size = round(r * .8), replace = FALSE)
data_train <- data[train_set,]
data_test <- data[-train_set,]
data_train_s = as.data.frame(scale(data_train))
data_test_s = as.data.frame(scale(data_test))
```

Now that we know there are no outliers to contend with, a stepwise regression will performed on predictors in our data set. The stepAIC function is used below, which performs a stepwise regression that uses AIC to determine what predictors should be used in the model.

Stepwise regression using stepAIC() forward and backward selection

```
full_model = lm(Crime~., data=data_train_s)
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 = data_train_s)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.02503 -0.28118  0.04972  0.32049  0.96245
##
## Coefficients:
##                           Estimate             Std. Error t value
## (Intercept)  0.0000000000000005485  0.0776676745981413896   0.000
## M            0.2235914405405521610  0.1157425855213351923   1.932
## Ed           0.4041674723530174762  0.1578393105302986210   2.561
## Po1          0.7003090125674605870  0.1396948514085850912   5.013
## M.F          0.2770793741279568567  0.1123044483055713472   2.467
## U1          -0.4543017108658310432  0.1885986294127475205  -2.409
## U2           0.5577113502325253824  0.2007386537720826092   2.778
## Ineq         0.5369013312382707737  0.1481287854624219469   3.625
## Prob        -0.2002637592094941577  0.0924698861679648931  -2.166
##               Pr(>|t|)
## (Intercept)   1.00000
## M             0.06321 .
## Ed            0.01592 *
## Po1         0.0000245 ***
## M.F           0.01976 *
## U1            0.02258 *
## U2            0.00948 **
## Ineq          0.00110 **
## Prob          0.03870 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4788 on 29 degrees of freedom
## Multiple R-squared:  0.8203, Adjusted R-squared:  0.7708
## F-statistic: 16.55 on 8 and 29 DF,  p-value: 0.000000006651
```

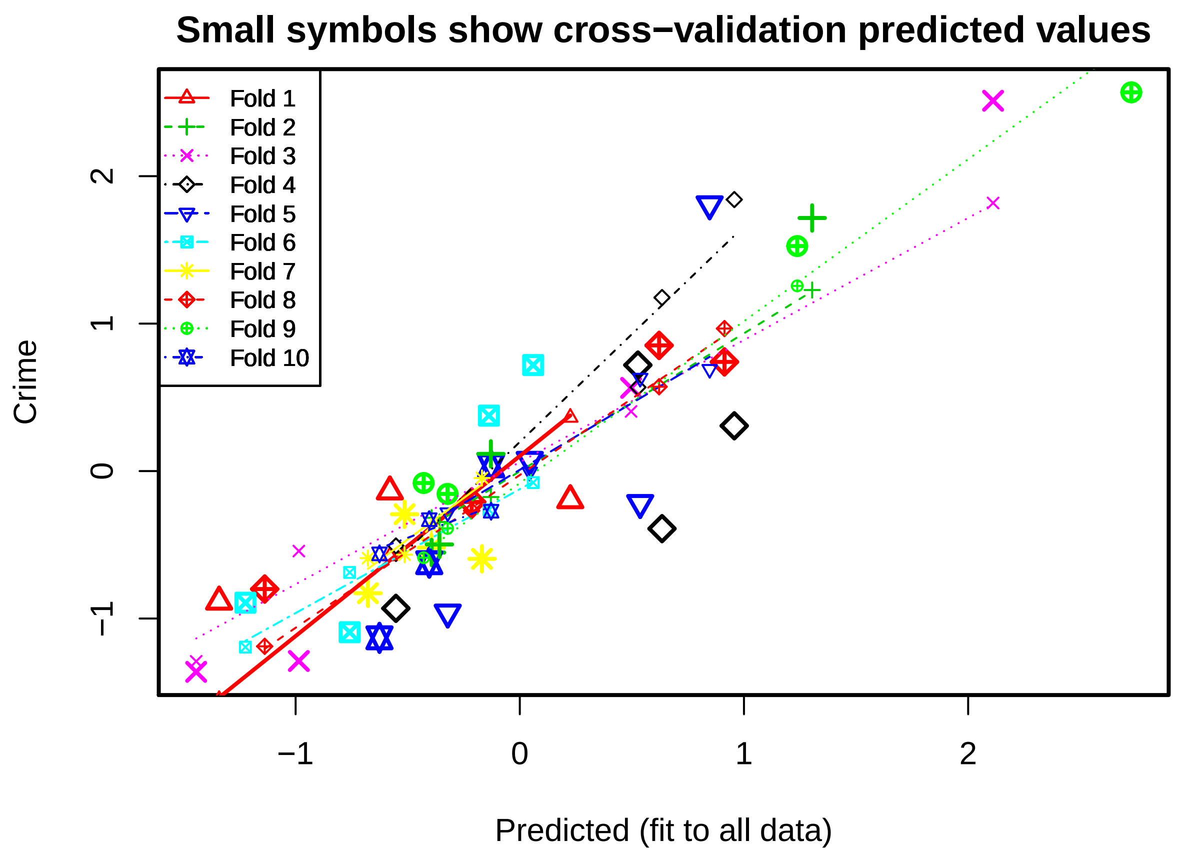
```
step_model$coef
```

```
##               (Intercept)                         M
##  0.00000000000000005485176  0.22359144054055216099982
##                        Ed                       Po1
##  0.4041674723530174762054  0.7003090125674605870287
##                       M.F                        U1
##  0.2770793741279568567215 -0.4543017108658310432290
##                        U2                      Ineq
##  0.5577113502325253824310  0.5369013312382707736603
##                      Prob
## -0.2002637592094941576537
```

Using the predictors from this model, perform K-fold cross validation to get a more accurate r^2.

```
# M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob
step_model_cv <- cv.lm(data_train_s, step_model, m=10)
```

```
## Analysis of Variance Table
##
## Response: Crime
##           Df Sum Sq Mean Sq F value        Pr(>F)
## M          1   0.74    0.74    3.21       0.08345 .
## Ed         1   3.40    3.40   14.84       0.00060 ***
## Po1        1  17.93   17.93   78.20 0.00000000099 ***
## M.F        1   1.29    1.29    5.62       0.02461 *
## U1         1   0.14    0.14    0.62       0.43861
## U2         1   3.22    3.22   14.04       0.00079 ***
## Ineq       1   2.56    2.56   11.19       0.00229 **
## Prob       1   1.08    1.08    4.69       0.03870 *
## Residuals 29   6.65    0.23
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



```
##
## fold 1
## Observations in test set: 3
##                   9     37     42
## Predicted    -0.580  0.225 -1.341
## cvpred       -0.576  0.363 -1.551
## Crime        -0.138 -0.198 -0.886
## CV residual   0.438 -0.561  0.665
##
## Sum of squares = 0.95    Mean square = 0.32    n = 3
##
## fold 2
## Observations in test set: 4
##                   7      6     10     2
## Predicted    -0.129 -0.393 -0.359 1.304
## cvpred       -0.176 -0.317 -0.345 1.228
## Crime         0.117 -0.552 -0.498 1.716
## CV residual   0.293 -0.236 -0.152 0.489
##
## Sum of squares = 0.4    Mean square = 0.1    n = 4
##
## fold 3
## Observations in test set: 4
##                 40      27     4     31
## Predicted    0.496 -1.4435 2.111 -0.986
## cvpred       0.404 -1.2905 1.818 -0.543
## Crime        0.564 -1.3618 2.512 -1.288
## CV residual 0.160 -0.0713 0.694 -0.745
##
## Sum of squares = 1.07    Mean square = 0.27    n = 4
##
## fold 4
## Observations in test set: 4
##                 25    28     19     29
## Predicted   -0.553 0.527  0.635  0.957
## cvpred      -0.508 0.568  1.177  1.841
## Crime       -0.931 0.719 -0.390  0.307
## CV residual -0.423 0.151 -1.567 -1.534
##
## Sum of squares = 5.01    Mean square = 1.25    n = 4
##
## fold 5
## Observations in test set: 4
##                 13       16     43    11
## Predicted   -0.321  0.04443  0.537 0.847
## cvpred      -0.281 -0.00564  0.631 0.691
## Crime       -0.959  0.07612 -0.217 1.809
## CV residual -0.678  0.08176 -0.848 1.118
##
## Sum of squares = 2.44    Mean square = 0.61    n = 4
##
## fold 6
## Observations in test set: 4
##                 45      23     33     17
## Predicted   -0.759  0.0596 -0.138 -1.224
## cvpred      -0.688 -0.0783 -0.263 -1.194
## Crime       -1.093  0.7189  0.376 -0.893
## CV residual -0.405  0.7972  0.639  0.301
##
## Sum of squares = 1.3    Mean square = 0.32    n = 4
##
## fold 7
## Observations in test set: 4
##                 38      14     30      1
## Predicted   -0.677 -0.1687 -0.395 -0.514
## cvpred      -0.590 -0.0469 -0.375 -0.567
## Crime       -0.829 -0.5952 -0.519 -0.293
## CV residual -0.238 -0.5483 -0.144  0.274
##
## Sum of squares = 0.45    Mean square = 0.11    n = 4
##
## fold 8
## Observations in test set: 4
##                  39      3    36     20
## Predicted   -0.2145 -1.138 0.621  0.913
## cvpred      -0.2687 -1.189 0.572  0.967
## Crime       -0.2096 -0.800 0.852  0.740
## CV residual  0.0592  0.389 0.280 -0.226
##
## Sum of squares = 0.28    Mean square = 0.07    n = 4
##
```

```
## fold 9
## Observations in test set: 4
##                 26    8     41     12
## Predicted    2.728 1.24 -0.429 -0.322
## cvpred       2.924 1.26 -0.588 -0.389
## Crime        2.569 1.53 -0.081 -0.155
## CV residual -0.355 0.27  0.507  0.234
##
## Sum of squares = 0.51    Mean square = 0.13    n = 4
##
## fold 10
## Observations in test set: 3
##                  18     22     35
## Predicted   -0.1279 -0.626 -0.404
## cvpred      -0.2723 -0.562 -0.330
## Crime        0.0356 -1.131 -0.621
## CV residual  0.3079 -0.569 -0.292
##
## Sum of squares = 0.5    Mean square = 0.17    n = 3
##
## Overall (Sum over all 3 folds)
##   ms
## 0.34
```

Now extract r^2, adjusted r^2 from the cross validated model.

```
sse <- attr(step_model_cv, 'ms') * nrow(data_train_s)
sst <- sum((data_train_s$Crime - mean(data_train_s$Crime)) ^ 2)
step_model_r2 <- 1 - sse/sst
step_model_r2
```

```
## [1] 0.651
```

```
step_model_adjr2 = 1 - ((1 - step_model_r2)* (nrow(data_train_s)-1))/(nrow(data_train_s)-8-1)
step_model_adjr2
```

```
## [1] 0.555
```

The stepwise regression model using stepAIC() has an r^2 of 0.651, and adjusted r^2 of 0.555.

The second stepwise regression is done below using the regsubsets() function from the leaps package. The parameter nvmax specifies the maximum number of predictors to use in the model. Since the stepwise regression using stepAIC() returned an optimal model using 8 predictors. The nvmax for regsubsets() was set to 8 as well. The methods performed are "forward", "backward", and "segrep". The "segrep" method does both forward and backward selections. The regsubsets() function will return the best 1, 2, . . . nvmax predictor models. Unlike stepAIC(), we need to run the models separately on the variables returned to determine the number of predictors that returns the best model.

Stepwise regression using regsubsets() both forward and backward selection.

```
models <- regsubsets(Crime~., data = data_train, nvmax = 8, method = "seqrep", intercept=FALSE)
summary(models)
```

```
## Subset selection object
## Call: regsubsets.formula(Crime ~ ., data = data_train, nvmax = 8, method = "seqrep",
##     intercept = FALSE)
## 15 Variables
##        Forced in Forced out
## So         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
## 1 subsets of each size up to 8
## Selection Algorithm: 'sequential replacement'
##          M   So  Ed  Po1 Po2 LF  M.F Pop NW  U1  U2  Wealth Ineq Prob Time
## 1  ( 1 ) " " " " " " "*" " " " " " " " " " " " " " " " "    " "  " "  " " 
## 2  ( 1 ) " " " " " " "*" " " " " " " " " " " " " " " " "    "*"  " "  " " 
## 3  ( 1 ) " " " " " " "*" " " " " " " " " " " " " " " " "    "*"  "*"  " " 
## 4  ( 1 ) " " " " " " "*" " " " " " " " " " " " " " " " "    "*"  "*"  "*" 
## 5  ( 1 ) " " " " " " "*" "*" " " " " " " " " " " " " " "    "*"  "*"  "*" 
## 6  ( 1 ) " " " " " " "*" "*" " " " " " " " " "*" " " " "    "*"  "*"  "*" 
## 7  ( 1 ) "*" "*" "*" "*" "*" "*" "*" " " " " " " " " " "    " "  " "  " " 
## 8  ( 1 ) "*" " " " " "*" "*" " " " " " " " " "*" "*" " "    "*"  "*"  "*" 
```

Since regsubsets() only returns the best model for each number k predictors, we will have to loop through the models and perform a regression on each one to get the results.

```
all_models <- summary(models)[[1]]
all_models <- lapply(1:nrow(all_models), function(x)as.formula(paste("Crime~", paste(names(which(all_mod
all_models_lm<-lapply(all_models, lm, data_train)
all_models_r2 <- sapply(all_models_lm, function(i) summary(lm(i))$r.squared)
```

Now extract the r^2, adjusted r^2 for each of the models.

```
all_models_r2
```

```
## [1] 0.516 0.621 0.659 0.680 0.709 0.723 0.675 0.751
```

```
all_models_adjr2 <- sapply(all_models_lm, function(i) summary(lm(i))$adj.r.squared)
all_models_adjr2
```

```
## [1] 0.503 0.599 0.629 0.641 0.663 0.670 0.599 0.682
```

```
min_adjr2 = min(all_models_adjr2)
min_adjr2
```

```
## [1] 0.503
```

```
all_models_best = which.min(all_models_adjr2)
all_models_best
```

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

The best model using regsubsets() has 1 predictors. Looking at adjusted r^2 across all our models so far, the best model to use is the model selected as best from rsubsets(), with an r^2 of 0.516 and an adjusted r^2 of 0.503. This model is the best one so far, prior to doing Lasso and Elastic Net stepwise regressions.

2. Lasso

The glmnet function was used to perform a stepwise regression using a lambda (alpha in the function parameters) equal to 1 results in a lasso regression being performed. We will perform a K-fold validation as well.

```
set.seed(3836)
scaled_predictors_train = as.matrix(data_train_s[,-16])
scaled_response_train = as.double(as.matrix(data_train_s[,16]))
lr_model = cv.glmnet(scaled_predictors_train, scaled_response_train, standardize=TRUE, alpha=1)
```

Get the resulting coefficients from our lasso model.

```
coefs = coef(lr_model$glmnet.fit, s=lr_model$lambda.min)
coefs
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                                  1
## (Intercept)  0.000000000000000342
## M            0.176190685812040115
## So           0.025971524880549206
## Ed           0.258103174030197224
## Po1          0.717394482670984046
## Po2          .
## LF           0.015758081144492824
## M.F          0.220699182416291828
## Pop          .
## NW           .
## U1          -0.246978917402515569
## U2           0.309638312123818149
## Wealth       0.095805690589544690
## Ineq         0.496744720057367362
## Prob        -0.182417368189575657
## Time         .
```

Calculate the r^2 and the adjusted r^2 from the lasso regression. We have 10 predictors in our resulting model.

```
n <- length(attributes(coefs)$x) - 1
k <- nrow(data_train_s)
lr_model_r2 = cbind(c(),lr_model$glmnet.fit$dev.ratio[which(lr_model$glmnet.fit$lambda == lr_model$lamb
lr_model_r2
```

```
##      [,1]
## [1,] 0.81
```

```
lr_model_adjr2 = 1 - ((1 -lr_model_r2)* (k-1))/(k-n-1)
lr_model_adjr2
```

```
##      [,1]
## [1,] 0.73
```

Get the mean crossvalidated error for the model

```
lr_model_cvm <- lr_model$cvm[lr_model$lambda == lr_model$lambda.min]
lr_model_cvm
```

```
## [1] 0.51
```

With an r^2 of 0.81. and an adjusted r^2 of 0.73' The mean consolidated error is 0.51. The lasso regression model has a higher r^2 and adjusted r^2, than the previous best model using regsubsets(). This is now the best model prior to performing an Elastic Net regression.

3. Elastic net

Perform a K-fold cross validation while looping through the alpha parameter of the gmlnet function(). By comparing various adjusted r^2 values, we can determine the best alpha value to use for Elastic Net Regression.

```r
set.seed(3836)
en_models <- matrix(1:11, nrow=11, ncol=5)
scaled_predictors_train = as.matrix(data_train_s[,-16])
scaled_response_train = as.double(as.matrix(data_train_s[,16]))
for (i in 0:10) {
  en_train <- cv.glmnet(scaled_predictors_train, scaled_response_train, standardize=TRUE, alpha=i/10)
  cvm = en_train$cvm[en_train$lambda == en_train$lambda.min]
  coefs = coef(en_train)
  n <- length(attributes(coefs)$x) - 1
  k <- nrow(data_train_s)
  r2 = cbind(c(),en_train$glmnet.fit$dev.ratio[which(en_train$glmnet.fit$lambda == en_train$lambda.min)]
  adjr2 =  1 - ((1 -r2)* (k-1))/(k-n-1)

  en_models[i+1,1] <- i
  en_models[i+1,2] <- i/10
  en_models[i+1,3] <- cvm
  en_models[i+1,4] <- r2
  en_models[i+1,5] <- adjr2

}
best_en = which.max(en_models[,5])
best_en
```

```
## [1] 5
```

```r
en_models[best_en,]
```

```
## [1] 4.000 0.400 0.584 0.792 0.780
```

The best Elastic Net Model used a lambda(glmnet alpha) of 0.4. It's mean consolidated error was 0.584. It's r^2 and adjusted r^2 were 0.792 and 0.78, respectively. This model is slightly better than our previous best model using lasso regression, and is the model that should be chosen.

Get the actual model so we can use it against the test set.

```r
en_train <- cv.glmnet(scaled_predictors_train, scaled_response_train, standardize=TRUE, alpha=0.5)
coef(en_train, s=en_train$lambda.min)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                               1
## (Intercept)  0.00000000000000031
## M            0.18213958493001681
## So           0.04877124650490266
## Ed           0.25404725711576071
## Po1          0.48294084292157263
## Po2          0.21459408118579287
## LF           0.03753273703016737
## M.F          0.22600118747652467
## Pop          .
## NW           .
## U1          -0.25165261059790917
## U2           0.33170534215167025
## Wealth       0.10932992711022167
## Ineq         0.48136829873804204
## Prob        -0.18735971885292274
## Time         .
```

The resulting Elastic Net model has 11 predictors. If our data set was large enough we could run this model against our test set. Since the test_set is only 9 data points. The model will be run against the entire data set.

```r
scaled_data = as.data.frame(scale(data))
final_model = lm(Crime~M+So+Ed+Po1+Po2+LF+M.F+NW+U2+Ineq+Prob, data=scaled_data)
summary(final_model)
```

```
##
## Call:
## lm(formula = Crime ~ M + So + Ed + Po1 + Po2 + LF + M.F + NW +
##     U2 + Ineq + Prob, data = scaled_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.0014 -0.2625 -0.0096  0.2561  1.4714
##
## Coefficients:
##                          Estimate             Std. Error t value Pr(>|t|)
## (Intercept) -0.0000000000000000831  0.0785141649174538186    0.00   1.0000
## M            0.2839964212150521417  0.1310067383810094577    2.17   0.0371
## So           0.1530676999076890366  0.1621263135812865686    0.94   0.3516
## Ed           0.5125120293894989132  0.1726596761007349878    2.97   0.0054
## Po1          1.3643745723727982444  0.7633654305360542125    1.79   0.0825
## Po2         -0.5262843615507021289  0.7885932281086628359   -0.67   0.5089
## LF           0.0399819413643276098  0.1345653729303318280    0.30   0.7681
## M.F          0.0821135658757902454  0.1125148967159027430    0.73   0.4704
## NW           0.0136992335691967677  0.1584882637836832198    0.09   0.9316
## U2           0.1749948168352014888  0.1107532094323024768    1.58   0.1231
## Ineq         0.5744945690273914884  0.1813870272185472421    3.17   0.0032
## Prob        -0.2628918212684396849  0.1053200537454658664   -2.50   0.0174
##
## (Intercept)
## M           *
## So
## Ed          **
## Po1         .
## Po2
## LF
## M.F
## NW
## U2
## Ineq        **
## Prob        *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.538 on 35 degrees of freedom
## Multiple R-squared:  0.78,   Adjusted R-squared:  0.71
## F-statistic: 11.3 on 11 and 35 DF,  p-value: 0.0000000161
```

```
final_r2 = summary(final_model)$r.squared
final_adjr2 = summary(final_model)$adj.r.squared
```

Our final model's r^2 is 0.78, and its adjusted r^2 is 0.71.