Question 1

Step 0 - Global Setup

Applicable to all three parts of question 1, I clear the environment, set the seed, load glmnet and DAAG libraries, read in and converted data to a matrix, and also created a helper function to calculate R^2 (since this will be calculated repeatedly):

```
# Clear the environment
rm(list = ls())

# Comment in set.seed(33) to repeat results
set.seed(33)

# Load glmnet and DAAG lib
require(glmnet)
require(DAAG)

# Load crime data into a data frame
data_df <- read.table("uscrime.txt", header=TRUE)

# Scale the data and convert it to a matrix for LASSO and ELNET
scaled_data_df <- as.data.frame(scale(data_df[,c(1,3:15)]))
scaled_data_df <- cbind(data_df[,2],scaled_data_df,data_df[,16])
colnames(scaled_data_df)[1] <- "So"
colnames(scaled_data_df)[16] <- "Crime"
data_mx <- as.matrix(scaled_data_df)

# Helper function to calculate R^2 - will be repeatedly
ComputeR2 <- function(yhat_df, data_df) {
  SSres <- sum((yhat_df - data_df$Crime)^2)
  SStot <- sum((data_df$Crime - mean(data_df$Crime))^2)
  R2 <- 1 - SSres/SStot
  return(R2)
}
```

Stepwise Regression

Step 1 - Identify Factors with Step()

For stepwise regression factor selection, I built a model with all factors and ran step() on it with direction set to “both”; this directs step() to perform “backward” and “forward” selection:

```
model_all <- lm(Crime ~., data_df)
step(model_all, direction = "both")

Step:  AIC=504
```

```
Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob

         Df Sum of Sq     RSS AIC
<none>                1453068 504
+ Wealth  1     26493 1426575 505
- M.F     1    103159 1556227 505
+ Pop     1     16697 1436371 505
+ Po2     1     14148 1438919 505
+ So      1      9329 1443739 506
+ LF      1      4374 1448694 506
+ NW      1      3799 1449269 506
+ Time    1      2293 1450775 506
- U1      1    127044 1580112 506
- Prob    1    247978 1701046 509
- U2      1    255443 1708511 510
- M       1    296790 1749858 511
- Ed      1    445788 1898855 515
- Ineq    1    738244 2191312 521
- Po1     1   1672038 3125105 538

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

Coefficients:
(Intercept)            M           Ed          Po1            M.F
U1            U2           Ineq          Prob
    -6426.1          93.3        180.1         102.7          22.3
-6086.6        187.3          61.3      -3796.0
```

<u>Step 2 - Retrain Model with Step() Identified Factors</u>

After using step() to identify the factors to include in our model, I retrained the regression model:

```
step_model <- lm(formula = Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq +
Prob, data = data_df)
summary(step_model)

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

Residuals:
```

```
   Min     1Q Median     3Q    Max
  -445   -111      3    122    483

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -6426.1     1194.6   -5.38  4.0e-06 ***
M               93.3       33.5    2.79   0.0083 **
Ed             180.1       52.8    3.41   0.0015 **
Po1            102.7       15.5    6.61  8.3e-08 ***
M.F             22.3       13.6    1.64   0.1087
U1           -6086.6     3339.3   -1.82   0.0762 .
U2             187.3       72.5    2.58   0.0137 *
Ineq            61.3       14.0    4.39  8.6e-05 ***
Prob         -3796.0     1490.6   -2.55   0.0151 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 196 on 38 degrees of freedom
Multiple R-squared:  0.789,   Adjusted R-squared:  0.744
F-statistic: 17.7 on 8 and 38 DF,  p-value: 1.16e-10
```

<u>Step 3 - Cross-Validate the Step-Model and Calculate R^2</u>

Due to the limited size of the dataset, I used cv.lm() to cross-validate the model and using the cv prediction values calculate R^2 (using the ComputeR2 function defined in the Global Step above):

```
# Cross-validate the step_model
cv_step_model <- cv.lm(data = data_df, form.lm = step_model, m = 10)

# Calculate R^2 for the cv_step_model
step_yhat <- as.data.frame(cv_step_model$cvpred)
cv_step_model_R2 <- ComputeR2(step_yhat, data_df)
cv_step_model_R2 # 0.62
```

LASSO

<u>Step 1 - Identify Factors using LASSO</u>

Using cv.glmnet with an alpha = 1, I determined that the optimized lambda.min value was equal to 4.82 and suggested using factors: So, M, Ed, Po1, M.F, Pop, NW, U1, U2, Wealth, Ineq, and Prop:

```
# Identify factors using LASSO
lasso_factors <- cv.glmnet(x = data_mx[,-16],
                           y = data_mx[,"Crime"],
                           alpha = 1,
```

```
                                nfolds = 5,
                                type.measure = "mse",
                                family = "gaussian")

# Display the lambda.min for lasso_factors
lasso_factors$lambda.min # 4.82

# Display coefficients for lambda.min
lasso_coeff <- coef(lasso_factors, s = lasso_factors$lambda.min)
lasso_coeff

16 x 1 sparse Matrix of class "dgCMatrix"
                1
(Intercept) 893.1
So           35.3
M           100.7
Ed          165.7
Po1         297.7
Po2           .
LF            .
M.F          53.8
Pop         -12.6
NW           12.1
U1          -62.6
U2          105.0
Wealth       41.0
Ineq        234.6
Prob        -87.5
Time          .
```

Step 2 - Retrain Model with LASSO Identified Factors

Using the factors recommended by LASSO, I retrain the model:

```
# Re-train model using lambda.min factors
lasso_model <- lm(formula = Crime ~ So + M + Ed + Po1 + M.F + Pop + NW + U1
+ U2 + Wealth + Ineq + Prob, data = data_df)
summary(lasso_model)

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

```
Residuals:
   Min     1Q Median     3Q    Max 
-434.2 -107.0   18.6  115.9  470.3 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -6.39e+03   1.41e+03   -4.52  7.1e-05 ***
So           2.29e+01   1.25e+02    0.18   0.8562    
M            8.97e+01   3.93e+01    2.28   0.0288 *  
Ed           1.75e+02   5.63e+01    3.11   0.0038 ** 
Po1          9.87e+01   2.19e+01    4.51  7.3e-05 ***
M.F          1.66e+01   1.63e+01    1.02   0.3166    
Pop         -8.74e-01   1.20e+00   -0.73   0.4711    
NW           1.86e+00   5.61e+00    0.33   0.7419    
U1          -4.98e+03   3.64e+03   -1.37   0.1807    
U2           1.67e+02   7.91e+01    2.11   0.0424 *  
Wealth       8.63e-02   9.90e-02    0.87   0.3893    
Ineq         7.16e+01   2.14e+01    3.35   0.0020 ** 
Prob        -4.08e+03   1.81e+03   -2.26   0.0307 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 203 on 34 degrees of freedom
Multiple R-squared:  0.797,   Adjusted R-squared:  0.726 
F-statistic: 11.1 on 12 and 34 DF,  p-value: 1.52e-08
```

Step 3 - Cross-Validate the LASSO Model and Calculate R^2

Due to the limited size of the dataset, I used cv.lm() to cross-validate the lasso_model and using the cv prediction values calculate R^2 (using the ComputeR2 function defined in the Global Step above):

```
# Cross-validate the lasso_model
cv_lasso_model <- cv.lm(data = data_df, form.lm = lasso_model, m = 10)
summary(cv_lasso_model)

# Calculate R^2 for the cv_step_model
lasso_yhat <- as.data.frame(cv_lasso_model$cvpred)
cv_lasso_model_R2 <- ComputeR2(lasso_yhat, data_df)
cv_lasso_model_R2 # 0.564
```

ELASTIC NET

Step 0 - Variety of Alpha Values

In an effort to vary the alpha setting, I chose values 0.25, 0.50, and 0.75 and ran the following process: run cv.glmnet with variety of alpha values, determine factors, retrain model with identified factors, cv and calculate R^2.

Step 1 - Alpha of 0.25

```
# Identify factors using Elastic Net and alpha of 0.25
elnet_factors <- cv.glmnet(x = data_mx[,-16],
                           y = data_mx[,"Crime"],
                           alpha = 0.25,
                           nfolds = 5,
                           type.measure = "mse",
                           family = "gaussian")

# Display the lambda.min for elnet_factors
elnet_factors$lambda.min

# Display the coefficients for lamdba.min
elnet_coeff <- coef(elnet_factors, s = elnet_factors$lambda.min)
elnet_coeff

# Re-train model using lambda.min factors
elnet_model <- lm(formula = Crime ~ So + M + Ed + Po1 + Po2 + LF + M.F +
Pop + NW + U1 + U2 + Wealth + Ineq + Prob, data = data_df)
summary(elnet_model)

# Cross-validate the elnet_model
cv_elnet_model <- cv.lm(data = data_df, form.lm = elnet_model, m = 10)
summary(cv_elnet_model)

# Calculate R^2 for the cv_elnet_model
elnet_yhat <- as.data.frame(cv_elnet_model$cvpred)
cv_elnet_model_R2 <- ComputeR2(elnet_yhat, data_df)
cv_elnet_model_R2 # 0.484
```

Step 2 - Alpha of 0.50

```
# Identify factors using Elastic Net and alpha of 0.50
elnet_factors <- cv.glmnet(x = data_mx[,-16],
                           y = data_mx[,"Crime"],
                           alpha = 0.50,
                           nfolds = 5,
                           type.measure = "mse",
```

```
                           family = "gaussian")

# Display the lambda.min for elnet_factors
elnet_factors$lambda.min

# Display the coefficients for lamdba.min
elnet_coeff <- coef(elnet_factors, s = elnet_factors$lambda.min)
elnet_coeff

# Re-train model using lambda.min factors
elnet_model <- lm(formula = Crime ~ So + M + Ed + Po1 + Po2 + M.F + Pop +
NW + U1 + U2 + Wealth + Ineq + Prob, data = data_df)
summary(elnet_model)

# Cross-validate the elnet_model
cv_elnet_model <- cv.lm(data = data_df, form.lm = elnet_model, m = 10)
summary(cv_elnet_model)

# Calculate R^2 for the cv_elnet_model
elnet_yhat <- as.data.frame(cv_elnet_model$cvpred)
cv_elnet_model_R2 <- ComputeR2(elnet_yhat, data_df)
cv_elnet_model_R2 # 0.529
```

Step 3 - Alpha of 0.75

```
# Identify factors using Elastic Net and alpha of 0.75
elnet_factors <- cv.glmnet(x = data_mx[,-16],
                           y = data_mx[,"Crime"],
                           alpha = 0.75,
                           nfolds = 5,
                           type.measure = "mse",
                           family = "gaussian")

# Display the lambda.min for elnet_factors
elnet_factors$lambda.min

# Display the coefficients for lamdba.min
elnet_coeff <- coef(elnet_factors, s = elnet_factors$lambda.min)
elnet_coeff

# Re-train model using lambda.min factors
elnet_model <- lm(formula = Crime ~ So + M + Ed + Po1 + M.F + Pop + NW + U1
```

```
+ U2 + Wealth + Ineq + Prob, data = data_df)
summary(elnet_model)

# Cross-validate the elnet_model
cv_elnet_model <- cv.lm(data = data_df, form.lm = elnet_model, m = 10)
summary(cv_elnet_model)

# Calculate R^2 for the cv_elnet_model
elnet_yhat <- as.data.frame(cv_elnet_model$cvpred)
cv_elnet_model_R2 <- ComputeR2(elnet_yhat, data_df)
cv_elnet_model_R2 # 0.564
```

CONCLUSION

In conclusion, I found that the LASSO and Elastic Net methods recommended an abundance of factors that would likely lead to overfitting. If I was developing this model for a production system, I would simplify it by removing some of the factors that have lower importance.