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
2. Lasso
3. Elastic net

For Parts 2 and 3, remember to scale the data first – otherwise, the regression coefficients will be on different scales and the constraint won't have the desired effect.

For Parts 2 and 3, use the glmnet function in R.

Notes on R:

- For the elastic net model, what we called λ in the videos, glmnet calls "alpha"; you can get a range of results by varying alpha from 1 (lasso) to 0 (ridge regression) [and, of course, other values of alpha in between].
- In a function call like glmnet(x,y,family="mgaussian",alpha=1) the predictors x need to be in R's matrix format, rather than data frame format. You can convert a data frame to a matrix using as.matrix – for example, x <- as.matrix(data[,1:n-1])
- Rather than specifying a value of T, glmnet returns models for a variety of values of T.

Method 1. Stepwise Regression

The stepwise regression uses the Akaike Information Criterion (AIC) to add or remove variables iteratively to find the best-fitting model.

Code:

```
uscrime <- read.table("E:/Study/ISyE 6501/hw7/uscrime.txt", header = TRUE)

initial_model <- lm(Crime ~ ., data = uscrime)

sink("E:/Study/ISyE 6501/hw7/11.1.1_Output.txt")
stepwise_model <- step(initial_model, direction = "both")
print(stepwise_model)
summary(stepwise_model)
sink()

for (i in 1:6) {
  filename <- paste0("E:/Study/ISyE 6501/hw7/plot", i, ".png")
  png(filename)
  plot(stepwise_model, which = i, main = paste("Stepwise Model Plot - Plot", i))
  dev.off()
}
```

Iterations:

```
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 = uscrime)

Coefficients:
(Intercept)          M         Ed        Po1        M.F         U1         U2       Ineq
  -6426.10      93.32     180.12     102.65      22.34   -6086.63     187.35      61.33
       Prob
  -3796.03


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

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
```

The initial model includes all variables: Crime ~ M + So + Ed + Po1 + Po2 + LF + M.F + Pop + NW + U1 + U2 + Wealth + Ineq + Prob + Time, and the starting AIC is 514.65. The stepwise procedure removes variables So, LF, Time, Pop, NW, M.F, Wealth, and Po2 one by one to see if AIC decreases.

At each step, the model tests whether removing a variable results in a lower AIC value. The variable is excluded if AIC decreases significantly after removing a variable (e.g., AIC reduces from 514.65 to 512.65 by removing So). The process continues until no further removal or addition of variables reduces AIC.

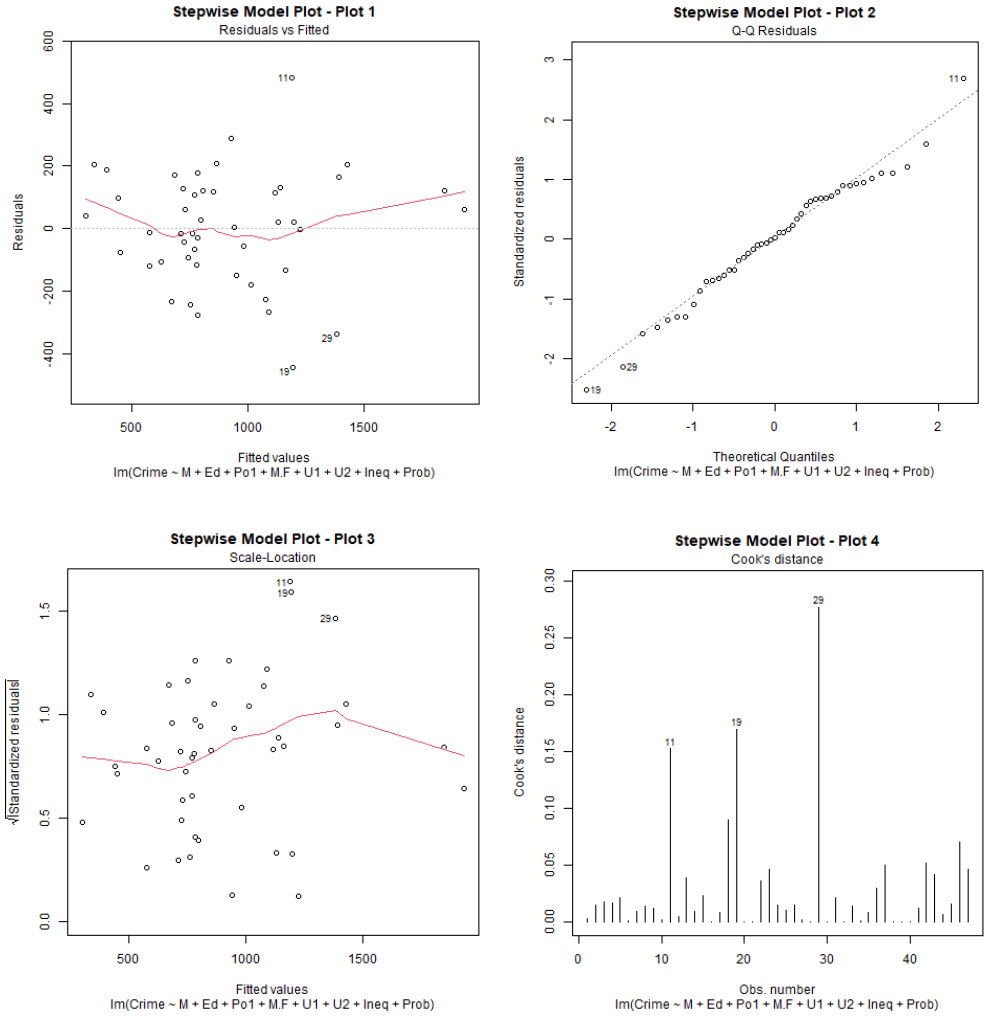
The final model includes Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob. The final AIC value is 503.93, which is much lower than the initial AIC (514.65), indicating an improved model fit.

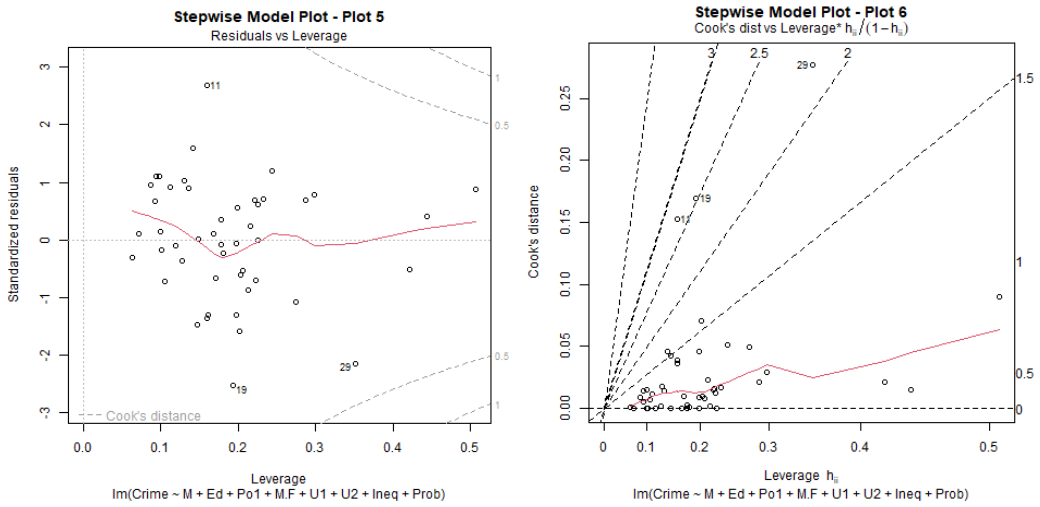
Thus, coefficients represent the estimated impact of each variable on the Crime rate. Significant variables include M, Ed, Po1, and Ineq, as shown by their low p-values (< 0.05), suggesting these variables strongly influence

the Crime rate. M.F and U1 have higher p-values, indicating weaker significance, but they are still included in the final model.

The R-squared value is 0.7888, meaning the model explains 78.88% of the variance in Crime rate. Adjusted R-squared is 0.7444, which considers the number of variables in the model and suggests an excellent overall fit. The F-statistic is 17.74, with a very small p-value, indicating the overall model is statistically significant.

The stepwise regression selected a subset of variables significantly impacting crime rate prediction. The final model has a high R-squared value and reduced AIC, indicating a more optimal model than the initial full-variable one.





Plot 1 checks the linearity and equal variance of the residuals. The residuals should be randomly scattered around the horizontal line at zero, and the outcomes fit.

Plot 2 checks if the residuals follow a normal distribution. The points lie along the diagonal line, indicating the residuals are normally distributed.

Plot 3 checks the constant variance of the residuals. The red line should ideally be flat. Overall, the variance of residuals in this model doesn't change across fitted values dramatically.

Plot 4 identifies observations that have a significant influence on the fitted model. Points with Cook's distance values above 0.5 or 1 (dashed lines) indicate influential observations. Thus, observations 11, 19, and 29 are the most influential in this model.

Plot 5 identifies influential points based on leverage and residual size. Observations in the top right or bottom right are points with high leverage and can significantly influence the model; thus, points 11, 19, and 29 should be reviewed closely for their impact.

Plot 6 highlights observations that are both outliers and influential. Observations outside the dashed lines indicate high influence; thus, points 11, 19, and 29 are again highlighted as significant.

Based on the plots, observations 11, 19, and 29 appear to be particularly influential. It may be worth investigating these specific data points to see if they should be removed or transformed or if the model should be adjusted to better account for them.

Method 2. Lasso

First, to make the values of the feature variables comparable, we normalized the feature variables to zero mean and unit variance to avoid the uneven influence of features of different dimensions on the model. Meanwhile, we used cross-validation to select the best lambda value and drew a graph of cross-validation error changes with lambda (figure 1) to help visualize the lambda corresponding to the smallest error. We find the regularization parameter (11.12694331514) that minimizes the model error to predict the data. The output based on the lasso model: 16 x 1 sparse Matrix of class "dgCMatrix"

	s0
(Intercept)	905.0851064
M	84.6490497
So	22.1160730
Ed	124.1054500
Po1	308.9322037
Po2	.
LF	0.8365183
M.F	52.2295659
Pop	.
NW	4.7549442
U1	-22.1731429
U2	55.6686025
Wealth	.
Ineq	180.8039382
Prob	-81.8184334
Time	.

The data shows that some coefficients are 0, that is, these features do not contribute to the model (Lasso models are characterized by feature selection). MSE =34608.4 represents the average squared variance between the model's prediction and the actual value. $R^2 = 0.763$ indicates that the model can explain 76.36% of the target variable fluctuations.

Finally, we plotted a scatter plot between the actual and predicted values (figure 2) to visualize the prediction effect. The red reference line indicates y=x, which is the ideal case where the predicted value is exactly equal to the actual value. If most of the data points are close to this line, the model prediction is good.

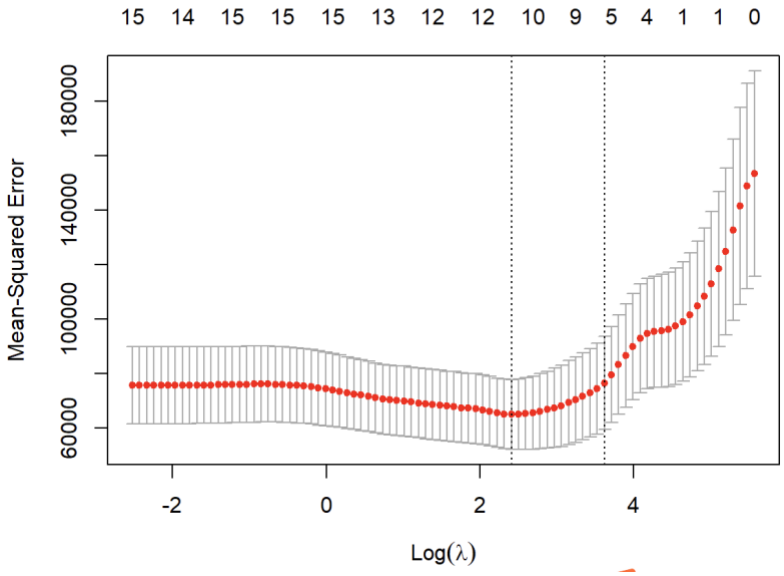


Figure 1 The best lambda value and the corresponding cross-validation error

Figure 2 Actual vs Predicted Crime Rates

Code:

```
library(glmnet)

crime_data <- read.table("C:/Users/Susie/Desktop/uscrime.txt", header = TRUE)

head(crime_data)

# Extract the feature variables (remove the last column)
x <- as.matrix(crime_data[, -ncol(crime_data)])  # 'Crime' is the target variable
# Extract the target variable
y <- crime_data$Crime

# Standardize the feature variables
x_scaled <- scale(x)

# Lasso regression (alpha=1 indicates Lasso regression)
lasso_model <- glmnet(x_scaled, y, alpha = 1)

print(lasso_model)

# Use cross-validation to select the optimal lambda value, determining the best regularization parameter
cv_lasso <- cv.glmnet(x_scaled, y, alpha = 1)

# Get the best lambda value (the optimal lambda selected by cross-validation)
best_lambda <- cv_lasso$lambda.min

# Print the best lambda value
print(paste("Best lambda from CV: ", best_lambda))

# Plot the cross-validation error as a function of lambda
plot(cv_lasso)

# Train the final Lasso model using the best lambda value
```

```
final_lasso_model <- glmnet(x_scaled, y, alpha = 1, lambda = best_lambda)

# Print the regression coefficients
print("Final Lasso Model Coefficients:")
print(coef(final_lasso_model))

# Make predictions using the model
predictions <- predict(final_lasso_model, s = best_lambda, newx = x_scaled)

# Calculate the Mean Squared Error
mse <- mean((predictions - y)^2)
print(paste("Mean Squared Error (MSE): ", mse))

# Define the R²
r_squared <- function(actuals, predictions) {
  ss_res <- sum((actuals - predictions)^2)  # Residual sum of squares
  ss_tot <- sum((actuals - mean(actuals))^2)  # Total sum of squares
  r2 <- 1 - (ss_res / ss_tot)  # Calculate R²
  return(r2)
}

# Calculate R²
r2_value <- r_squared(y, predictions)
print(paste("R²: ", r2_value))

# Visualize the relationship between predicted and actual values
plot(y, predictions, main="Actual vs Predicted Crime Rates",
     xlab="Actual Crime Rates", ylab="Predicted Crime Rates", pch=19, col="blue")
abline(0, 1, col="red")  # Add a reference line y=x
```

Method 3. Elastic net

We use the crime data set to build a regression model with elastic net. Before we build the model, we scale the predictors and convert it into matrix for data preparation. Then we use cross-validation to extract the best alpha value and the best lambda, based on the cross-validation error. In this case, alpha control the Lasso regularization and Ridge regularization, the lambda is regularization parameter to control the penalty.

The figure 1 shows the range of alpha with different C.V. error.

The figure 2 shows the range of lambda with different C.V. error.

The best alpha is 0.7, which means the model uses 70% Lasso and 30% Ridge.

Also, the best lambda is 6.88084, which means the coefficients will be shrunk towards zero according to both Lasso and Ridge penalties. This combination helps reduce model complexity, prevent overfitting

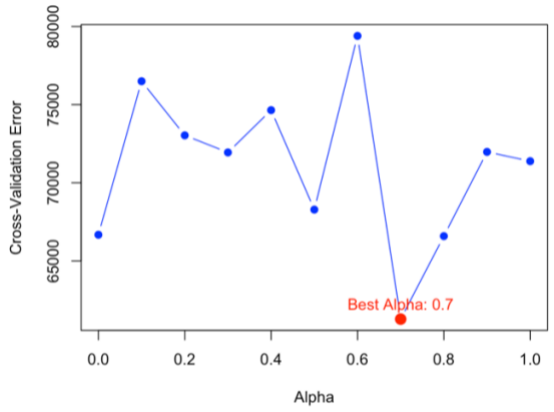


Figure 1 Cross-Validation Error vs Alpha

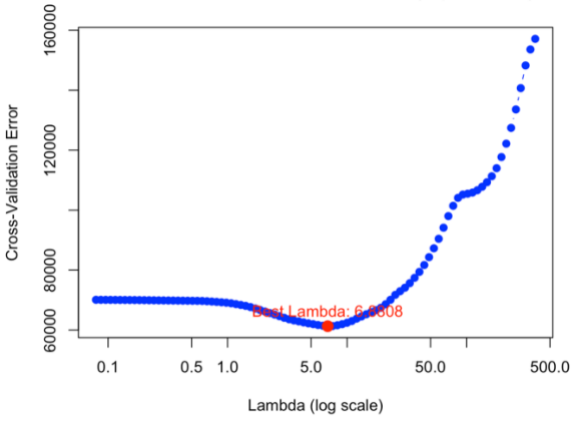


Figure 2 Cross-Validation Error vs Lambda (Alpha=0.7)

The best coefficients are as follow:

16 x 1 sparse Matrix of class "dgCMatrix"

s1

(Intercept) 905.085106

M	*97.928827*
So	*19.117967*
Ed	*160.177692*
Po1	*291.352395*
Po2	*.*
LF	*.*
M.F	*57.203184*
Pop	*-9.181431*
NW	*15.074001*
U1	*-61.931326*
U2	*102.400419*
Wealth	*39.088210*
Ineq	*222.964742*
Prob	*-88.071994*
Time	*.*

The data shows that some coefficients are 0, that is, these features do not contribute to the model. The R-squared is 0.7882172, which means the model can explain 78.82% of the target variable fluctuations. The figure 3 show the plot between the actual and predicted values to visualize the prediction effect. So, the model prediction is good.

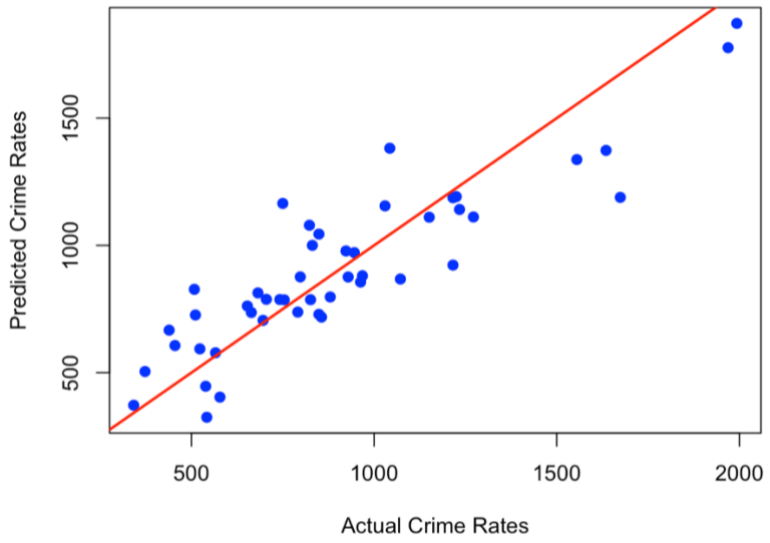


Figure 3 Actual vs Predicted Crime Rates

CODE:

```
# Load necessary libraries
library(glmnet)
library(readr)

# Load the data
crime_data <- read.table("http://www.statsci.org/data/general/uscrime.txt", header = TRUE)

# Prepare predictors (X) and response (y)
x <- as.matrix(scale(crime_data[, 1:ncol(crime_data) - 1])) # Scaling the predictors and converting to matrix
y <- crime_data$Crime # Assuming 'Crime' is the dependent variable

# Cross-validation for Elastic Net model with varying alpha
cv_model <- list()
alphas <- seq(0, 1, by = 0.1)

# Loop over alpha values and store cross-validation results
cv_errors <- numeric(length(alphas))
for (i in seq_along(alphas)) {
  alpha_val <- alphas[i]
  cv_model[[as.character(alpha_val)]] <- cv.glmnet(x, y, alpha = alpha_val)
  cv_errors[i] <- min(cv_model[[as.character(alpha_val)]]$cvm)  # Store the minimum cross-validation error for each alpha
}

# Find the best alpha based on cross-validation error
best_alpha_index <- which.min(cv_errors)
best_alpha <- alphas[best_alpha_index]
cat("Best alpha:", best_alpha, "\n")

# Plot cross-validation error vs. alpha
plot(alphas, cv_errors, type = "b", pch = 19, col = "blue", xlab = "Alpha", ylab = "Cross-Validation Error",
     main = "Cross-Validation Error vs. Alpha")
points(best_alpha, cv_errors[best_alpha_index], col = "red", pch = 19, cex = 1.5)
```

```
text(best_alpha, cv_errors[best_alpha_index], labels = paste("Best Alpha:", best_alpha), pos = 3, col = "red")

# Fit Elastic Net model using the best alpha and perform cross-validation to find the best lambda
cv_model_best_alpha <- cv_model[[as.character(best_alpha)]]

# Optimal lambda for the best alpha
best_lambda <- cv_model_best_alpha$lambda.min
cat("Best lambda:", best_lambda, "\n")

# Plot cross-validation error vs. lambda for the best alpha
plot(cv_model_best_alpha$lambda, cv_model_best_alpha$cvm, type = "b", log = "x", pch = 19, col = "blue",
    xlab = "Lambda (log scale)", ylab = "Cross-Validation Error",
    main = paste("Cross-Validation Error vs. Lambda (Alpha =", best_alpha, ")"))
points(best_lambda, min(cv_model_best_alpha$cvm), col = "red", pch = 19, cex = 1.5)
text(best_lambda, min(cv_model_best_alpha$cvm), labels = paste("Best Lambda:", round(best_lambda, 4)), pos = 3, col = "red")

# Fit the final Elastic Net model using the best alpha and lambda
elastic_net_model <- glmnet(x, y, alpha = best_alpha, family = "gaussian")

# Predict the response for the training data
y_pred <- predict(cv_model_best_alpha, s = "lambda.min", newx = x)

# Calculate R^2
ss_res <- sum((y - y_pred)^2)   # Residual sum of squares
ss_tot <- sum((y - mean(y))^2)  # Total sum of squares
r_squared <- 1 - (ss_res / ss_tot)
cat("R-squared:", r_squared, "\n")

# Plot actual vs predicted values
plot(y, y_pred, xlab = "Actual Crime Rates", ylab = "Predicted Crime Rates",
    main = "Actual vs Predicted Crime Rates", pch = 19, col = "blue")
abline(0, 1, col = "red", lwd = 2)  # Add 45-degree line
```

```
# Extract and print the coefficients at the best lambda
best_coefficients <- coef(cv_model_best_alpha, s = "lambda.min")
cat("Best Coefficients at Best Lambda:\n")
print(best_coefficients)
```