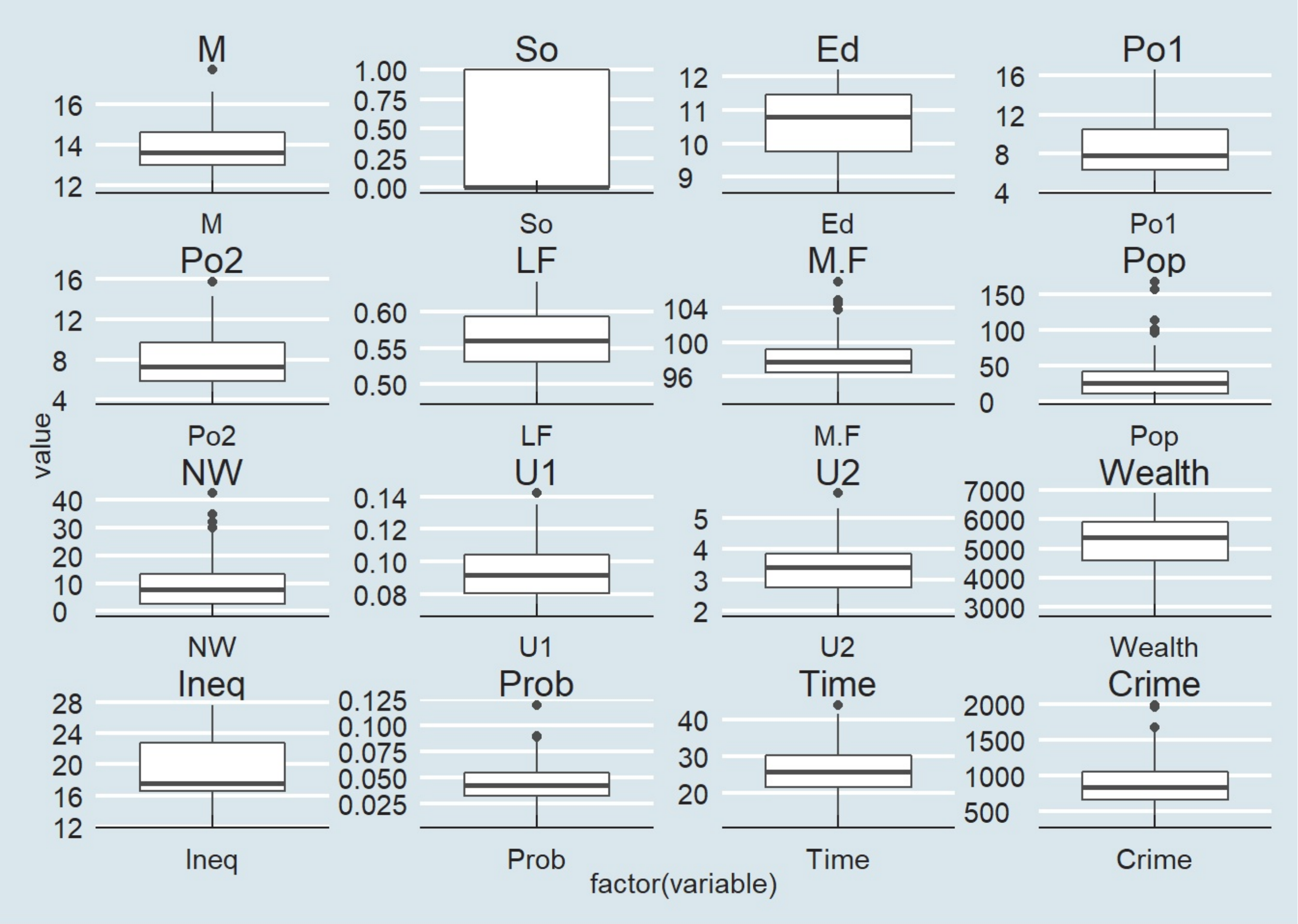
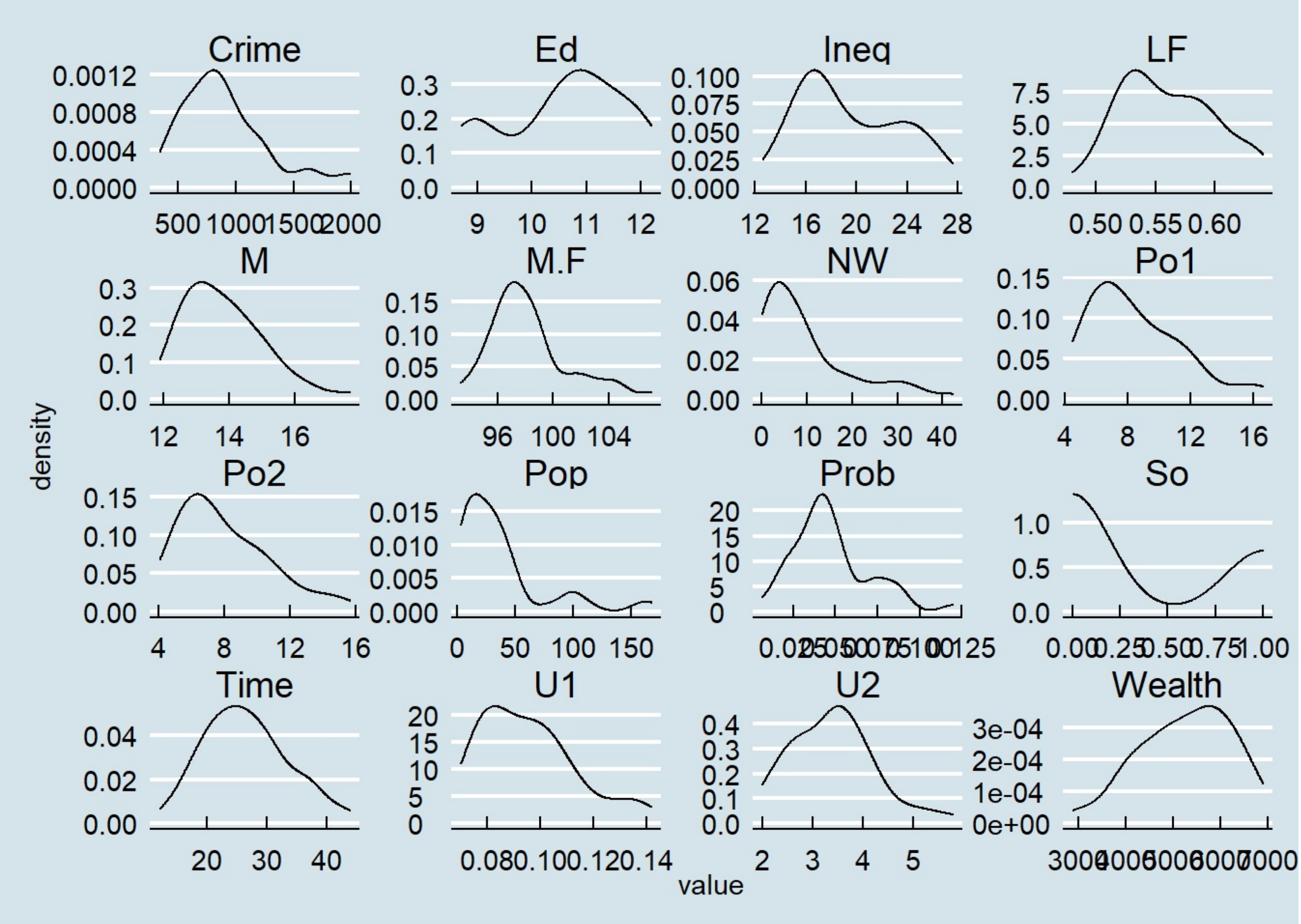
DEMO 8 - STEP-WISE REGRESSION, LASSO AND RIDGE REGRESSION, AND ELASTIC NET MODELS

PRELIMINARY DATA EXPLORATION AND PREPARATION

Box plot visualization for each predictor -



Density distribution visualization for each predictor -



QUESTION 1 - STEP-WISE REGRESSION MODEL

Step-By-Step Analysis ~

1 - We first build a stepwise regression model using 10-fold cross-validation and check the $finalModel attribute of the model with the set of attributes that give a good result.

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

Residuals:
    Min      1Q  Median      3Q     Max
-439.19 -116.92   -4.76  127.15  474.12

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -6557.63    1359.17  -4.825 3.09e-05 ***
M              87.10      34.76   2.506 0.017312 *
Ed            173.41      55.87   3.104 0.003903 **
Po1            98.34      16.22   6.064 7.99e-07 ***
M.F            27.00      15.20   1.777 0.084851 .
U1          -7394.41    3702.00  -1.997 0.054080 .
U2            206.41      77.94   2.648 0.012312 *
Ineq           56.57      15.02   3.766 0.000651 ***
Prob        -3489.88    1578.65  -2.211 0.034100 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 200.7 on 33 degrees of freedom
Multiple R-squared:  0.7873, Adjusted R-squared:  0.7358
F-statistic: 15.27 on 8 and 33 DF,  p-value: 4.342e-09
```

2 - Now we try to improve our quality of model by pruning the lesser significant predictors and seeing what results our train-test set give us -

Results of full data being used -

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

```
Residuals:
    Min      1Q  Median      3Q     Max 
-520.80  -80.81   -9.98  156.62  511.52 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -3803.59     872.91  -4.357 0.000105 ***
M              76.04      33.96   2.239 0.031423 *  
Ed            146.12      46.92   3.115 0.003604 ** 
Po1           119.88      14.76   8.122 1.18e-09 ***
Ineq           64.54      15.61   4.135 0.000203 ***
Prob        -3622.43    1696.34  -2.135 0.039600 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 216.3 on 36 degrees of freedom
Multiple R-squared:  0.7305,	Adjusted R-squared:  0.693 
F-statistic: 19.51 on 5 and 36 DF,  p-value: 2.303e-09
```

Please note that while the F-statistic and p-value show an improvement, the R-squared measures have decreased. But considering this is a joint predictor model, we will favor the new pruned model. The individual metrics for test and train divisions stay the same except for the residual standard error -

Train - 200.27 Test - 162.05

3 - This is maybe a rare case where RMSE value of the test data is better than train data. :)

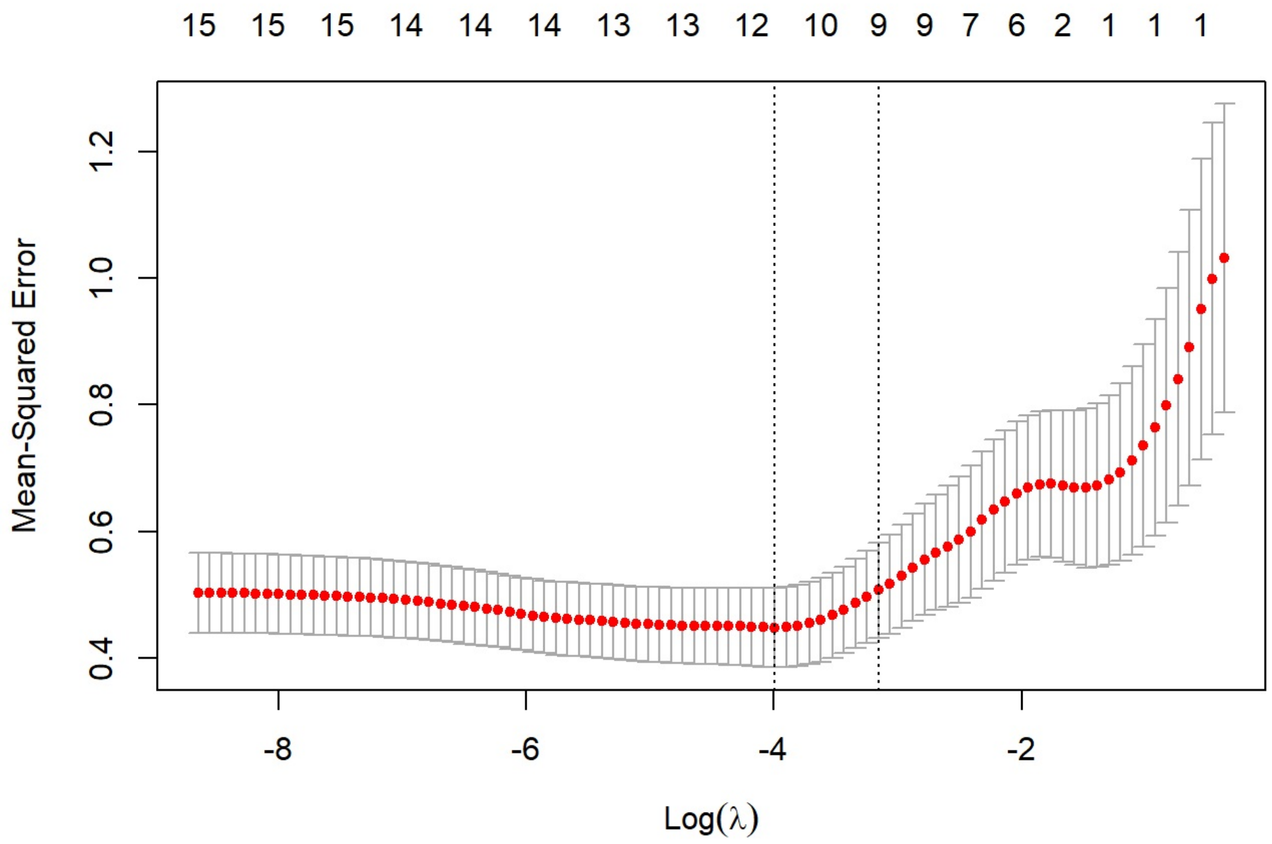
QUESTION 1 - CODE

In [1]:
```
#Code - Step-Wise Regression

#Initial unpruned model
set.seed(123)
#Generate a random sample
random_row<- sample(1:nrow(df ),as.integer(0.8*nrow(df),replace=F))
traindata = df[random_row,]
testdata = df [-random_row,]
train.control <- trainControl(method = "cv", number = 10)
step.model <- train(Crime ~., data = traindata ,
                    method = "lmStepAIC",
                    trControl = train.control,trace=F
                    )
step.model$results
step.model$finalModel
summary(step.model$finalModel)

#evaluation after pruning of full set
full1.model <- lm(Crime ~ M + Ed + Po1 + Ineq + Prob,data = (traindata)) # on train data set
stepfinal1.model <- stepAIC(full1.model, direction = "both",
                      trace = FALSE,k=2)
summary(stepfinal1.model)

#evaluation after pruning of training and testing set
eval_metrics = function(model, df, predictions, target){
    resids = df[,target] - predictions
    resids2 = resids**2
    N = length(predictions)
    r2 = as.character(round(summary(model)$r.squared, 2))
    adj_r2 = as.character(round(summary(model)$adj.r.squared, 2))
    print(adj_r2)
    print(as.character(round(sqrt(sum(resids2)/N), 2)))
}
predictions.train = predict(stepfinal1.model, newdata = (traindata))
predictions.test = predict(stepfinal1.model, newdata = testdata)

#training
eval_metrics(stepfinal1.model, traindata, predictions.train, target = 'Crime')

#testing
eval_metrics(stepfinal1.model, testdata, predictions.test, target = 'Crime')
```

QUESTION 2 - LASSO MODEL

Step-By-Step Analysis ~

1 - We first do a few data adjustments and then apply the lasso function with the underlying family distribution as 'gaussian' (very parametric approach) and try an alpha value of 1. We get this plot -



2 - We retreive coefficents of this model and get our best lambda value to be reapplied to our updated lasso model to get this -

Best Value: 0.01844124

Model:

```
16 x 1 sparse Matrix of class "dgCMatrix"
                       s0
(Intercept) -2.304740e-16
M            2.248093e-01
So           9.661256e-02
Ed           3.637371e-01
Po1          7.720783e-01
Po2          .
LF           2.177071e-02
M.F          1.563956e-01
Pop          .
NW           5.887326e-03
U1          -1.563660e-01
U2           2.607918e-01
Wealth       3.499338e-02
Ineq         4.674476e-01
Prob        -2.103825e-01
Time         .
```

3 - Now we apply this model to train and test data -

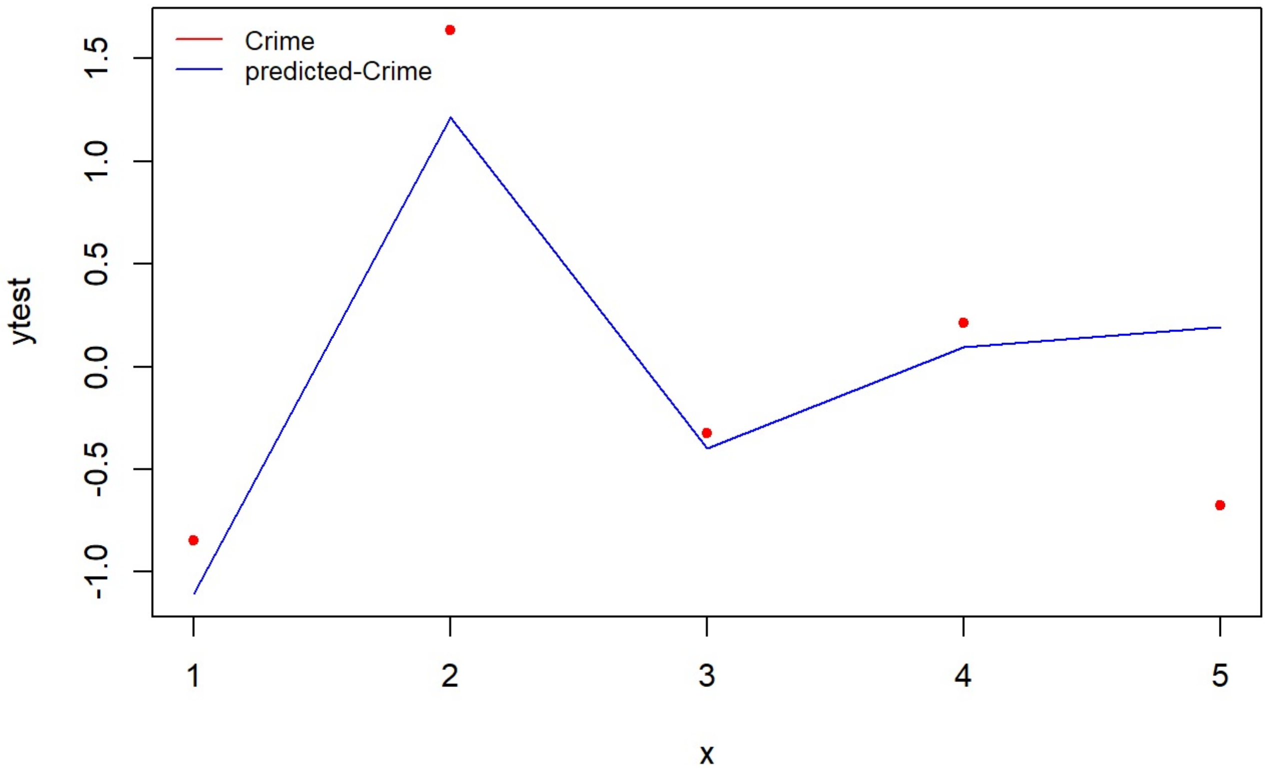
Training -

```
       RMSE   Rsquare
1 0.4634677 0.7799586
```

Testing -

```
       RMSE  Rsquare
1 0.4510459 0.745697
```

The lasso plot -



Optimal Lambda: 0.02221254

4 - The R-sqaured and RMSE are almost the same for both sets. We cannot compare the RMSE of lasso and stepwise due to the scaling element in this one, hence we look at the R-squared and see there isn't much difference in the quality in those those terms either.

QUESTION 2 - CODE

In [2]:
```
#Code - Lasso

#scaling
xtrain<-scale(as.matrix(traindata)[,-16], center = TRUE, scale = TRUE)
ytrain<-scale(as.matrix(traindata)[,16], center = TRUE, scale = TRUE)
xtest<-scale(as.matrix(testdata)[,-16], center = TRUE, scale = TRUE)
ytest<-scale(as.matrix(testdata)[,16], center = TRUE, scale = TRUE)

#lasso model
lasso_cv <- cv.glmnet(xtrain, ytrain, family="gaussian", alpha=1)
plot(lasso_cv)
coef(lasso_cv)
best_lambda <- lasso_cv$lambda.min
cat(best_lambda) #best lambda value to reapply

#Using new lambda value
lasso_mod = glmnet(xtrain, ytrain, family = "gaussian", alpha = 1, lambda = best_lambda)
coef(lasso_mod)

#prediction on test and train
eval_results <- function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
  R_square <- 1 - SSE / SST
  RMSE = sqrt(SSE/nrow(df))


  # Model performance metrics
data.frame(
  RMSE = RMSE,
  Rsquare = R_square
)

}

#for training
yhat.train = predict(lasso_mod, xtrain)
eval_results(ytrain, yhat.train, traindata)
```

```
#for testing
yhat.test = predict(lasso_mod, xtest)
eval_results(ytest, yhat.test, testdata)

#final lasso plot of new model
x = 1:length(ytest)
plot(x, ytest, ylim=c(min(yhat.test), max(ytest)), pch=20, col="red")
lines(x, yhat.test, lwd="1", col="blue")
legend("topleft", legend=c("Crime", "predicted-Crime"),
       col=c("red", "blue"), lty=1,cex = 0.8, lwd=1, bty='n')
```

QUESTION 3 - ELASTIC NET MODEL

Step-By-Step Analysis ~

1 - For elastic net, we use the glm parameterization, and derive our best tuning parameters for our training set -

```
       alpha     lambda
1 0.00381962 0.09804711
```

2 - Now we run this model with the tuned parameters on the train and test data and check the quality metrics -

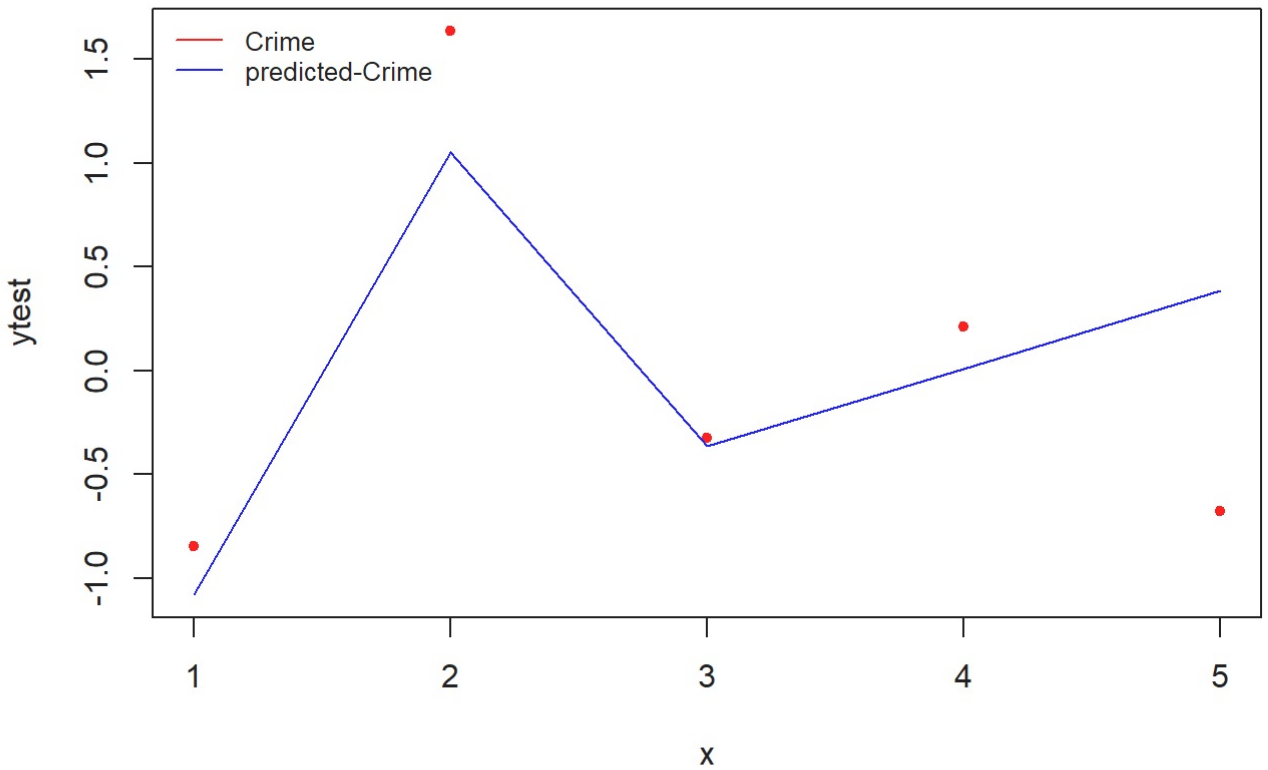
Training -

```
      RMSE  Rsquare
1 0.477752 0.766186
```

Testing -

```
       RMSE   Rsquare
1 0.5595539 0.6086243
```

Elastic net plot -



3 - As can be seen, the R-squared values have become higher (better) compared to the linear regression models which is a good thing. The test data behaves as expected - a little worse than training. And all three model qualities can be put within the same ballpark of R-squared values and good enough in terms of quality.

QUESTION 3 - CODE

In [3]:

```
#Code - Elastic Net

#training set for best tuning parameters
train_cont <- trainControl(method = "repeatedcv",
                              number = 10,
                              repeats = 5,
                              search = "random",
                              verboseIter = F)

elastic_reg <- train(Crime ~ .,data = as.matrix(scale(traindata)),method = "glmnet",preProcess = c("center", "sca
                           tuneLength = 20,#tested 20 alpha-lambda combinations!
                           trControl = train_cont)

elastic_reg$bestTune

#training set predict
predictions_train <- predict(elastic_reg, xtrain)
eval_results(ytrain, predictions_train, as.matrix(traindata))

#testing set predict
predictions_test <- predict(elastic_reg, xtest)
eval_results(ytest, predictions_test, as.matrix(testdata))

#final plot of new model
x = 1:length(ytest)
plot(x, ytest, ylim=c(min(predictions_test), max(ytest)), pch=20, col="red")
lines(x, predictions_test, lwd="1", col="blue")
legend("topleft", legend=c("Crime", "predicted-Crime"),
       col=c("red", "blue"), lty=1,cex = 0.8, lwd=1, bty='n')
```

THE END---

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js