Comparing the predictive power of Lasso and Ridge Regression

Jonas Øren

9/16/2020

Introduction

Lasso and Ridge regression are two methods for handling large dimensional data in regression analysis. Both methods penalize large values of predictors through a penalty term in the likelihood maximizing procedure. Ridge regression uses a quadratic penalty

$$\hat{\beta}^{\text{ridge}} = \text{argmin}\Big\{\sum_{i=1}^{n}\big(Y_i - \beta_0 + \sum_{j=1}^{p} X_{ij}\beta_j\big)^2 + \lambda\sum_{j=1}^{p}\beta_p^2\Big\}.$$

Lasso on the other hand penalizes the absolute value of the parameters

$$\hat{\beta}^{\text{lasso}} = \text{argmin}\Big\{\sum_{i=1}^{n}\big(Y_i - \beta_0 + \sum_{j=1}^{p} X_{ij}\beta_j\big)^2 + \lambda\sum_{j=1}^{p}|\beta_p|\Big\}$$

The penalty boundary of the absolute value function will usually cause some parameter estimates to be zero, leading to automatic variable selection as well.

The goal of this analysis is to test the predictive power of Lasso vs Ridge Regression on the data of crime rates in selected American cities.

About the data

The data is collected to describe the people in each area, their living situation and economic situation. For example `racePctHisp`: percentage of population that is of hispanic, `perCapInc`: per capita income, and `pctUrban`: percentage of people living in areas classified as urban.

The covariates and response variable come normalized between zero and one. Certain extreme outliers are truncated to zero or one manually. For details see the `communities.names` file.

Missing data

Many of the variables have much missing data. We simply remove variables with a lot of missing data, as they are not useful to us.

Collinearity

We will not check variables for colinearity because we will be using penalizing methods that mitigate the effect of colinearity by preventing large opposite estimated values for correlated variables. In addition, as our focus is prediction and not interpretation colinearity is not as big a concern in this case.

Model

We assume the outcomes are a linear function of the covariates, plus noise,

$$Y_i = \beta_0 + \sum_{j=1}^{p} \beta_j X_j + \epsilon$$

We will check if this assumption holds by inspecting the residuals, and then assess the models prediction accuracy by splitting the data into a training- and test set, and check prediction accuracy on the data not used for model fitting. This is to ensure the prediction evaluation is not biased by the correlation between the training data and the fitted parameters.

Analysis

We read the data and give the parameters their correct names.

```
data = read.csv("./data/communities.data")
# Set correct names for variables
names(data) = c("state", "county", "community", "communityname", "fold", "population",
"householdsize", "racepctblack", "racePctWhite", "racePctAsian", "racePctHisp",
"agePct12t21", "agePct12t29", "agePct16t24", "agePct65up", "numbUrban", "pctUrban",
"medIncome", "pctWWage", "pctWFarmSelf", "pctWInvInc", "pctWSocSec", "pctWPubAsst",
"pctWRetire", "medFamInc", "perCapInc", "whitePerCap", "blackPerCap", "indianPerCap",
"AsianPerCap", "OtherPerCap", "HispPerCap", "NumUnderPov", "PctPopUnderPov",
"PctLess9thGrade", "PctNotHSGrad", "PctBSorMore", "PctUnemployed", "PctEmploy",
"PctEmplManu", "PctEmplProfServ", "PctOccupManu", "PctOccupMgmtProf", "MalePctDivorce",
"MalePctNevMarr", "FemalePctDiv", "TotalPctDiv", "PersPerFam", "PctFam2Par",
"PctKids2Par", "PctYoungKids2Par", "PctTeen2Par", "PctWorkMomYoungKids", "PctWorkMom",
"NumIlleg", "PctIlleg", "NumImmig", "PctImmigRecent", "PctImmigRec5", "PctImmigRec8",
"PctImmigRec10", "PctRecentImmig", "PctRecImmig5", "PctRecImmig8", "PctRecImmig10",
"PctSpeakEnglOnly", "PctNotSpeakEnglWell", "PctLargHouseFam", "PctLargHouseOccup",
"PersPerOccupHous", "PersPerOwnOccHous", "PersPerRentOccHous", "PctPersOwnOccup",
"PctPersDenseHous", "PctHousLess3BR", "MedNumBR", "HousVacant", "PctHousOccup",
"PctHousOwnOcc", "PctVacantBoarded", "PctVacMore6Mos", "MedYrHousBuilt", "PctHousNoPhone",
"PctWOFullPlumb", "OwnOccLowQuart", "OwnOccMedVal", "OwnOccHiQuart", "RentLowQ",
"RentMedian", "RentHighQ", "MedRent", "MedRentPctHousInc", "MedOwnCostPctInc",
"MedOwnCostPctIncNoMtg", "NumInShelters", "NumStreet", "PctForeignBorn",
"PctBornSameState", "PctSameHouse85", "PctSameCity85", "PctSameState85", "LemasSwornFT",
"LemasSwFTPerPop", "LemasSwFTFieldOps", "LemasSwFTFieldPerPop", "LemasTotalReq",
"LemasTotReqPerPop", "PolicReqPerOffic", "PolicPerPop", "RacialMatchCommPol",
"PctPolicWhite", "PctPolicBlack", "PctPolicHisp", "PctPolicAsian", "PctPolicMinor",
"OfficAssgnDrugUnits", "NumKindsDrugsSeiz", "PolicAveOTWorked", "LandArea", "PopDens",
"PctUsePubTrans", "PolicCars", "PolicOperBudg", "LemasPctPolicOnPatr",
"LemasGangUnitDeploy", "LemasPctOfficDrugUn", "PolicBudgPerPop", "ViolentCrimesPerPop")
```

Missing data

We then remove variables with a lot of missing data, meaning variables that have more than 100 missing observations. There are still some observations with missing data. We will remove these. As there is only one observation left with missing data, wchich is 0.05% of the total number of observations, we will remove it without fear of biasing the predictions.

```r
# Change missing values to NA data type
data <-  naniar::replace_with_na_all(data, condition = ~.x == "?" )

# Drop columns with more than 100 NA values
omitted <- (names(data[, colSums(is.na(data)) > 100]))
data <- data[, colSums(is.na(data)) < 100]
cat(paste("Omitting variables with > 100 missing data. Omitted:\n",
          paste(omitted, collapse = "\n ")))
```

```
## Omitting variables with > 100 missing data. Omitted:
##  county
##  community
##  LemasSwornFT
##  LemasSwFTPerPop
##  LemasSwFTFieldOps
##  LemasSwFTFieldPerPop
##  LemasTotalReq
##  LemasTotReqPerPop
##  PolicReqPerOffic
##  PolicPerPop
##  RacialMatchCommPol
##  PctPolicWhite
##  PctPolicBlack
##  PctPolicHisp
##  PctPolicAsian
##  PctPolicMinor
##  OfficAssgnDrugUnits
##  NumKindsDrugsSeiz
##  PolicAveOTWorked
##  PolicCars
##  PolicOperBudg
##  LemasPctPolicOnPatr
##  LemasGangUnitDeploy
##  PolicBudgPerPop
```

```r
# Remove observations with missing data
n_omitted <- nrow(data)-nrow(na.omit(data))
omitted <- paste(names(data[, colSums(is.na(data)) > 0]), collapse = "\n ")
data <- na.omit(data)
cat(paste("Omitting observations with missing data. Number of obs. removed: ",
          n_omitted,
          "\n-Variables that contained missing data: \n",
          omitted
          ))
```

```
## Omitting observations with missing data. Number of obs. removed:  1
## -Variables that contained missing data:
##  OtherPerCap
```

We then ensure that all variables have the correct datatype. All variables are numerical

```r
# Convert to numeric datatype (covariates only)
data[-c(1,2,3)] <- lapply(data[-c(1,2,3)], function(x) {
    if(is.factor(x)) as.numeric(as.character(x)) else x
})
```

Finally, we store the response variable and covariates for later use

```r
## Store variables
response <- data$ViolentCrimesPerPop
# Remove first five variables, and the response variable
covariates <- data[-c(1,2,3, which(names(data)=="ViolentCrimesPerPop"))]
```

Split into train and test data

We then spilt the data into a training- and test test, in order to test the predictive power of our models on new data not used in the fitting process.

```r
set.seed(432)    # For reproducability

# Indexes
n <- nrow(data)
train <- sample(1:n, size = round(n*0.75), replace = FALSE)
test <- (1:n)[-train]
```

Fitting the models

We set up a general function for finding the tuning parameter and fitting a model. The tuning parameter λ will be chosen using cross-validation, by the `cv.glmnet`-function. The set was be chosen by trial and error to ensure it contains a minimizing point. The model will be fitted by `glmnet` where the parameter `alpha` decides if it is a lasso or ridge-regression fitting procedure.

```r
glmnet_analysis <- function(alpha) {
  ## Perform glmnet analysis
  ## alpha = 1  -> lasso
  ## alpha = 0  -> ridge regression

  mm <- as.matrix(covariates[train, ])

  # Find lambda with leave one out cross validation
  cv_fit <- glmnet::cv.glmnet(x = mm,
                              y = response[train],
                              alpha = alpha,
                              lambda = exp(seq((-10), (-2), length.out = 100))
  )

  # Fit model using the best lambda
  fit <- glmnet::glmnet(x = mm,
                        y = response[train],
                        alpha = alpha,
                        lambda = cv_fit$lambda.min
  )
  # predict(fit, newx=as.matrix(covariates[test, ]))
  predicted <- predict(fit, newx=as.matrix(covariates[test, ]))[,1]
  error <- predicted-response[test]
  return(list(error=error, cv_fit=cv_fit, fit=fit))
}
```

Fit the models

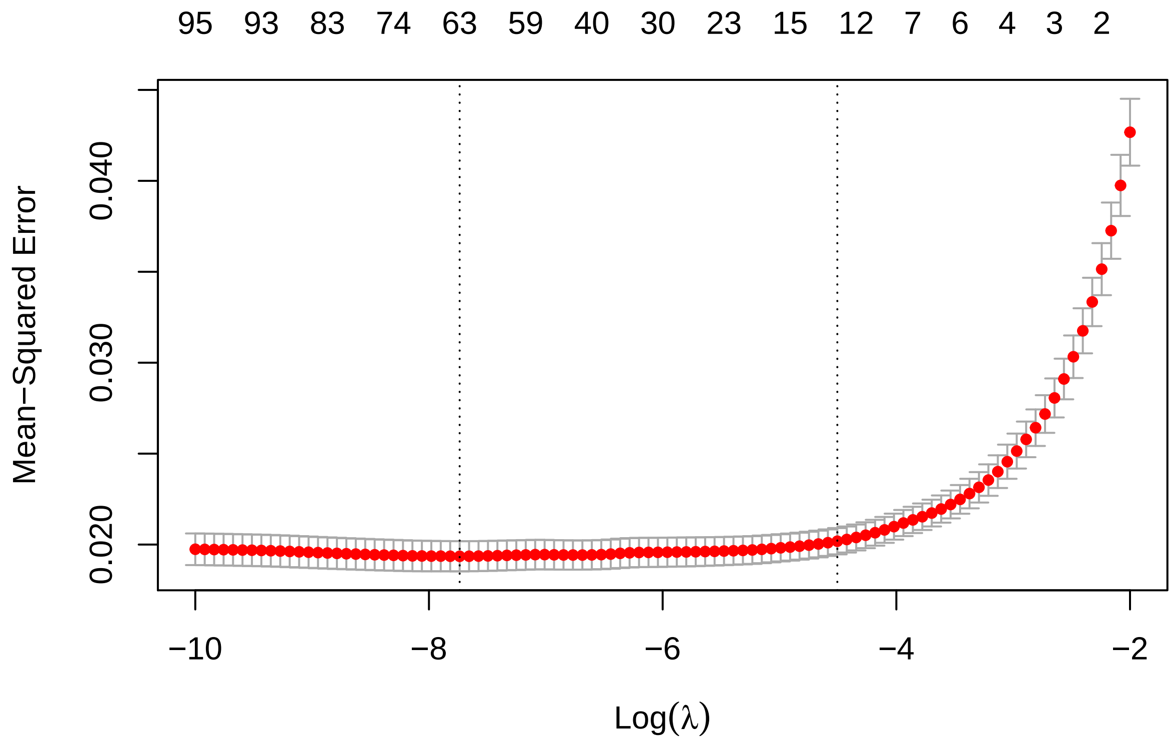
We then fit the models

```
lasso <- glmnet_analysis(alpha = 1)
ridge <- glmnet_analysis(alpha = 0)
```

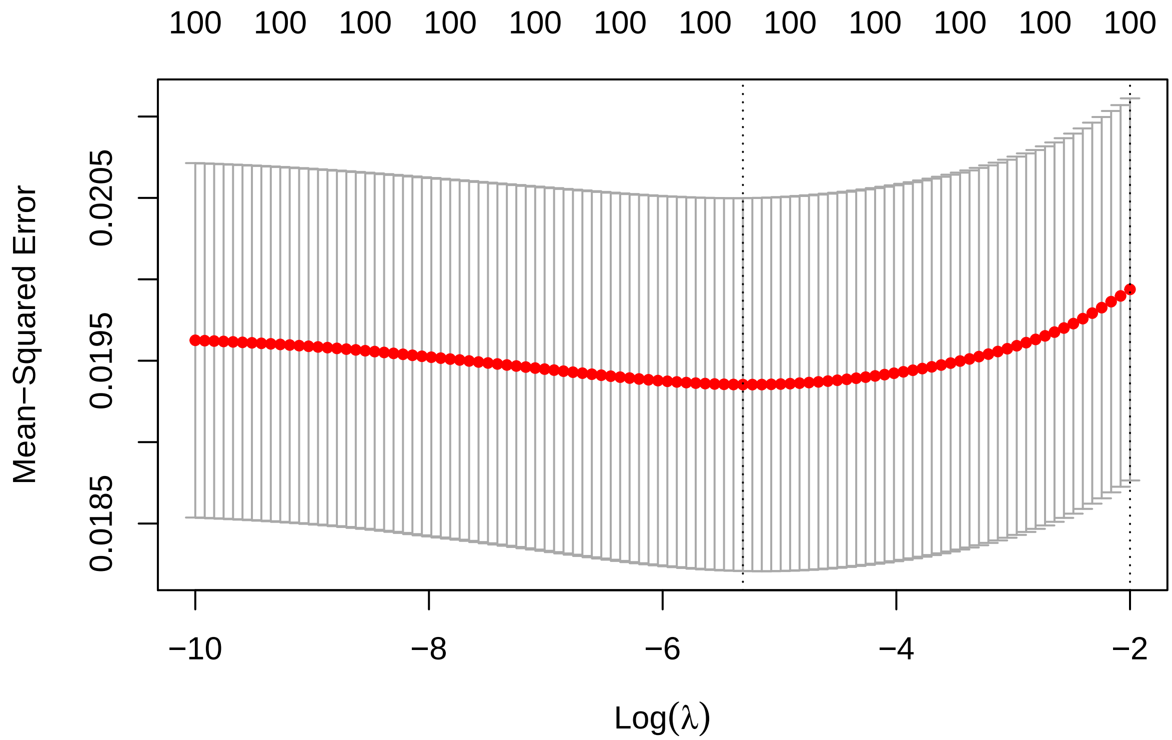
Lambda

Now we must inspect the cross-validation mean squared error for the sequence of lambdas, to make sure a good value was chosen.

```
plot(lasso$cv_fit)
```




```
plot(ridge$cv_fit)
```



The plots indicate that the procedure has found reasonable λ values. The chosen $\log(\lambda)$ values were

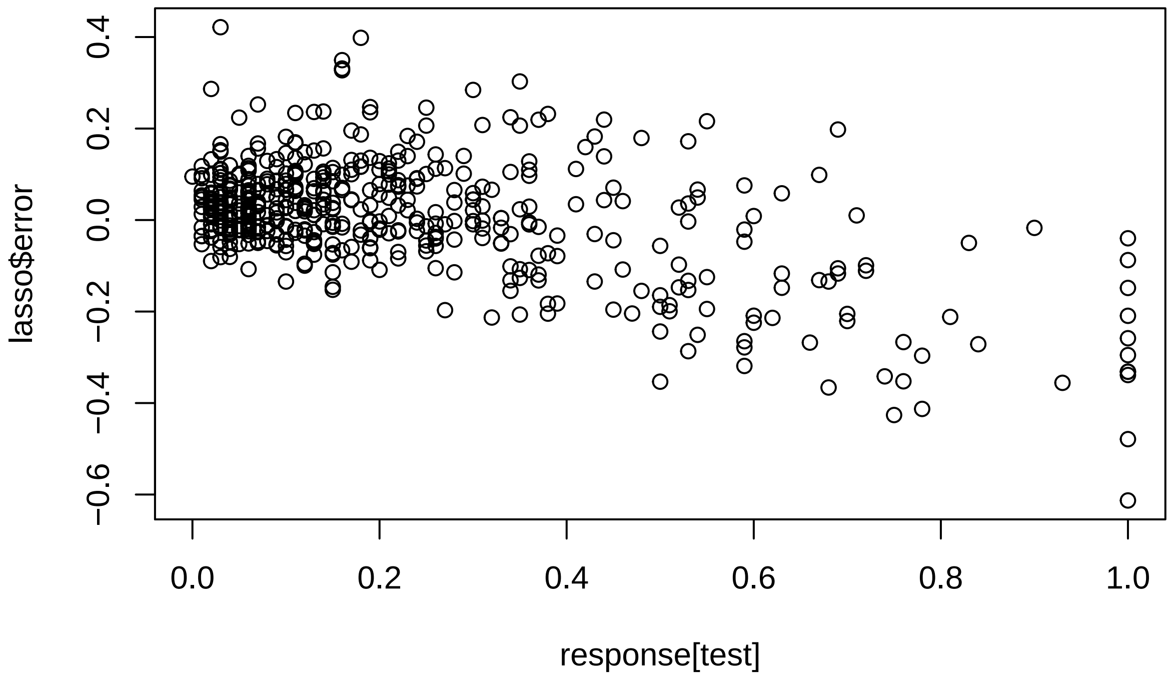
```r
cat(paste(
  "Lasso log-lambda: ",
  round(log(lasso$fit$lambda), 2),
  "\nRidge Regression log-lambda: ",
  round(log(ridge$fit$lambda),2)
))
```

```
## Lasso log-lambda:  -7.74
## Ridge Regression log-lambda:  -5.31
```

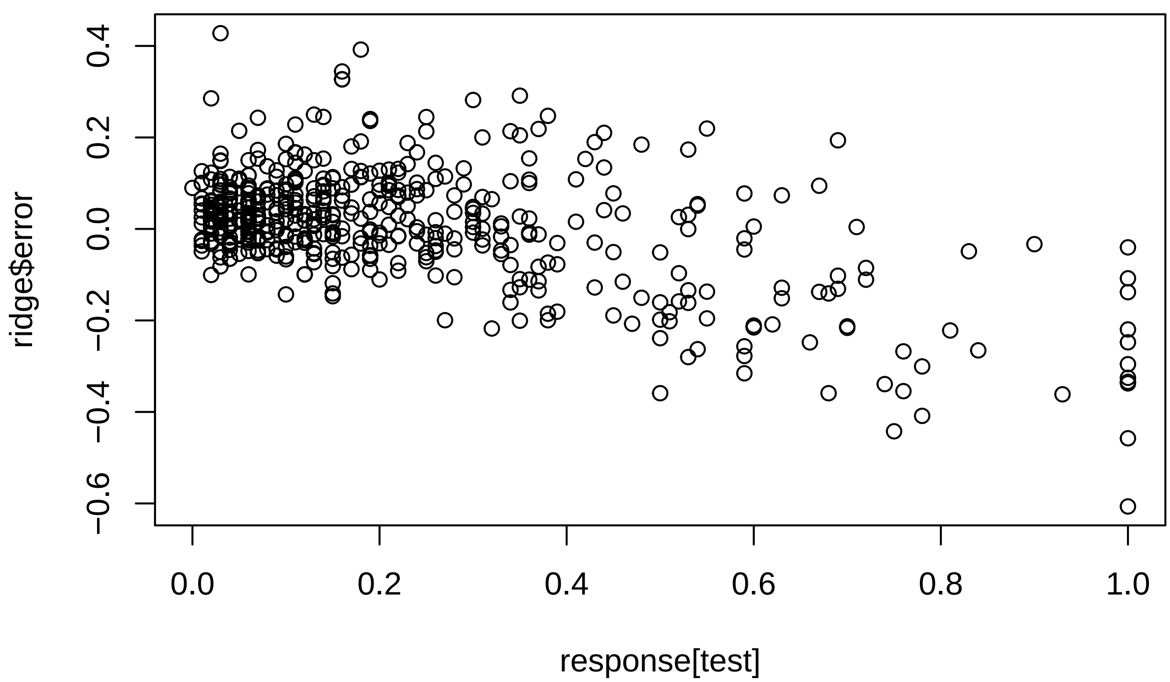
Check linearity assumption

We then inspect the residuals, to see if they look linear.

```r
plot(response[test], lasso$error)
```




```r
plot(response[test], ridge$error)
```



The residuals do indeed appear to have a linear relationship with the response. This supports our assumption of a linear model.

Coefficients

We now inspect the coefficients chosen by the two methods, by printing the 20 largest chosen coefficients in absolute value.

```r
ridge_coef <- coef(ridge$fit)
lasso_coef <- coef(lasso$fit)
ridge_varname <- head(ridge_coef@Dimnames[[1]][order(abs(ridge_coef@x), decreasing = T)], 20)
ridge_var <- head(sort(abs(ridge_coef@x), decreasing = T), 20)

index_of_lasso_coefs <- head((lasso_coef@i+1)[order(lasso_coef@x, decreasing = T)], 20)
lasso_varname <- head(lasso_coef@Dimnames[[1]][index_of_lasso_coefs], 20)
lasso_var <- head(sort(lasso_coef@x, decreasing = T), 20)
data.frame(
  lasso_varname = lasso_varname,
  lasso_var = lasso_var,
  ridge_varname = ridge_varname,
  ridge_var = ridge_var
)
```

```
##          lasso_varname  lasso_var    ridge_varname ridge_var
## 1          (Intercept) 0.54890040      (Intercept) 0.5382677
## 2              MedRent 0.26314628      agePct12t29 0.2175177
## 3     PersPerOccupHous 0.26044113 PersPerOccupHous 0.1931322
## 4         racepctblack 0.23208999     racepctblack 0.1909272
## 5           HousVacant 0.17877315       PctWorkMom 0.1826961
## 6            NumStreet 0.17238261         RentLowQ 0.1819902
## 7       MalePctDivorce 0.16275286        NumStreet 0.1788708
## 8       MalePctNevMarr 0.15738554      PctKids2Par 0.1713745
## 9             PctIlleg 0.14563447       HousVacant 0.1707146
## 10           PctEmploy 0.11799214          MedRent 0.1591046
## 11    PctPersDenseHous 0.11020019   MalePctDivorce 0.1542707
## 12        NumInShelters 0.10047971         PctIlleg 0.1445438
## 13      PctHousLess3BR 0.09223216        pctWInvInc 0.1438674
## 14         racePctHisp 0.09117551          pctWWage 0.1389201
## 15 PctWorkMomYoungKids 0.06120525   MalePctNevMarr 0.1380150
## 16          agePct12t21 0.05490706         NumImmig 0.1365804
## 17         PctRecImmig8 0.04781808    PctPopUnderPov 0.1316242
## 18    PctVacantBoarded 0.04598487       whitePerCap 0.1252659
## 19         PctOccupManu 0.04430768         NumIlleg 0.1247601
## 20           HispPerCap 0.04034210        PctEmploy 0.1209167
```

We see that the two methods choose roughly the same variables to be the most important. The lasso method has discarded 36 of the 101 variables.

```r
lasso_coef@Dim[1] - lasso_coef@p[2]
```

```
## [1] 36
```

Prediction efficiency

We will assess the prediction efficiency by two metrics. First the mean absolute error (MAE)

$$\frac{1}{n}\sum_{i=1}^{n}|\hat{Y}_i - Y_i|$$

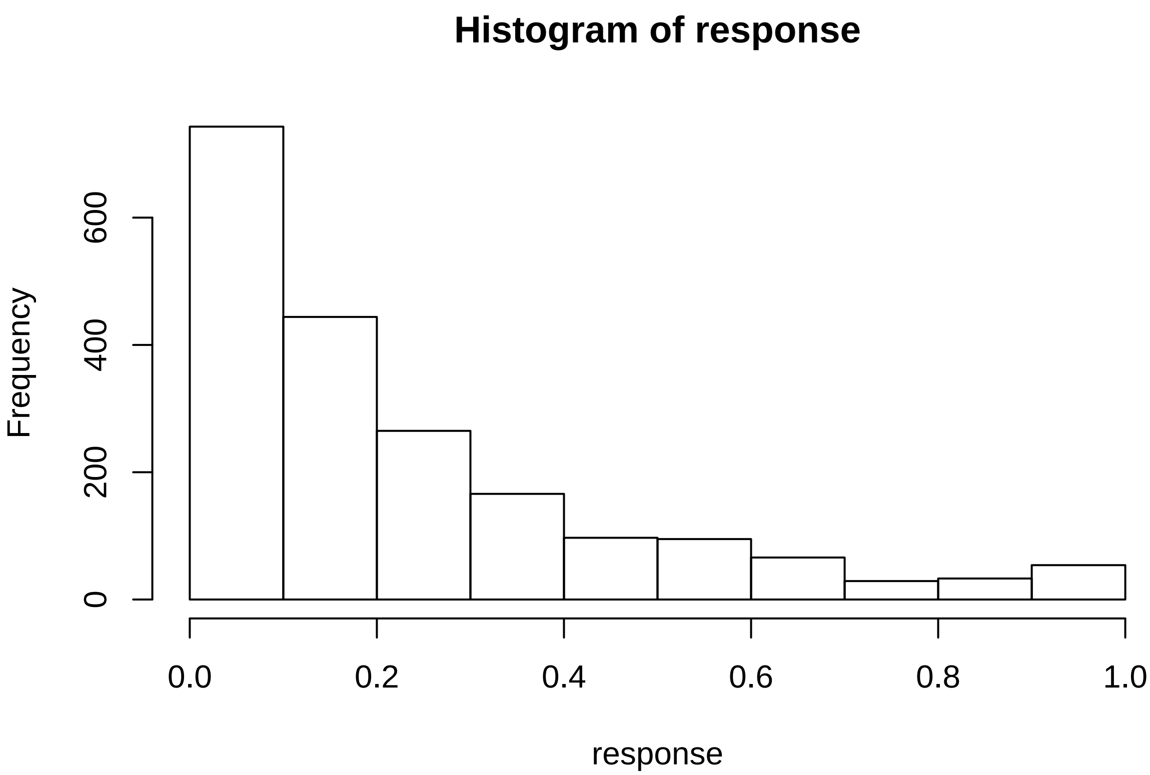
where $\hat{Y}_i$ is the prediction. Because the response variable ''violent chrime per capita" is normalized between zero and one we interpret absolute error as how close the prediction is in terms of the range of "reasonable" values. If the error is larger than one, we are far outside of reasonable predictions. The first and third quantiles are 0.07 and 0.33.

```r
summary(response)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.070   0.150   0.238   0.330   1.000
```

This is the range of the bulk of the response variable. As we see from the histogram of the response.

```r
hist(response)
```



Second we will inspect the mean squared error (MSE)

$$\frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

The MSE is heavily influenced by large errors. If there is a large difference between MAE and MSE we deduce that there there are some errors that are very large compared to the MAE.

```r
n <- length(test)
cat(paste(
  "MAE lasso:             ",
  round(sum(abs(lasso$error))/n, 5),
  "\nMAE Ridge Regression: ",
  round(sum(abs(ridge$error))/n, 5),
  "\nMSE lasso:             ",
  round(sqrt(sum((ridge$error^2))/n), 5),
  "\nMSE Ridge Regression: ",
  round(sqrt(sum((lasso$error^2))/n), 5)
))
```

```
## MAE lasso:              0.09213
## MAE Ridge Regression:  0.092
## MSE lasso:              0.12694
```

```
## MSE Ridge Regression:  0.12731
```

On average both methods predictions are about 10% off the mark. But Ridge regression seems to perform slightly better. From MSE we see that there does not seem to be many extreme errors, but here lasso seem to make somewhat less of relatively large errors.

Conclution

Ridge regression and Lasso both handle seem to handle prediction of crime rates reasonably well. For prediction the choice between the two is nearly arbitrary, as they performed very similarly. If one is to be recommended the author would suggest lasso, as it seems to be slightly more robust in that it makes less large errors.