STAT 702 - Homework 3

Noah Javadi

2025-03-09

##Setup

```r
#install.packages('ISLR2','boot','glmnet','pls')
library(ISLR2)
```

```
## Warning: package 'ISLR2' was built under R version 4.3.3
```

```r
library(boot)
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.3.3
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```r
library(pls)
```

```
## Warning: package 'pls' was built under R version 4.3.3
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

##Problem 5-9

```r
#Problem 9a
u <- mean(Boston$medv)
u
```

```
## [1] 22.53281
```

```r
#Problem 9b
se1 <- sd(Boston$medv)/sqrt(nrow(Boston))
se1
```

```
## [1] 0.4088611
```

```r
#Problem 9c - The bootstrap estimation is very close to the manual calculation from b
se.fn <- function(Boston, index) {
  x <- Boston$medv[index]
  y <- length(index)

  se <- sd(x)/sqrt(y)
  se
}
se.fn(Boston,1:500)
```

```
## [1] 0.4130334
```

```r
boot(Boston,se.fn,R=1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston, statistic = se.fn, R = 1000)
##
##
## Bootstrap Statistics :
##      original       bias    std. error
## t1* 0.4088611 -0.001018028  0.01657769
```

```r
#Problem 9d - The results from the manual calculation and t.test are very close as well
CI.u <- c(u - 1.96*boot(Boston,se.fn,R=1000)[[1]],u + 1.96*boot(Boston,se.fn,R=1000)[[1]])
CI.u
```

```
## [1] 21.73144 23.33417
```

```r
t.test(Boston$medv)
```

```
##
##  One Sample t-test
##
## data:  Boston$medv
## t = 55.111, df = 505, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  21.72953 23.33608
## sample estimates:
## mean of x
##  22.53281
```

```r
#Problem 9e
median(Boston$medv)
```

```
## [1] 21.2
```

```r
#Problem 9f -
se.fn(Boston,1:quantile(Boston$medv,probs = c(0.5)))
```

```
## [1] 1.375661
```

```r
#Problem 9g
u0.1 <- mean(quantile(Boston$medv),probs = c(0.1))
u0.1
```

```
## [1] 23.645
```

```r
#Problem 9h -
se.fn(Boston,1:quantile(Boston$medv,probs = c(0.1)))
```

```
## [1] 2.080688
```

##Problem 6-9

```r
#Problem 9a
College.Train <- College[1:388,]
College.Test <- College[389:777,]

College.Trainy <- sample(1:nrow(College),nrow(College)/2)
College.Testy <- (-College.Trainy)
Apps.test <- College$Apps[College.Testy]

#Problem 9b
model.train <- lm(Apps ~ .,data = College.Train)
summary(model.train)
```

```
##
## Call:
## lm(formula = Apps ~ ., data = College.Train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2721.8  -337.1   -30.6   253.5  6341.8
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7.681e+02  4.895e+02  -1.569  0.11748
## PrivateYes  -4.834e+02  1.748e+02  -2.766  0.00596 **
## Accept       1.202e+00  7.875e-02  15.267  < 2e-16 ***
## Enroll       8.369e-02  2.780e-01   0.301  0.76358
## Top10perc    4.062e+01  6.950e+00   5.845 1.11e-08 ***
```

```
## Top25perc   -1.353e+01  5.692e+00  -2.377  0.01795 *
## F.Undergrad  3.144e-02  4.338e-02   0.725  0.46902
## P.Undergrad  6.651e-03  5.587e-02   0.119  0.90530
## Outstate    -2.514e-02  2.306e-02  -1.090  0.27638
## Room.Board   1.872e-01  5.813e-02   3.220  0.00139 **
## Books       -1.958e-01  2.599e-01  -0.753  0.45166
## Personal     1.062e-01  8.226e-02   1.291  0.19756
## PhD          7.152e-01  6.115e+00   0.117  0.90695
## Terminal    -1.088e+01  6.780e+00  -1.604  0.10956
## S.F.Ratio    1.168e+01  1.508e+01   0.774  0.43915
## perc.alumni -6.553e+00  5.098e+00  -1.285  0.19944
## Expend       8.440e-02  1.415e-02   5.963 5.78e-09 ***
## Grad.Rate    7.445e+00  3.515e+00   2.118  0.03483 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 884.6 on 370 degrees of freedom
## Multiple R-squared:  0.9072, Adjusted R-squared:  0.9029
## F-statistic: 212.7 on 17 and 370 DF,  p-value: < 2.2e-16
```

```
#Problem 9c
grid <- 10^seq(10, -2, length = 388)
ridge.train <- glmnet(College.Train[,c(3:18)],College.Train$Apps,alpha=0,thresh = 1e-12)

ridge.mod <- glmnet(model.matrix(Apps ~ .,College.Train),College.Trainy,alpha = 0,lambda = grid)

cv.out <- cv.glmnet(model.matrix(Apps ~ .,College.Train),College.Trainy,alpha=0)

bestlam <- cv.out$lambda.min

ridge.pred <- predict(ridge.mod, s=bestlam, newx = model.matrix(Apps ~ .,College.Train))
#ridge.pred

mean((ridge.pred - College.Trainy)^2)
```

```
## [1] 50486.42
```

```
#Problem 9d
lasso.train <- glmnet(College.Train[,c(1,3:18)],College.Train$Apps,alpha=1)

lasso.mod <- glmnet(model.matrix(Apps ~ .,College.Train),College.Trainy,alpha = 1,lambda = grid)

cv.out <- cv.glmnet(model.matrix(Apps ~ .,College.Train),College.Trainy,alpha=1)

bestlam <- cv.out$lambda.min

lasso.pred <- predict(lasso.mod, s=bestlam, newx = model.matrix(Apps ~ .,College.Train))
#lasso.pred

lasso.pred <- predict(lasso.mod, s=bestlam, type = 'coefficients')[1:19,]
#lasso.pred

mean((lasso.pred - College.Trainy)^2)
```

```
## Warning in lasso.pred - College.Trainy: longer object length is not a multiple
## of shorter object length

## [1] 200053.7
```

```
#Problem 9e - M = 10
pcr.fit <- pcr(Apps ~ .,data = College,scale = TRUE, validation = "CV")
summary(pcr.fit)
```

```
## Data:    X dimension: 777 17
##  Y dimension: 777 1
## Fit method: svdpc
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            3873     3836     2024     2030     1748     1587     1579
## adjCV         3873     3837     2022     2030     1651     1579     1577
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1570     1540     1498      1491      1495      1497      1502
## adjCV     1571     1534     1495      1489      1492      1494      1499
##        14 comps  15 comps  16 comps  17 comps
## CV         1503      1443      1172      1131
## adjCV      1500      1424      1165      1125
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X      31.670    57.30    64.30    69.90    75.39    80.38    83.99    87.40
## Apps    2.316    73.06    73.07    82.08    84.08    84.11    84.32    85.18
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X       90.50     92.91     95.01     96.81      97.9     98.75     99.36
## Apps    85.88     86.06     86.06     86.10      86.1     86.13     90.32
##       16 comps  17 comps
## X        99.84    100.00
## Apps     92.52     92.92
```

```
pcr.pred <- predict(pcr.fit,College.Test,ncomp = 10)
#pcr.pred

mean((pcr.pred - Apps.test)^2)
```

```
## [1] 31777007
```

```
#Problem 9f - M = 10
pls.fit <- plsr(Apps ~ ., data = College, subset = College.Trainy, scale = TRUE, validation = "CV")
summary(pls.fit)
```

```
## Data:    X dimension: 388 17
##  Y dimension: 388 1
## Fit method: kernelpls
```

```
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            3631     1607     1473     1279     1234     1154     1103
## adjCV         3631     1604     1470     1275     1222     1133     1094
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1082     1080     1080      1076      1075      1075      1074
## adjCV     1077     1076     1075      1071      1070      1070      1069
##        14 comps  15 comps  16 comps  17 comps
## CV         1073      1073      1073      1073
## adjCV      1068      1068      1068      1068
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       26.75    53.38    62.95    65.66    67.76    72.06    76.33    80.62
## Apps    81.18    84.72    88.85    90.58    92.08    92.36    92.46    92.50
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X       82.68     84.69     87.67     90.74     92.67     95.26     96.89
## Apps    92.58     92.63     92.64     92.64     92.65     92.65     92.65
##       16 comps  17 comps
## X        99.13    100.00
## Apps     92.65     92.65
```

```r
pls.pred <- predict(pls.fit,College.Test,ncomp = 10)
#pcr.pred

mean((pls.pred - Apps.test)^2)
```

```
## [1] 33543289
```

#Problem 9g - It does not seem like we are able to accurately predict the number of college applications received based on the data provided. The MSE squared is quite large for the prediction vs test data. The test errors do seem similar between the 5 approaches.

##Problem 6-11

```r
#Problem 11a
#Linear
Boston.lm <- lm(crim ~ .,data = Boston)
summary(Boston.lm)
```

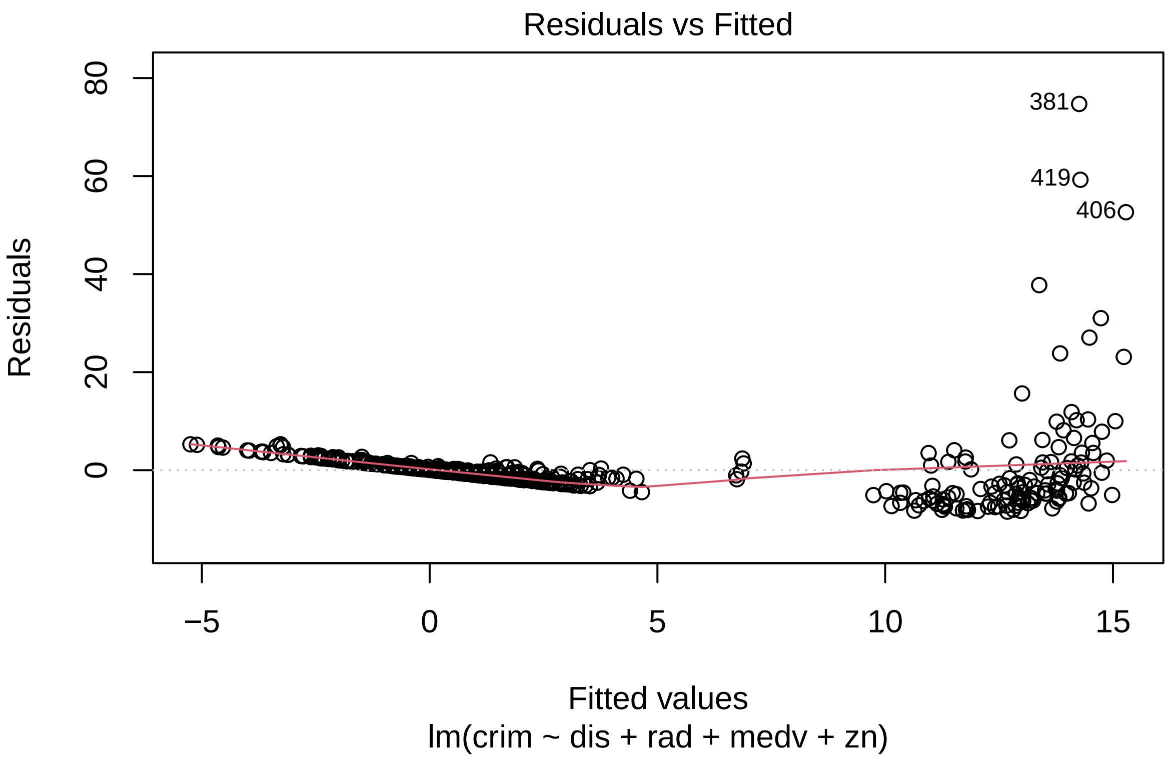
```
##
## Call:
## lm(formula = crim ~ ., data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.534 -2.248 -0.348  1.087 73.923
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
```

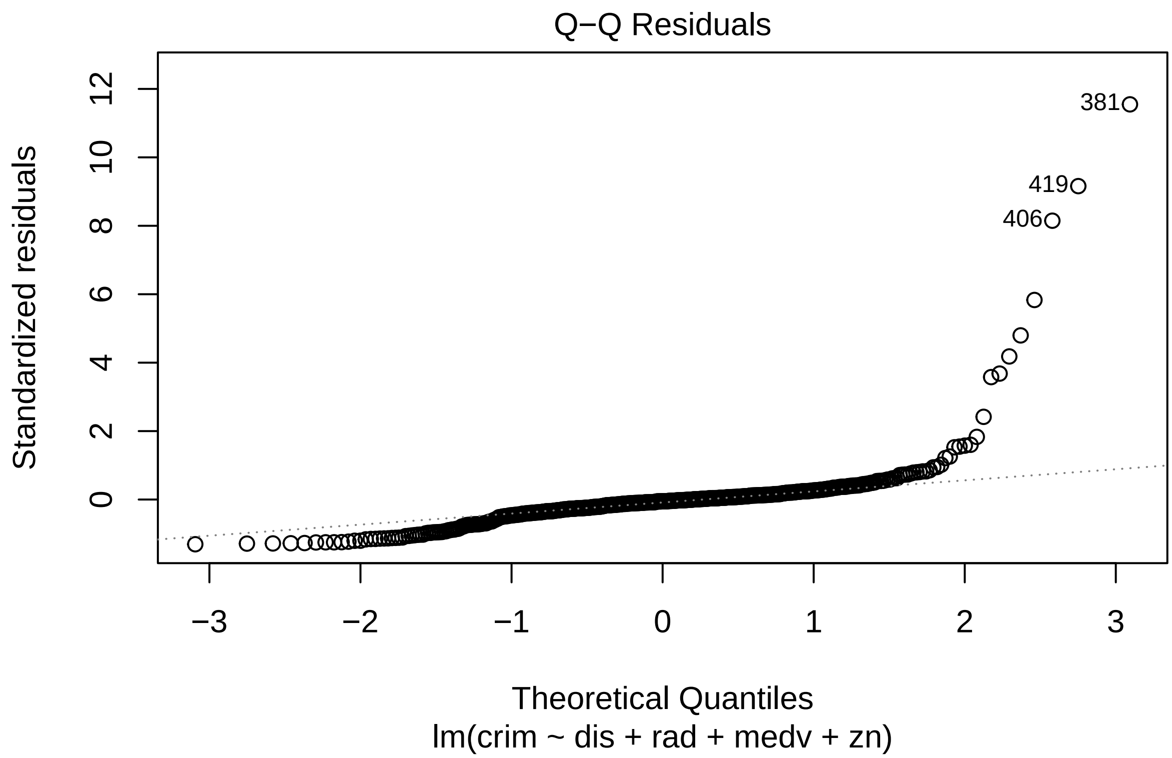
```
## (Intercept) 13.7783938  7.0818258   1.946 0.052271 .
## zn           0.0457100  0.0187903   2.433 0.015344 *
## indus       -0.0583501  0.0836351  -0.698 0.485709
## chas        -0.8253776  1.1833963  -0.697 0.485841
## nox         -9.9575865  5.2898242  -1.882 0.060370 .
## rm           0.6289107  0.6070924   1.036 0.300738
## age         -0.0008483  0.0179482  -0.047 0.962323
## dis         -1.0122467  0.2824676  -3.584 0.000373 ***
## rad          0.6124653  0.0875358   6.997 8.59e-12 ***
## tax         -0.0037756  0.0051723  -0.730 0.465757
## ptratio     -0.3040728  0.1863598  -1.632 0.103393
## lstat        0.1388006  0.0757213   1.833 0.067398 .
## medv        -0.2200564  0.0598240  -3.678 0.000261 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.46 on 493 degrees of freedom
## Multiple R-squared:  0.4493, Adjusted R-squared:  0.4359
## F-statistic: 33.52 on 12 and 493 DF,  p-value: < 2.2e-16
```

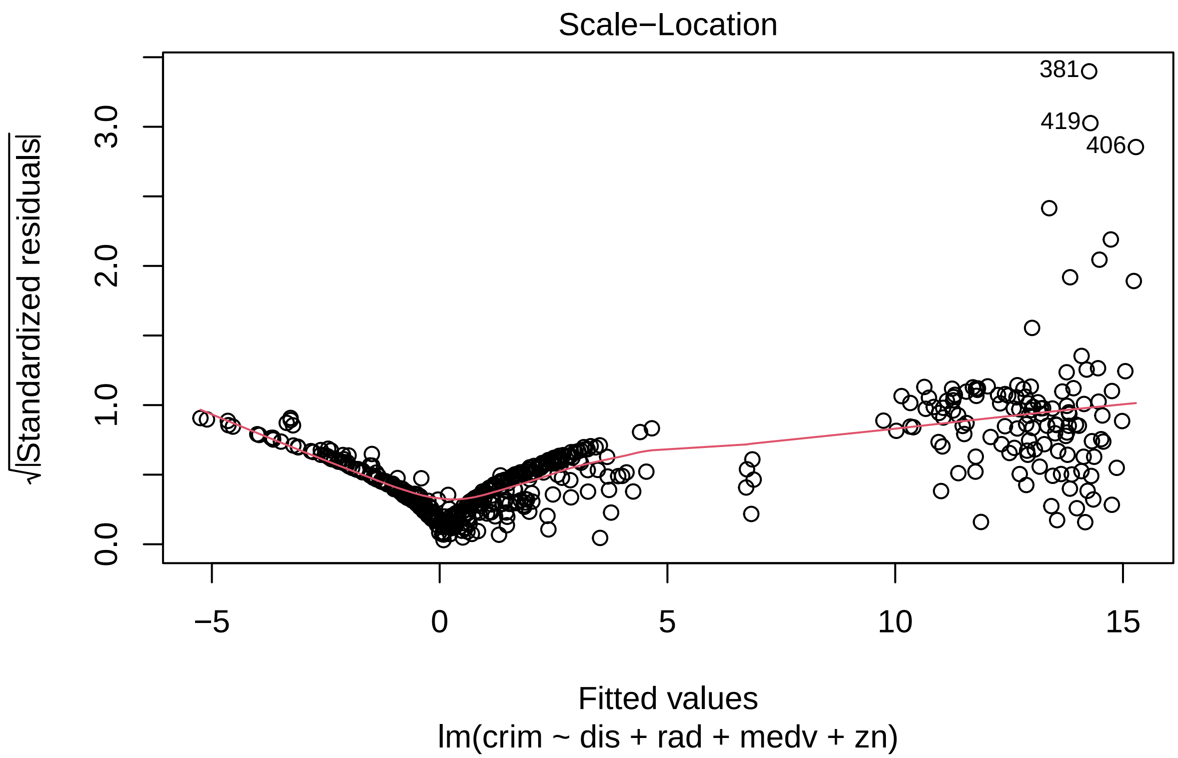
```r
Boston.red.lm <- lm(crim ~ dis + rad + medv + zn, data = Boston)
summary(Boston.red.lm)
```

```
##
## Call:
## lm(formula = crim ~ dis + rad + medv + zn, data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.459 -1.960 -0.331  0.857 74.718
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.26548    1.34674   3.910 0.000105 ***
## dis         -0.72291    0.20254  -3.569 0.000393 ***
## rad          0.50021    0.04044  12.370  < 2e-16 ***
## medv        -0.19122    0.03566  -5.362 1.26e-07 ***
## zn           0.05487    0.01735   3.163 0.001658 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.5 on 501 degrees of freedom
## Multiple R-squared:  0.4335, Adjusted R-squared:  0.429
## F-statistic: 95.84 on 4 and 501 DF,  p-value: < 2.2e-16
```

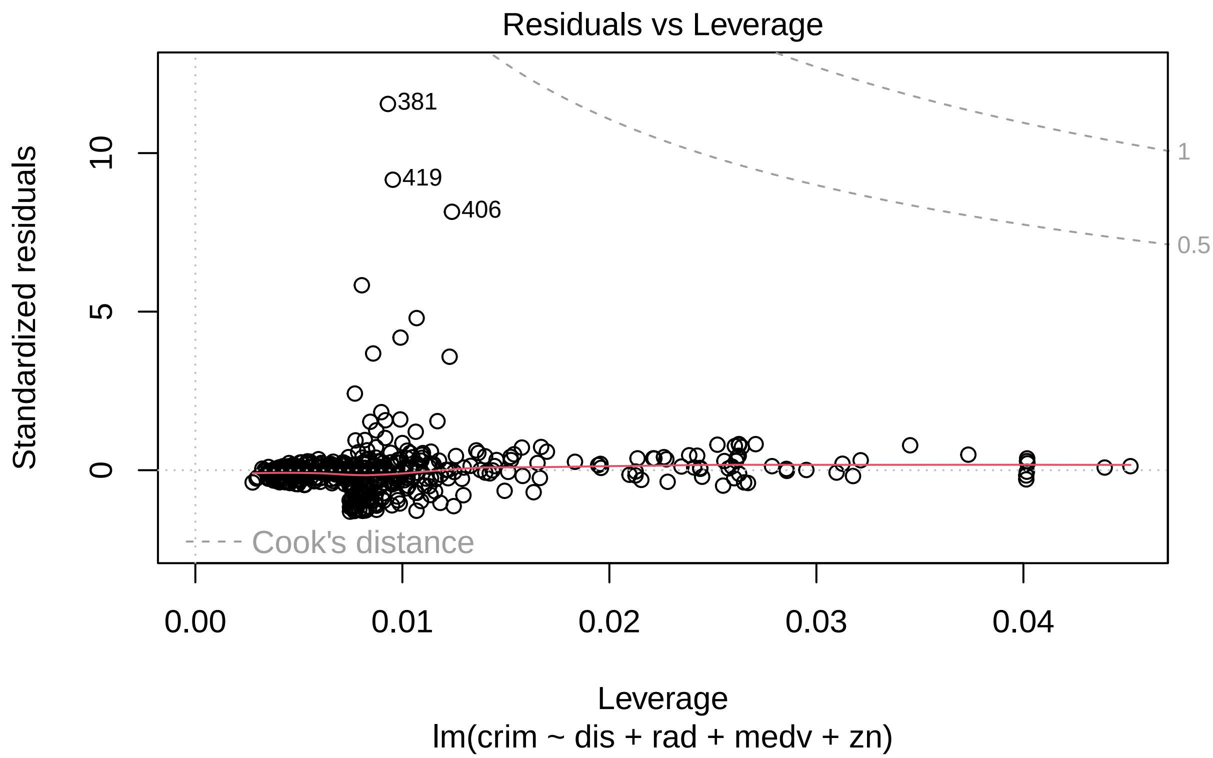
```r
plot(Boston.red.lm)
```


Residuals vs Fitted
Residuals
80
60
40
20
0
381
419
406
−5
0
5
10
15
Fitted values
lm(crim ~ dis + rad + medv + zn)


Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(crim ~ dis + rad + medv + zn)
381
419
406
−3
−2
−1
0
1
2
3
0
2
4
6
8
10
12



Scale−Location
381
419
406
√|Standardized residuals|
0.0
1.0
2.0
3.0
−5
0
5
10
15
Fitted values
lm(crim ~ dis + rad + medv + zn)




```
#Ridge
Boston.x <- model.matrix(crim ~ .,Boston)[,-1]
Boston.y <- Boston$crim
Boston.lambda <- 10^seq(10, -2, length = 100)

Boston.train <- sample(1:nrow(Boston),nrow(Boston)/2)
Boston.test <- (-Boston.train)
Boston.ytest <- Boston.y[Boston.test]

Boston.ridge.mod <- glmnet(Boston.x,Boston.y,alpha = 0,lambda = Boston.lambda)
predict(Boston.ridge.mod,s=0, type = 'coefficients')
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept) 13.4473681349
## zn           0.0452630596
## indus       -0.0605142483
## chas        -0.8223648009
## nox         -9.7788562633
## rm           0.6281672462
## age         -0.0008916473
## dis         -1.0038844850
## rad          0.6051810433
## tax         -0.0034139067
## ptratio     -0.2990248708
## lstat        0.1401134551
```

```
## medv        -0.2178870802
```

```r
Boston.ridge.mod <- glmnet(Boston.x[Boston.train,], Boston.y[Boston.train], alpha = 0, lambda = Boston.l
Boston1.cv.out <- cv.glmnet(Boston.x[Boston.train,],Boston.y[Boston.train],alpha=0)
summary(Boston.ridge.mod)
```

```
##           Length Class     Mode
## a0         100   -none-    numeric
## beta      1200   dgCMatrix S4
## df         100   -none-    numeric
## dim          2   -none-    numeric
## lambda     100   -none-    numeric
## dev.ratio  100   -none-    numeric
## nulldev      1   -none-    numeric
## npasses      1   -none-    numeric
## jerr         1   -none-    numeric
## offset       1   -none-    logical
## call         5   -none-    call
## nobs         1   -none-    numeric
```

```r
Boston1.bestlam <- Boston1.cv.out$lambda.min

Boston.ridge.pred <- predict(Boston.ridge.mod,s=Boston1.bestlam,newx = Boston.x[Boston.test,])
Boston.s.pred <- predict(Boston.lm,newdata = Boston[Boston.test,])
summary(Boston.ridge.pred)
```

```
##        s1
##  Min.   :-3.14605
##  1st Qu.: 0.08169
##  Median : 1.21295
##  Mean   : 3.36367
##  3rd Qu.: 3.82836
##  Max.   :14.84202
```

```r
summary(Boston$crim)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  0.00632  0.08204  0.25651  3.61352  3.67708 88.97620
```

```r
mean((Boston.s.pred - Boston.ytest)^2)
```

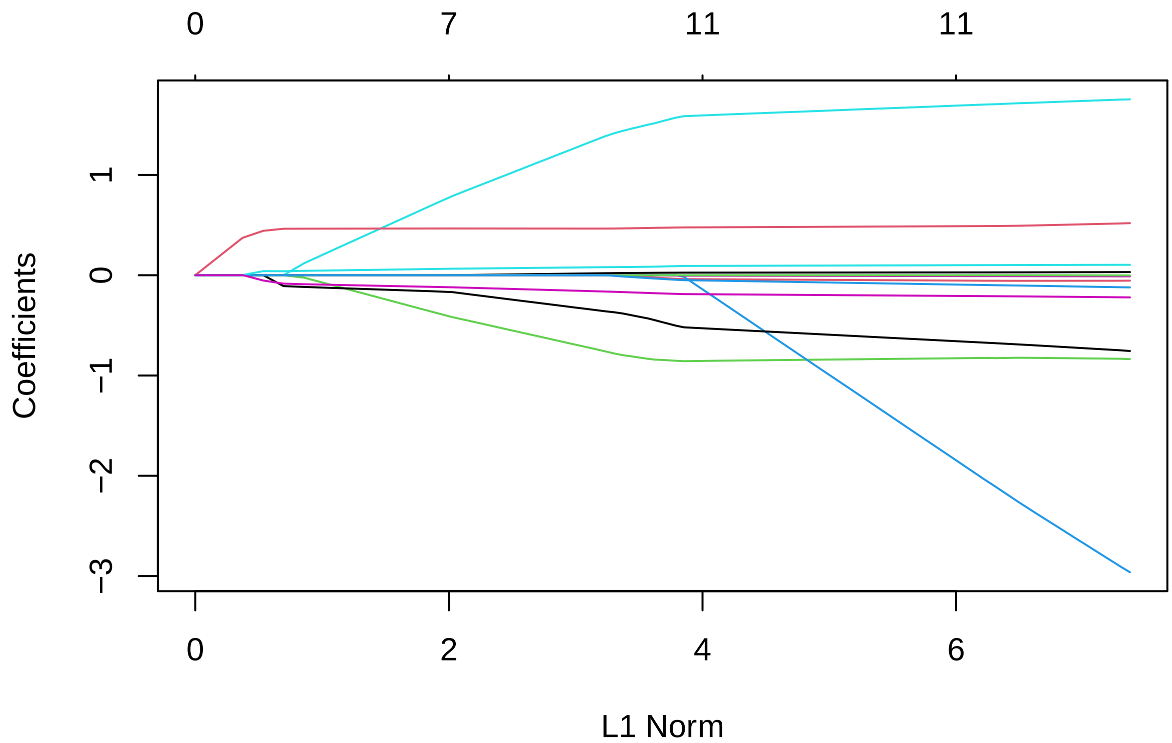
```
## [1] 47.13302
```

```r
mean((Boston.ridge.pred - Boston.ytest)^2)
```

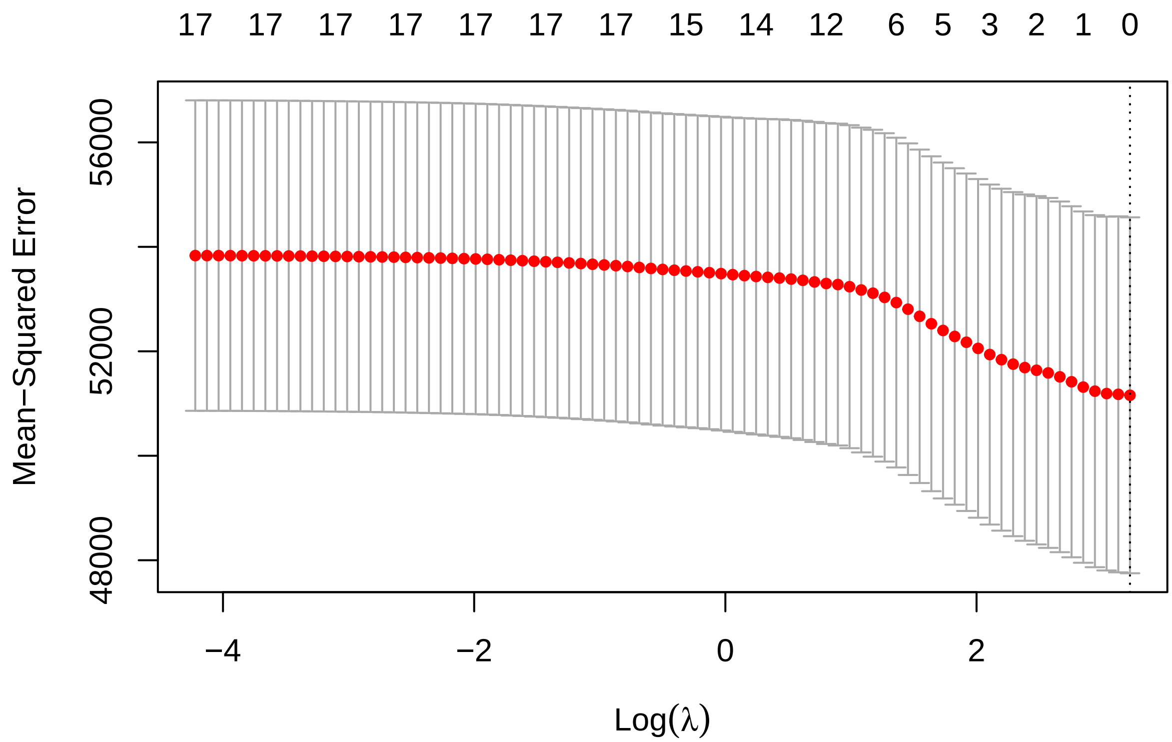
```
## [1] 50.61527
```

```r
#Lasso
Boston.lasso.mod <- glmnet(Boston.x[Boston.train,],Boston.y[Boston.train],alpha = 1,lambda = grid)
plot(Boston.lasso.mod)
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```




```
Boston2.cv.out <- cv.glmnet(Boston.x[Boston.train,],Boston.y[Boston.train],alpha=1)
plot(cv.out)
```




```
Boston2.bestlam <- Boston2.cv.out$lambda.min
bestlam
```

```
## [1] 25.07386
```

```
Boston.lasso.pred <- predict(Boston.lasso.mod, s=Boston2.bestlam, newx = Boston.x[Boston.test,])
#Boston.lasso.pred

mean((Boston.lasso.pred - Boston.ytest)^2)
```

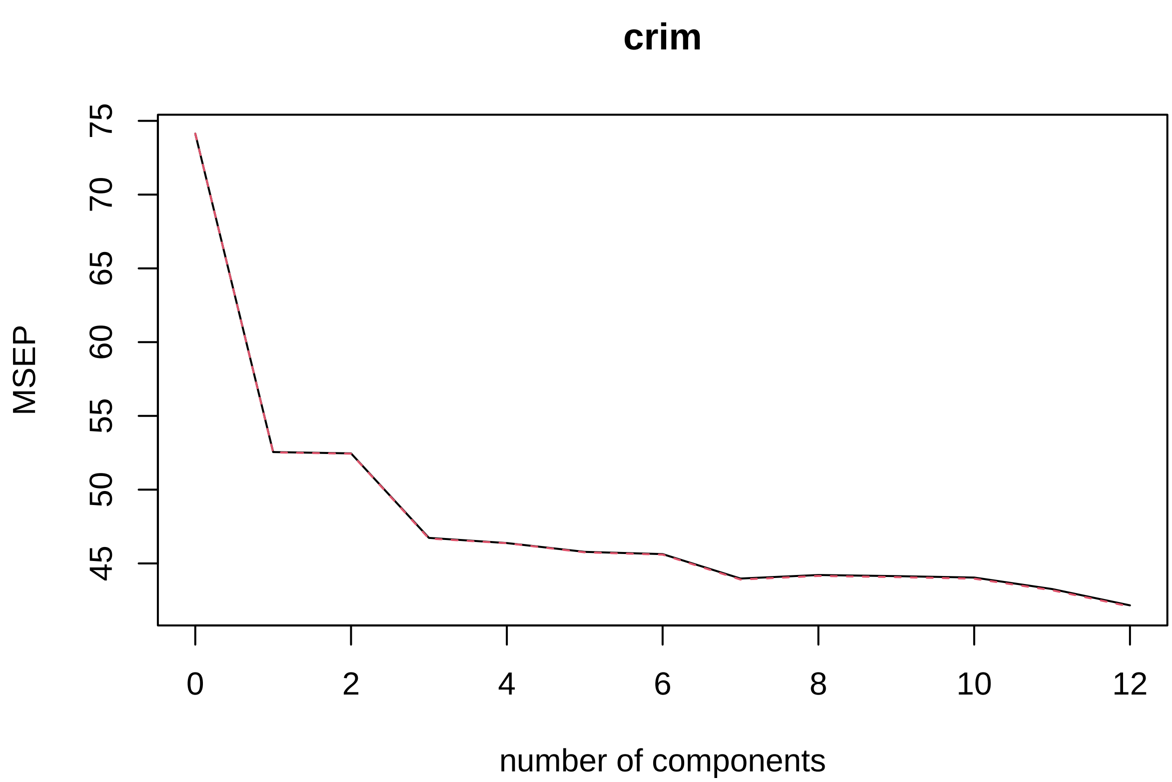
```
## [1] 50.00845
```

```
#PCR
Boston.pcr.fit <- pcr(crim ~ .,data = Boston,scale = TRUE, validation = "CV")
summary(Boston.pcr.fit)
```

```
## Data:    X dimension: 506 12
##  Y dimension: 506 1
## Fit method: svdpc
## Number of components considered: 12
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
```

```
## CV            8.61    7.249    7.243    6.836     6.81    6.766    6.755
## adjCV         8.61    7.247    7.241    6.833     6.81    6.764    6.752
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps
## CV       6.631    6.649    6.644     6.636     6.578     6.493
## adjCV    6.626    6.645    6.639     6.631     6.571     6.487
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       49.93    63.64    72.94    80.21    86.83    90.26    92.79    94.99
## crim    29.39    29.55    37.39    37.85    38.85    39.23    41.73    41.82
##       9 comps  10 comps  11 comps  12 comps
## X       96.78     98.33     99.48    100.00
## crim    42.12     42.43     43.58     44.93
```

```r
validationplot(Boston.pcr.fit,val.type = "MSEP")
```




```r
Boston.pcr.pred <- predict(Boston.pcr.fit,Boston.x[Boston.test,],ncomp = 12)
#Boston.pcr.pred

Boston.pcr.fit1 <- pcr(Boston.y ~ Boston.x,scale = TRUE,ncomp =12)
summary(Boston.pcr.fit1$fitted.values)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -8.0696 -0.2981  1.4915  3.6135  8.4130 17.7592
```

```r
mean((Boston.pcr.pred - Boston.ytest)^2)
```

```
## [1] 47.13302
```

#Problem 11b - The PCR model seems to more accurately fit the data and is slightly improved from the linear model. The PCR model also better describes the data based on plots of the Boston crime rate per capita.

#Problem 11c - The PCR model does contain all components of the data set because that is where we achieve the most variability accounted for in Boston crime rate per capita.