PQHS471 HW 2

Gregory Powers (gxp145)

Chapter 5 Q9

(A)

```
library(MASS)
library(Matrix)
library(knitr)
library(kableExtra)
library(MVN)
library(corrplot)
attach(Boston)
set.seed(1)
medv.Mean <- mean(medv)
medv.Mean
```

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

(B) As I am learning R, I'm going to do this a few ways.

```
sum(medv > 0)
```

```
## [1] 506
```

```
length(medv)
```

```
## [1] 506
```

```
medv.Error <- sd(medv)/sqrt(506)
medv.Error
```

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

```
print(sd(medv)/sqrt(length(medv)))
```

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

(C)

```
library(boot)
mean.fn <- function (x ,id) {
           return(mean(x[id]))
}
boot.M <- boot(medv, mean.fn, 1000)
boot.M
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = medv, statistic = mean.fn, R = 1000)
##
##
## Bootstrap Statistics :
```

```
##     original        bias    std. error
## t1* 22.53281 0.008517589   0.4119374
```

```r
boot.SD <- sd(boot.M$t)/length(t)
boot.SD - medv.Error
```

```
## [1] 0.003076292
```

The difference between the bootstrapped estimate and the original is about 0.01.

(D)

```r
c(boot.M$t0 - 2 *sd(boot.M$t), boot.M$t0 + 2 *sd(boot.M$t))
```

```
## [1] 21.70893 23.35668
```

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

```
##
##  One Sample t-test
##
## data:  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
```

(E)

```r
medv.Median <- median(Boston$medv); medv.Median
```

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

(F)

```r
median.fn <- function (x ,id) {
           return(median(x[id]))
}

boot.Median <- boot(medv, median.fn, 1000)
boot.Median
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = medv, statistic = median.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*     21.2 -0.0098   0.3874004
```

The estimated SE of the median is 0.3801

(G)

```r
print(medv.muTen <- quantile(medv, 0.1))
```

```
##   10%
## 12.75
```

```r
quantile.fn <- function (x ,id) {
           return(quantile(x[id], 0.1))
}

boot.Quantile10 <- boot(medv, quantile.fn, 1000)
boot.Quantile10
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = medv, statistic = quantile.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    12.75 0.00515   0.5113487
```

The estimated SE is 0.4826.

Chapter 6 Q9

(A)

```r
library(ISLR)
detach(Boston)
attach(College)
```

```r
library(dplyr)
summary(College)
```

```
##  Private        Apps           Accept          Enroll       Top10perc
##  No :212   Min.   :   81   Min.   :   72   Min.   :  35   Min.   : 1.00
##  Yes:565   1st Qu.:  776   1st Qu.:  604   1st Qu.: 242   1st Qu.:15.00
##            Median : 1558   Median : 1110   Median : 434   Median :23.00
##            Mean   : 3002   Mean   : 2019   Mean   : 780   Mean   :27.56
##            3rd Qu.: 3624   3rd Qu.: 2424   3rd Qu.: 902   3rd Qu.:35.00
##            Max.   :48094   Max.   :26330   Max.   :6392   Max.   :96.00
##    Top25perc      F.Undergrad     P.Undergrad         Outstate
##  Min.   :  9.0   Min.   :  139   Min.   :    1.0   Min.   : 2340
##  1st Qu.: 41.0   1st Qu.:  992   1st Qu.:   95.0   1st Qu.: 7320
##  Median : 54.0   Median : 1707   Median :  353.0   Median : 9990
##  Mean   : 55.8   Mean   : 3700   Mean   :  855.3   Mean   :10441
##  3rd Qu.: 69.0   3rd Qu.: 4005   3rd Qu.:  967.0   3rd Qu.:12925
##  Max.   :100.0   Max.   :31643   Max.   :21836.0   Max.   :21700
##    Room.Board       Books           Personal         PhD
##  Min.   :1780   Min.   :  96.0   Min.   : 250   Min.   :  8.00
##  1st Qu.:3597   1st Qu.: 470.0   1st Qu.: 850   1st Qu.: 62.00
```

```
##  Median :4200   Median : 500.0   Median :1200   Median : 75.00  
##  Mean   :4358   Mean   : 549.4   Mean   :1341   Mean   : 72.66  
##  3rd Qu.:5050   3rd Qu.: 600.0   3rd Qu.:1700   3rd Qu.: 85.00  
##  Max.   :8124   Max.   :2340.0   Max.   :6800   Max.   :103.00  
##     Terminal       S.F.Ratio      perc.alumni        Expend     
##  Min.   : 24.0   Min.   : 2.50   Min.   : 0.00   Min.   : 3186  
##  1st Qu.: 71.0   1st Qu.:11.50   1st Qu.:13.00   1st Qu.: 6751  
##  Median : 82.0   Median :13.60   Median :21.00   Median : 8377  
##  Mean   : 79.7   Mean   :14.09   Mean   :22.74   Mean   : 9660  
##  3rd Qu.: 92.0   3rd Qu.:16.50   3rd Qu.:31.00   3rd Qu.:10830  
##  Max.   :100.0   Max.   :39.80   Max.   :64.00   Max.   :56233  
##    Grad.Rate     
##  Min.   : 10.00  
##  1st Qu.: 53.00  
##  Median : 65.00  
##  Mean   : 65.46  
##  3rd Qu.: 78.00  
##  Max.   :118.00
```

```r
str(College)
```

```
## 'data.frame':    777 obs. of  18 variables:
##  $ Private    : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Apps       : num  1660 2186 1428 417 193 ...
##  $ Accept     : num  1232 1924 1097 349 146 ...
##  $ Enroll     : num  721 512 336 137 55 158 103 489 227 172 ...
##  $ Top10perc  : num  23 16 22 60 16 38 17 37 30 21 ...
##  $ Top25perc  : num  52 29 50 89 44 62 45 68 63 44 ...
##  $ F.Undergrad: num  2885 2683 1036 510 249 ...
##  $ P.Undergrad: num  537 1227 99 63 869 ...
##  $ Outstate   : num  7440 12280 11250 12960 7560 ...
##  $ Room.Board : num  3300 6450 3750 5450 4120 ...
##  $ Books      : num  450 750 400 450 800 500 500 450 300 660 ...
##  $ Personal   : num  2200 1500 1165 875 1500 ...
##  $ PhD        : num  70 29 53 92 76 67 90 89 79 40 ...
##  $ Terminal   : num  78 30 66 97 72 73 93 100 84 41 ...
##  $ S.F.Ratio  : num  18.1 12.2 12.9 7.7 11.9 9.4 11.5 13.7 11.3 11.5 ...
##  $ perc.alumni: num  12 16 30 37 2 11 26 37 23 15 ...
##  $ Expend     : num  7041 10527 8735 19016 10922 ...
##  $ Grad.Rate  : num  60 56 54 59 15 55 63 73 80 52 ...
```

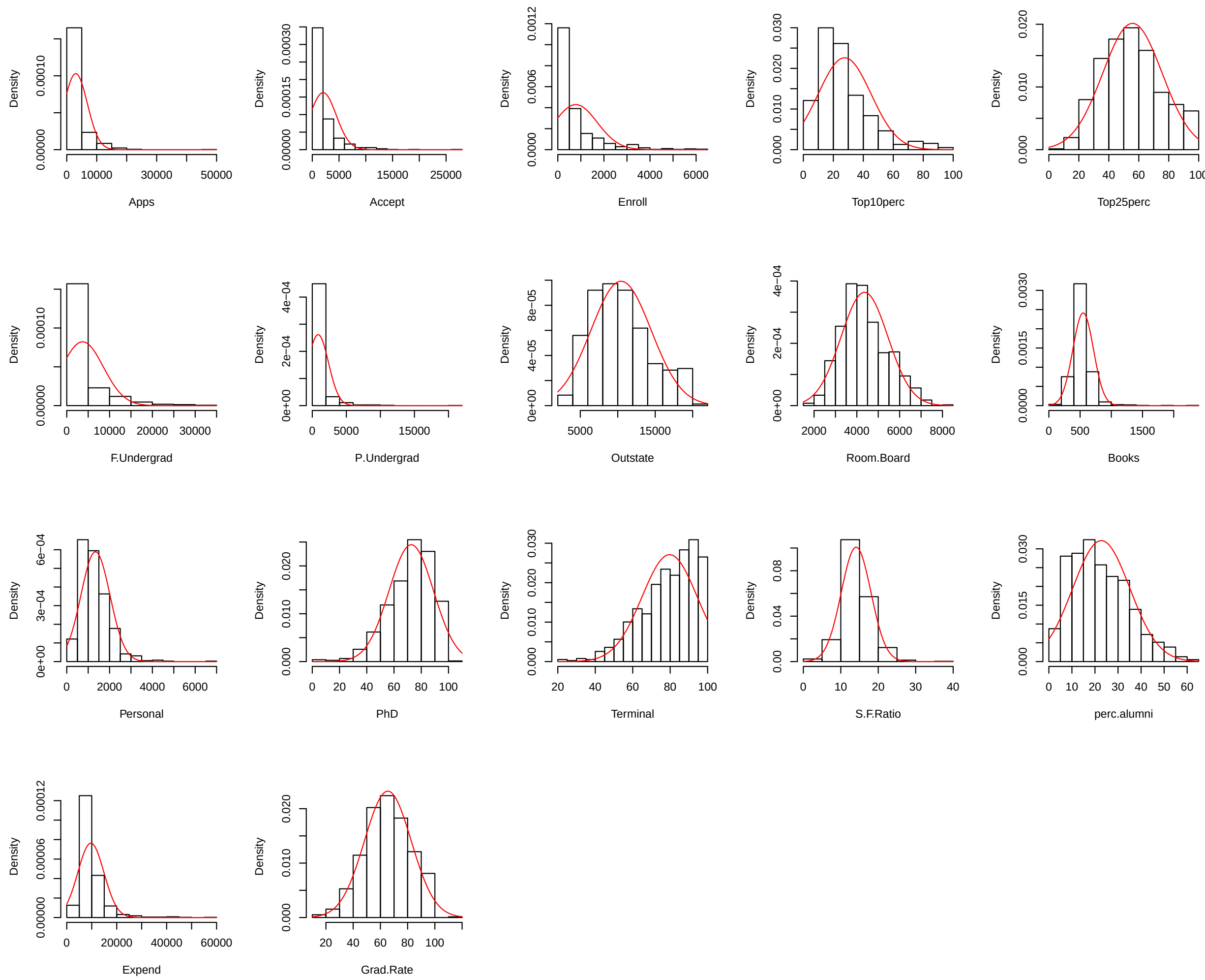
```r
anyDuplicated(College)
```

```
## [1] 0
```

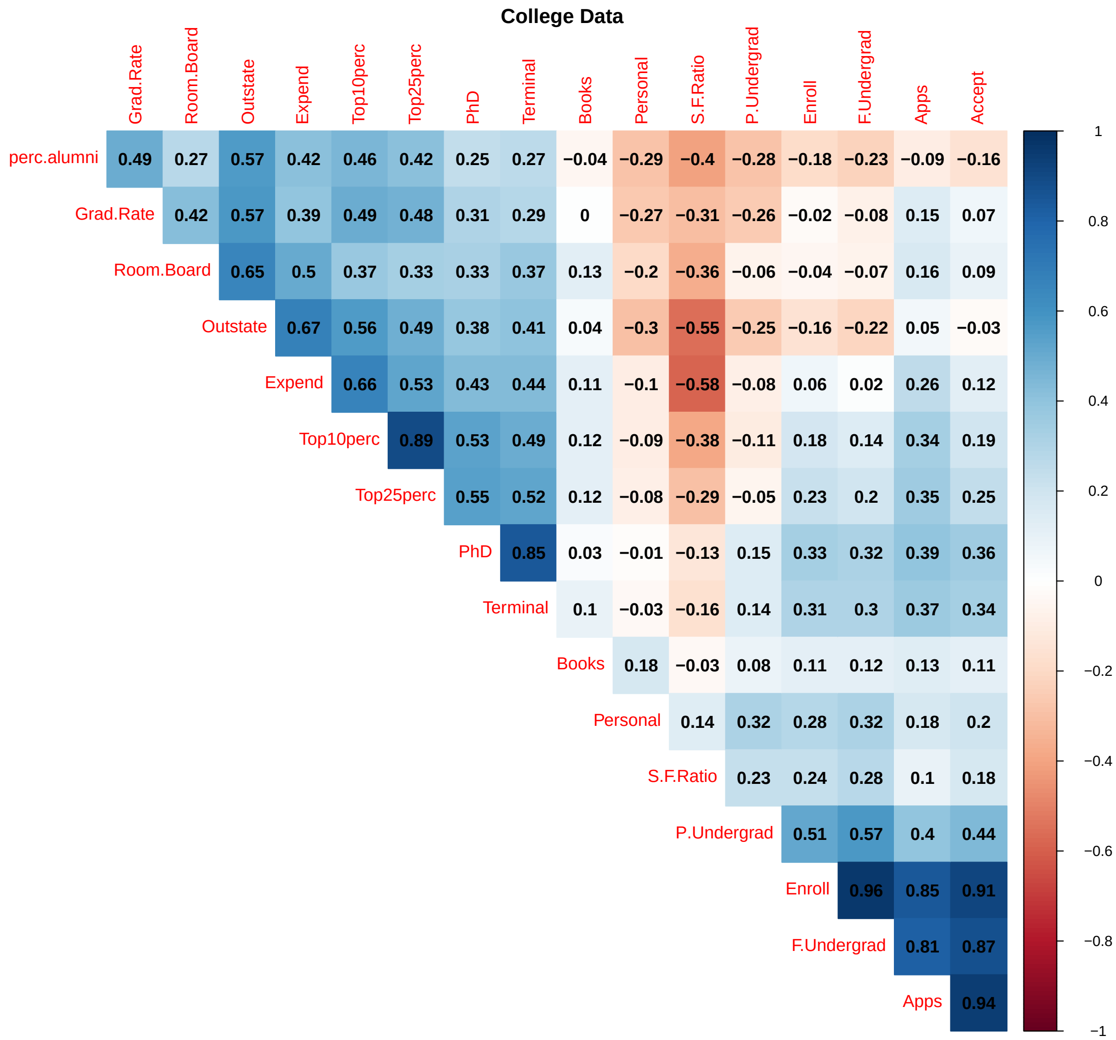
```r
sum(is.na(College))
```

```
## [1] 0
```

```r
uniPlot(College[2:18], type = "histogram")
```




```r
r <- cor(College[2:18])
title <- 'College Data'
corrplot(r, method = "color", type = 'upper', diag = FALSE, addCoef.col = "black",
         order = "hclust", title = title, mar=c(0,0,1,0))
```



Using dplyr to split into test and train.

```
col.train <- sample_frac(College, 0.8)
col.test = setdiff(College, col.train)
nrow(col.train) + nrow(col.test) == nrow(College)
```

```
## [1] TRUE
```

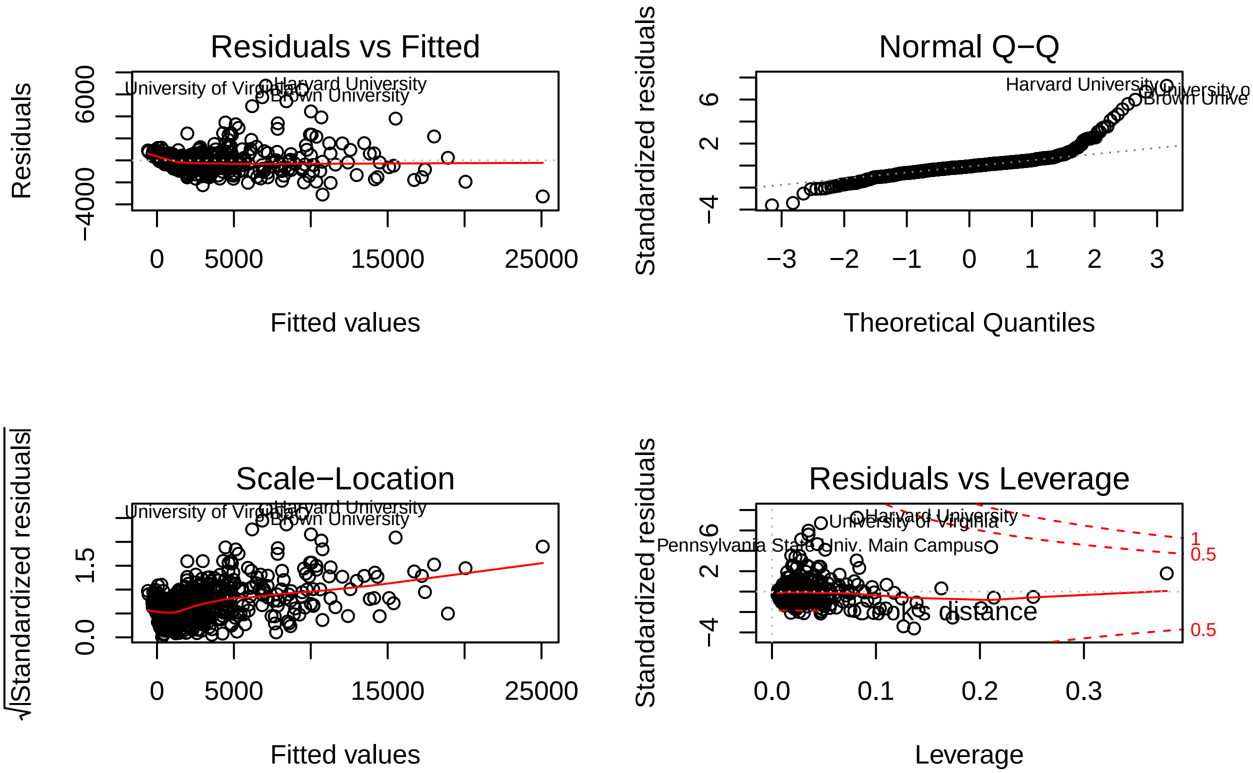
(B)

```
lm.Apps <- lm(Apps ~ ., data = col.train)
summary(lm.Apps)
```

```
##
## Call:
## lm(formula = Apps ~ ., data = col.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3280.5  -436.5   -72.1   291.8  6792.1
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) -4.573e+02  4.208e+02  -1.087 0.277607    
## PrivateYes  -6.156e+02  1.470e+02  -4.187 3.24e-05 ***
## Accept       1.309e+00  5.261e-02  24.878  < 2e-16 ***
## Enroll      -4.010e-01  2.045e-01  -1.961 0.050346 .  
## Top10perc    3.967e+01  6.017e+00   6.592 9.46e-11 ***
## Top25perc   -9.843e+00  4.846e+00  -2.031 0.042688 *  
## F.Undergrad  8.041e-02  3.461e-02   2.323 0.020501 *  
## P.Undergrad  3.213e-02  3.148e-02   1.021 0.307793    
## Outstate    -6.510e-02  2.054e-02  -3.170 0.001603 ** 
## Room.Board   1.777e-01  5.032e-02   3.531 0.000446 ***
## Books        5.272e-03  2.436e-01   0.022 0.982742    
## Personal    -1.720e-02  6.511e-02  -0.264 0.791696    
## PhD         -6.635e+00  5.033e+00  -1.318 0.187896    
## Terminal    -5.347e+00  5.466e+00  -0.978 0.328344    
## S.F.Ratio    1.980e+00  1.409e+01   0.141 0.888253    
## perc.alumni -7.617e+00  4.308e+00  -1.768 0.077519 .  
## Expend       8.415e-02  1.261e-02   6.672 5.72e-11 ***
## Grad.Rate    1.310e+01  3.268e+00   4.009 6.87e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 977.5 on 604 degrees of freedom
## Multiple R-squared:  0.9292, Adjusted R-squared:  0.9272 
## F-statistic:   466 on 17 and 604 DF,  p-value: < 2.2e-16
```

```r
par(mfrow=c(2,2))
plot(lm.Apps)
```

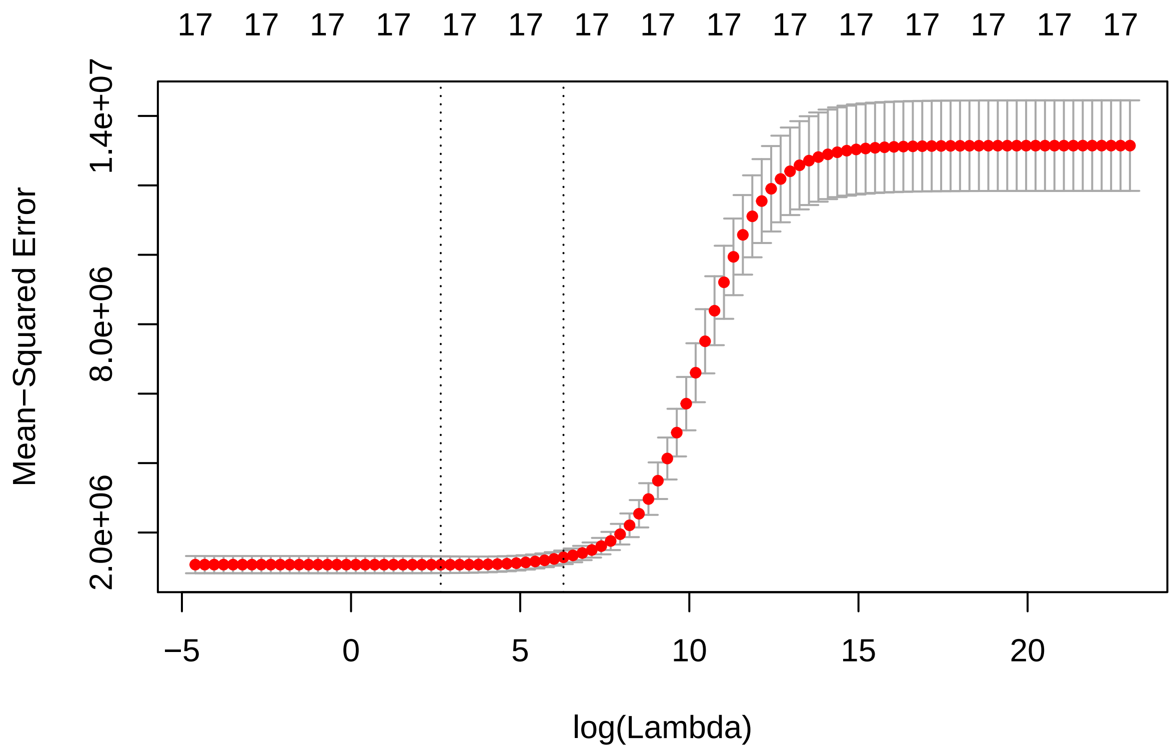



```
lm.pred <- predict(lm.Apps, col.test)
lm.er <- mean((col.test[, 'Apps'] - lm.pred)^2) #doing it the book way once. Referencing cols by [] is
```

Test RSS = 944829

(C)

```
library(glmnet)
library(foreach)
train.mat <- model.matrix(Apps ~ ., data = col.train) # Reminder to self: no missing [,-1]
test.mat <- model.matrix(Apps ~ ., data = col.test) # [,-1]
grid = 10^seq(10, -2, length = 100)
ridge.cv <- cv.glmnet(train.mat, col.train$Apps, alpha = 0, lambda = grid, thresh = 1e-12) # can also r
plot(ridge.cv)
```




```
ridge.cv$lambda.min
```

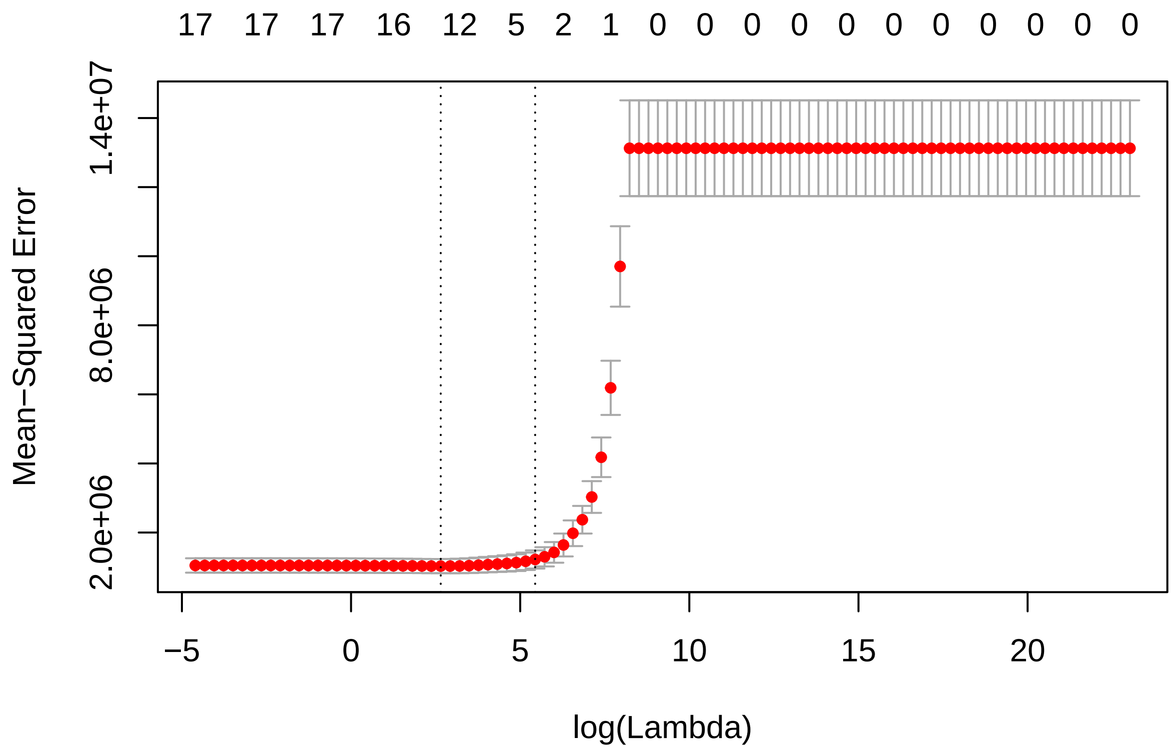
```
## [1] 14.17474
```

```
ridge.pred <- predict(ridge.cv, newx = test.mat, s = ridge.cv$lambda.min)
ridge.er <- mean((col.test$Apps - ridge.pred)^2)
```

The RSS improves only slightly.

(D)

```
lass.cv <- cv.glmnet(train.mat, col.train$Apps, alpha = 1, lambda = grid, thresh = 1e-12)
plot(lass.cv)
```



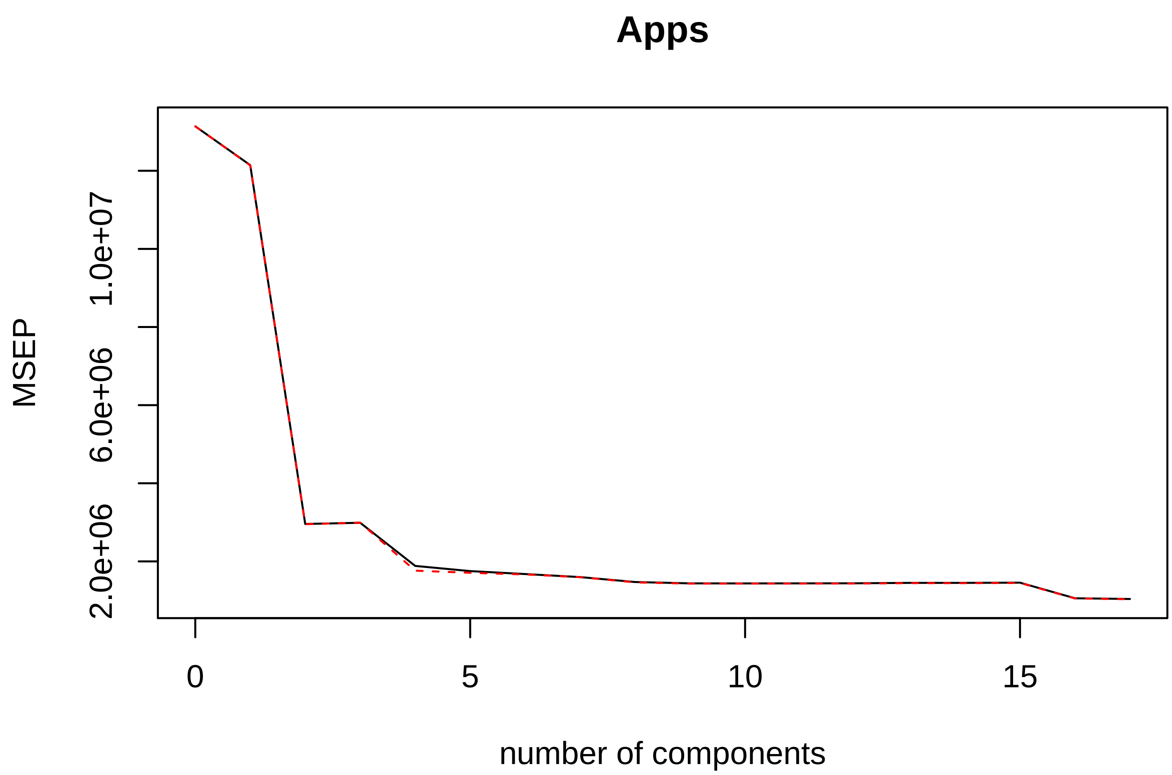

```r
lass.pred <- predict(lass.cv, s = lass.cv$lambda.min, newx = test.mat)
lass.er <- mean((col.test$Apps - lass.pred)^2)
mod.lass = glmnet(model.matrix(Apps~., data=College), College$Apps, alpha=1)
mod.lass.p <-predict(mod.lass, s=lass.cv$lambda.min, type="coefficients"); mod.lass.p
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept) -556.01427783
## (Intercept)    .
## PrivateYes  -458.50058120
## Accept         1.49947647
## Enroll        -0.34285887
## Top10perc     39.10796866
## Top25perc     -6.17240648
## F.Undergrad    .
## P.Undergrad    0.03388810
## Outstate      -0.06812801
## Room.Board     0.13486936
## Books          .
## Personal       0.01173185
## PhD           -6.71936715
## Terminal      -3.15826314
## S.F.Ratio      8.61896397
## perc.alumni   -0.72038115
## Expend         0.07249557
## Grad.Rate      6.28499717
```

The test RSS is 2136982. There are 14 nonzero coefficient estimates, though some are quite small.

(E)

```
library(pls)
pcr.mod <- pcr(Apps ~ ., data = col.train, scale = TRUE, validation = "CV")
validationplot(pcr.mod, val.type = 'MSEP')
```




```
summary(pcr.mod)
```

```
## Data:    X dimension: 622 17
##  Y dimension: 622 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            3625     3485     1720     1729     1373     1323     1294
## adjCV         3625     3484     1718     1730     1327     1307     1291
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1264     1213     1199      1198      1199      1200      1204
## adjCV     1263     1209     1196      1196      1197      1197      1201
##        14 comps  15 comps  16 comps  17 comps
## CV         1204      1206      1028      1019
## adjCV      1201      1203      1025      1016
##
## TRAINING: % variance explained
```

```
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X      31.540    57.13    64.18    69.91    75.36    80.53    84.52
## Apps    7.907    77.82    77.90    87.08    87.61    87.77    88.38
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X       88.04    90.96     93.17     95.27     97.07     98.16     98.94
## Apps    89.24    89.70     89.74     89.82     89.86     89.86     89.88
##       15 comps  16 comps  17 comps
## X        99.45     99.86    100.00
## Apps     90.17     92.60     92.92
```

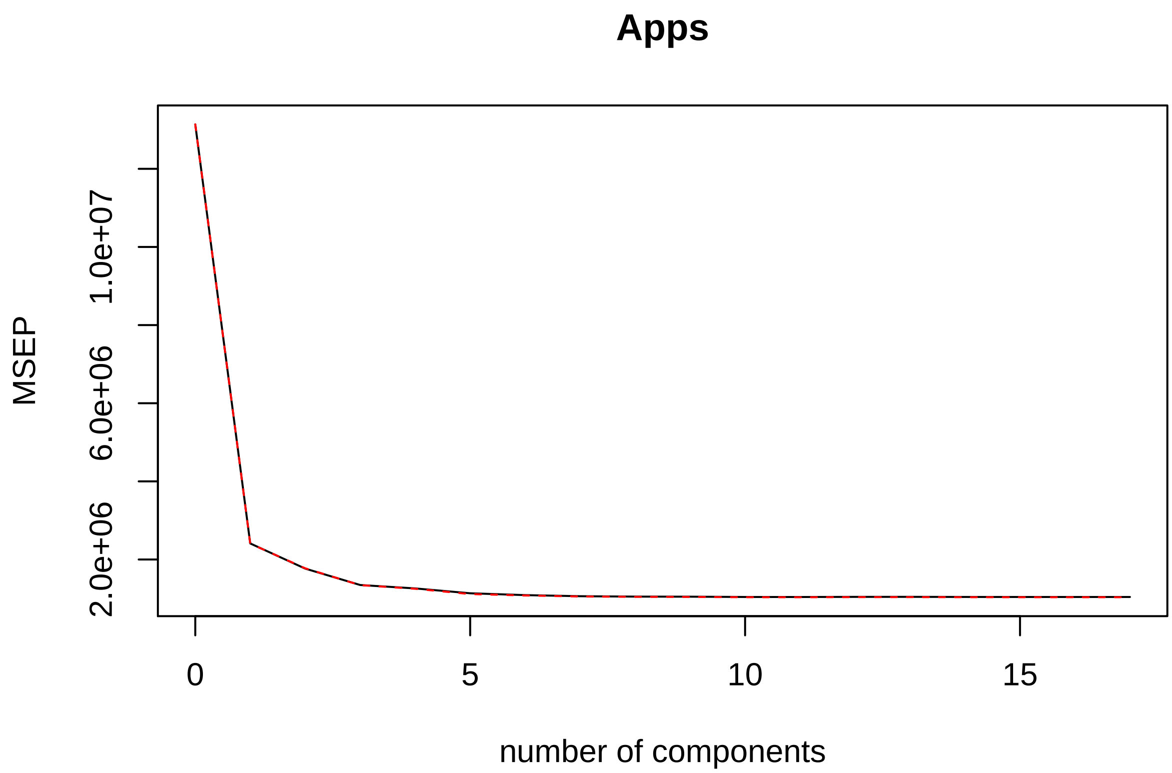
```
pcr.pred <- predict(pcr.mod, col.test, ncomp = 17)
pcr.er <- mean((col.test$Apps - pcr.pred)^2)
```

M = 17 (no reduction in dimensions), test RSS = 1969505

```
pls.mod <- plsr(Apps ~ ., data = col.train, scale = TRUE, validation = "CV")
summary(pls.mod)
```

```
## Data:    X dimension: 622 17
##  Y dimension: 622 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            3625     1553     1329     1161     1122     1065     1043
## adjCV         3625     1551     1332     1159     1118     1056     1038
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1029     1025     1023      1020      1020      1021      1021
## adjCV     1026     1022     1021      1017      1017      1018      1018
##        14 comps  15 comps  16 comps  17 comps
## CV         1020      1020      1020      1020
## adjCV      1017      1017      1017      1017
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       26.32    47.39    62.60    65.86    68.51    73.43    76.87
## Apps    81.95    86.86    90.23    91.21    92.35    92.71    92.80
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X       80.61    83.59     85.50     88.59     91.60     94.26     96.03
## Apps    92.83    92.86     92.89     92.90     92.91     92.91     92.92
##       15 comps  16 comps  17 comps
## X        96.87     98.81    100.00
## Apps     92.92     92.92     92.92
```

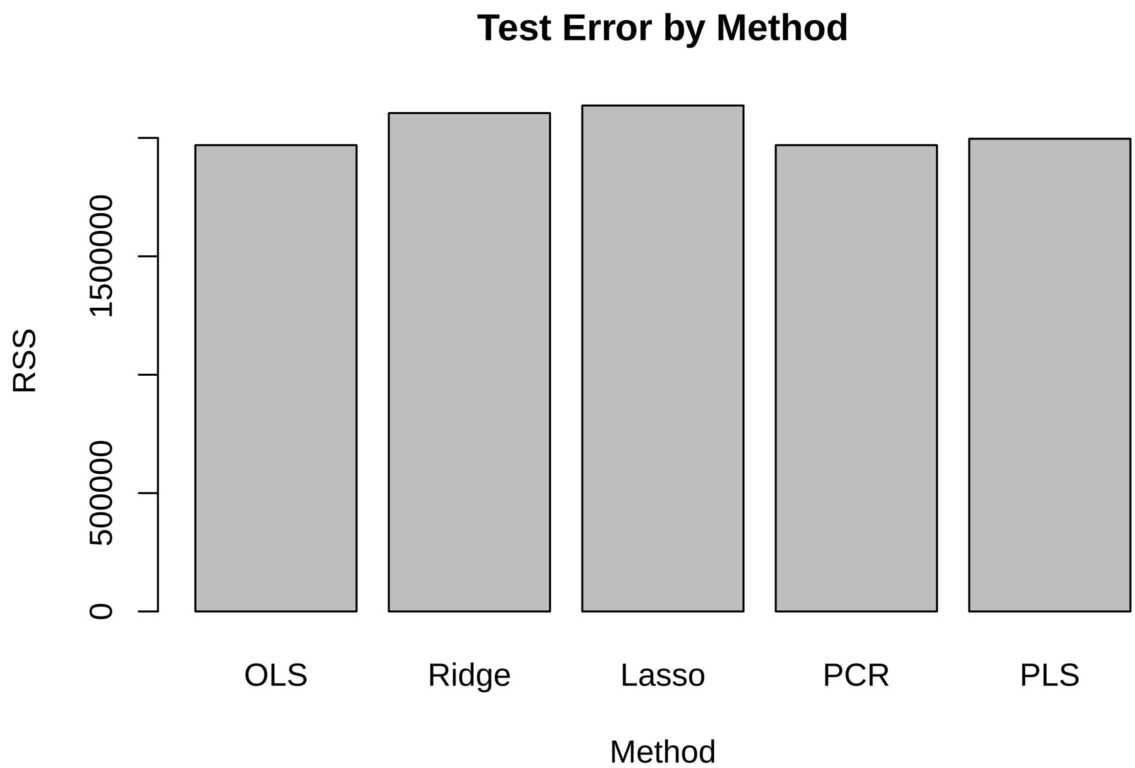
```
validationplot(pls.mod, val.type = 'MSEP')
```



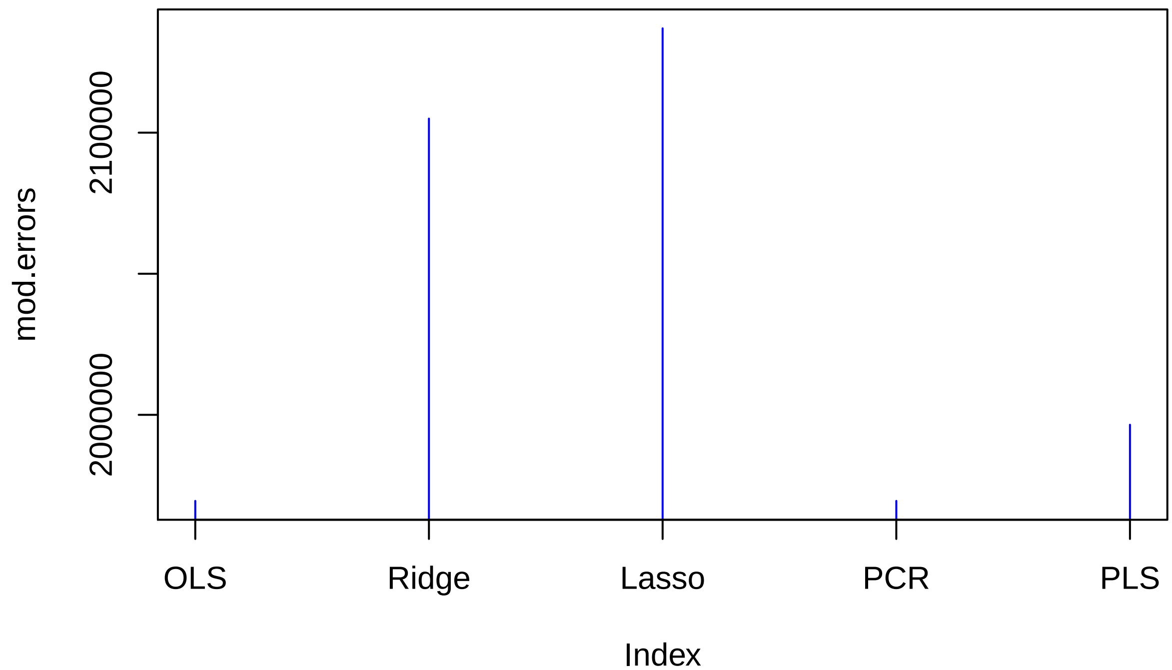

```
pls.pred <- predict(pls.mod, col.test, ncomp = 10)
pls.er <- mean((col.test$Apps - pls.pred)^2)
```

Min CV is where M = 10. Test error RSS = 1996495

```
mod.errors <- c(lm.er, ridge.er, lass.er, pcr.er, pls.er)
names(mod.errors) <- c('OLS', 'Ridge', 'Lasso', 'PCR', 'PLS')
barplot(mod.errors, main = 'Test Error by Method', xlab = 'Method', ylab = 'RSS')
```




```
plot(mod.errors, type = 'h', col='blue', xaxt='n')
axis(1, at=1:5, lab=c('OLS', 'Ridge', 'Lasso', 'PCR', 'PLS'))
```



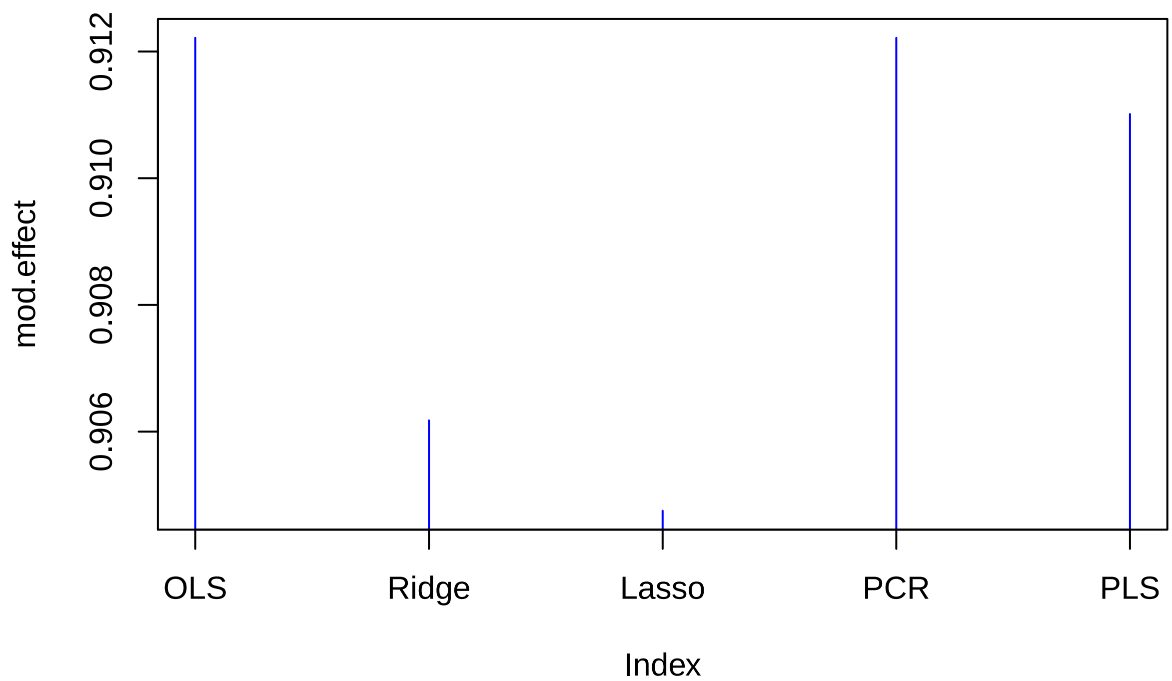

```
kable(mod.errors, col.names = 'RSS')%>%
  kable_styling()
```

	RSS
OLS	1969505
Ridge	2104991
Lasso	2136982
PCR	1969505
PLS	1996495

```
#The idea to do the below came from an R forum
t.avg <- mean(col.test$Apps)
ols.r2 = 1 - mean((lm.pred - col.test$Apps)^2) / mean((t.avg - col.test$Apps)^2)
ridge.r2 = 1 - mean((ridge.pred - col.test$Apps)^2) / mean((t.avg - col.test$Apps)^2)
lass.r2 = 1 - mean((lass.pred - col.test$Apps)^2) / mean((t.avg - col.test$Apps)^2)
pcr.r2 = 1 - mean((pcr.pred - col.test$Apps)^2) / mean((t.avg - col.test$Apps)^2)
pls.r2 = 1 - mean((pls.pred - col.test$Apps)^2) / mean((t.avg - col.test$Apps)^2)

mod.effect <- c(ols.r2, ridge.r2, lass.r2, pcr.r2, pls.r2)
names(mod.effect) <- c('OLS', 'Ridge', 'Lasso', 'PCR', 'PLS')

plot(mod.effect, type = 'h', col='blue', xaxt='n')
axis(1, at=1:5, lab=c('OLS', 'Ridge', 'Lasso', 'PCR', 'PLS'))
```




```
kable(mod.effect, col.names = 'r2')%>%
  kable_styling()
```

	r2
OLS	0.9122158
Ridge	0.9061770
Lasso	0.9047511
PCR	0.9122158
PLS	0.9110128

There is very little difference in RSS and all models account for the variation in applications quite well (r2 < .9). OLS followed by PLS produce the models with the smallest test error and largest r2, though the absolute differences are tiny. That PCR is not considered: as no dimensions were reduced, it is equivalent to OLS.

Chapter 6 Q11

(A)

```
#Splitting into test and train via dplyr. Making test Mats for Ridge & Lasso
set.seed(16565)
detach(College)
attach(Boston)
bos.train <- sample_frac(Boston, 0.8)
```

```r
bos.test = setdiff(Boston, bos.train)
nrow(bos.train) + nrow(bos.test) == nrow(Boston)
```

```
## [1] TRUE
```

```r
rownames(bos.train) <- c()
rownames(bos.test) <- c()
mat.train <- model.matrix(crim ~ . , data = bos.train)[,-1]
mat.test <- model.matrix(crim ~ ., data=bos.test)[,-1]
summary(Boston)
```

```
##       crim                zn             indus            chas
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000
##  1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000
##       nox               rm             age              dis
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127
##       rad              tax           ptratio          black
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90
##      lstat            medv
##  Min.   : 1.73   Min.   : 5.00
##  1st Qu.: 6.95   1st Qu.:17.02
##  Median :11.36   Median :21.20
##  Mean   :12.65   Mean   :22.53
##  3rd Qu.:16.95   3rd Qu.:25.00
##  Max.   :37.97   Max.   :50.00
```

```r
str(Boston)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
```

```
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

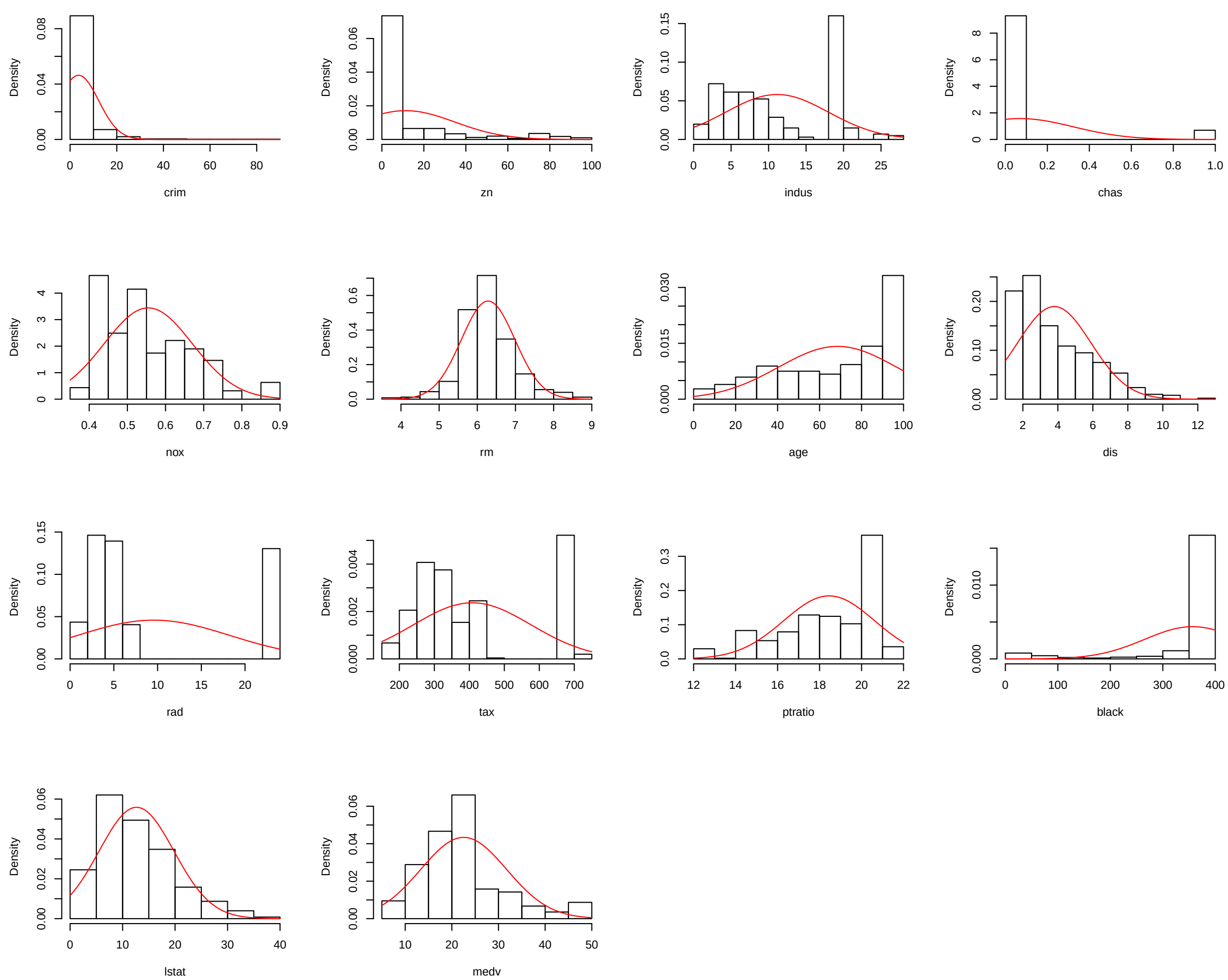
```r
anyDuplicated(Boston)
```

```
## [1] 0
```

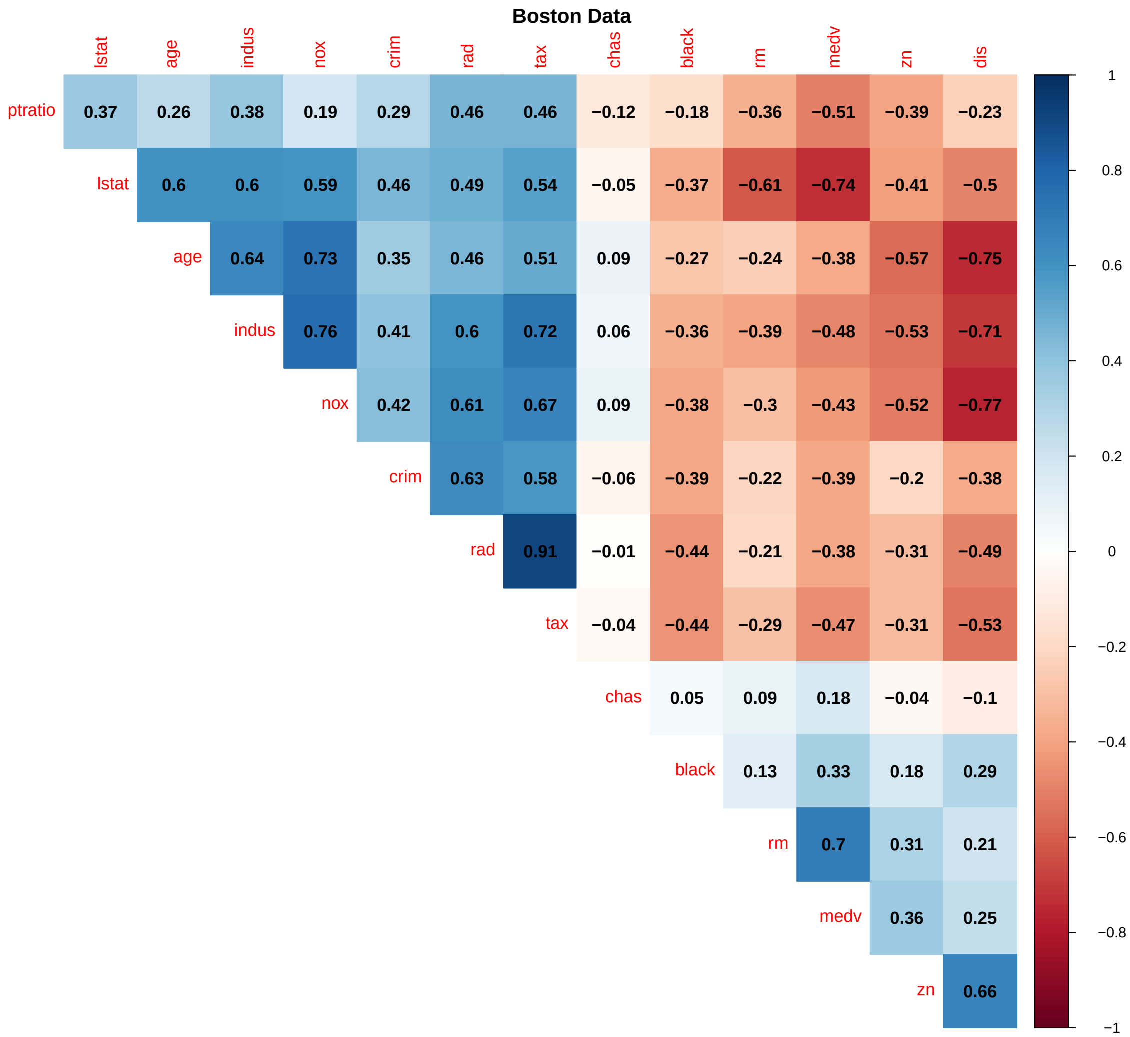
```r
sum(is.na(Boston))
```

```
## [1] 0
```

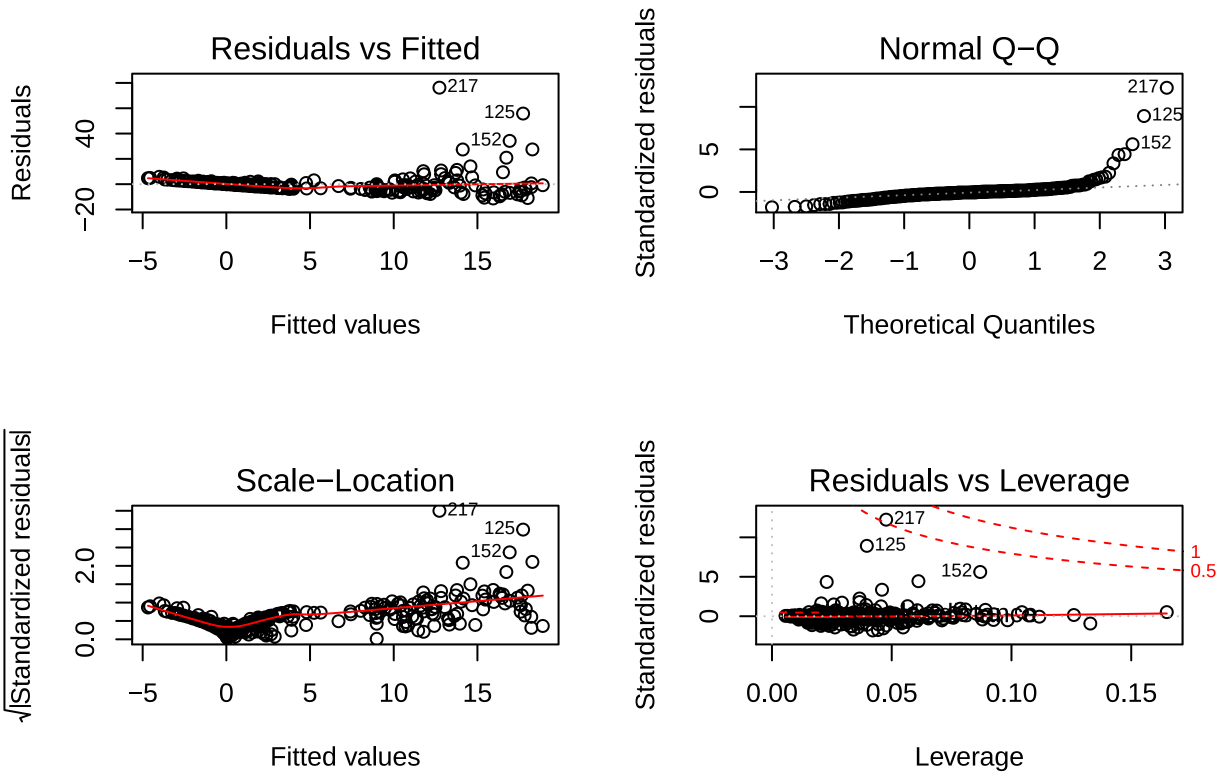
```r
uniPlot(Boston, type = "histogram")
```




```r
r <- cor(Boston)
title <- 'Boston Data'
corrplot::corrplot(r, method = "color", type = 'upper', diag = FALSE, addCoef.col = "black",
            order = "hclust", title = title, mar=c(0,0,1,0))
```



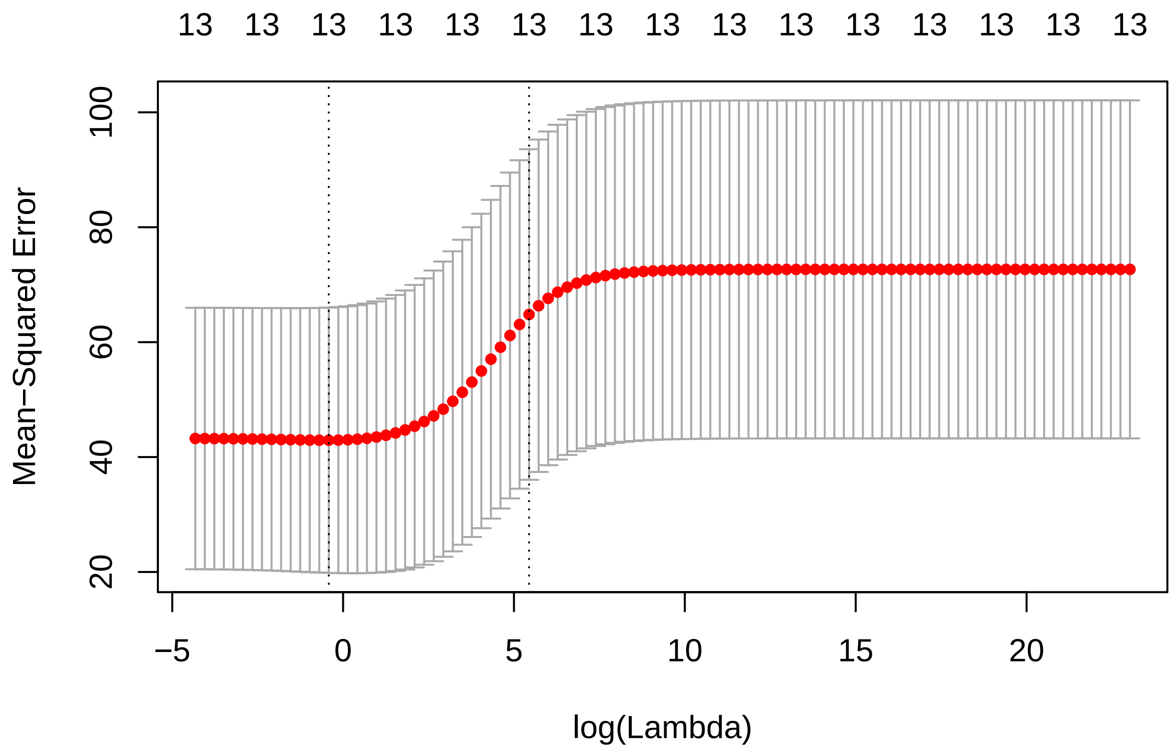

```r
lm.fit <- lm(crim ~ ., data = bos.train)
par(mfrow=c(2,2))
plot(lm.fit)
```




```
lm.pred <- predict(lm.fit, bos.test)
lm.er <- mean((bos.test$crim - lm.pred)^2); lm.er
```

```
## [1] 45.90214
```

```
grid = 10^seq(10, -2, length = 100)
ridge.fit <- cv.glmnet(mat.train, bos.train$crim, alpha = 0, lambda = grid, thresh = 1e-12)
plot(ridge.fit)
```




```r
ridge.fit$lambda.min
```

```
## [1] 0.6579332
```

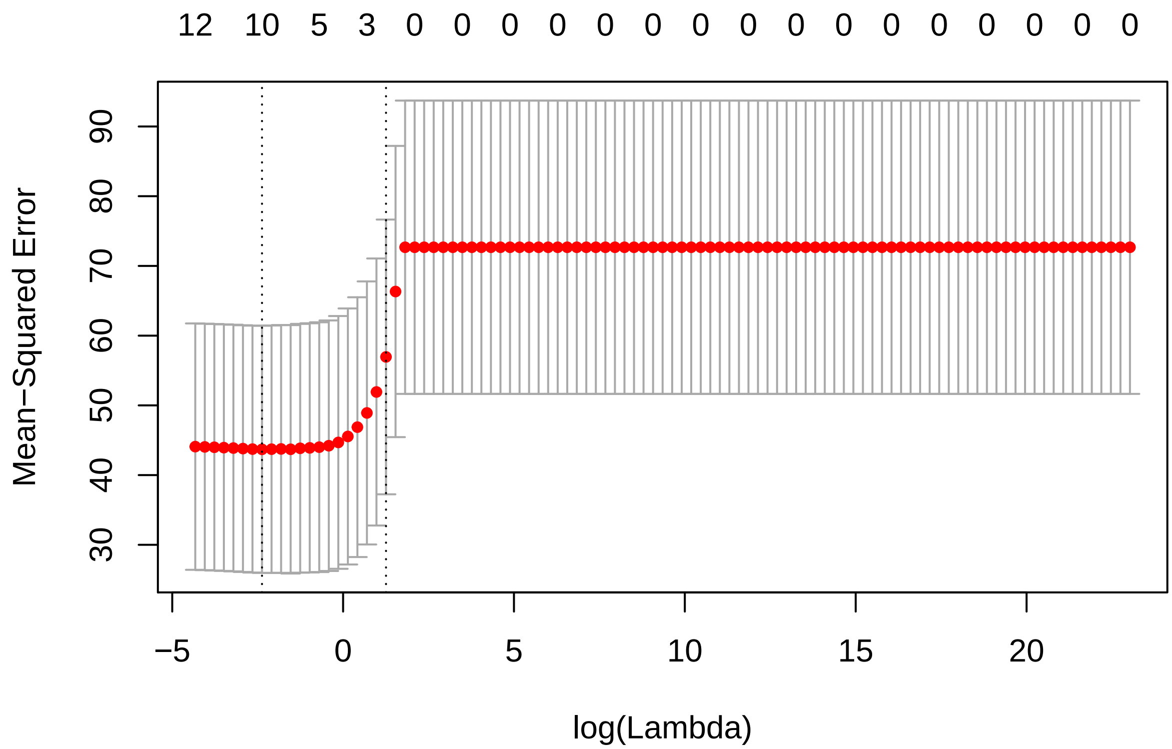
```r
ridge.pred <- predict(ridge.fit, newx = mat.test, s = ridge.fit$lambda.min)
ridge.er <- mean((bos.test$crim - ridge.pred)^2); ridge.er
```

```
## [1] 47.24654
```

```r
predict(ridge.fit, s = ridge.fit$lambda.min, type = "coefficients")
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  9.674512475
## zn           0.026115285
## indus       -0.061593845
## chas        -0.874509207
## nox         -4.886593189
## rm           0.211496504
## age          0.002191499
## dis         -0.633453558
## rad          0.387455204
## tax          0.003302351
## ptratio     -0.066763814
## black       -0.014351994
## lstat        0.117045867
## medv        -0.096758718
```

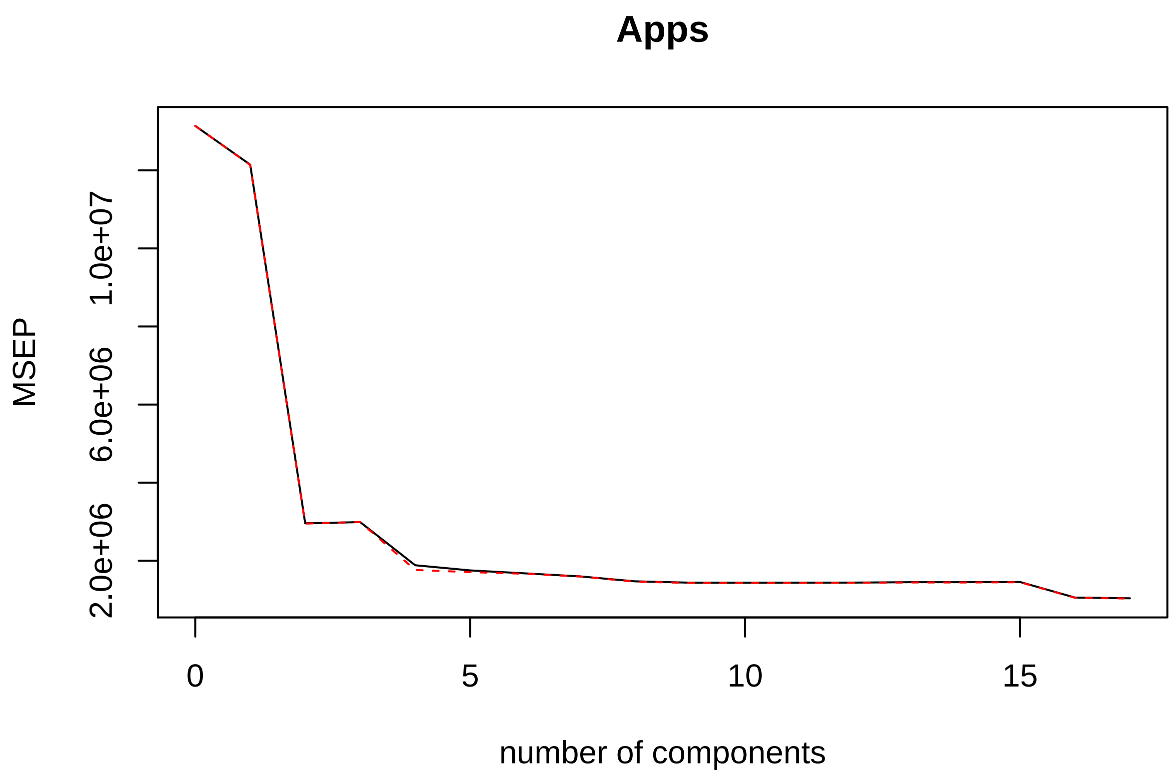
```r
lass.fit = cv.glmnet(mat.train, bos.train$crim, alpha = 1, lambda = grid, thresh = 1e-12)
plot(lass.fit)
```




```r
lass.pred = predict(lass.fit, s = lass.fit$lambda.min, newx = mat.test)
lass.er = mean((bos.test$crim - lass.pred)^2)
predict(lass.fit, s = lass.fit$lambda.min, type = "coefficients")
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) 10.25170685
## zn           0.02477619
## indus       -0.03161461
## chas        -0.71055271
## nox         -4.10731848
## rm           .
## age          .
## dis         -0.56965751
## rad          0.46463310
## tax          .
## ptratio     -0.04737249
## black       -0.01407236
## lstat        0.09664408
## medv        -0.08298947
```

```r
library(pls)
pcr.fit <- pcr(crim ~ ., data = bos.train, scale = TRUE, validation = "CV")
validationplot(pcr.mod, val.type = 'MSEP')
```




```r
summary(pcr.mod)
```

```
## Data:    X dimension: 622 17
##  Y dimension: 622 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            3625     3485     1720     1729     1373     1323     1294
## adjCV         3625     3484     1718     1730     1327     1307     1291
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1264     1213     1199      1198      1199      1200      1204
## adjCV     1263     1209     1196      1196      1197      1197      1201
##        14 comps  15 comps  16 comps  17 comps
## CV         1204      1206      1028      1019
## adjCV      1201      1203      1025      1016
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X      31.540    57.13    64.18    69.91    75.36    80.53    84.52
## Apps    7.907    77.82    77.90    87.08    87.61    87.77    88.38
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X       88.04    90.96     93.17     95.27     97.07     98.16     98.94
```

```
## Apps    89.24    89.70     89.74     89.82     89.86     89.86     89.88
##       15 comps  16 comps  17 comps
## X        99.45     99.86    100.00
## Apps     90.17     92.60     92.92
```

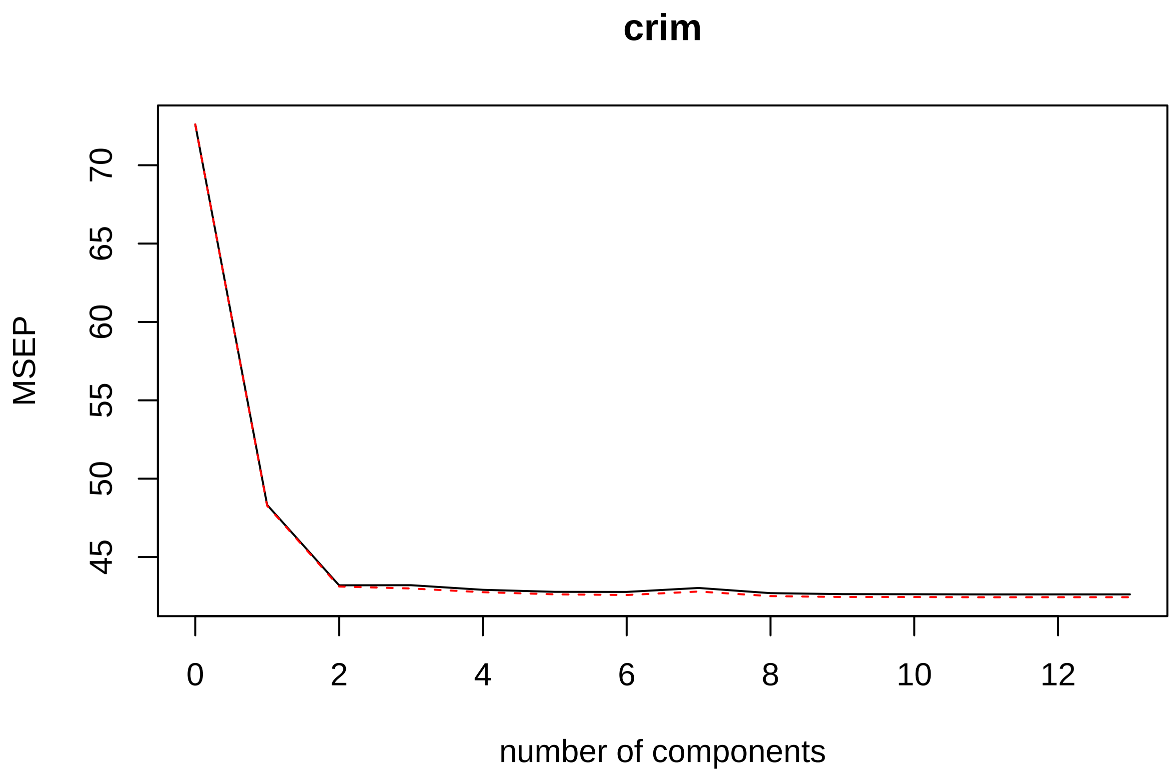
```r
pcr.pred <- predict(pcr.fit, bos.test, ncomp = 10)
pcr.er <- mean((bos.test$crim - pcr.pred)^2); pcr.er
```

```
## [1] 49.54782
```

```r
pls.fit <- plsr(crim ~ ., data = bos.train, scale = TRUE, validation = "CV")
summary(pls.fit)
```

```
## Data:    X dimension: 405 13
##  Y dimension: 405 1
## Fit method: kernelpls
## Number of components considered: 13
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           8.521    6.951    6.573    6.573    6.551    6.541    6.540
## adjCV        8.521    6.948    6.567    6.557    6.539    6.529    6.525
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       6.560    6.534    6.530     6.529     6.528     6.528     6.528
## adjCV    6.542    6.520    6.516     6.515     6.514     6.515     6.515
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       47.89    57.48    61.20    71.80    77.24    80.44    82.82
## crim    34.64    42.65    44.53    44.87    45.18    45.47    45.55
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X       86.54    90.25     94.84     97.26     98.51    100.00
## crim    45.57    45.57     45.57     45.57     45.57     45.57
```

```r
validationplot(pls.fit, val.type = 'MSEP')
```

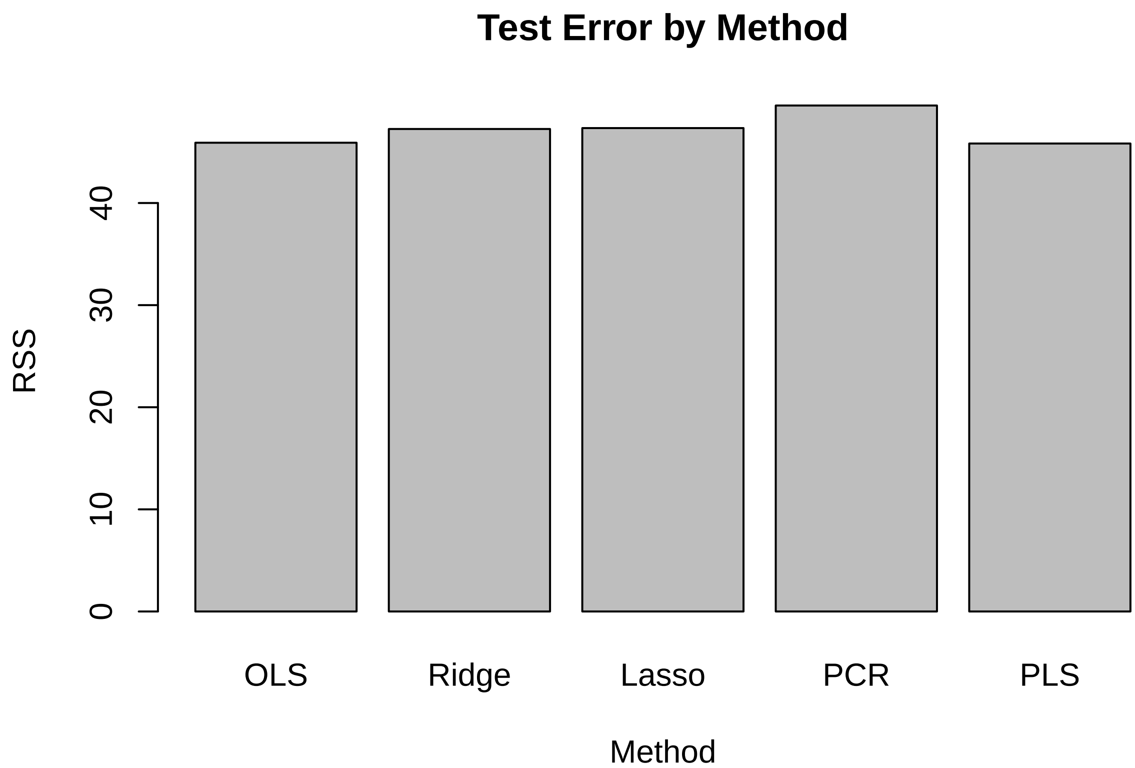



```r
pls.pred <- predict(pls.fit, bos.test, ncomp = 9)
pls.er <- mean((bos.test$crim - pls.pred)^2); pls.er
```

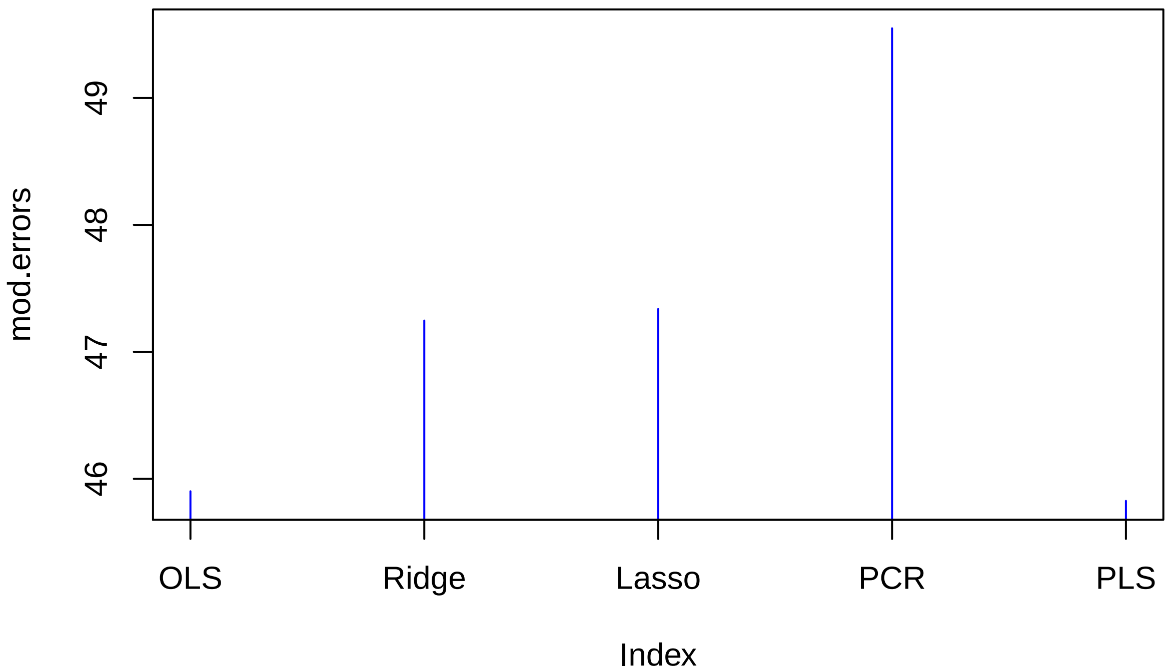
```
## [1] 45.82656
```

(B)

```r
mod.errors <- c(lm.er, ridge.er, lass.er, pcr.er, pls.er)
names(mod.errors) <- c('OLS', 'Ridge', 'Lasso', 'PCR', 'PLS')
barplot(mod.errors, main = 'Test Error by Method', xlab = 'Method', ylab = 'RSS')
```




```r
plot(mod.errors, type = 'h', col='blue', xaxt='n')
axis(1, at=1:5, lab=c('OLS', 'Ridge', 'Lasso', 'PCR', 'PLS'))
```

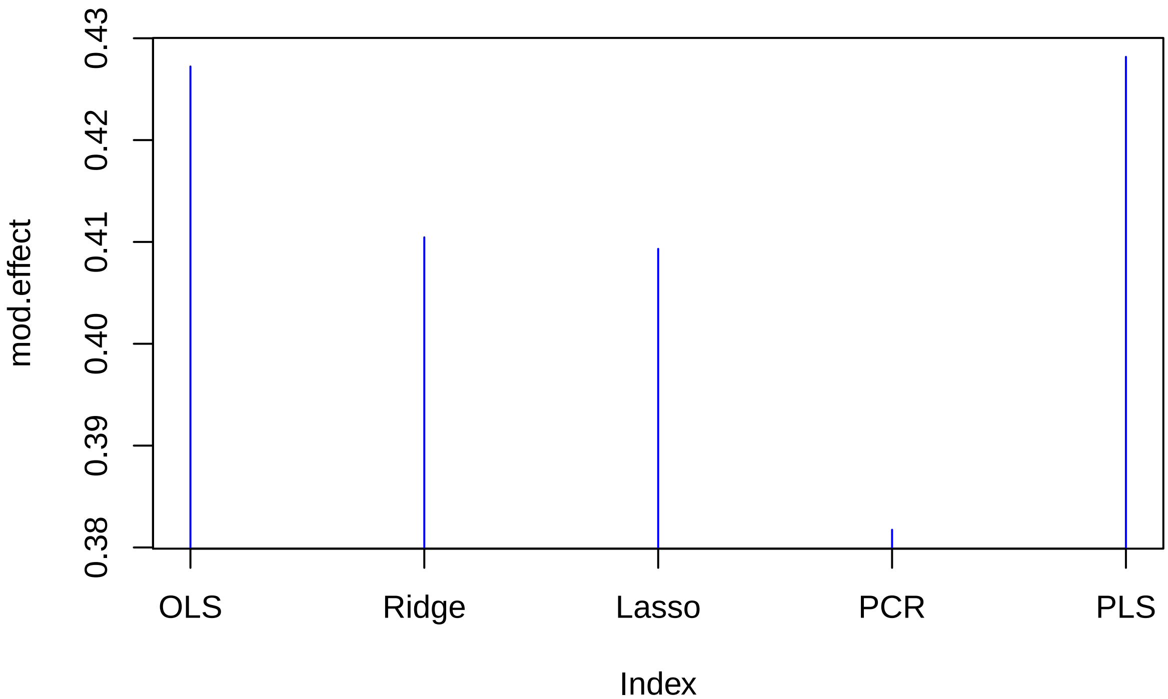



```
kable(mod.errors, col.names = 'RSS')%>%
  kable_styling()
```

	RSS
OLS	45.90214
Ridge	47.24654
Lasso	47.33744
PCR	49.54782
PLS	45.82656

```
#Again, calculating all the r2 in this manner was inspired by a forum post
t.avg <- mean(bos.test$crim)
ols.r2 = 1 - mean((lm.pred - bos.test$crim)^2) / mean((t.avg - bos.test$crim)^2)
ridge.r2 = 1 - mean((ridge.pred - bos.test$crim)^2) / mean((t.avg - bos.test$crim)^2)
lass.r2 = 1 - mean((lass.pred - bos.test$crim)^2) / mean((t.avg - bos.test$crim)^2)
pcr.r2 = 1 - mean((pcr.pred - bos.test$crim)^2) / mean((t.avg - bos.test$crim)^2)
pls.r2 = 1 - mean((pls.pred - bos.test$crim)^2) / mean((t.avg - bos.test$crim)^2)

mod.effect <- c(ols.r2, ridge.r2, lass.r2, pcr.r2, pls.r2)
names(mod.effect) <- c('OLS', 'Ridge', 'Lasso', 'PCR', 'PLS')
plot(mod.effect, type = 'h', col='blue', xaxt='n')
axis(1, at=1:5, lab=c('OLS', 'Ridge', 'Lasso', 'PCR', 'PLS'))
```




```
kable(mod.effect, col.names = 'r2')%>%
  kable_styling()
```

	r2
OLS	0.4272306
Ridge	0.4104551
Lasso	0.4093208
PCR	0.3817395
PLS	0.4281737

OLS was used as a basis for comparison. All of the models perform fairly similarly, save PCR which stood out as the worst model (i.e. largest test error, lowest r2). PLS had the smallest test error, though not smaller than OLS which had marginally better test error and r2. Of the methods from chapter 6 (excepting step wise and subset selection), the PLS regression model performs the best, accounting for 39% of the variation in test set crime.

(C)

The PLS model that performed best predicting Boston's crime fit 9 linear combinations of variables; while this reduced dimensions, it nonetheless utilized much of the information available. Thus, I would not remove any features.