Stats 202: HW 5 Solutions

Probem 1: Chapter 6, Exercise 1

(a) *Short answer* (sufficient for full credit). For $k = 0, \ldots, p$, the best subset model has equal or better training RSS than the other methods.
Long answer. For $k = 0$ and $k = p$, best subset, forward stepwise, and backward stepwise have the same model and therefore the same RSS.
For $k = 1$, best subset has the same model as forward stepwise and therefore the same RSS. Backward stepwise may have a different model, and a worse RSS.
For $k = p - 1$, best subset has the same model as backward stepwise and therefore the same RSS. Forward stepwise may have a different model, and a worse RSS.
For $k \in \{2, \ldots, p-2\}$, best subset has equal or better training RSS than forward stepwise and backward stepwise.

(b) *Short answer* (sufficient for full credit). Any of the three methods could have the smallest test RSS.
Long answer. As mentioned above, for $k = 0$ and $k = p$ all three models are the same and hence have the same test RSS. Likwise, for $k = 1$, best subset and forward stepwise have the same model, and for $k = p - 1$, best subset and backward stepwise have the same model. Apart from these cases, it is possible for any of the three methods to yield the best test RSS because randomness in the data results in randomness in the selected model and hence in the test RSS of the selected model.

(c) i. True, ii. True, iii. False, iv. False, v. False

Problem 2: Chapter 6, Exercise 3

Consider the lasso in its constraint form:

$$\begin{aligned} &\text{minimize} && \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 \\ &\text{subject to} && \sum_{j=1}^{p}|\beta_j| \leq s. \end{aligned}$$

As s increases from 0:

(a) Training RSS will (iv) steadily decrease
Justification: As s increases, the set of feasible solutions $\{\beta : \sum_j |\beta_j| \leq s\}$ becomes larger and hence the objective value (the training RSS) can only get smaller

(b) Test RSS will (ii) decrease initially and then start increasing in a U shape
Justification: By the usual considerations of the bias-variance tradeoff (see below)

(c) Variance will (iii) steadily increase
Justification: Increasing s results in a larger feasible set (as explained above), hence a more flexible model and larger variance

(d) Squared bias (iv) steadily decrease
Justification: Increasing s results in a larger feasible set (as explained above), hence a more flexible model and smaller bias

(e) Irreducible error will (v) remain constant
Justification: Irreducible error does not depend on the model, only on the distribution of the data

Problem 3: Chapter 6, Exercise 8

Part a

Generate a predictor:

```
set.seed(2000)
n = 100
x = rnorm(n)
epsilon = rnorm(n)
```

Part b

Generate a response variable (using $\beta_0 = \beta_1 = \beta_2 = \beta_3 = 1$):

```
beta = rep(1,4)
X = cbind(1, x, x^2, x^3)
y = X %*% beta + epsilon
```

Part c

Perform best subset selection using the `regsubsets` function. The selected models for each size are:

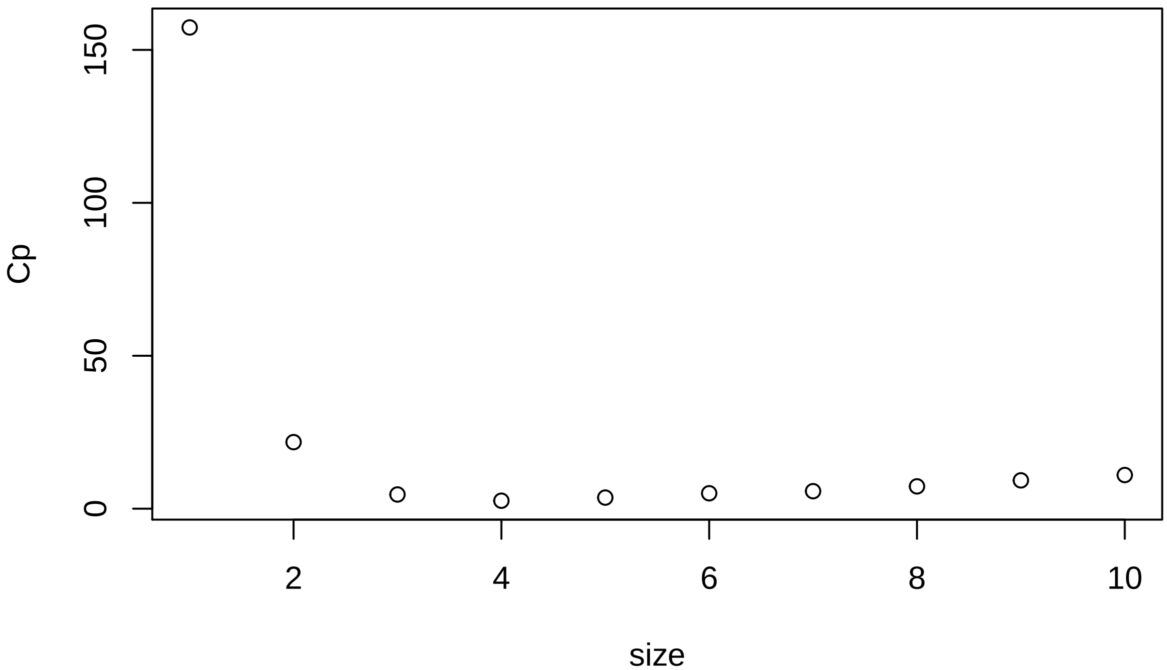
```
library(leaps)

features = data.frame(cbind(x, x^2, x^3, x^4, x^5, x^6, x^7, x^8, x^9, x^10))
names(features) = paste0('X^', 1:10)
bss = regsubsets(x=features, y=y, method='exhaustive', nvmax=10)
summary(bss)
```

```
## Subset selection object
## 10 Variables  (and intercept)
##      Forced in Forced out
## X^1      FALSE      FALSE
## X^2      FALSE      FALSE
## X^3      FALSE      FALSE
## X^4      FALSE      FALSE
## X^5      FALSE      FALSE
## X^6      FALSE      FALSE
## X^7      FALSE      FALSE
## X^8      FALSE      FALSE
## X^9      FALSE      FALSE
## X^10     FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: exhaustive
##           X^1 X^2 X^3 X^4 X^5 X^6 X^7 X^8 X^9 X^10
## 1  ( 1 )  " " " " "*" " " " " " " " " " " " " " " 
## 2  ( 1 )  " " " " "*" "*" " " " " " " " " " " " " 
## 3  ( 1 )  "*" "*" "*" " " " " " " " " " " " " " " 
## 4  ( 1 )  "*" "*" "*" " " " " " " " " " " " " "*" 
## 5  ( 1 )  "*" "*" "*" "*" " " "*" " " " " " " " " 
## 6  ( 1 )  "*" "*" "*" "*" "*" "*" " " " " " " " " 
## 7  ( 1 )  "*" "*" " " "*" "*" "*" "*" " " "*" " " 
```

```
## 8  ( 1 )  "*" "*" "*" "*" "*" " " "*" "*" "*" " "
## 9  ( 1 )  "*" "*" "*" "*" "*" "*" "*" " " "*" "*"
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

Best model according to C_p

```r
plot(summary(bss)$cp, xlab="size", ylab="Cp")
```




```r
which.min(summary(bss)$cp)
```

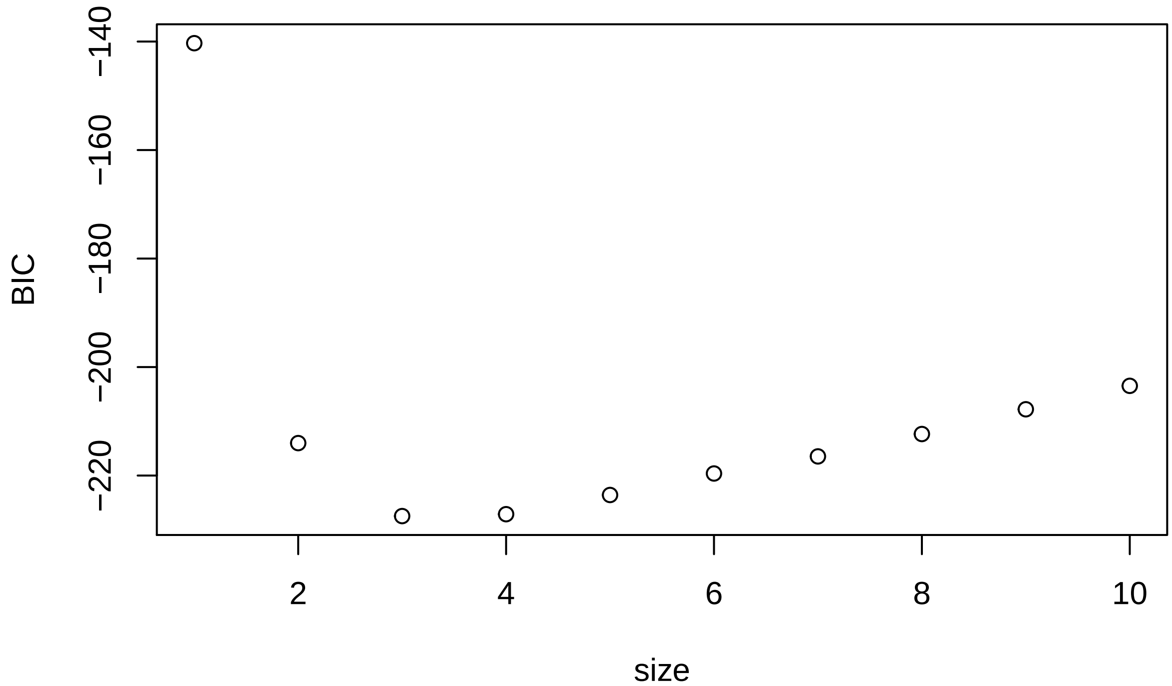
```
## [1] 4
```

```r
coef(bss, which.min(summary(bss)$cp))
```

```
##  (Intercept)          X^1          X^2          X^3         X^10 
## 1.0427564245 0.7854533475 0.9191223050 1.0890995581 0.0004590147
```

Best model accoding to BIC

```r
plot(summary(bss)$bic, xlab="size", ylab="BIC")
```




```r
which.min(summary(bss)$bic)
```

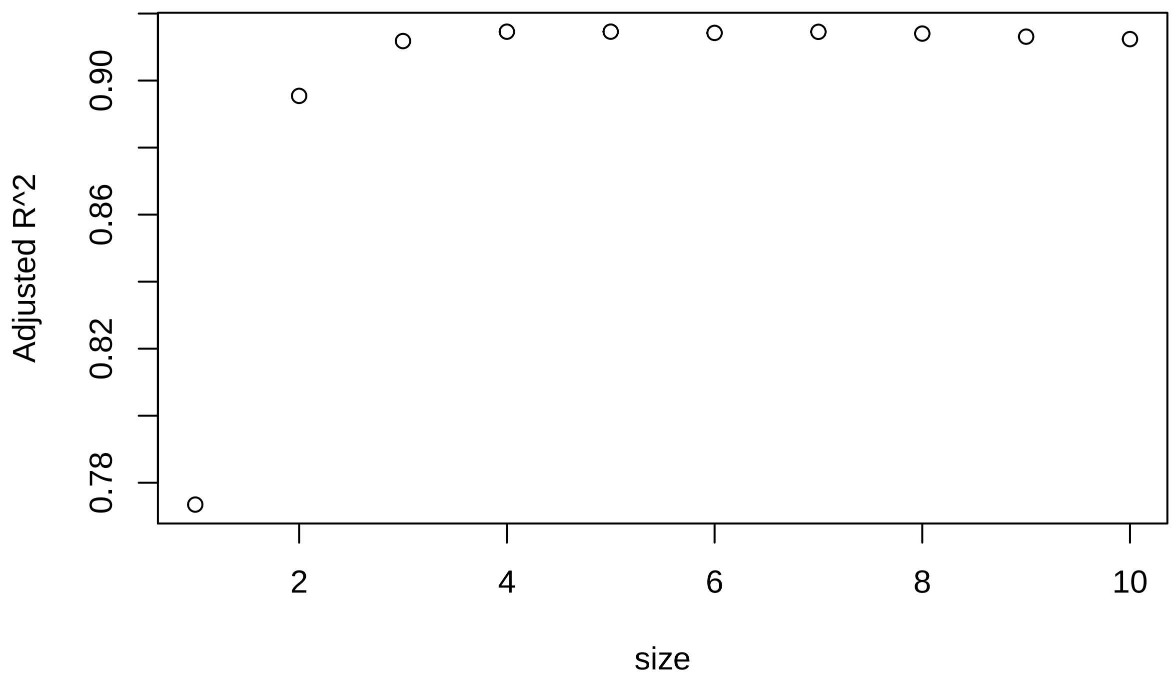
```
## [1] 3
```

```r
coef(bss, which.min(summary(bss)$bic))
```

```
## (Intercept)         X^1         X^2         X^3
##   0.9478377   0.9426868   1.0754135   0.9981607
```

Best model according to adjusted R^2:

```r
plot(summary(bss)$adjr2, xlab="size", ylab="Adjusted R^2")
```




```r
which.max(summary(bss)$adjr2)
```

```
## [1] 5
```

```r
coef(bss, which.max(summary(bss)$adjr2))
```

```
## (Intercept)         X^1         X^2         X^3         X^4         X^6
##  0.93130325  0.74999185  1.42932630  1.10560295 -0.34534931  0.06178997
```

Thus, only BIC finds the correct model. The other criteria produce models that include additional predictors.

Part d

Forward selection with BIC finds the same model as best subset selection with BIC (the correct one), but the other criteria find larger models:

```r
fwd = regsubsets(x=features, y=y, method='forward', nvmax=10)
coef(fwd, which.min(summary(fwd)$cp))
```

```
## (Intercept)         X^1         X^2         X^3         X^4         X^6
##  0.93130325  0.74999185  1.42932630  1.10560295 -0.34534931  0.06178997
```

```r
coef(fwd, which.min(summary(fwd)$bic))
```

```
## (Intercept)         X^1         X^2         X^3
##   0.9478377   0.9426868   1.0754135   0.9981607
```

```r
coef(fwd, which.max(summary(fwd)$adjr2))
```

```
## (Intercept)         X^1         X^2         X^3         X^4         X^6
##  0.93130325  0.74999185  1.42932630  1.10560295 -0.34534931  0.06178997
```

The results for backward selection are similar:

```r
bwd = regsubsets(x=features, y=y, method='backward', nvmax=10)
coef(bwd, which.min(summary(bwd)$cp))
```

```
## (Intercept)         X^1         X^2         X^5         X^7
##  0.99953611  1.36693521  0.98862439  0.47386194 -0.05774149
```

```r
coef(bwd, which.min(summary(bwd)$bic))
```

```
## (Intercept)         X^1         X^2         X^3
##   0.9478377   0.9426868   1.0754135   0.9981607
```

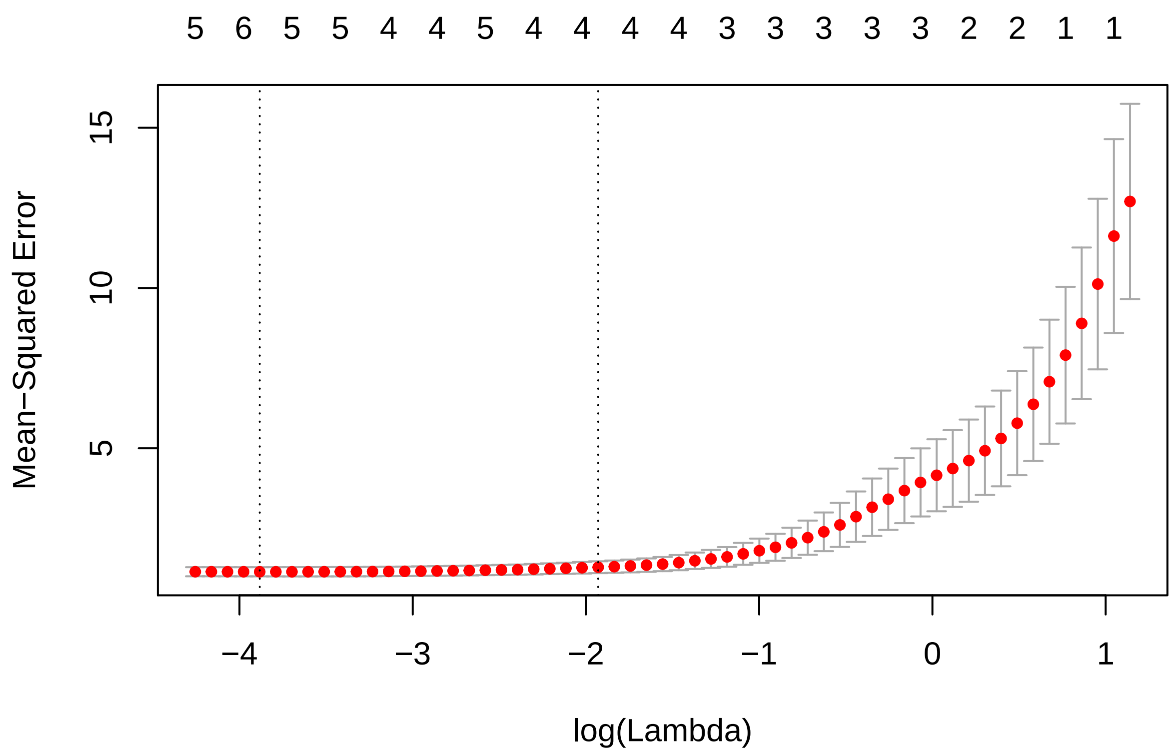
```r
coef(bwd, which.max(summary(bwd)$adjr2))
```

```
## (Intercept)         X^1         X^2         X^4         X^5         X^6
##  0.89120040  1.15952915  1.70930130 -0.60021299  0.80867739  0.11167937
##         X^7         X^9
## -0.22397388  0.02089738
```

Part e

Next, we fit a lasso model, using cross-validation to choose the penalty parameter.

```r
library(glmnet)

cv.out = cv.glmnet(as.matrix(features), y, alpha=1)
plot(cv.out)
```



The resulting model includes all the true predictors but also two spurious ones.

```r
s = cv.out$lambda.min
lasso.out = glmnet(as.matrix(features), y, alpha=1)
predict(lasso.out, type="coefficients", s=s)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                         1
```

```
## (Intercept) 1.0604321982
## X^1         0.7982693267
## X^2         0.8929514717
## X^3         1.0690449987
## X^4         .
## X^5         .
## X^6         .
## X^7         .
## X^8         0.0013040240
## X^9         .
## X^10        0.0001942767
```

Part f

Finally, we generate new data $Y = \beta_0 + \beta_7 X^7 + \epsilon$:

```
y = 1 + x^7 + epsilon
```

We run best subset selection.

```
bss = regsubsets(x=features, y=y, method='exhaustive', nvmax=10)
summary(bss)
```

```
## Subset selection object
## 10 Variables  (and intercept)
##      Forced in Forced out
## X^1      FALSE      FALSE
## X^2      FALSE      FALSE
## X^3      FALSE      FALSE
## X^4      FALSE      FALSE
## X^5      FALSE      FALSE
## X^6      FALSE      FALSE
## X^7      FALSE      FALSE
## X^8      FALSE      FALSE
## X^9      FALSE      FALSE
## X^10     FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: exhaustive
##           X^1 X^2 X^3 X^4 X^5 X^6 X^7 X^8 X^9 X^10
## 1  ( 1 )  " " " " " " " " " " " " "*" " " " " " "
## 2  ( 1 )  " " " " " " " " " " " " "*" " " " " "*"
## 3  ( 1 )  " " " " " " " " " " "*" "*" "*" " " " "
## 4  ( 1 )  "*" " " " " " " " " "*" "*" "*" " " " "
## 5  ( 1 )  "*" "*" " " "*" " " "*" "*" " " " " " "
## 6  ( 1 )  " " " " "*" " " "*" "*" "*" " " "*" "*"
## 7  ( 1 )  "*" " " "*" " " "*" " " "*" "*" "*" "*"
## 8  ( 1 )  "*" "*" "*" "*" "*" " " "*" "*" "*" " "
## 9  ( 1 )  "*" "*" "*" "*" "*" "*" "*" " " "*" "*"
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

```
coef(bss, which.min(summary(bss)$cp))
```

```
## (Intercept)         X^7        X^10
## 0.971390101 1.001739314 0.000394677
```

```r
coef(bss, which.min(summary(bss)$bic))
```

```
## (Intercept)         X^7
##    1.005768    0.997857
```

```r
coef(bss, which.max(summary(bss)$adjr2))
```

```
## (Intercept)         X^7        X^10
## 0.971390101 1.001739314 0.000394677
```

Again, only BIC selects the correct model.

We also run the lasso as in Part e:

```r
cv.out = cv.glmnet(as.matrix(features), y, alpha=1)
s = cv.out$lambda.min
lasso.out = glmnet(as.matrix(features), y, alpha=1)
predict(lasso.out, type="coefficients", s=s)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) 0.894319093
## X^1         .
## X^2         .
## X^3         .
## X^4         .
## X^5         0.003490764
## X^6         .
## X^7         0.970672998
## X^8         .
## X^9         .
## X^10        .
```

Again, the lasso model includes the correct predictor but also a spurious predictor.

Problem 4: Chapter 6, Exercise 9

We predict the number of applications received using the other variables in the `College` data set.

```r
College = read.csv("College.csv")
College = College[,-1]
College$Private = ifelse(College$Private == 'Yes', 1, 0)
head(College)
```

```
##   Private Apps Accept Enroll Top10perc Top25perc F.Undergrad P.Undergrad
## 1       1 1660   1232    721        23        52        2885         537
## 2       1 2186   1924    512        16        29        2683        1227
## 3       1 1428   1097    336        22        50        1036          99
## 4       1  417    349    137        60        89         510          63
## 5       1  193    146     55        16        44         249         869
## 6       1  587    479    158        38        62         678          41
##   Outstate Room.Board Books Personal PhD Terminal S.F.Ratio perc.alumni
## 1     7440       3300   450     2200  70       78      18.1          12
## 2    12280       6450   750     1500  29       30      12.2          16
## 3    11250       3750   400     1165  53       66      12.9          30
## 4    12960       5450   450      875  92       97       7.7          37
```

```
## 5     7560       4120   800     1500  76        72       11.9            2
## 6    13500       3335   500      675  67        73        9.4           11
##   Expend Grad.Rate
## 1   7041        60
## 2  10527        56
## 3   8735        54
## 4  19016        59
## 5  10922        15
## 6   9727        55
```

Part a

We randomly split the data into training (70%) and test (30%).

```r
set.seed(1101)

n = nrow(College)
train = sample(n, floor(n*0.7))
```

Part b

Test error for OLS:

```r
lm.fit = lm(Apps~., data=College[train,])
lm.pred = predict(lm.fit, newdata=College[-train,])
mean((lm.pred - College[-train,'Apps'])^2)
```

```
## [1] 1123168
```

Part c

Test error for ridge regression with penalty chosen by cross-validation:

```r
library(glmnet)

X = as.matrix(College[,-2])
y = College[[2]]

cv.out = cv.glmnet(X[train,], y[train], alpha=0)
ridge.out = glmnet(X[train,], y[train], alpha=0)
ridge.pred = predict(ridge.out, newx=X[-train,], s=cv.out$lambda.min)
mean((ridge.pred - y[-train])^2)
```

```
## [1] 1068678
```

Part d

Test error for lasso with penalty chosen by cross-validation:

```r
library(glmnet)

cv.out = cv.glmnet(X[train,], y[train], alpha=1)
lasso.out = glmnet(X[train,], y[train], alpha=1)
```

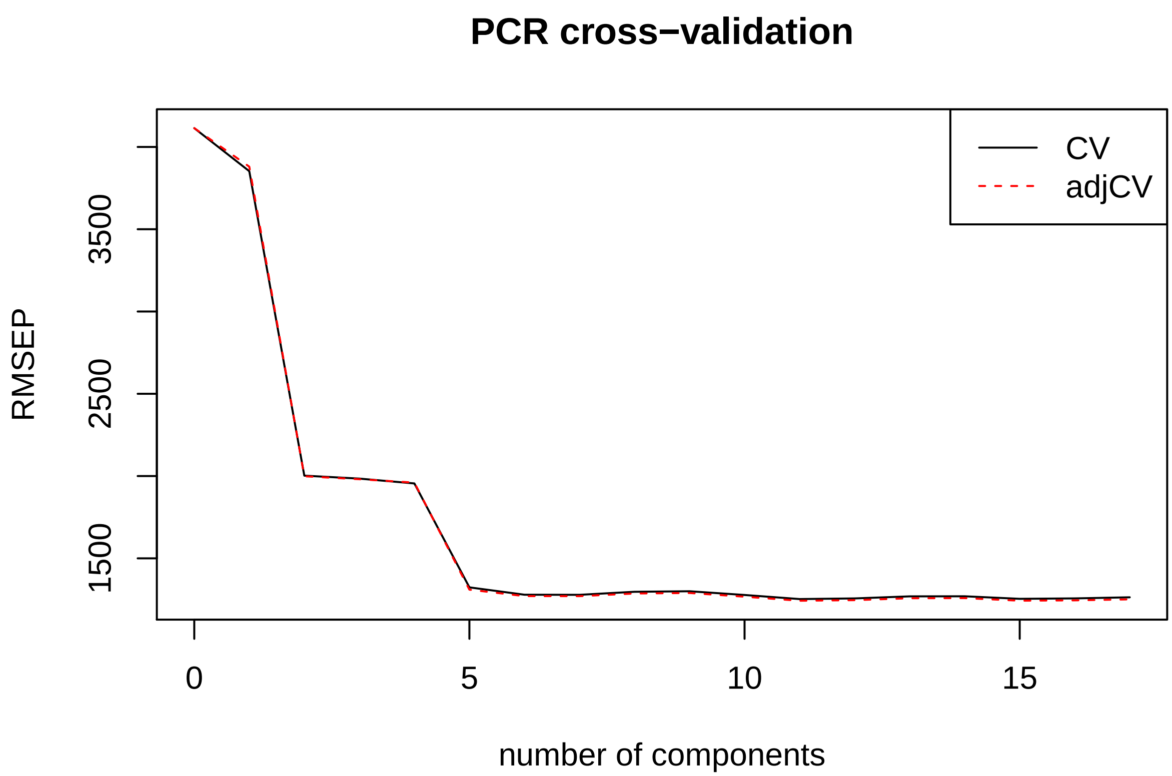
```r
lasso.pred = predict(lasso.out, newx=X[-train,], s=cv.out$lambda.min)
mean((lasso.pred - y[-train])^2)
```

```
## [1] 1083430
```

Part e

Test error for PCR with number of components chosen by cross-validation:

```r
library(pls)

pcr.fit = pcr(Apps~., data=College[train,], validation='CV')
plot(RMSEP(pcr.fit), legendpos='topright', main='PCR cross-validation')
```



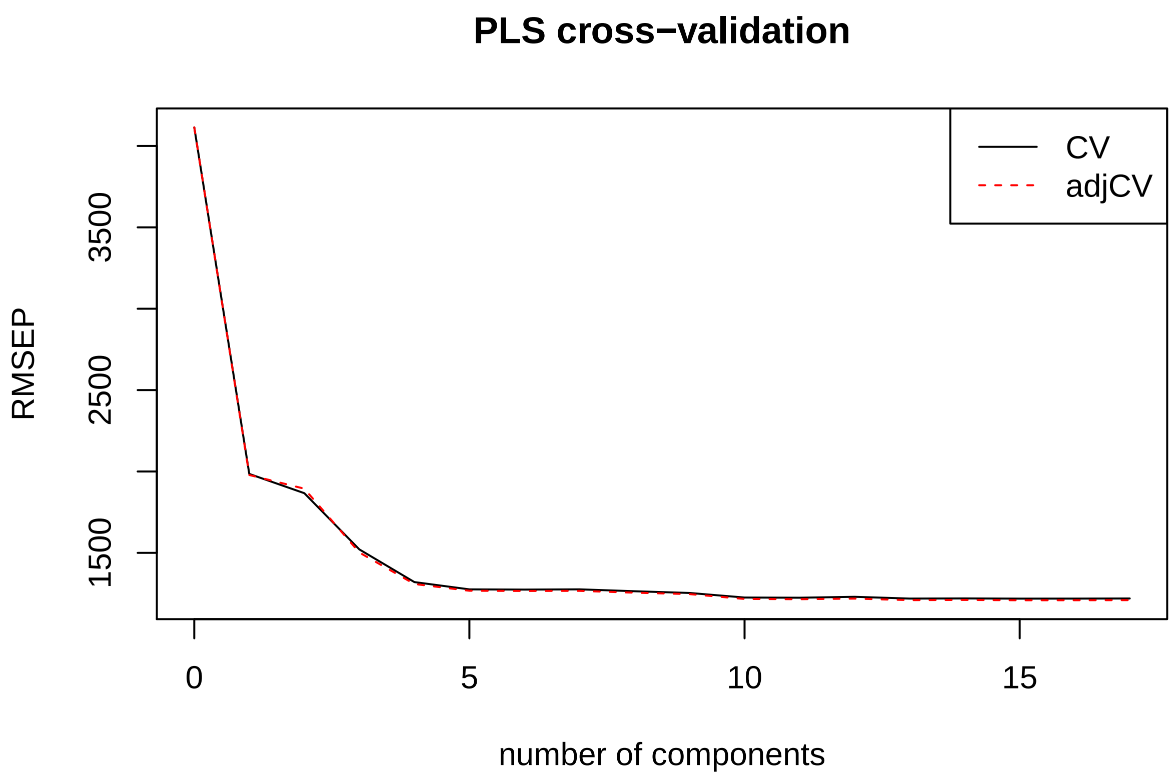

```r
pcr.pred = predict(pcr.fit, newdata=College[-train,], ncomp=5)
mean((pcr.pred - College[-train,'Apps'])^2)
```

```
## [1] 1487490
```

Part f

Test error for PLS with number of components chosen by cross-validation:

```r
pls.fit = plsr(Apps~., data=College[train,], validation='CV')
plot(RMSEP(pls.fit), legendpos='topright', main='PLS cross-validation')
```




```
pls.pred = predict(pls.fit, newdata=College[-train,], ncomp=5)
mean((pls.pred - College[-train,'Apps'])^2)
```

```
## [1] 1508866
```

Part g

None of the test errors are too different. (Possible exceptions are PCR and PLS, but we could have picked the number of components more aggressively in Parts e and f, yielding the full model. In this case, the selected model would be identical to OLS and, in particular, would have the same test error.)

Let's take the OLS model as representative. Its root-mean-square (RMS) test error is

```
sqrt(mean((lm.pred - College[-train,'Apps'])^2))
```

```
## [1] 1059.796
```

For interpretability, we might compare this to the standard deviation of the response:

```
sd(College$Apps)
```

```
## [1] 3870.201
```

Informally, the expected prediction error is about 25-30% of a standard deviation. Whether this is "good" will depend on how the predictions are to be used.

Problem 5: Chapter 6, Exercise 11

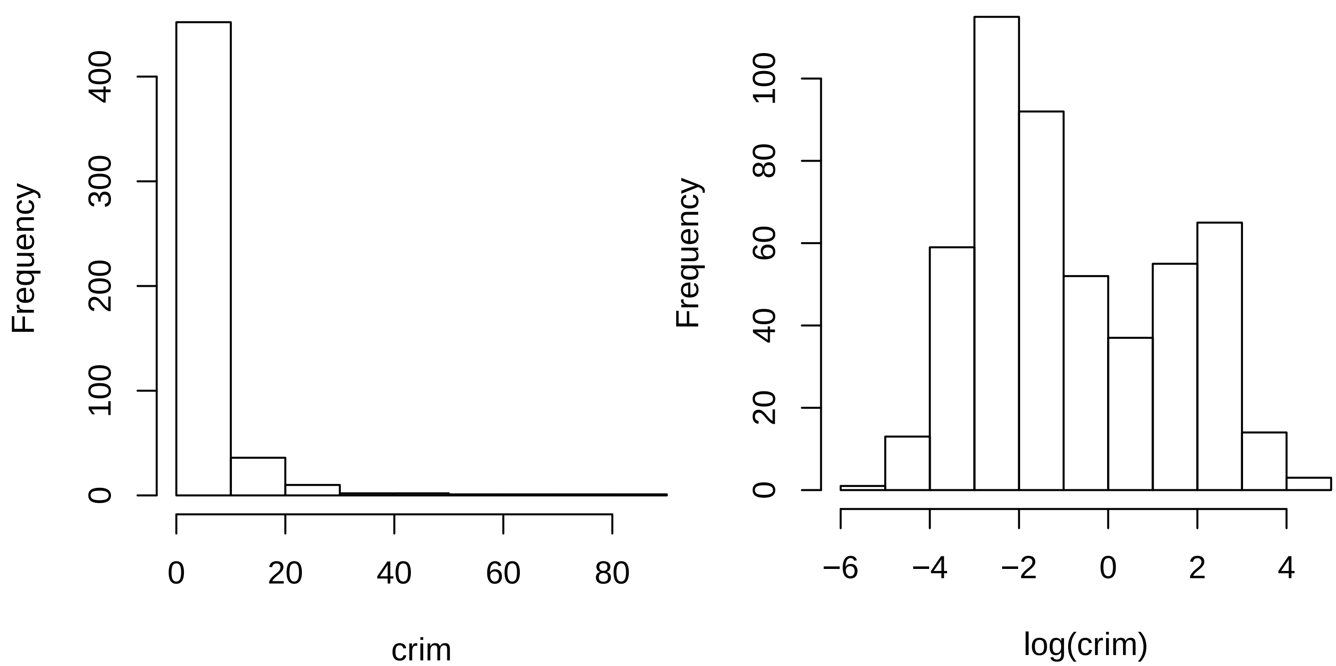
We try to predict the per capita crime rate (`crim`) in the `Boston` dataset.

```r
library(MASS)

data(Boston)
knitr::kable(head(Boston))
```

crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0.00632	18	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
0.02985	0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	28.7

As is often the case with rates, the crime rate is heavily skewed to the right. We work with its logarithm, which is better behaved. [In your solution, either choice is fine.]

```r
hist(Boston$crim, xlab='crim', main='')
hist(log(Boston$crim), xlab='log(crim)', main='')
```



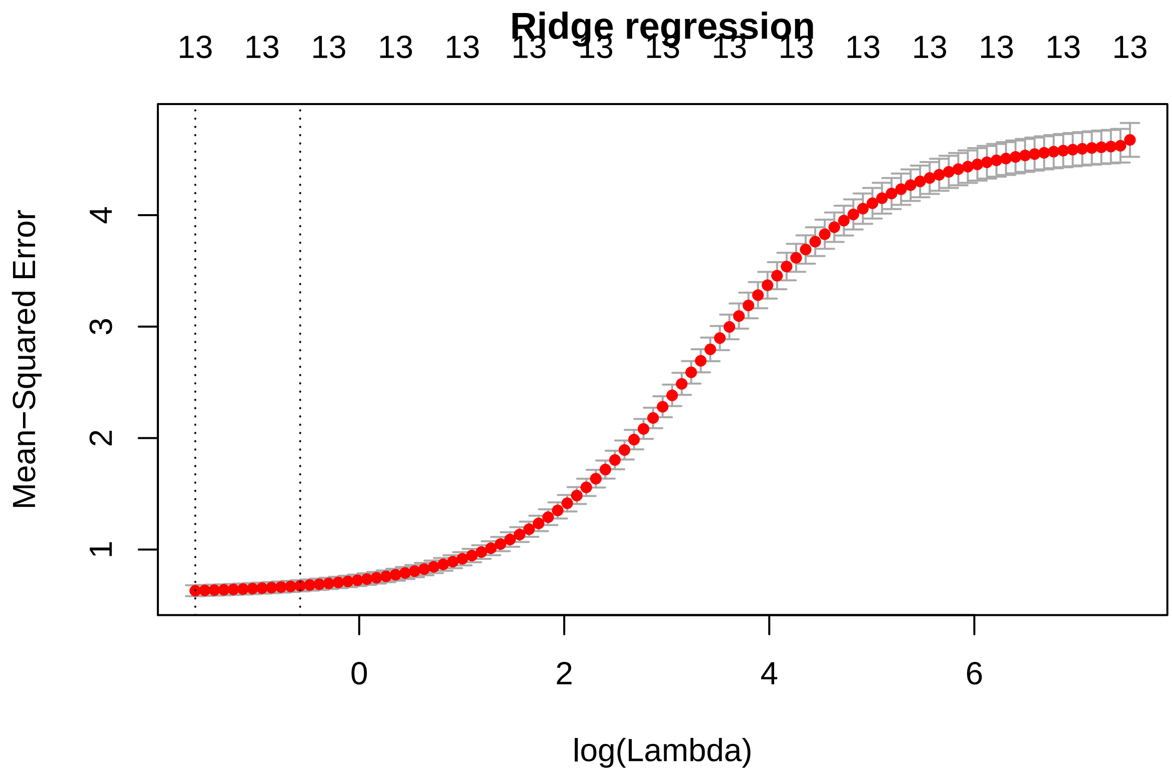
Part a

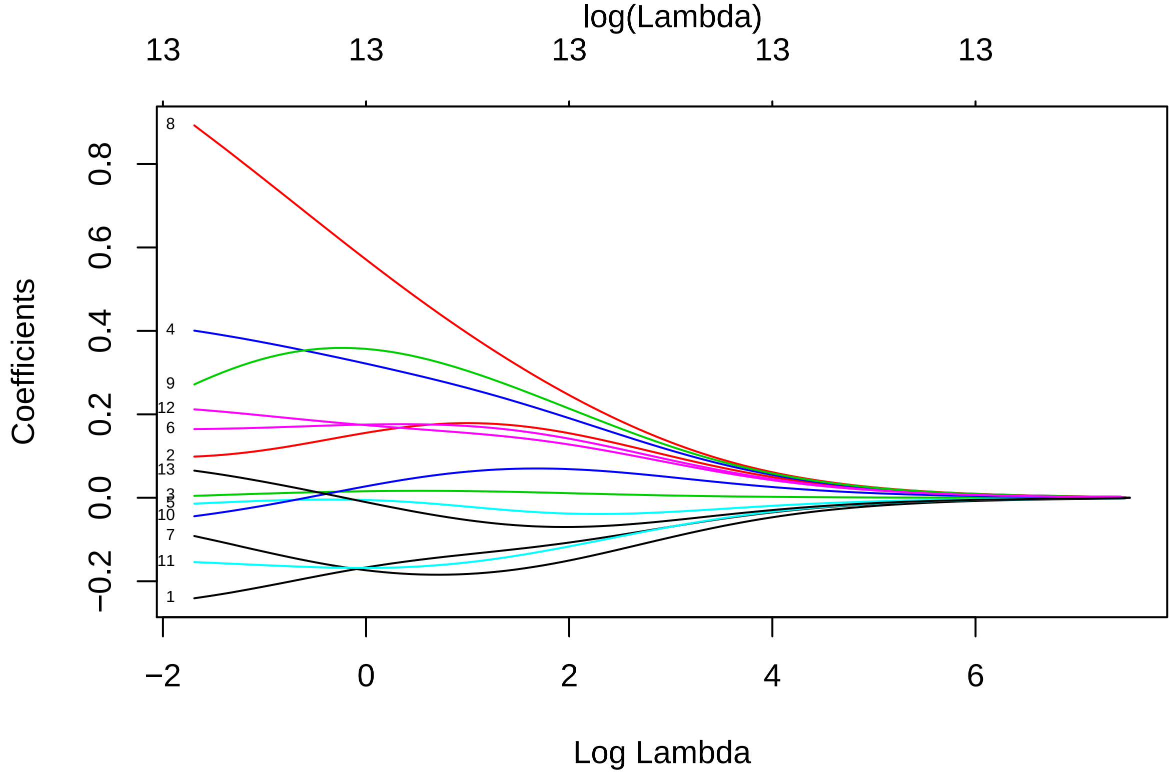
We'll compare ridge regression, the lasso, principal components regression (PCR), and partial least squares (PLS). [Any choice of at least three methods described in this chapter is adequate.]

First, we try ridge regression. To get more interpretable ridge and lasso paths, we standardize the predictors. That puts the regression coefficients on the same scale, but doesn't change the quality of the fit.

```r
library(glmnet)

X = subset(Boston, select=-c(crim))
X = scale(as.matrix(X))
```

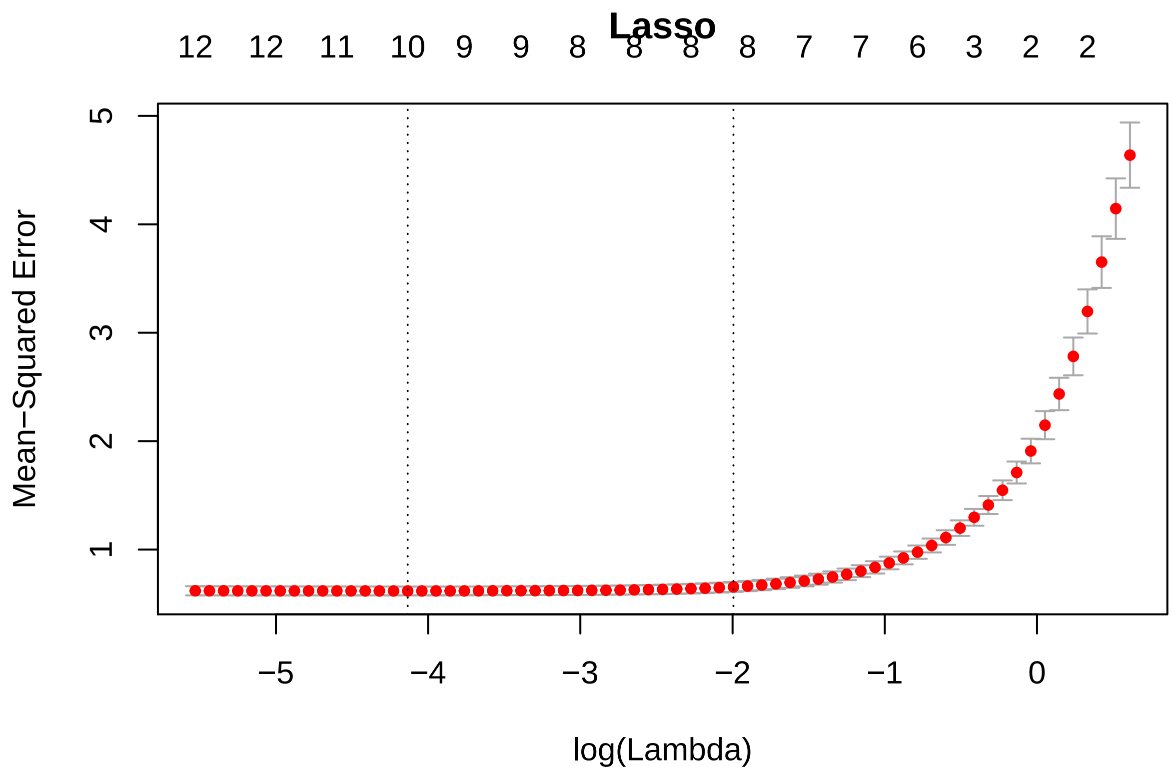
```r
y = log(Boston$crim)

cv.ridge = cv.glmnet(X, y, alpha=0, standardize=FALSE)
plot(cv.ridge, main='Ridge regression')
plot(cv.ridge$glmnet.fit, xvar='lambda', label=TRUE)
```

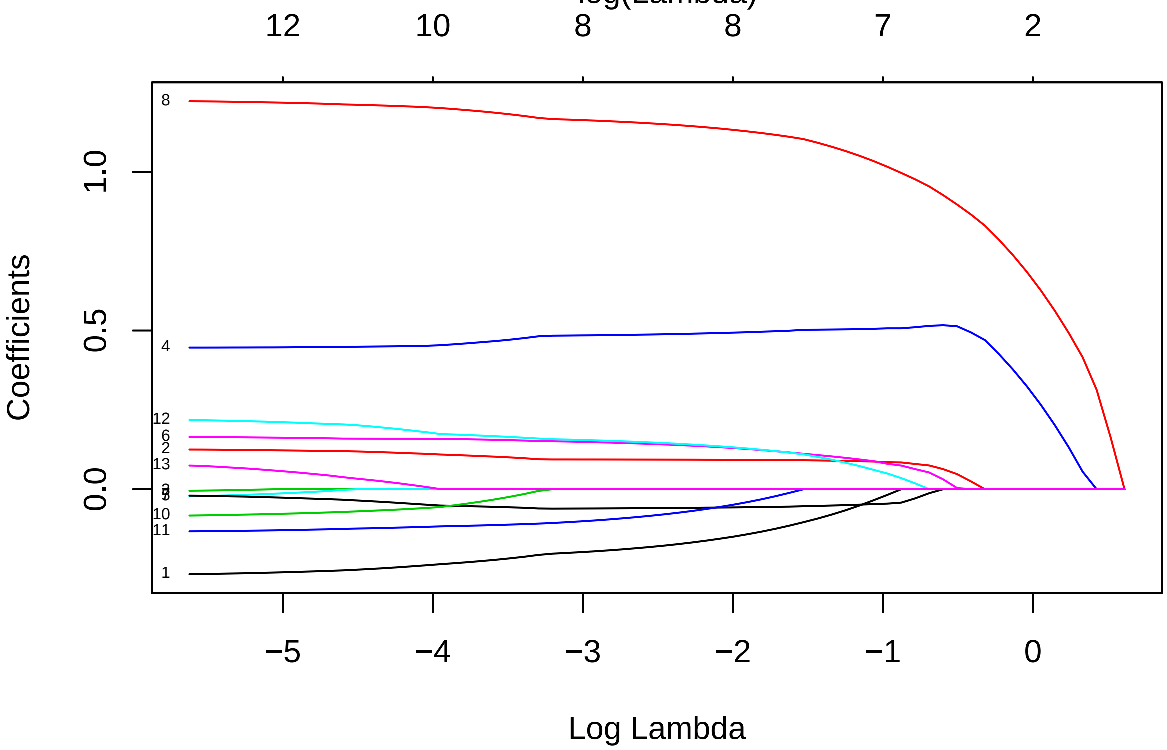




Next, we try the lasso:

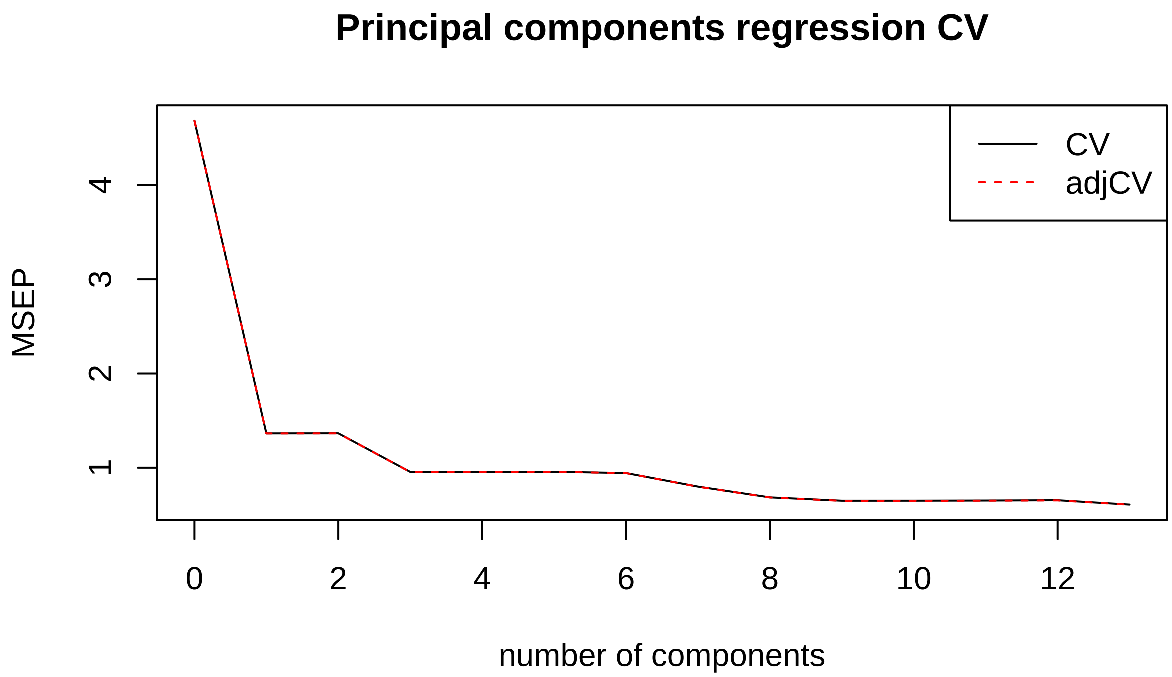
```r
cv.lasso = cv.glmnet(X, y, alpha=1, standardize=FALSE)
plot(cv.lasso, main='Lasso')
plot(cv.lasso$glmnet.fit, xvar='lambda', label=TRUE)
```



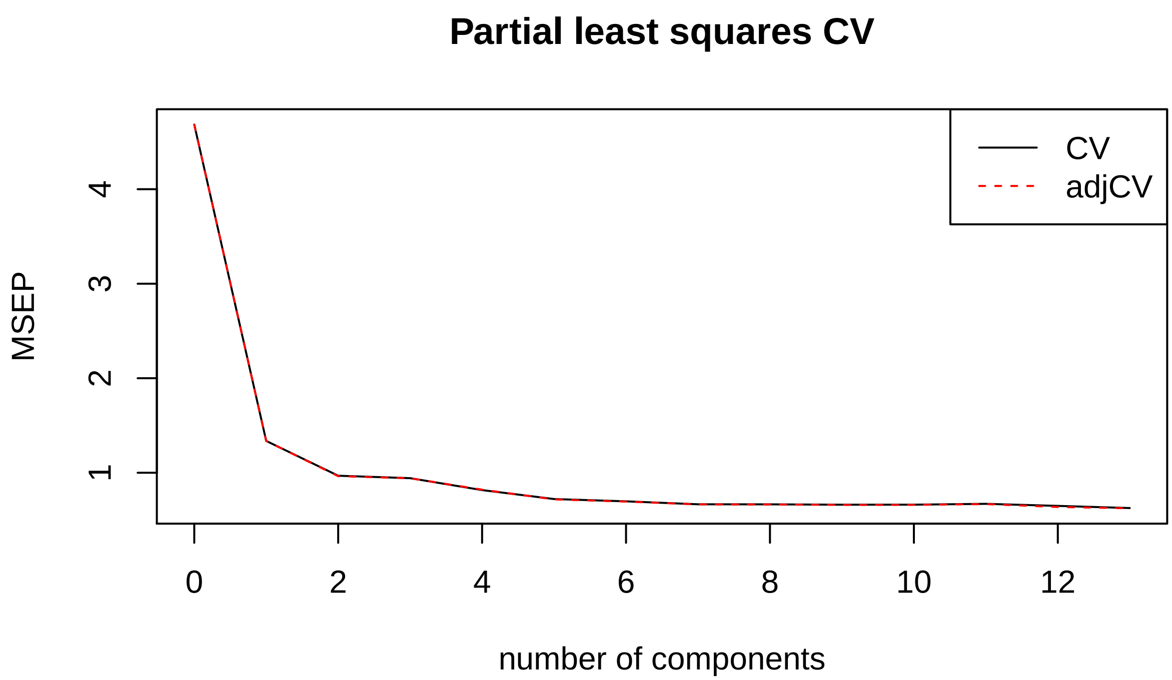


Finally, we try principal components regression (PCR) and partial least squares (PLS).

```r
library(pls)

pcr.fit = pcr(log(crim) ~ ., data=Boston, validation='CV')
plot(MSEP(pcr.fit), legendpos='topright', main='Principal components regression CV')
```




```r
pls.fit = plsr(log(crim) ~ ., data=Boston, validation='CV')
plot(MSEP(pls.fit), legendpos='topright', main='Partial least squares CV')
```



Part b

[To receive full credit, it's enough to choose a model using a valid estimate of the test error, e.g., from a validation set or cross-validation. A more elaborate answer is given below.]

All four methods have similar minima of the CV estimates of MSE:

```r
c(ridge=min(cv.ridge$cvm), lasso=min(cv.lasso$cvm),
  pcr=min(MSEP(pcr.fit, 'CV')$val), pls=min(MSEP(pls.fit, 'CV')$val))
```

```
##     ridge     lasso       pcr       pls
## 0.6305996 0.6182300 0.6084220 0.6262458
```

It's more informative to look at the cross-validation curves shown above for each method. With the exception of ridge regression, the error curves flatten out long before the absolute minimum is achieved. Since

- the CV estimates have *uncertainty*, quantified by their standard errors, and
- among roughly equally predictive models, simpler or more interpretable models are preferable,

we will not simply choose the model with absolute minimum (estimated) test error. First, we pick the lasso over the other models for its superior interpretability. There is a clear "elbow" in the lasso CV curve near $\log \lambda = 2$. The elbow also happens to be near the largest value of λ within one standard deviation of the absolute minimum CV error (another useful rule of thumb). For all these reasons, we select the lasso model with penalty parameter $\log \lambda$ equal to

```r
log(cv.lasso$lambda.1se)
```

```
## [1] -1.994385
```

Part c

The coefficients of our selected model are

```r
coef(cv.lasso$glmnet.fit, s=cv.lasso$lambda.1se)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) -0.78043626
## zn          -0.14920722
## indus        0.09267219
## chas         .
## nox          0.49335728
## rm           .
## age          0.12965804
## dis         -0.05724052
## rad          1.13200969
## tax          .
## ptratio      .
## black       -0.04846501
## lstat        0.13183959
## medv         .
```

Because the lasso promotes sparsity, only 8 of the 13 predictors are included in the final model. Moreover, two of the predictors stand out as particularly important: nitrogen oxide concentration (`nox`) and accessibility to radial highways (`rad`). This is evident from both the ridge path and the lasso path shown above.