Linear Model Selection and Regularization Ch. 6 Exercises

Lucy L.

2024-01-28

```
library(ISLR2)
library(leaps)
library(glmnet)
library(pls)
library(tidyverse)
library(caret)
library(leaps)
set.seed(1)
```

Exercise 8

In this exercise, we will generate simulated data, and will then use this data to perform best subset selection.

(a) Use the `rnorm()` function to generate a predictor `X` of length `n = 100`, as well as a noise vector ϵ of length `n = 100`.

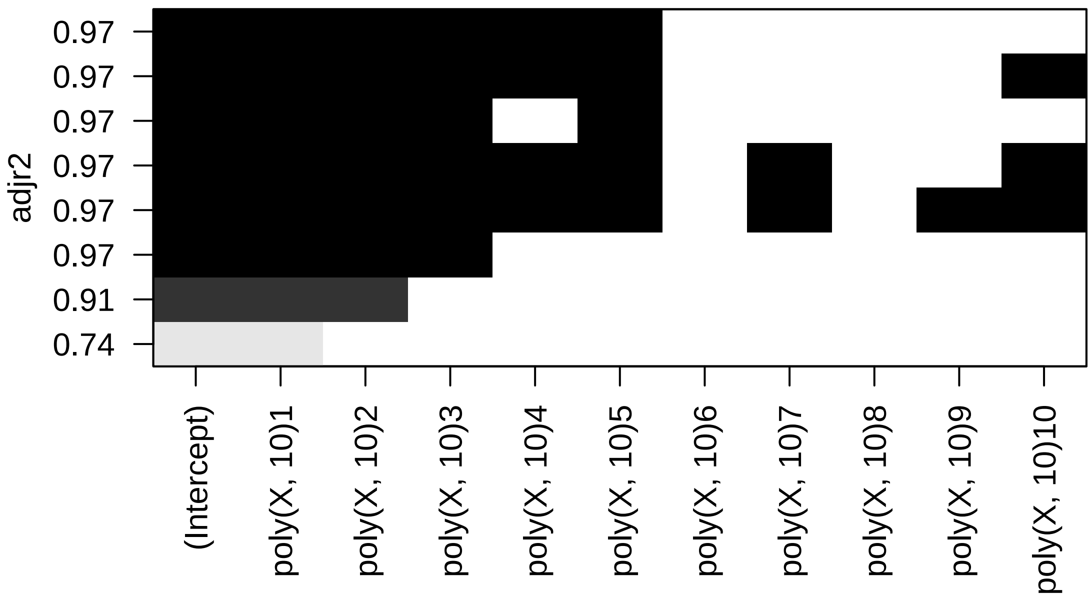
```
X = rnorm(100)
epsilon = rnorm(100)
```

(b) Generate a response vector `Y` of length `n = 100` according to the model $Y = \beta_0 + beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon$ where $\beta_0, ..., \beta_3$ are constants of your choice.

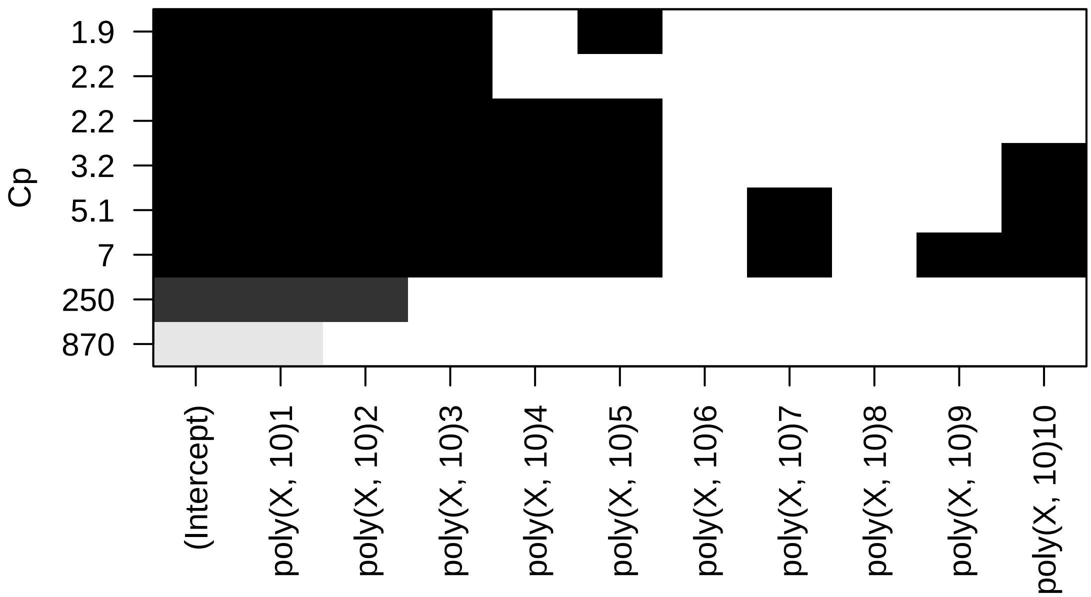
```
Y = 5 + 3*X + 2*(X^2) + X^3 + epsilon
data_xy = data.frame(X, Y)
```

(c) Use the `regsubsets()` function to perform best subset selection in order to choose the best model containing the predictors $X, X^2, ..., X^{10}$. What is the best model obtained according to C_p, BIC, and adjusted R^2? Show some plots to provide evidence for your answer, and report the coefficients of the best model obtained. Note you will need to use the `data.frame()` function to create a single data set containing both `X` and `Y`.

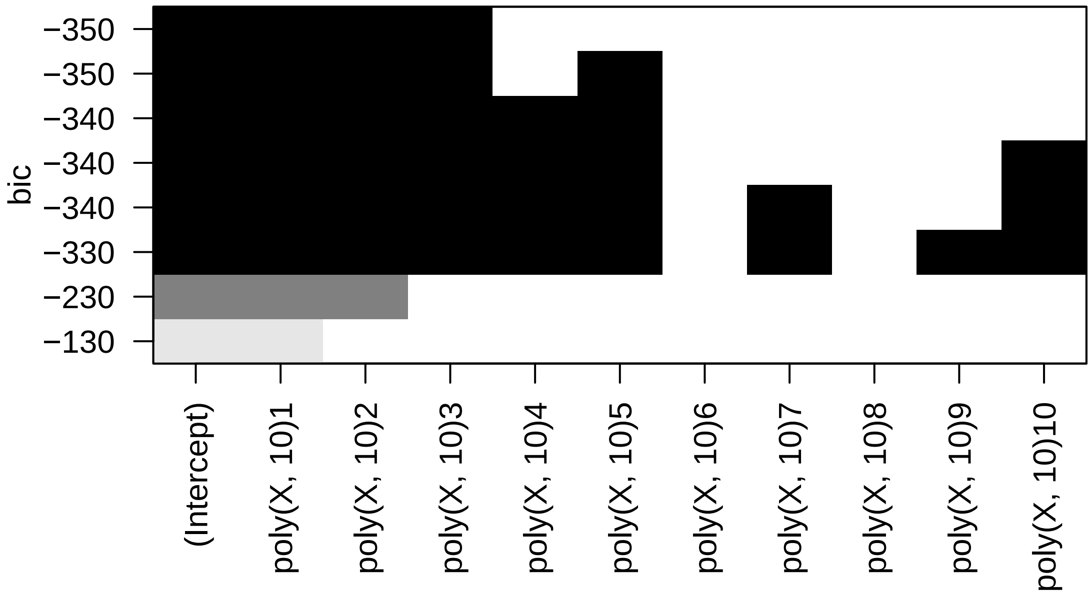
```
# fitting for best subset selection
regfit.full = regsubsets(Y ~ poly(X, 10), data = data_xy)

# plots
plot(regfit.full, scale = "adjr2")
```




```
plot(regfit.full, scale = "Cp")
```




```
plot(regfit.full, scale = "bic")
```




```r
# reporting coefficients
regfit_adjr = lm(Y ~ X + I(X^2) + I(X^3) + I(X^5) + I(X^6)
                        + I(X^8), data = data_xy)
regfit_bic = lm(Y ~ X + I(X^2) + I(X^3)+ I(X^6) + I(X^8),
                        data = data_xy)
regfit_cp = lm(Y ~ X + I(X^2) + I(X^3), data = data_xy)

regfit_adjr$coefficients
```

```
##  (Intercept)            X       I(X^2)       I(X^3)       I(X^5)       I(X^6)
##  5.117263516  3.360184202  1.731721481  0.607135374  0.069588354  0.011331687
##       I(X^8)
## -0.001016896
```

```r
regfit_bic$coefficients
```

```
##  (Intercept)            X       I(X^2)       I(X^3)       I(X^6)       I(X^8)
##  5.113909391  3.053217562  1.778847934  0.973332603 -0.013122156  0.003936642
```

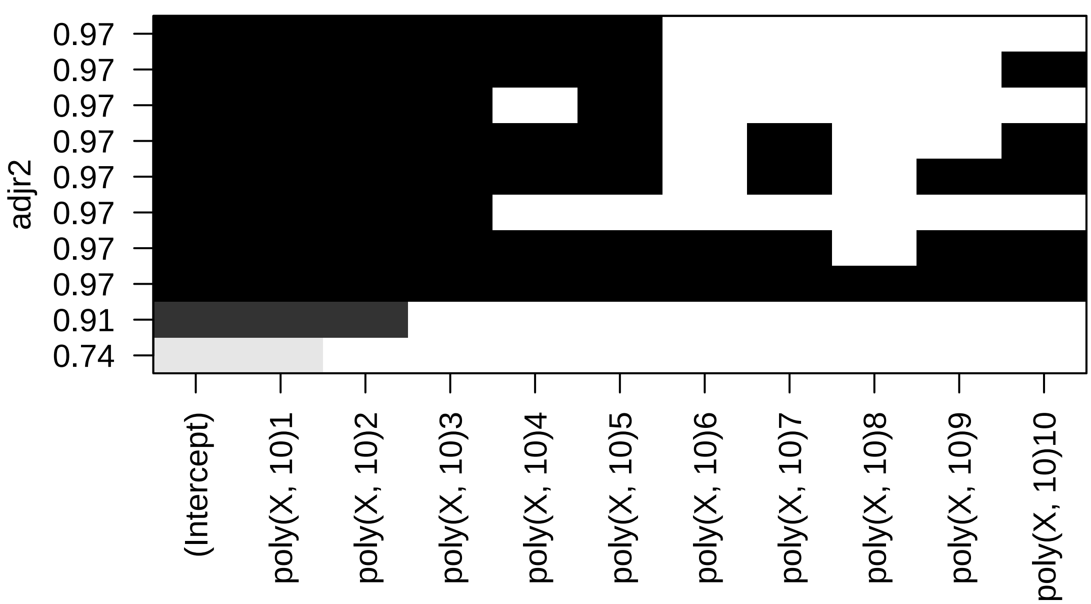
```r
regfit_cp$coefficients
```

```
## (Intercept)           X      I(X^2)      I(X^3)
##    5.061507    2.975280    1.876209    1.017639
```

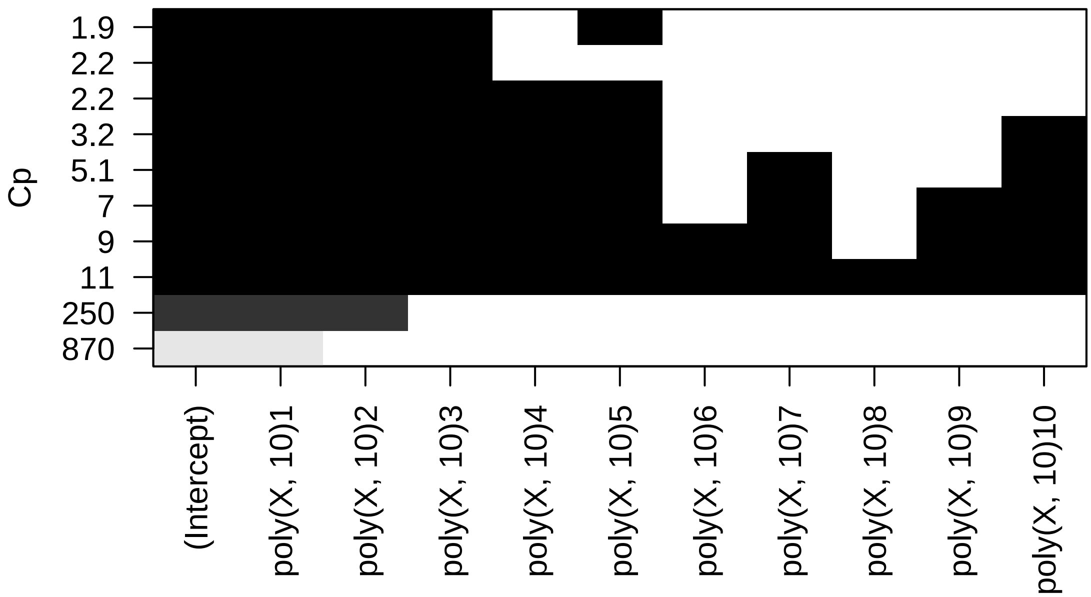
The best models according to C_p, BIC, and adjusted R^2 are as shown.

(d) Repeat (c), using forward stepwise selection and also using backwards stepwise selection. How does your answer compare to the results in (c)?

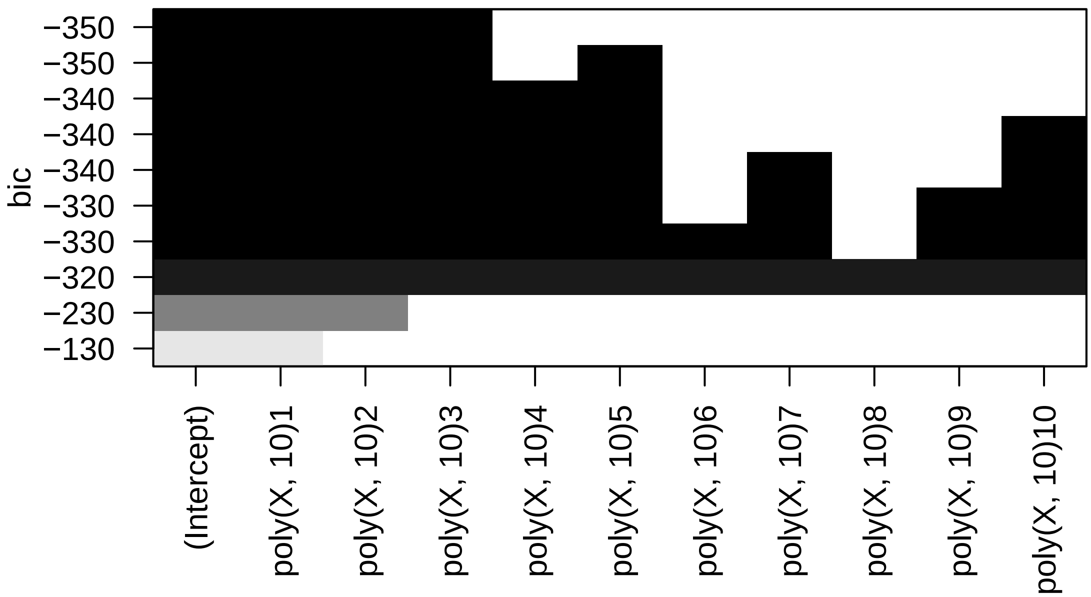
```r
# fitting for forward and backward selection
regfit.fwd = regsubsets(Y ~ poly(X, 10), data = data_xy, nvmax = 19,
                        method = "forward")
regfit.bwd = regsubsets(Y ~ poly(X, 10), data = data_xy, nvmax = 19,
                        method = "backward")

# examining plots for best model
plot(regfit.fwd, scale = "adjr2")
```



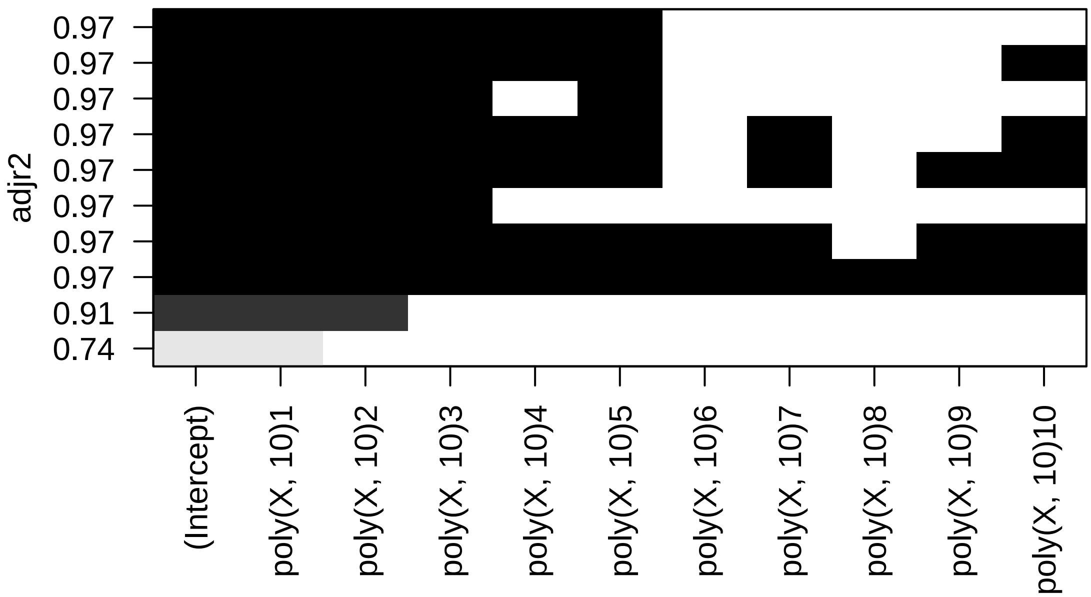

```r
plot(regfit.fwd, scale = "Cp")
```



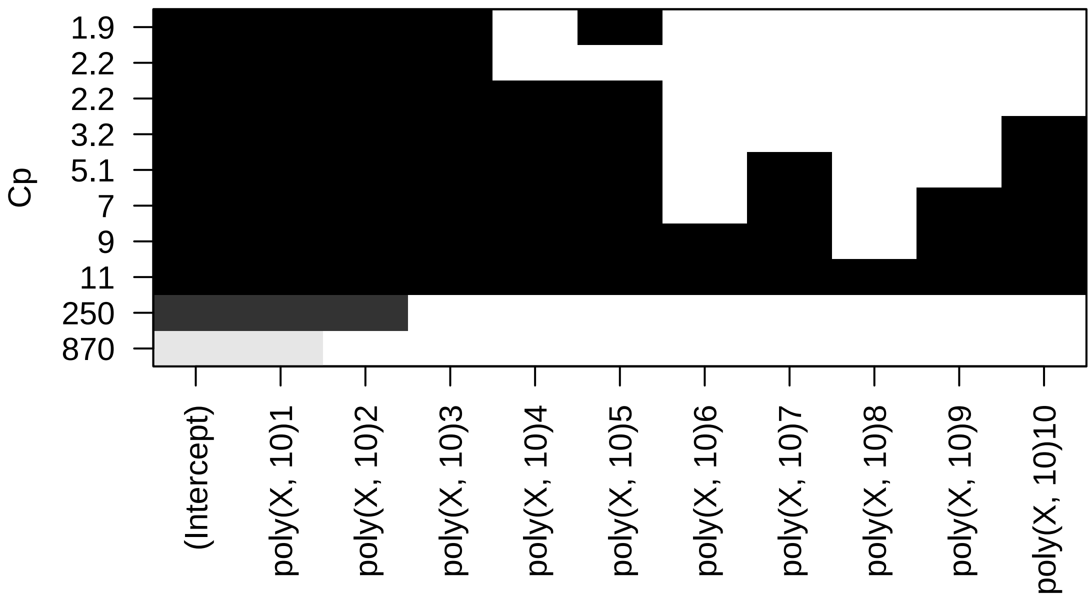

```r
plot(regfit.fwd, scale = "bic")
```



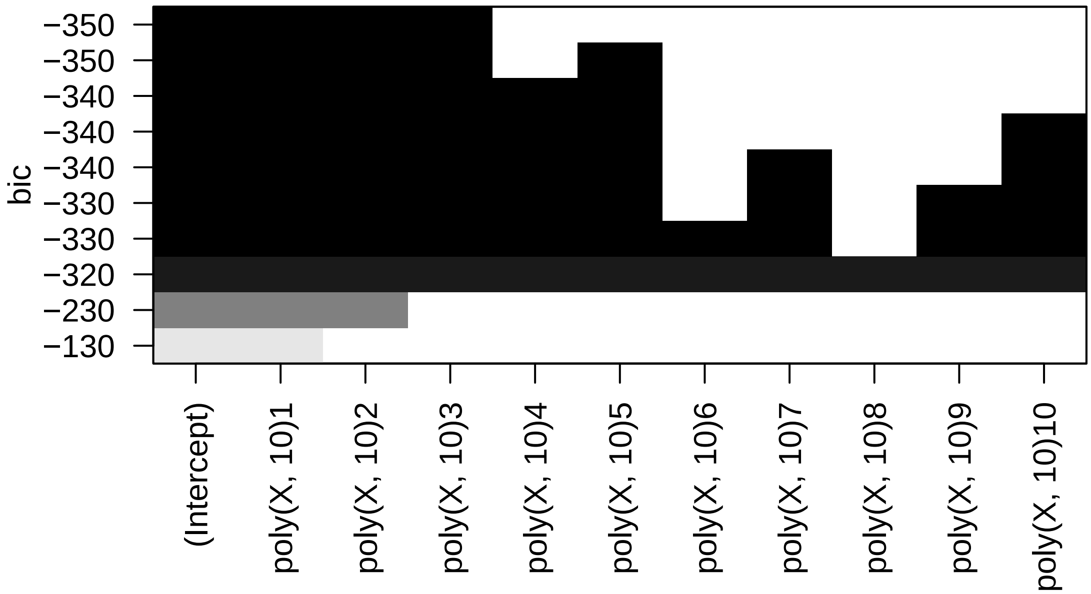

```r
plot(regfit.bwd, scale = "adjr2")
```




```r
plot(regfit.bwd, scale = "Cp")
```



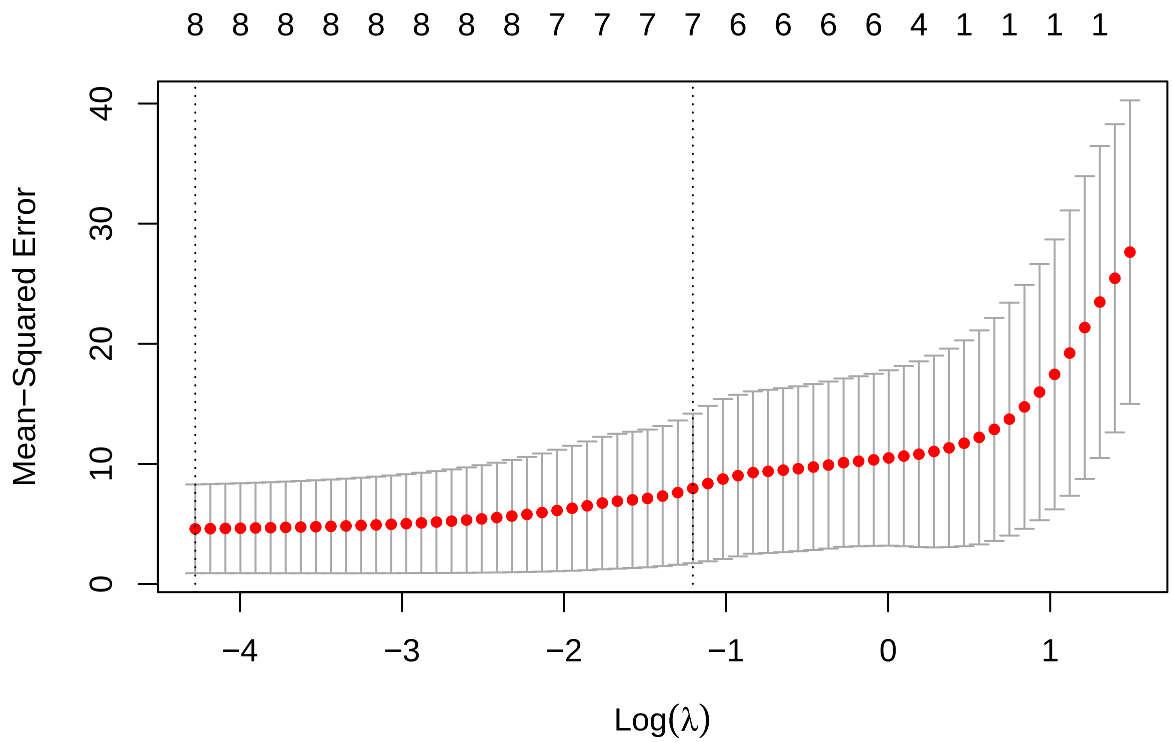

```
plot(regfit.bwd, scale = "bic")
```



The plots reveal that the forwards and backwards stepwise selection processes selected for the same predictors across all three models.

(e) Now ft a lasso model to the simulated data, again using $X, X^2, ..., X^{10}$ as predictors. Use cross-validation to select the optimal value of λ. Create plots of the cross-validation error as a function of λ. Report the resulting coefficient estimates, and discuss the results obtained.

```
set.seed(1)
x_lasso =  model.matrix(Y ~ poly(X, 10), data = data_xy)
grid = 10^seq(10, -2, length = 100)

# splitting into train and test set to estimate test error
train = sample(1:nrow(x_lasso), nrow(x_lasso)/2)
test = (-train)
y.test = Y[test]

# setting up model
cv.out = cv.glmnet(x_lasso[train, ], Y[train], alpha = 1)
plot(cv.out)
```




```
(best_lambda = cv.out$lambda.min)
```

```
## [1] 0.01390017
```

```
# determining coefficients
(lasso.coef = predict(cv.out, type = "coefficients", s = best_lambda))
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)     6.969034
## (Intercept)     .
## poly(X, 10)1   48.173049
## poly(X, 10)2   20.974823
## poly(X, 10)3   12.861193
## poly(X, 10)4   -0.946507
## poly(X, 10)5    .
## poly(X, 10)6    .
## poly(X, 10)7   -1.700703
## poly(X, 10)8   -1.900094
## poly(X, 10)9   -2.342676
## poly(X, 10)10  -2.978259
```

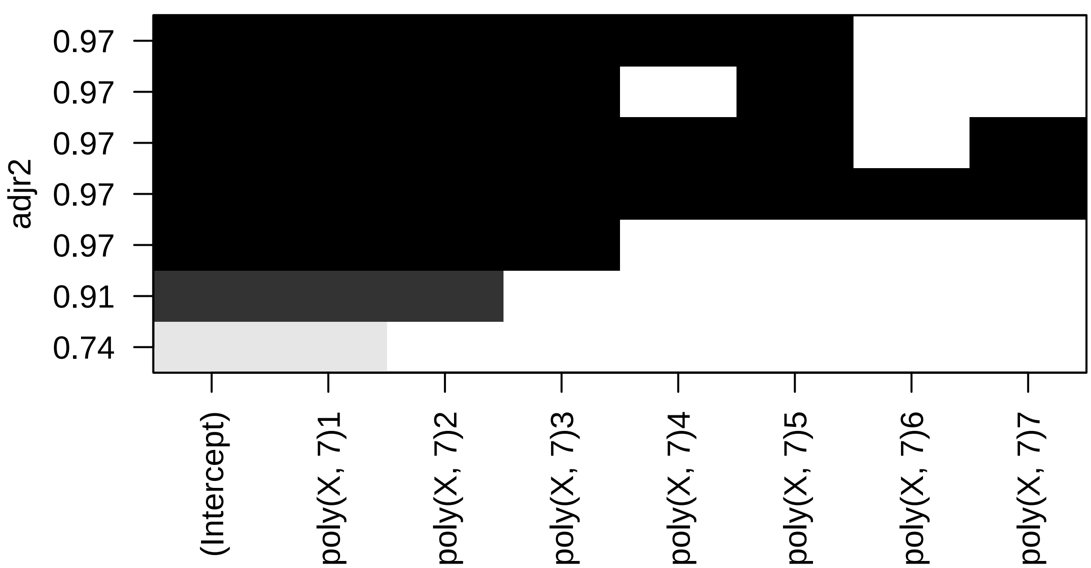
```
lasso.coef[lasso.coef != 0]
```

```
## [1]  6.969034 48.173049 20.974823 12.861193 -0.946507 -1.700703 -1.900094
## [8] -2.342676 -2.978259
```

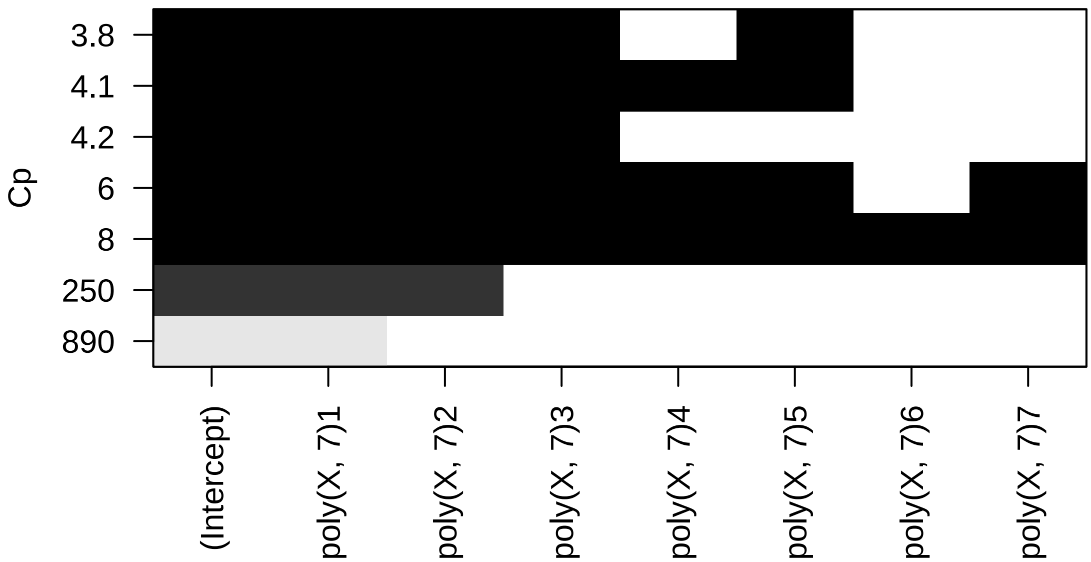
Cross validation chose $\lambda = 0.0139$ and $p = 8$. This includes much more predictors than the previous methods.

(f) Now generate a response vector `Y` according to the model $Y = \beta_0 + \beta_7 X^7 + \epsilon$, and perform best subset selection and the lasso. Discuss the results obtained.

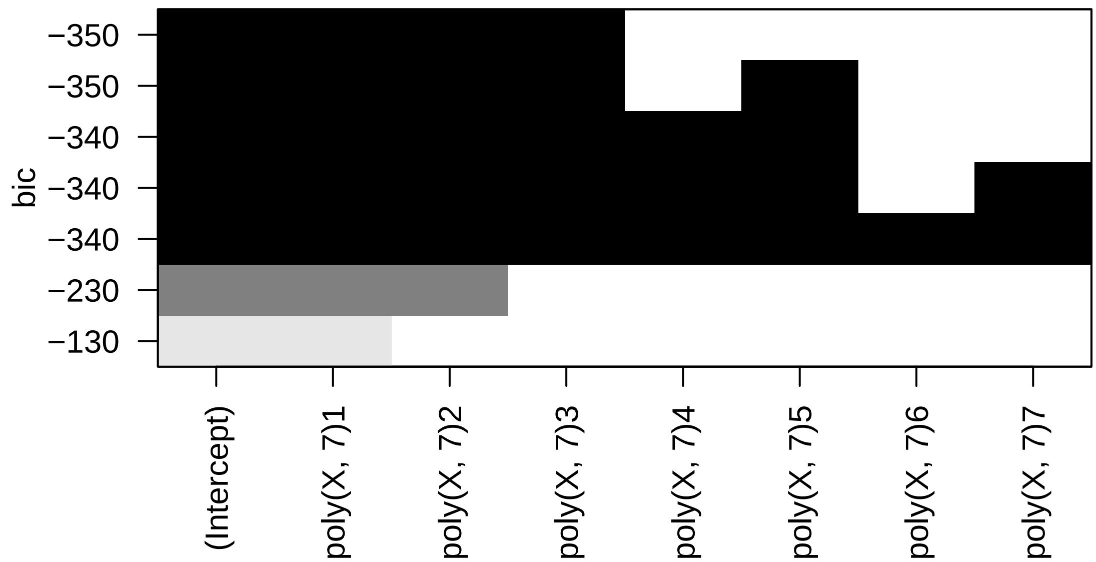
```r
# BSS
regfit.x7 = regsubsets(Y ~ poly(X, 7), data = data_xy)
plot(regfit.x7, scale = "adjr2")
```



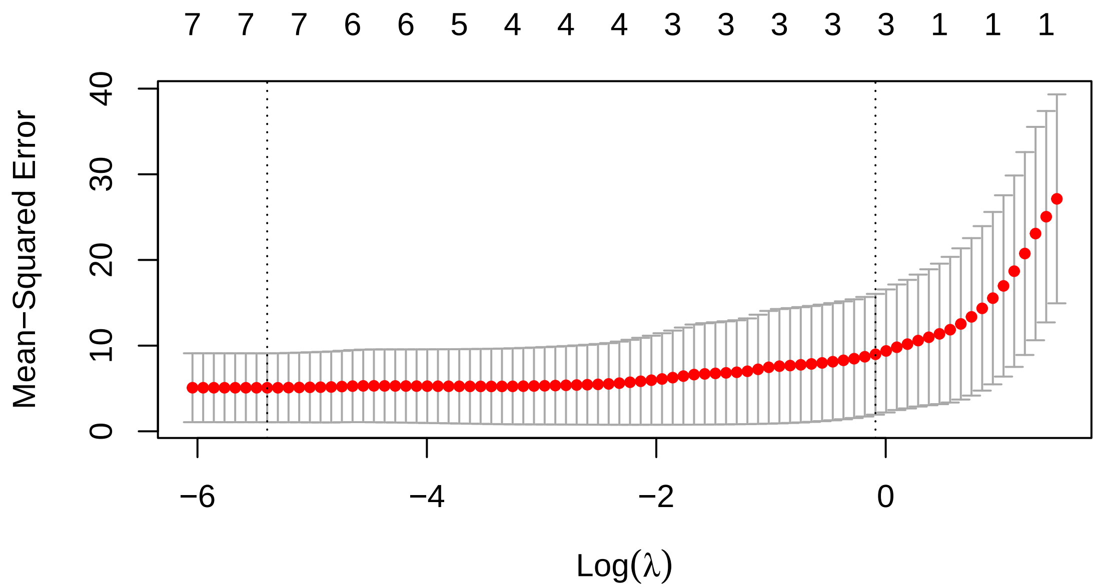

```r
plot(regfit.x7, scale = "Cp")
```




```
plot(regfit.x7, scale = "bic")
```




```r
# Lasso
x_lasso_7 =  model.matrix(Y ~ poly(X, 7), data = data_xy)[, -1]
cv.out2 = cv.glmnet(x_lasso_7[train, ], Y[train], alpha = 1)
plot(cv.out2)
```




```r
(best_lambda2 = cv.out2$lambda.min)
```

```
## [1] 0.004551678
```

```r
(lasso.coef2 = predict(cv.out2, type = "coefficients", s = best_lambda2))
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept)   7.1289323
## poly(X, 7)1 51.4935965
## poly(X, 7)2 26.0424239
## poly(X, 7)3 17.1294758
## poly(X, 7)4  1.7062480
## poly(X, 7)5  2.0296464
## poly(X, 7)6  1.9542330
## poly(X, 7)7  0.6786227
```

For best subset selection, the same predictors are picked again across the criterion. For lasso, all coefficients are included this time. This time, the X^5, X^6 predictors are included. Lambda is significantly decreased.

Exercise 9

In this exercise, we will predict the number of applications received using the other variables in the `College` data set.

(a) Split the data set into a training set and a test set.

```
college = College
train_subset = sample(1:nrow(college), nrow(college)/2)
test_subset = (-train_subset)
college_train = college[train_subset, ]
college_test = college[-train_subset, ]
```

(b) Fit a linear model using least squares on the training set, and report the test error obtained.

```
college_mod = lm(Apps ~ ., data = college_train)
college_pred = predict(college_mod, college_test)
mean((college_test$Apps - college_pred)^2)
```

```
## [1] 1202357
```

(c) Fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test error obtained.

```
x_college =  model.matrix(Apps ~ ., college)
y_college = college$Apps
grid = 10^seq(10, -2, length = 100)

# picking lambda
cv.out.college = cv.glmnet(x_college[train_subset, ],
                           y_college[train_subset],
                           alpha = 0)
bestlam.college = cv.out.college$lambda.min

# fitting RR on training set with lambda = 4
ridge.model.college = glmnet(x_college[train_subset, ],
                             y_college[train_subset],
                             alpha = 0, lambda = grid,
                             thresh = 1e-12)

# obtaining predictions
ridge.pred = predict(ridge.model.college, s = bestlam.college,
                     newx = x_college[test_subset, ])

# test MSE
mean((y_college[test_subset] - ridge.pred)^2 )
```

```
## [1] 1022341
```

(d) Fit a lasso model on the training set, with λ chosen by cross validation. Report the test error obtained, along with the number of non-zero coefficient estimates.

```
lasso.mod = glmnet(x_college[train_subset, ],
                           y_college[train_subset],
                   alpha = 1, lambda = grid)
cv.out = cv.glmnet(x_college[train_subset, ],
```

```r
                            y_college[train_subset], alpha = 1)
bestlam3 = cv.out$lambda.min

lasso.pred = predict(lasso.mod, s = bestlam3,
                      newx = x_college[test_subset, ])
mean((lasso.pred - y_college[test_subset])^2)
```

```
## [1] 1189178
```

(e) Fit a PCR model on the training set, with `M` chosen by cross validation. Report the test error obtained, along with the value of `M` selected by cross-validation.

```r
# apply PCR to data
pcr.fit = pcr(Apps ~ ., data = college, subset = train_subset, scale = TRUE,
              validation = "CV")
# summary(pcr.fit)
# validationplot(pcr.fit, val.type = "MSEP")

# 5-9 components explains a reasonable proportion of variance in the data
# (between 75-90 percent)

pcr.pred = predict(pcr.fit, college_test, ncomp = 9)
mean((pcr.pred - y_college[test_subset])^2)
```

```
## [1] 1531532
```

(f) Fit a PLS model on the training set, with `M` chosen by cross validation. Report the test error obtained, along with the value of `M` selected by cross-validation.

```r
# implementing PLS
pls.fit = plsr(Apps ~ ., data = college, subset = train_subset, 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            4125     2220     1906     1710     1533     1257     1186
## adjCV         4125     2209     1891     1704     1506     1242     1177
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1178     1175     1168      1165      1167      1164      1166
## adjCV     1170     1167     1159      1156      1157      1156      1158
##        14 comps  15 comps  16 comps  17 comps
## CV         1164      1163      1163      1163
## adjCV      1156      1154      1154      1154
##
```

```
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       26.08    38.17    63.93    66.52    70.04    74.32    78.66    81.78
## Apps    74.67    83.88    86.37    91.11    93.12    93.63    93.69    93.74
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X       83.87     85.87     88.17     91.30     93.37     95.56     96.91
## Apps    93.79     93.80     93.82     93.84     93.84     93.85     93.85
##       16 comps  17 comps
## X        98.94    100.00
## Apps     93.85     93.85
```

```
# lowest CV error occurs at M = 10

# evaluating test set MSE
pls.pred = predict(pls.fit, college_test, ncomp = 10)
mean((pls.pred - y_college[test_subset])^2)
```

```
## [1] 1223369
```

Exercise 10

We have seen that as the number of features used in a model increases, the training error will necessarily decrease, but the test error may not. We will now explore this in a simulated data set.

(a) Generate a data set with p = 20 features, n = 1,000 observations, and an associated quantitative response vector generated according to the model $Y = X\beta + \epsilon$ where β has some elements that are exactly equal to zero.

```
X = matrix(rnorm(1000 * 20, mean = 0, sd = 3), nrow = 1000, ncol = 20)
epsilon = rnorm(1000, mean = 0, sd = 5)
beta = c(5, 3.8, 2.9, 25, 6.50, 15, -9.60, 2.5, 0, -2, 9.1, 0.2,
         -3, 0, 23, 5, -14.3, 8, 0, -0.1)
Y = (X %*% beta) + epsilon
```

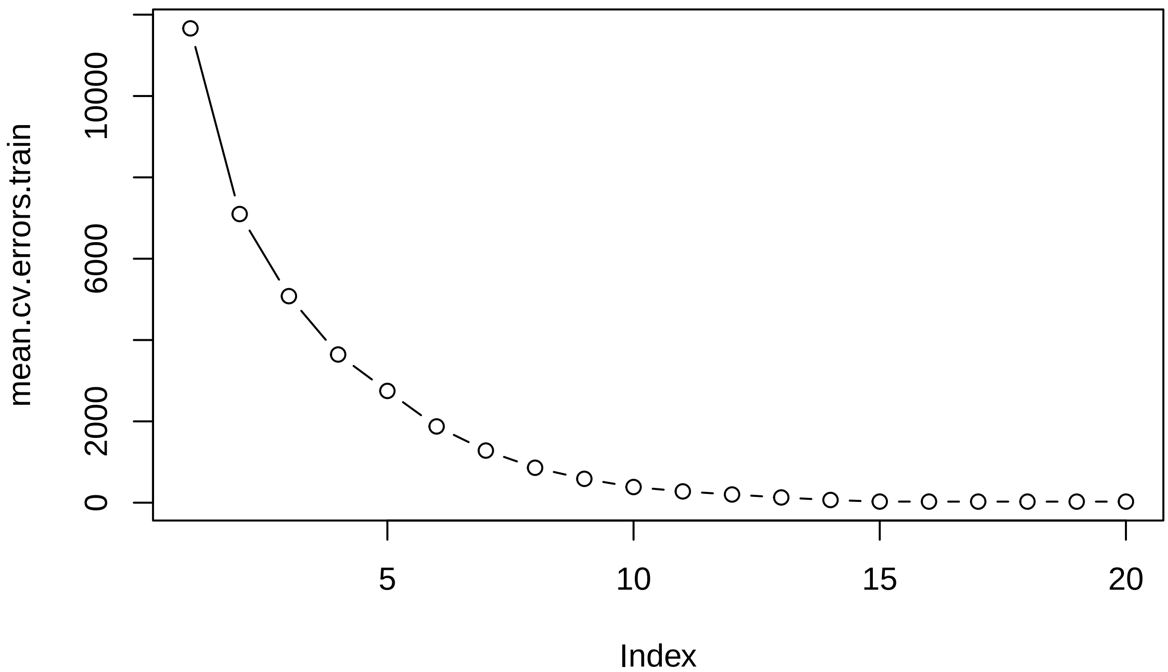
(b) Split your data set into a training set containing 100 observations and a test set containing 900 observations.

```
xy_df = data.frame(X, Y)

training_indices <- sample(seq_len(nrow(xy_df)), 100)

train_xy <- xy_df[training_indices, ]
test_xy <- xy_df[-training_indices, ]
```

(c) Perform best subset selection on the training set, and plot the training set MSE associated with the best model of each size.

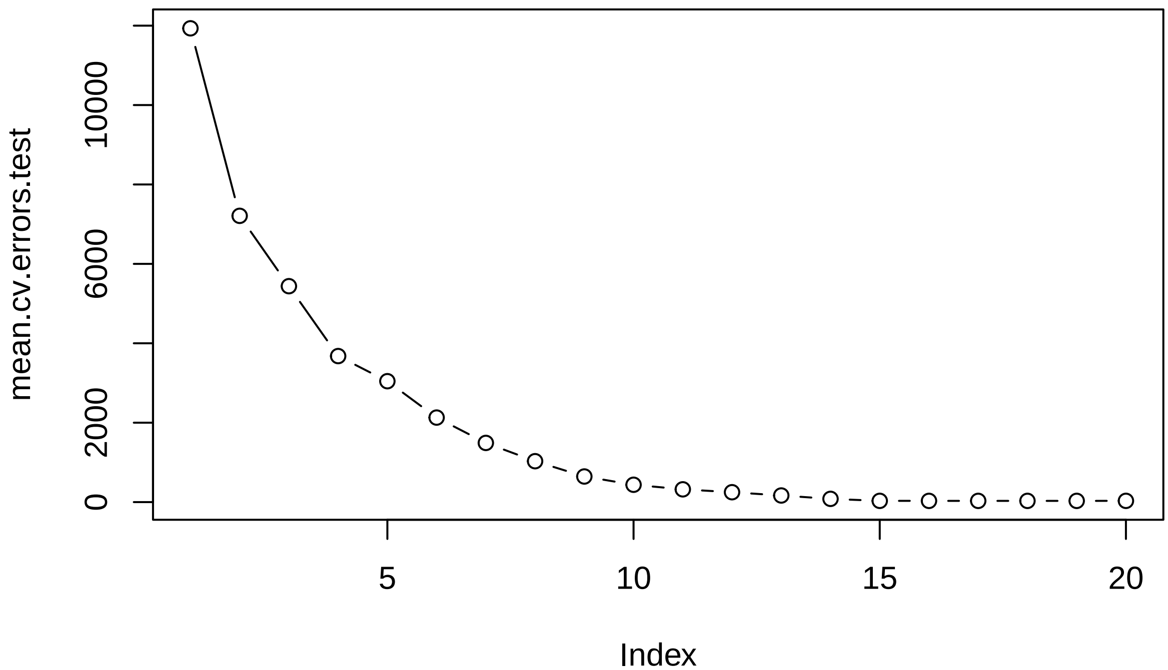
```
regfit.bss = regsubsets(Y ~ ., data = train_xy, nvmax = 20)

train.mat = model.matrix(Y ~ ., data = train_xy)
```

```r
test.mat = model.matrix(Y ~ ., data = test_xy)

predict.regsubsets = function(object, newdata, id, ...) {
  form = as.formula(object$cal[[2]])
  mat = model.matrix(form, newdata)
  coefi = coef(object, id = id)
  xvars = names(coefi)
  mat[, xvars] %*% coefi
}

# picking models via k-fold cross-validation
k = 10
n = nrow
folds = sample(rep(1:k, length = n))
cv.errors = matrix(NA, k, 20, dimnames = list(NULL, paste(1:20)))

for (j in 1:k) {
  best.fit = regsubsets(Y ~ ., data = xy_df[folds != j, ], nvmax = 20)
  for (i in 1:20) {
    pred = predict(best.fit, xy_df[folds == j, ], id = i)
    cv.errors[j, i] = mean((xy_df$Y[folds == j] - pred)^2)
  }
}

mean.cv.errors.train = apply(cv.errors, 2, mean)

plot(mean.cv.errors.train, type = "b")
```



(d) Plot the test set MSE associated with the best model of each size.

```
cv.errors = matrix(NA, k, 20, dimnames = list(NULL, paste(1:20)))

for (j in 1:k) {
  best.fit = regsubsets(Y ~ ., data = train_xy[folds != j, ], nvmax = 20)
  for (i in 1:20) {
    pred = predict(best.fit, test_xy[folds == j, ], id = i)
    cv.errors[j, i] = mean((test_xy$Y[folds == j] - pred)^2)
  }
}

mean.cv.errors.test = apply(cv.errors, 2, mean)
plot(mean.cv.errors.test, type = "b")
```



(e) For which model size does the test set MSE take on its minimum value? Comment on your results. If it takes on its minimum value for a model containing only an intercept or a model containing all of the features, then play around with the way that you are generating the data in (a) until you come up with a scenario in which the test set MSE is minimized for an intermediate model size.

```
coef(regfit.bss, which.min(mean.cv.errors.test))
```

```
##  (Intercept)           X1           X2           X3           X4           X5 
##  -0.65143994   4.98106596   3.92993557   2.75883846  24.96218696   6.62955036 
##           X6           X7           X8          X10          X11          X12 
##  14.82918139  -9.60637317   2.84040702  -2.06021140   9.21448999   0.18795786 
##          X13          X14          X15          X16          X17          X18 
##  -3.25449214   0.02982196  22.90786950   4.96403536 -14.15162036   7.64395403 
##          X20 
##  -0.23830236
```