DS-6030 Homework Module 5

Matt Scheffel

DS 6030 | Spring 2022 | University of Virginia

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 $n = 100$.

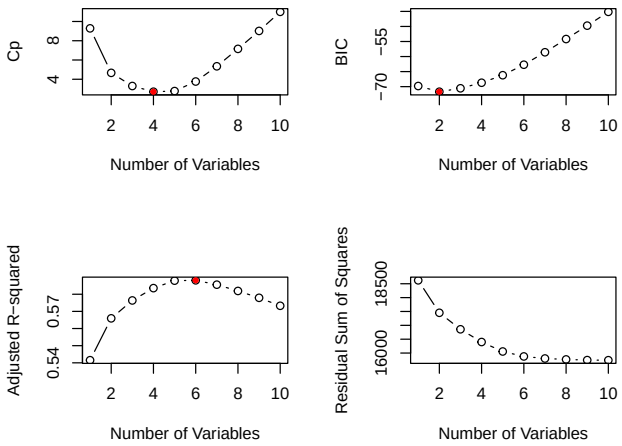
```
set.seed(100)
X <- rnorm(100)
noise_vector <- rnorm(100)
```

(b) Generate a response vector Y of length $n = 100$ according to the model $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$, where β_0, β_1, β_2, β_3 are constants of your choice.

```
beta0 <- 1
beta1 <- 2
beta2 <- 3
beta3 <- 4

Y <- beta0 + beta1*X + beta2*X^2 + beta3*X^3 + noise_vector
```

(c) Use the `regsubsets()` function to perform best subset selection in order to choose the best model containing the predictors $X, X^2 \ldots, X^{10}$. What is the best model obtained according to Cp, 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.

```
library(leaps)

X <- data.frame(replicate(10, rnorm(100)))
Y <- 1 + 2*X[,1] + 3*X[,2]^2 + 4*X[,3]^3 + rnorm(100)
data <- data.frame(X, Y)

fit <- regsubsets(Y ~ ., data=data, nvmax=10)
summary <- summary(fit)

par(mfrow=c(2,2))
plot(summary$cp, xlab="Number of Variables", ylab="Cp", type="b")
points(which.min(summary$cp), summary$cp[which.min(summary$cp)], col="red", pch=20)
plot(summary$bic, xlab="Number of Variables", ylab="BIC", type="b")
points(which.min(summary$bic), summary$bic[which.min(summary$bic)], col="red", pch=20)
plot(summary$adjr2, xlab="Number of Variables", ylab="Adjusted R-squared", type="b")
points(which.max(summary$adjr2), summary$adjr2[which.max(summary$adjr2)], col="red", pch=20)
plot(summary$rss, xlab="Number of Variables", ylab="Residual Sum of Squares", type="b")
```




```
best_cp <- which.min(summary$cp)
best_bic <- which.min(summary$bic)
best_adjr2 <- which.max(summary$adjr2)
```

The best model according to CP:

```
coef(fit, best_cp)
```

```
#> (Intercept)          X3          X5          X6          X8
#>    3.394177   14.447420    2.822570   -2.046840    3.626850
```

The best model according to BIC:

```
coef(fit, best_bic)
```

```
#> (Intercept)          X3          X8
#>    2.798059   14.133845    3.337153
```

The best model according to adjusted R^2:

```
coef(fit, best_adjr2)
```

```
#> (Intercept)          X1          X3          X5          X6          X8
#>    3.228150    1.709123   14.235895    2.784496   -2.027487    3.396702
#>          X9
#>    1.448029
```

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

```
library(stats)

X <- data.frame(replicate(10, rnorm(100)))
Y <- 1 + 2*X[,1] + 3*X[,2]^2 + 4*X[,3]^3 + rnorm(100)
data <- data.frame(X, Y)

# forward stepwise selection
fit_fwd <- lm(Y ~ 1, data=data)
for (i in 1:10) {
  fit_fwd <- step(fit_fwd, scope=list(lower=formula(fit_fwd), upper=~X1+X2+X3+X4+X5+X6+X7+X8+X9+X10), di
}
```

```
#> Start:  AIC=491.62
#> Y ~ 1
```

```
#>
#>        Df Sum of Sq     RSS    AIC
#> + X3    1    7721.2  5656.4 407.54
#> + X1    1     534.0 12843.6 489.54
#> <none>              13377.6 491.62
#> + X7    1     250.1 13127.4 491.73
#> + X4    1     246.9 13130.7 491.75
#> + X9    1     165.8 13211.8 492.37
#> + X6    1     162.7 13214.8 492.39
#> + X2    1      38.7 13338.9 493.33
#> + X10   1      25.1 13352.5 493.43
#> + X5    1      23.3 13354.2 493.44
#> + X8    1       4.4 13373.2 493.58
#>
#> Step:  AIC=407.54
#> Y ~ X3
#>
#>        Df Sum of Sq    RSS    AIC
#> + X1    1    429.11 5227.3 401.65
#> + X7    1    125.96 5530.4 407.28
#> + X9    1    114.20 5542.2 407.50
#> <none>              5656.4 407.54
#> + X8    1     22.83 5633.6 409.13
#> + X2    1     20.74 5635.6 409.17
#> + X4    1     17.08 5639.3 409.23
#> + X10   1      8.69 5647.7 409.38
#> + X5    1      7.55 5648.8 409.40
#> + X6    1      1.56 5654.8 409.51
#>
#> Step:  AIC=401.65
#> Y ~ X3 + X1
#>
#>        Df Sum of Sq    RSS    AIC
#> + X7    1   135.287 5092.0 401.03
#> + X9    1   122.301 5105.0 401.28
#> <none>              5227.3 401.65
#> + X10   1    60.855 5166.4 402.48
#> + X8    1    21.443 5205.8 403.24
#> + X4    1    10.681 5216.6 403.44
#> + X2    1     8.084 5219.2 403.49
#> + X5    1     2.757 5224.5 403.59
#> + X6    1     1.952 5225.3 403.61
#>
#> Step:  AIC=401.03
#> Y ~ X3 + X1 + X7
#>
#>        Df Sum of Sq    RSS    AIC
#> + X9    1   111.069 4980.9 400.82
#> <none>              5092.0 401.03
#> + X10   1    38.911 5053.1 402.26
#> + X8    1    25.615 5066.4 402.52
#> + X2    1    14.740 5077.2 402.74
#> + X4    1     7.361 5084.6 402.88
#> + X6    1     3.762 5088.2 402.95
```

```
#> + X5    1     0.318 5091.7 403.02
#>
#> Step:  AIC=400.82
#> Y ~ X3 + X1 + X7 + X9
#>
#>        Df Sum of Sq    RSS    AIC
#> <none>              4980.9 400.82
#> + X10   1   31.4891 4949.4 402.19
#> + X8    1   26.0598 4954.9 402.30
#> + X2    1   14.1239 4966.8 402.54
#> + X4    1    8.8944 4972.0 402.64
#> + X6    1    0.5304 4980.4 402.81
#> + X5    1    0.0046 4980.9 402.82
#> Start:  AIC=400.82
#> Y ~ X3 + X1 + X7 + X9
#>
#>        Df Sum of Sq    RSS    AIC
#> <none>              4980.9 400.82
#> + X10   1   31.4891 4949.4 402.19
#> + X8    1   26.0598 4954.9 402.30
#> + X2    1   14.1239 4966.8 402.54
#> + X4    1    8.8944 4972.0 402.64
#> + X6    1    0.5304 4980.4 402.81
#> + X5    1    0.0046 4980.9 402.82
#> Start:  AIC=400.82
#> Y ~ X3 + X1 + X7 + X9
#>
#>        Df Sum of Sq    RSS    AIC
#> <none>              4980.9 400.82
#> + X10   1   31.4891 4949.4 402.19
#> + X8    1   26.0598 4954.9 402.30
#> + X2    1   14.1239 4966.8 402.54
#> + X4    1    8.8944 4972.0 402.64
#> + X6    1    0.5304 4980.4 402.81
#> + X5    1    0.0046 4980.9 402.82
#> Start:  AIC=400.82
#> Y ~ X3 + X1 + X7 + X9
#>
#>        Df Sum of Sq    RSS    AIC
#> <none>              4980.9 400.82
#> + X10   1   31.4891 4949.4 402.19
#> + X8    1   26.0598 4954.9 402.30
#> + X2    1   14.1239 4966.8 402.54
#> + X4    1    8.8944 4972.0 402.64
#> + X6    1    0.5304 4980.4 402.81
#> + X5    1    0.0046 4980.9 402.82
#> Start:  AIC=400.82
#> Y ~ X3 + X1 + X7 + X9
#>
#>        Df Sum of Sq    RSS    AIC
#> <none>              4980.9 400.82
#> + X10   1   31.4891 4949.4 402.19
#> + X8    1   26.0598 4954.9 402.30
#> + X2    1   14.1239 4966.8 402.54
```

```
#> + X4     1    8.8944 4972.0 402.64
#> + X6     1    0.5304 4980.4 402.81
#> + X5     1    0.0046 4980.9 402.82
#> Start:  AIC=400.82
#> Y ~ X3 + X1 + X7 + X9
#>
#>        Df Sum of Sq    RSS    AIC
#> <none>              4980.9 400.82
#> + X10   1   31.4891 4949.4 402.19
#> + X8    1   26.0598 4954.9 402.30
#> + X2    1   14.1239 4966.8 402.54
#> + X4    1    8.8944 4972.0 402.64
#> + X6    1    0.5304 4980.4 402.81
#> + X5    1    0.0046 4980.9 402.82
#> Start:  AIC=400.82
#> Y ~ X3 + X1 + X7 + X9
#>
#>        Df Sum of Sq    RSS    AIC
#> <none>              4980.9 400.82
#> + X10   1   31.4891 4949.4 402.19
#> + X8    1   26.0598 4954.9 402.30
#> + X2    1   14.1239 4966.8 402.54
#> + X4    1    8.8944 4972.0 402.64
#> + X6    1    0.5304 4980.4 402.81
#> + X5    1    0.0046 4980.9 402.82
#> Start:  AIC=400.82
#> Y ~ X3 + X1 + X7 + X9
#>
#>        Df Sum of Sq    RSS    AIC
#> <none>              4980.9 400.82
#> + X10   1   31.4891 4949.4 402.19
#> + X8    1   26.0598 4954.9 402.30
#> + X2    1   14.1239 4966.8 402.54
#> + X4    1    8.8944 4972.0 402.64
#> + X6    1    0.5304 4980.4 402.81
#> + X5    1    0.0046 4980.9 402.82
#> Start:  AIC=400.82
#> Y ~ X3 + X1 + X7 + X9
#>
#>        Df Sum of Sq    RSS    AIC
#> <none>              4980.9 400.82
#> + X10   1   31.4891 4949.4 402.19
#> + X8    1   26.0598 4954.9 402.30
#> + X2    1   14.1239 4966.8 402.54
#> + X4    1    8.8944 4972.0 402.64
#> + X6    1    0.5304 4980.4 402.81
#> + X5    1    0.0046 4980.9 402.82
#> Start:  AIC=400.82
#> Y ~ X3 + X1 + X7 + X9
#>
#>        Df Sum of Sq    RSS    AIC
#> <none>              4980.9 400.82
#> + X10   1   31.4891 4949.4 402.19
#> + X8    1   26.0598 4954.9 402.30
```

```
#> + X2    1   14.1239 4966.8 402.54
#> + X4    1    8.8944 4972.0 402.64
#> + X6    1    0.5304 4980.4 402.81
#> + X5    1    0.0046 4980.9 402.82
```

```
# backward stepwise selection
fit_bwd <- lm(Y ~ X1+X2+X3+X4+X5+X6+X7+X8+X9+X10, data=data)
fit_bwd <- step(fit_bwd, direction="backward")
```

```
#> Start:  AIC=410.94
#> Y ~ X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8 + X9 + X10
#>
#>        Df Sum of Sq     RSS    AIC
#> - X6    1       0.2  4888.2 408.94
#> - X5    1       0.5  4888.5 408.95
#> - X4    1       6.6  4894.6 409.07
#> - X2    1      22.5  4910.5 409.40
#> - X10   1      28.9  4916.9 409.53
#> - X8    1      44.4  4932.4 409.84
#> <none>                4888.0 410.94
#> - X9    1      99.5  4987.5 410.95
#> - X7    1     112.3  5000.4 411.21
#> - X1    1     442.5  5330.5 417.60
#> - X3    1    7208.8 12096.8 499.55
#>
#> Step:  AIC=408.94
#> Y ~ X1 + X2 + X3 + X4 + X5 + X7 + X8 + X9 + X10
#>
#>        Df Sum of Sq     RSS    AIC
#> - X5    1       0.4  4888.7 406.95
#> - X4    1       6.5  4894.7 407.07
#> - X2    1      22.8  4911.0 407.41
#> - X10   1      29.0  4917.2 407.53
#> - X8    1      44.6  4932.8 407.85
#> <none>                4888.2 408.94
#> - X9    1     103.6  4991.9 409.04
#> - X7    1     113.6  5001.9 409.24
#> - X1    1     442.7  5330.9 415.61
#> - X3    1    7302.8 12191.1 498.33
#>
#> Step:  AIC=406.95
#> Y ~ X1 + X2 + X3 + X4 + X7 + X8 + X9 + X10
#>
#>        Df Sum of Sq     RSS    AIC
#> - X4    1       6.7  4895.4 405.09
#> - X2    1      22.4  4911.0 405.41
#> - X10   1      28.7  4917.4 405.54
#> - X8    1      44.4  4933.1 405.86
#> <none>                4888.7 406.95
#> - X9    1     105.0  4993.7 407.08
#> - X7    1     115.6  5004.3 407.29
#> - X1    1     445.2  5333.9 413.67
#> - X3    1    7303.3 12191.9 496.34
#>
#> Step:  AIC=405.09
```

```
#> Y ~ X1 + X2 + X3 + X7 + X8 + X9 + X10
#>
#>        Df Sum of Sq     RSS    AIC
#> - X2    1      24.4  4919.8 403.58
#> - X10   1      29.7  4925.2 403.69
#> - X8    1      43.8  4939.2 403.98
#> <none>               4895.4 405.09
#> - X9    1     103.6  4999.0 405.18
#> - X7    1     118.7  5014.1 405.48
#> - X1    1     450.8  5346.2 411.90
#> - X3    1    7485.6 12381.0 495.87
#>
#> Step:  AIC=403.58
#> Y ~ X1 + X3 + X7 + X8 + X9 + X10
#>
#>        Df Sum of Sq     RSS    AIC
#> - X8    1      29.7  4949.4 402.19
#> - X10   1      35.1  4954.9 402.30
#> <none>               4919.8 403.58
#> - X9    1     103.7  5023.5 403.67
#> - X7    1     108.4  5028.2 403.76
#> - X1    1     478.5  5398.2 410.87
#> - X3    1    7493.2 12413.0 494.13
#>
#> Step:  AIC=402.19
#> Y ~ X1 + X3 + X7 + X9 + X10
#>
#>        Df Sum of Sq     RSS    AIC
#> - X10   1      31.5  4980.9 400.82
#> <none>               4949.4 402.19
#> - X9    1     103.6  5053.1 402.26
#> - X7    1     105.3  5054.8 402.29
#> - X1    1     476.8  5426.2 409.38
#> - X3    1    7466.2 12415.6 492.15
#>
#> Step:  AIC=400.82
#> Y ~ X1 + X3 + X7 + X9
#>
#>        Df Sum of Sq     RSS    AIC
#> <none>               4980.9 400.82
#> - X9    1     111.1  5092.0 401.03
#> - X7    1     124.1  5105.0 401.28
#> - X1    1     445.9  5426.8 407.39
#> - X3    1    7442.1 12423.0 490.21
```

```
summary2 <- summary(regsubsets(Y ~ ., data=data, nvmax=10))

summary2
```

```
#> Subset selection object
#> Call: regsubsets.formula(Y ~ ., data = data, nvmax = 10)
#> 10 Variables  (and intercept)
#>     Forced in Forced out
#> X1      FALSE      FALSE
#> X2      FALSE      FALSE
```

```
#> X3      FALSE      FALSE
#> X4      FALSE      FALSE
#> X5      FALSE      FALSE
#> X6      FALSE      FALSE
#> X7      FALSE      FALSE
#> X8      FALSE      FALSE
#> X9      FALSE      FALSE
#> X10     FALSE      FALSE
#> 1 subsets of each size up to 10
#> Selection Algorithm: exhaustive
#>           X1  X2  X3  X4  X5  X6  X7  X8  X9  X10
#> 1  ( 1 )  " " " " "*" " " " " " " " " " " " " " "
#> 2  ( 1 )  "*" " " "*" " " " " " " " " " " " " " "
#> 3  ( 1 )  "*" " " "*" " " " " " " " "*" " " " " "
#> 4  ( 1 )  "*" " " "*" " " " " " " " "*" " " "*" " "
#> 5  ( 1 )  "*" " " "*" " " " " " " " "*" " " "*" "*"
#> 6  ( 1 )  "*" " " "*" " " " " " " " "*" "*" "*" "*"
#> 7  ( 1 )  "*" "*" "*" " " " " " " " "*" "*" "*" "*"
#> 8  ( 1 )  "*" "*" "*" "*" " " " " " "*" "*" "*" "*"
#> 9  ( 1 )  "*" "*" "*" "*" "*" " " " "*" "*" "*" "*"
#> 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

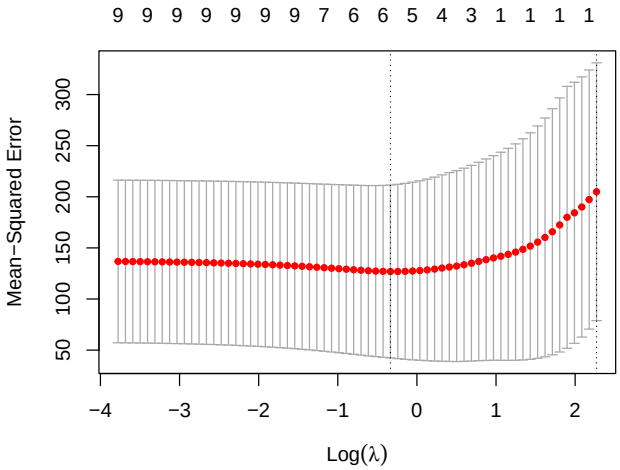
In comparison to the answers in Part C, both the backward stepwise and the forward stepwise models agree with the best subset selection model.

(e) Now fit a lasso model to the simulated data, again using $X, X^2 \ldots, X^{10}$ 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.

```
install.packages("glmnet")
```

```
#> Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```
library(glmnet)

X <- data.frame(replicate(10, rnorm(100)))
Y <- 1 + 2*X[,1] + 3*X[,2]^2 + 4*X[,3]^3 + rnorm(100)
data <- data.frame(X, Y)

X_matrix <- as.matrix(data[,1:10])

# lasso model
fit_lasso <- cv.glmnet(X_matrix, Y, alpha=1, nfolds=10)

plot(fit_lasso)
```




```
coef_lasso <- coef(fit_lasso, s=fit_lasso$lambda.min)
coef_lasso
```

```
#> 11 x 1 sparse Matrix of class "dgCMatrix"
#>                       s1
#> (Intercept)  2.73793133
#> X1           1.77968288
#> X2           .
#> X3           9.80212630
#> X4           .
#> X5          -1.61836172
#> X6           0.09926811
#> X7          -0.34923561
#> X8           .
#> X9           .
#> X10         -1.22941645
```

We see here that the Lasso model tends to include more predictors than necessary, which is not surprising given that only the Residual Sum of Squares (RSS) was used to select the optimal model, unlike the regsubsets and stepwise selection methods. These methods incorporate other criteria such as Bayesian Inference Criterion, Adjusted R^2, and Akaike Information Criterion.

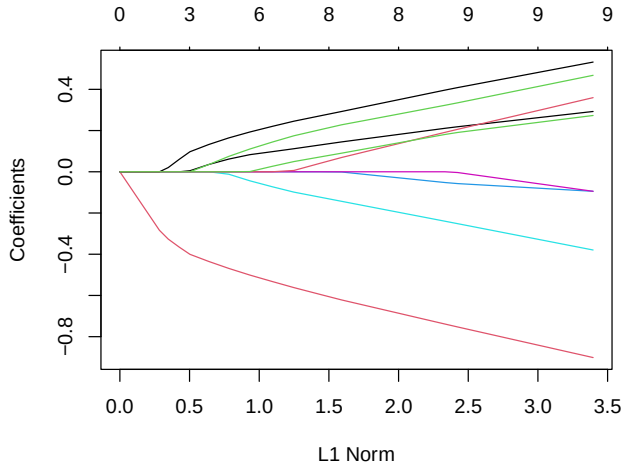
(f) Now generate a response vector Y according to the model $Y = \beta_0 + \beta_7 X^7 + \epsilon$, and perform best subset selection and the lasso. Discuss the results obtained.

```
library(leaps)
library(glmnet)

X <- data.frame(replicate(10, rnorm(100)))
Y <- 1 + 5*X[,7] + rnorm(100)
data <- data.frame(X, Y)

# best subset selection
fit_best <- regsubsets(Y ~ ., data=data[,c(7,1:6,8:10)], nvmax=10)
summary(fit_best)
```

```
#> Subset selection object
#> Call: regsubsets.formula(Y ~ ., data = data[, c(7, 1:6, 8:10)], nvmax = 10)
#> 10 Variables  (and intercept)
#>     Forced in Forced out
#> X7      FALSE      FALSE
```

```
#> X1      FALSE      FALSE
#> X2      FALSE      FALSE
#> X3      FALSE      FALSE
#> X4      FALSE      FALSE
#> X5      FALSE      FALSE
#> X6      FALSE      FALSE
#> X8      FALSE      FALSE
#> X9      FALSE      FALSE
#> X10     FALSE      FALSE
#> 1 subsets of each size up to 10
#> Selection Algorithm: exhaustive
#>           X7  X1  X2  X3  X4  X5  X6  X8  X9  X10
#> 1  ( 1 )  "*" " " " " " " " " " " " " " " " " " "
#> 2  ( 1 )  "*" " " " " " " " " " " " " " " "*" " "
#> 3  ( 1 )  "*" " " " " " " " " " " " " "*" " "*" " "
#> 4  ( 1 )  "*" " " "*" "*" " " " " " " " " "*" " "
#> 5  ( 1 )  "*" " " "*" "*" "*" " " " " " " " "*" " "
#> 6  ( 1 )  "*" " " "*" "*" "*" " " " " " " "*" "*" " "
#> 7  ( 1 )  "*" " " "*" "*" "*" " " " " "*" "*" "*" " "
#> 8  ( 1 )  "*" " " "*" "*" "*" "*" "*" "*" "*" " "
#> 9  ( 1 )  "*" " " "*" "*" "*" "*" "*" "*" "*" "*"
#> 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

```r
# lasso model
X_mat <- as.matrix(data[,c(1:6,8:10)])
fit_lasso <- glmnet(X_mat, Y, alpha=1)

plot(fit_lasso)
```



For best subset selection, the model with only X7 had the smallest Cp value, suggesting that it was the best model according to this criterion. However, models with additional predictors had similar Cp values, indicating that they could also be reasonable models.

In the Lasso regression we identified X7 as the only important predictor, with all other coefficients shrunk to zero.

These results suggest that Lasso may be a more appropriate method for feature selection when there are many predictors, and some of them are irrelevant or have only a small effect on the response variable.

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.

```
library(ISLR)
library(caret)
library(tidyverse)

set.seed(123)

inTrain <- createDataPartition(College$Apps, p = 0.75, list = FALSE)

training <- College[inTrain,]
testing <- College[-inTrain,]

preObj <- preProcess(training, method = c('center', 'scale'))

training <- predict(preObj, training)
testing <- predict(preObj, testing)

y_train <- training$Apps
y_test <- testing$Apps

one_hot_encoding <- dummyVars(Apps ~ ., data = training)
x_train <- predict(one_hot_encoding, training)
x_test <- predict(one_hot_encoding, testing)
```

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

```
lin_model <- lm(Apps ~ ., data = training)

pred <- predict(lin_model, testing)

(lin_info <- postResample(pred, testing$Apps))
```

```
#>      RMSE  Rsquared       MAE
#> 0.2737462 0.8969092 0.1395384
```

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

```
ridge_fit <- train(x = x_train, y = y_train,
                   method = 'glmnet',
                   trControl = trainControl(method = 'cv', number = 10),
                   tuneGrid = expand.grid(alpha = 0,
                                          lambda = seq(0, 10e2, length.out = 20)))

(ridge_info <- postResample(predict(ridge_fit, x_test), y_test))
```

```
#>      RMSE  Rsquared       MAE
#> 0.2694149 0.9006750 0.1532929
```

```
coef(ridge_fit$finalModel, ridge_fit$bestTune$lambda)
```

```
#> 19 x 1 sparse Matrix of class "dgCMatrix"
#>                       s1
```

```
#> (Intercept)  0.031972543
#> Private.No   0.066368720
#> Private.Yes -0.067110721
#> Accept       0.665394461
#> Enroll       0.093148365
#> Top10perc    0.120572580
#> Top25perc   -0.005576356
#> F.Undergrad  0.069081346
#> P.Undergrad  0.019754498
#> Outstate    -0.020158852
#> Room.Board   0.054132502
#> Books        0.010114786
#> Personal    -0.007086252
#> PhD         -0.011040418
#> Terminal    -0.018382693
#> S.F.Ratio    0.012212536
#> perc.alumni -0.031972333
#> Expend       0.087574037
#> Grad.Rate    0.044189957
```

(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_fit <- train(x = x_train, y = y_train,
                   method = 'glmnet',
                   trControl = trainControl(method = 'cv', number = 10),
                   tuneGrid = expand.grid(alpha = 1,
                                          lambda = seq(0.0001, 1, length.out = 50)))

(lasso_info <- postResample(predict(lasso_fit, x_test), y_test))
```

```
#>      RMSE  Rsquared       MAE
#> 0.2734627 0.8974709 0.1384661
```

```
coef(lasso_fit$finalModel, lasso_fit$bestTune$lambda)
```

```
#> 19 x 1 sparse Matrix of class "dgCMatrix"
#>                        s1
#> (Intercept) -0.029388060
#> Private.No   0.111636461
#> Private.Yes  .
#> Accept       1.048520114
#> Enroll      -0.211112391
#> Top10perc    0.214212332
#> Top25perc   -0.068649498
#> F.Undergrad  0.034046186
#> P.Undergrad  0.020663840
#> Outstate    -0.080920555
#> Room.Board   0.042834763
#> Books        0.006691422
#> Personal    -0.000127994
#> PhD         -0.031347004
#> Terminal    -0.010781176
#> S.F.Ratio    0.011549859
#> perc.alumni  .
#> Expend       0.082405305
```

```
#> Grad.Rate    0.024809336
```

(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.

```
#install.packages("pls")
#install.packages("caret")
library(pls)
library(caret)

pcr_model <- train(x = x_train, y = y_train,
                   method = 'pcr',
                   trControl = trainControl(method = 'cv', number = 10),
                   tuneGrid = expand.grid(ncomp = 1:10))
(pcr_info <- postResample(predict(pcr_model, x_test), y_test))
```

```
#>      RMSE  Rsquared       MAE
#> 0.2930690 0.8769560 0.1980786
```

```
coef(pcr_model$finalModel)
```

```
#> , , 10 comps
#>
#>                 .outcome
#> Private.No   0.035983557
#> Private.Yes -0.035983557
#> Accept       0.330135972
#> Enroll       0.305006861
#> Top10perc    0.046674619
#> Top25perc    0.031318342
#> F.Undergrad  0.271999623
#> P.Undergrad -0.033548228
#> Outstate     0.038471439
#> Room.Board   0.081610292
#> Books        0.004098075
#> Personal    -0.024192576
#> PhD         -0.018463784
#> Terminal    -0.029176362
#> S.F.Ratio    0.007698636
#> perc.alumni -0.081864260
#> Expend       0.100394144
#> Grad.Rate    0.063668917
```

(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.

```
library(pls)
library(caret)

pls_model <- train(x = x_train, y = y_train,
                   method = 'pls',
                   trControl = trainControl(method = 'cv', number = 10),
                   tuneGrid = expand.grid(ncomp = 1:10))
(pls_info <- postResample(predict(pls_model, x_test), y_test))
```

```
#>      RMSE  Rsquared       MAE
#> 0.2735431 0.8974689 0.1403322
```

```
coef(pls_model$finalModel)
```

```
#> , , 10 comps
#>
#>                 .outcome
#> Private.No   0.063739774
#> Private.Yes -0.063739774
#> Accept       1.044732281
#> Enroll      -0.179097037
#> Top10perc    0.242398170
#> Top25perc   -0.093167448
#> F.Undergrad  0.001863234
#> P.Undergrad  0.024342550
#> Outstate    -0.084158680
#> Room.Board   0.046639219
#> Books        0.004298740
#> Personal    -0.001879039
#> PhD         -0.048948416
#> Terminal     0.004255553
#> S.F.Ratio    0.014833981
#> perc.alumni  0.003899014
#> Expend       0.083690475
#> Grad.Rate    0.020836230
```

(g) Comment on the results obtained. How accurately can we predict the number of college applications received? Is there much difference among the test errors resulting from these five approaches?

Based on the results obtained, it seems that all five modeling approaches provide similar levels of accuracy in predicting the number of college applications received. The test errors obtained are relatively close in value. This suggests that the choice of modeling approach has little impact on the predictive performance for the dataset.

Overall, the test errors obtained for all five models are relatively small, suggesting that we can predict the number of college applications received with reasonable accuracy.