Nonlinear Modeling Ch. 7 Exercises

Lucy L.

2024-01-31

```
library(ISLR2)
library(splines)
library(gam)
library(glmnet)
library(akima)
library(interp)
library(caret)
library(boot)
library(dplyr)
library(gridExtra)
library(ggplot2)
library(leaps)

wage = Wage
set.seed(1)
```

Exercise 6

In this exercise, you will further analyze the `Wage` data set considered throughout this chapter.

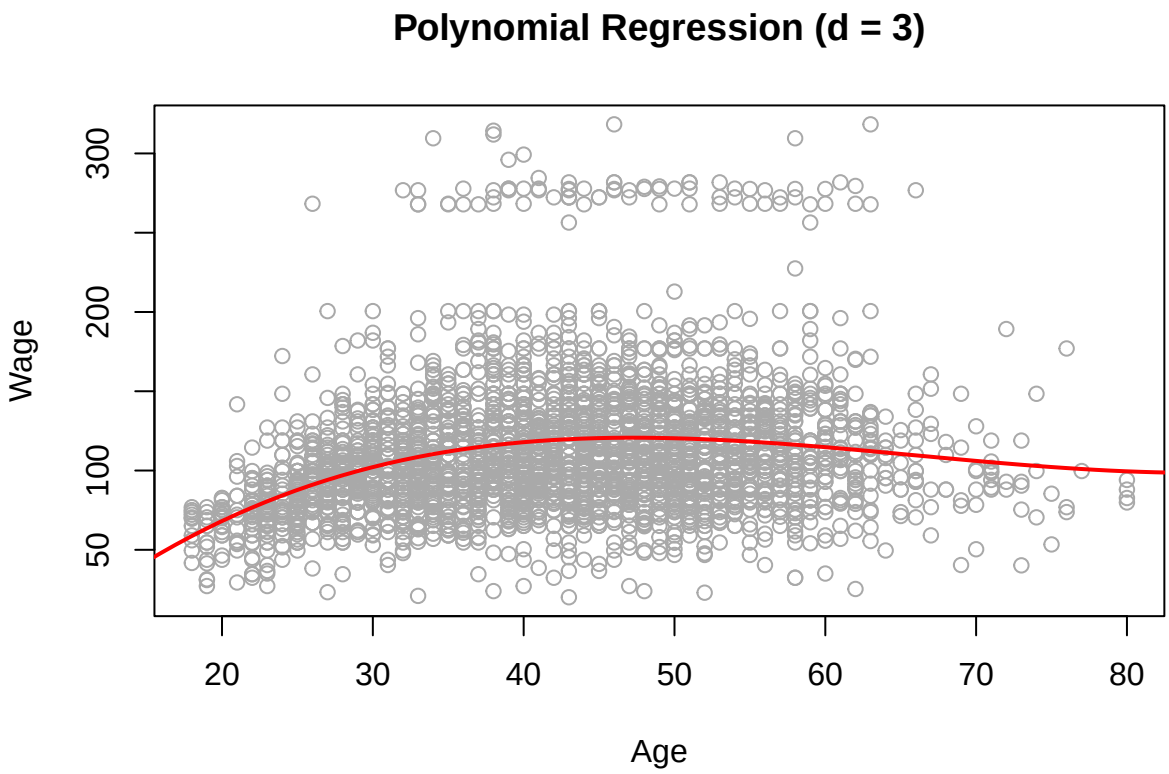
(a) Perform polynomial regression to predict `wage` using `age`. Use cross-validation to select the optimal degree `d` for the polynomial. What degree was chosen, and how does this compare to the results of hypothesis testing using ANOVA? Make a plot of the resulting polynomial fit to the data.

```
attach(Boston)
ctrl <- trainControl(method = "cv", number = 10)

model1 <- train(wage ~ age, data = Wage, method = "lm", trControl = ctrl)
model2 <- train(wage ~ poly(age, 2), data = Wage, method = "lm", trControl = ctrl)
model3 <- train(wage ~ poly(age, 3), data = Wage, method = "lm", trControl = ctrl)
model4 <- train(wage ~ poly(age, 4), data = Wage, method = "lm", trControl = ctrl)
model5 <- train(wage ~ poly(age, 5), data = Wage, method = "lm", trControl = ctrl)

# print(model1)
# print(model2)
# print(model3)
# print(model4)
# print(model5)

reg.model1 <- lm(wage ~ age, data = Wage)
```

```r
reg.model2 <- lm(wage ~ poly(age, 2), data = Wage)
reg.model3 <- lm(wage ~ poly(age, 3), data = Wage)
reg.model4 <- lm(wage ~ poly(age, 4), data = Wage)
reg.model5 <- lm(wage ~ poly(age, 5), data = Wage)

anova(reg.model1, reg.model2, reg.model3, reg.model4, reg.model5)
```

```
## Analysis of Variance Table
##
## Model 1: wage ~ age
## Model 2: wage ~ poly(age, 2)
## Model 3: wage ~ poly(age, 3)
## Model 4: wage ~ poly(age, 4)
## Model 5: wage ~ poly(age, 5)
##   Res.Df     RSS Df Sum of Sq        F    Pr(>F)
## 1   2998 5022216
## 2   2997 4793430  1    228786 143.5931 < 2.2e-16 ***
## 3   2996 4777674  1     15756   9.8888  0.001679 **
## 4   2995 4771604  1      6070   3.8098  0.051046 .
## 5   2994 4770322  1      1283   0.8050  0.369682
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

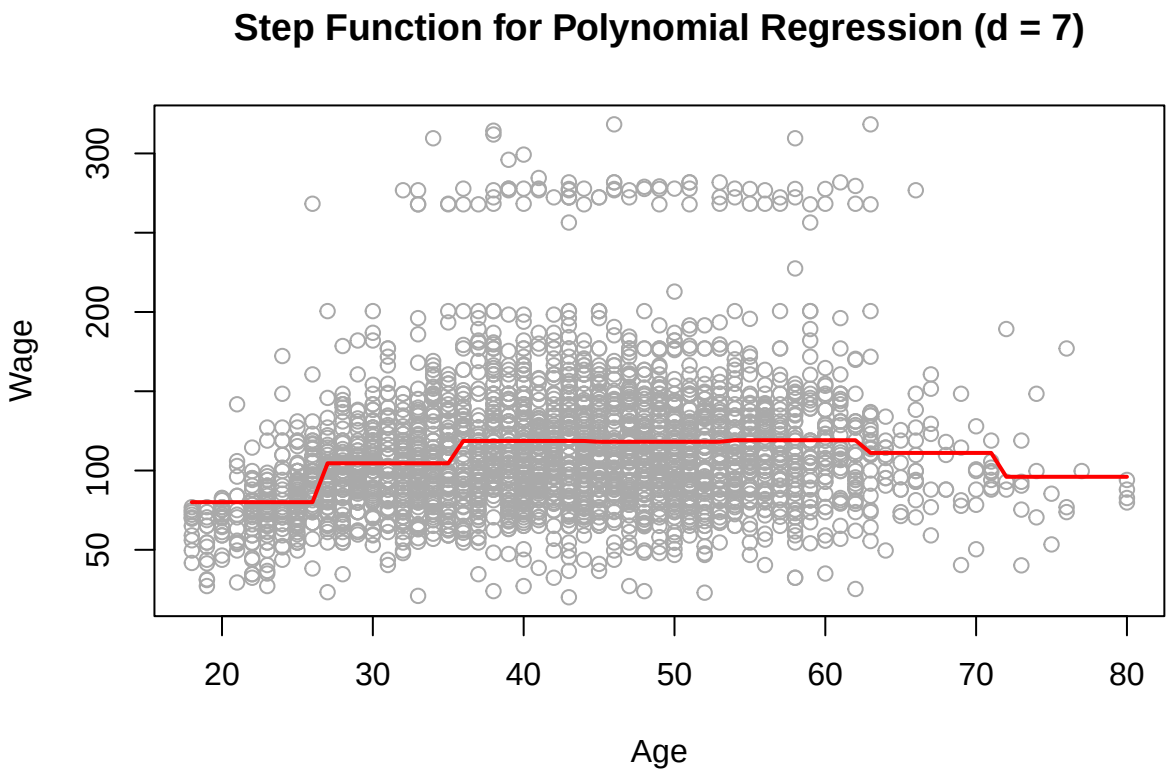
```r
agelims = range(age)
age.grid = seq(from = agelims[1], to = agelims[2])
preds = predict(reg.model3, newdata = list(age = age.grid), se = TRUE )


plot(wage$age, wage$wage,
     xlab = "Age", ylab = "Wage",
     main = "Polynomial Regression (d = 3)", col = "darkgrey")
lines(age.grid, preds$fit, lwd = 2, col = "red")
```



The k-fold cross validation with $k = 10$ indicates that the R-squared of the model doesn't improve much past a cubic fit. The ANOVA verifies this, as the p-value for the power-4 fit is slightly over 0.05. Thus we pick $d = 3$.

(b) Fit a step function to predict `wage` using `age`, and perform cross validation to choose the optimal number of cuts. Make a plot of the fit obtained.

```
cv.error <- rep(0,12)
for (i in 2:13){
  Wage$tmp <- cut(Wage$age, i)
  step.fit = glm(wage ~ tmp, data = Wage)
  cv.error[i] <- cv.glm(Wage, step.fit, K= 10)$delta[1]
}

cut_model = lm(wage ~ cut(age, which.min(cv.error[2:12])), data = Wage)

plot(Wage$age, Wage$wage, xlab = "Age", ylab = "Wage",
     main = "Step Function for Polynomial Regression (d = 7)",
     col = "darkgrey")
lines(sort(Wage$age), cut_model$fitted.values[order(Wage$age)],
        col='red', lwd=2)
```



At 7 cuts, the cross validation error is minimized.

Exercise 9

This question uses the variables `dis` (the weighted mean of distances to five Boston employment centers) and `nox` (nitrogen oxides concentration in parts per 10 million) from the `Boston` data. We will treat `dis` as the predictor and `nox` as the response.

(a) Use the `poly()` function to fit a cubic polynomial regression to predict `nox` using `dis`. Report the regression output, and plot the resulting data and polynomial fits.

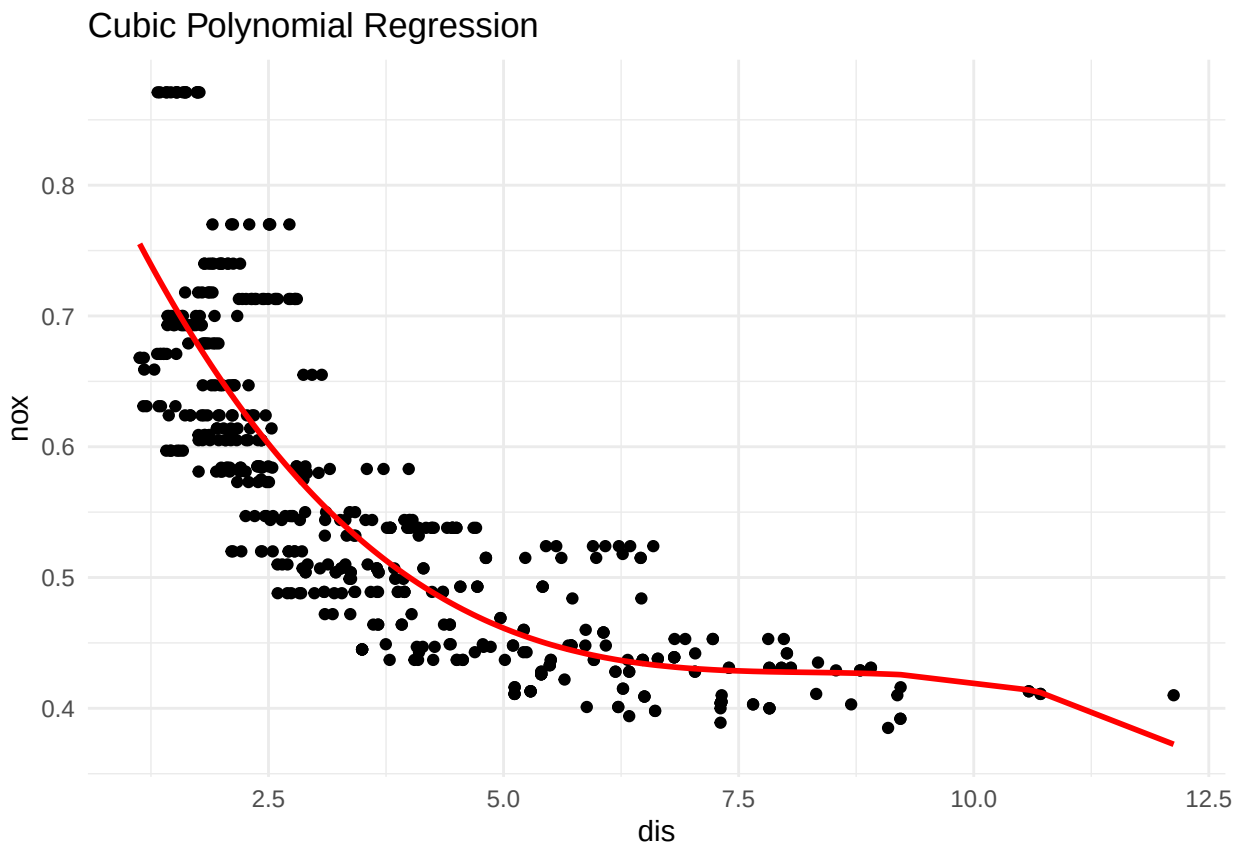
```
cub_mod = lm(nox ~ poly(dis, 3), data = Boston)
summary(cub_mod)
```

```
##
## Call:
## lm(formula = nox ~ poly(dis, 3), data = Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.121130 -0.040619 -0.009738  0.023385  0.194904
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)    0.554695   0.002759 201.021  < 2e-16 ***
## poly(dis, 3)1 -2.003096   0.062071 -32.271  < 2e-16 ***
## poly(dis, 3)2  0.856330   0.062071  13.796  < 2e-16 ***
## poly(dis, 3)3 -0.318049   0.062071  -5.124 4.27e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06207 on 502 degrees of freedom
## Multiple R-squared:  0.7148, Adjusted R-squared:  0.7131
## F-statistic: 419.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

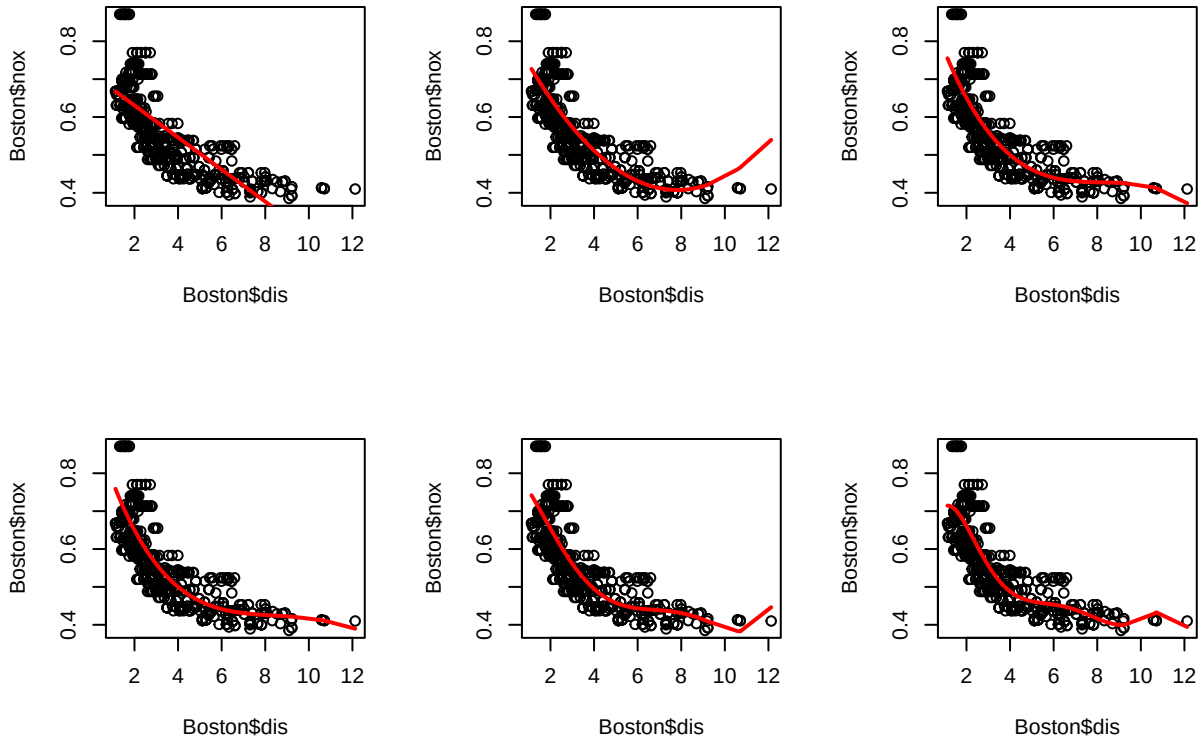
```r
ggplot() +
  geom_point(data = Boston, aes(x = dis, y = nox)) +
  geom_line(aes(x = Boston$dis,
                y = cub_mod$fit),
                color = "red",
                size = 1) +
  labs(title = "Cubic Polynomial Regression") +
  theme_minimal()
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```



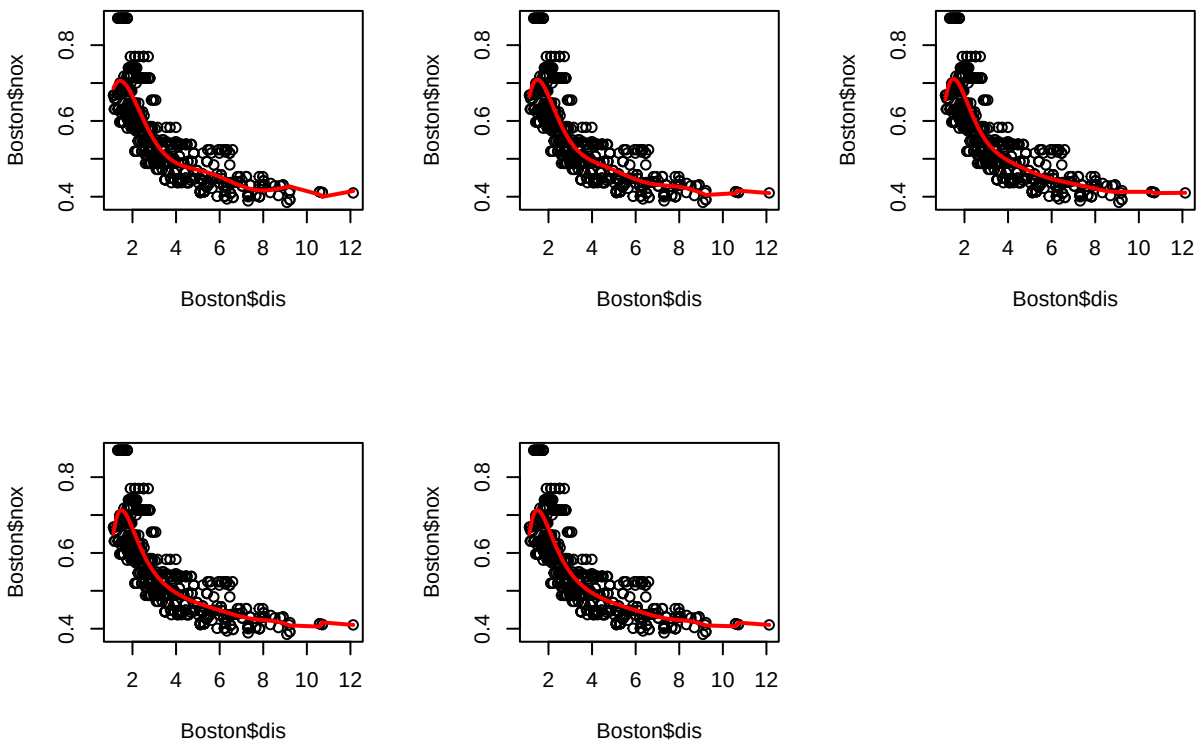
(b) Plot the polynomial fits for a range of different polynomial degrees (say, from 1 to 10), and report the associated residual sum of squares.

```
par(mfrow = c(2,3))
rss_list = rep(0, 10)
plot_list <- list()

for (i in 1:11) {
  model = lm(nox ~ poly(dis, i), data = Boston)
  rss_list[i] = sum(model$residuals ** 2)

  plot(Boston$dis, Boston$nox)
  lines(sort(Boston$dis), model$fitted.values[order(Boston$dis)],
        col='red', lwd=2)
}
```




```
rss_list
```

```
##  [1] 2.768563 2.035262 1.934107 1.932981 1.915290 1.878257 1.849484 1.835630
##  [9] 1.833331 1.832171 1.832168
```



(c) Perform cross-validation or another approach to select the optimal degree for the polynomial, and explain your results.

```
K <- 10
degree <- 10
folds <- cut(seq(1, nrow(Boston)), breaks=K, labels=FALSE)
mse_bin = matrix(data=NA,nrow=K,ncol=degree)

# K-fold cross validation
for (i in 1:K){
    testIndexes <- which(folds==i, arr.ind=TRUE)
    test <- Boston[testIndexes, ]
    train <- Boston[-testIndexes, ]

    for (j in 1:degree){
        fit.train = lm(nox ~ poly(dis, j), data=train)
        fit.test = predict(fit.train, newdata = test)
        mse_bin[i,j] = mean((fit.test - test$nox)^2)
    }
}

colMeans(mse_bin)
```

```
##  [1] 0.005941554 0.004329040 0.004202228 0.004571789 0.005181032 0.006033520
##  [7] 0.016158575 0.024727535 0.135321693 0.232052974
```

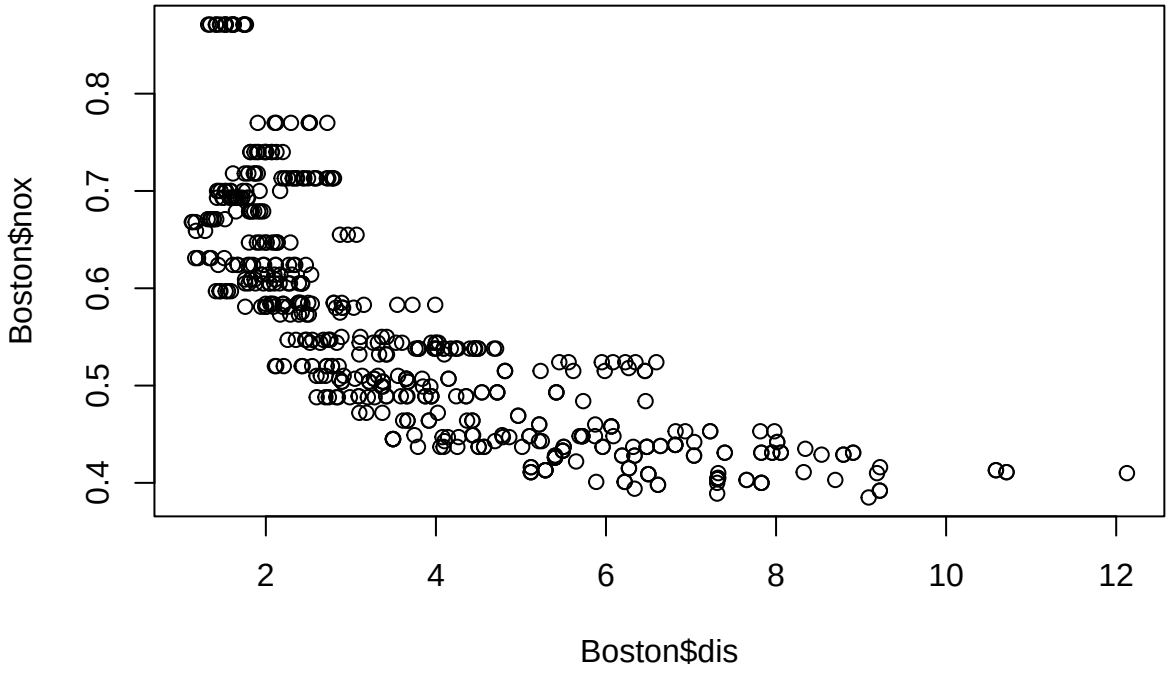
```
which.min(colMeans(mse_bin))
```

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

I performed $k = 10$ fold cross-validation on the `Boston` data, which indicates that a cubic fit results in the lowest MSE when using `dis` to predict `nox`.

(d) Use the `bs()` function to fit a regression spline to predict `nox` using `dis`. Report the output for the fit using four degrees of freedom. How did you choose the knots? Plot the resulting fit.

```
# fitting a cubic spline
plot(Boston$dis, Boston$nox)
```




```
bs_fit = lm(nox ~ bs(dis, knots = c(3, 5, 7)), data = Boston)
summary(bs_fit)
```

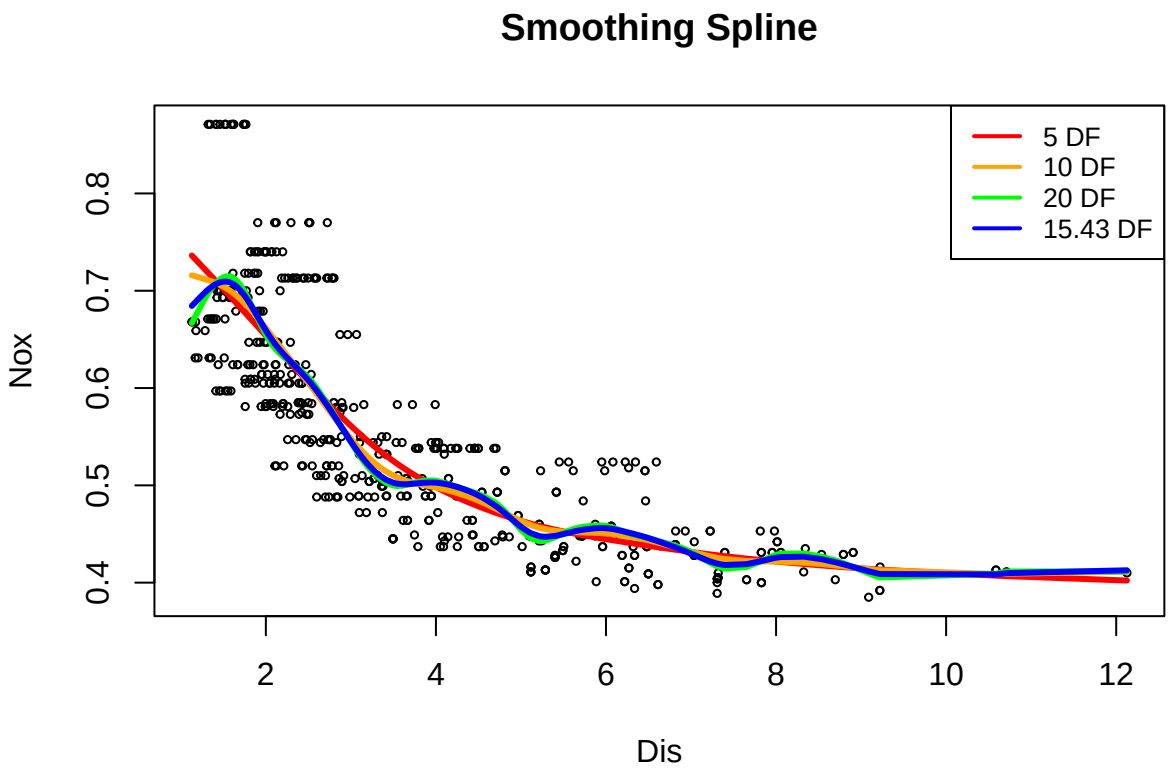
```
##
## Call:
## lm(formula = nox ~ bs(dis, knots = c(3, 5, 7)), data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.13394 -0.03823 -0.00945  0.02613  0.19095
```

```
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   0.70087    0.01668  42.020  < 2e-16 ***
## bs(dis, knots = c(3, 5, 7))1  0.02864    0.02809   1.019    0.308
## bs(dis, knots = c(3, 5, 7))2 -0.21802    0.01807 -12.065  < 2e-16 ***
## bs(dis, knots = c(3, 5, 7))3 -0.21995    0.02471  -8.901  < 2e-16 ***
## bs(dis, knots = c(3, 5, 7))4 -0.31010    0.03045 -10.184  < 2e-16 ***
## bs(dis, knots = c(3, 5, 7))5 -0.27291    0.05526  -4.939 1.07e-06 ***
## bs(dis, knots = c(3, 5, 7))6 -0.29709    0.05755  -5.163 3.52e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.061 on 499 degrees of freedom
## Multiple R-squared:  0.7262, Adjusted R-squared:  0.7229
## F-statistic: 220.5 on 6 and 499 DF,  p-value: < 2.2e-16
```

The knots were chosen based on the three points where the plot of `nox` vs. `dis` appeared to change the most.

(e) Now fit a regression spline for a range of degrees of freedom, and plot the resulting fits and report the resulting RSS. Describe the results obtained.

```r
# performing spline regression with smoothing splines
spline_rss_bin = rep(0, 5)

# generating fits by df
fit1 = smooth.spline(Boston$dis, Boston$nox, df = 5)
fit2 = smooth.spline(Boston$dis, Boston$nox, df = 10)
fit3 = smooth.spline(Boston$dis, Boston$nox, df = 20)
fit4 = smooth.spline(Boston$dis, Boston$nox, cv = TRUE)

# plotting
plot(Boston$dis, Boston$nox, xlim = range(Boston$dis), cex = .5,
     xlab = "Dis", ylab = "Nox", main = "Smoothing Spline")
lines(fit1, col = "red", lwd = 3)
lines(fit2, col = "orange", lwd = 3)
lines(fit3, col = "green", lwd = 3)
lines(fit4, col = "blue", lwd = 3)
legend("topright", legend = c("5 DF", "10 DF", "20 DF", "15.43 DF"),
       col = c("red", "orange", "green", "blue"),
       lty = 1, lwd = 2, cex = 0.8)
```




```
# calculating rss per fit
rss1 = sum((fit1$y - Boston$nox)^2)
rss2 = sum((fit2$y - Boston$nox)^2)
rss3 = sum((fit3$y - Boston$nox)^2)
rss4 = sum((fit4$y - Boston$nox)^2)

print(rss1); print(rss2); print(rss3); print(rss4)
```

```
## [1] 12.17785
```

```
## [1] 12.25768
```

```
## [1] 12.50689
```

```
## [1] 12.40535
```

As expected, lower DF causes the fit to be nearly linear and higher DF causes the fit to interpolate the data. The fits with the lowest DFs appear to sufficiently model the data. This is corroborated by the RSS, which is lowest at the lowest and highest DFs listed.

(f) Perform cross-validation or another approach in order to select the best degrees of freedom for a regression spline on this data. Describe your results.

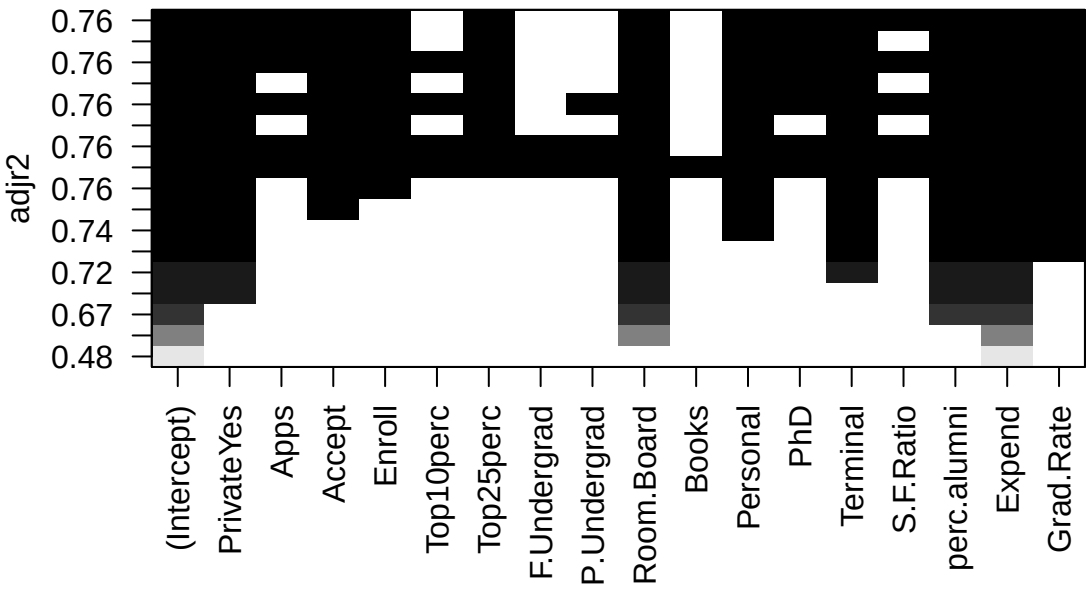
It was already done in part (e). The result of the cross-validation returned a degrees of freedom of about 15.43, which produced a smooth spline which appears to slightly overinterpolate the data.

Exercise 10

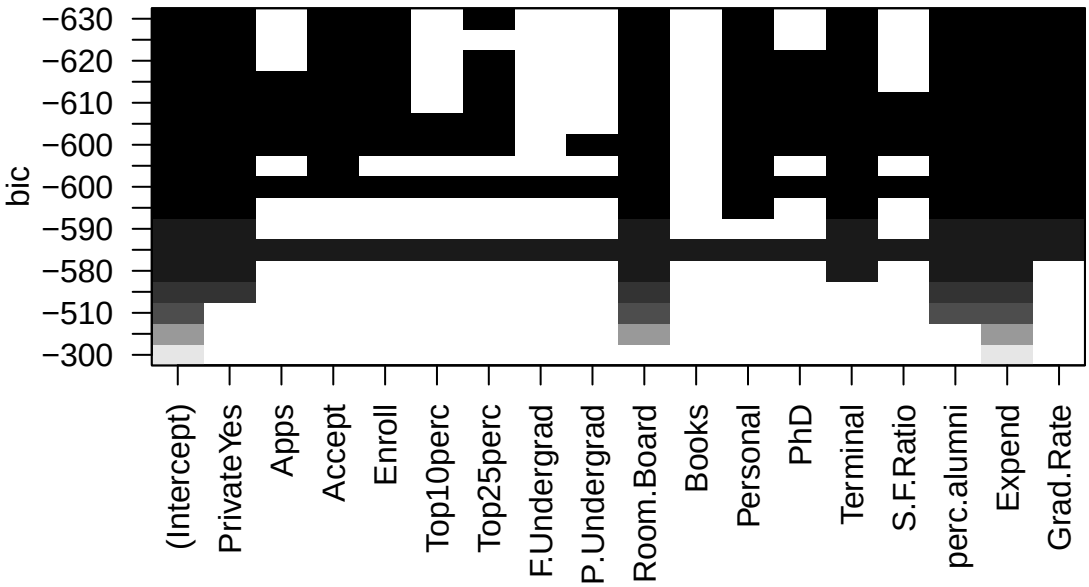
This question relates to the `College` data set.

(a) Split the data into a training set and a test set. Using out-of-state tuition as the response and the other variables as the predictors, perform forward stepwise selection on the training set in order to identify a satisfactory model that uses just a subset of the predictors.

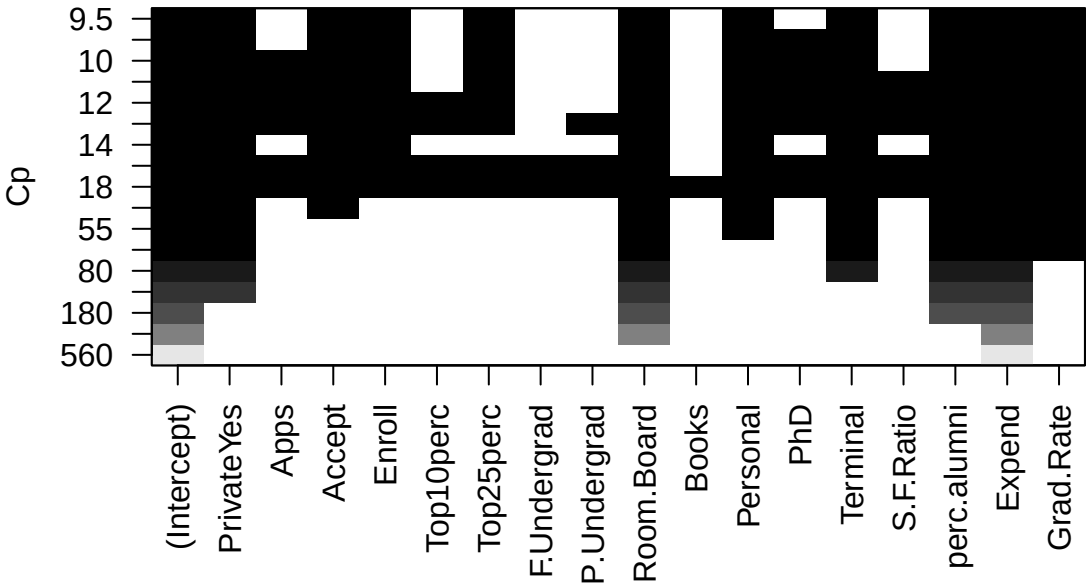
```
college = College

# training and test splits
train_split = 1:477
college_train = college[train_split, ]
college_test = college[-train_split, ]

# performing forward subset selection
regfit.fwd = regsubsets(Outstate ~ ., data = college_train,
                        nvmax = 18, method = "forward")
plot(regfit.fwd, scale = "adjr2")
```




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




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

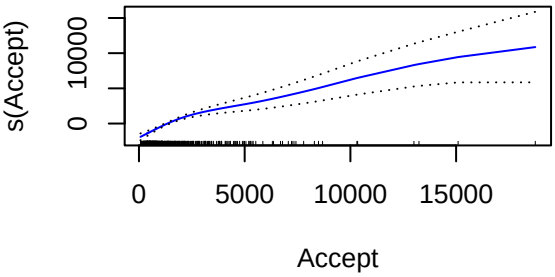


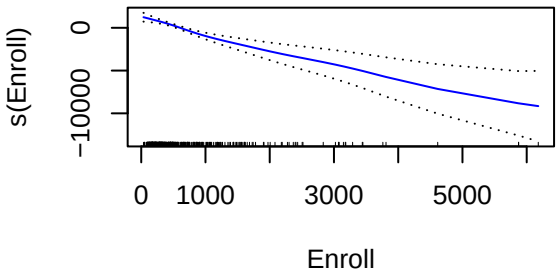
Based on the BIC and Cp criterion, we will fit a model that includes the following: `Private`, `Accept`, `Enroll`, `Top25perc`, `Room.Board`, `Personal`, `Terminal`, `perc.alumni`, `Expend`, and `Grad.Rate`.

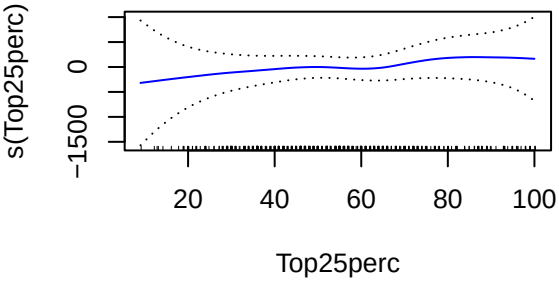
(b) Fit a GAM on the training data, using out-of-state tuition as the response and the features selected in the previous step as the predictors. Plot the results, and explain your findings.

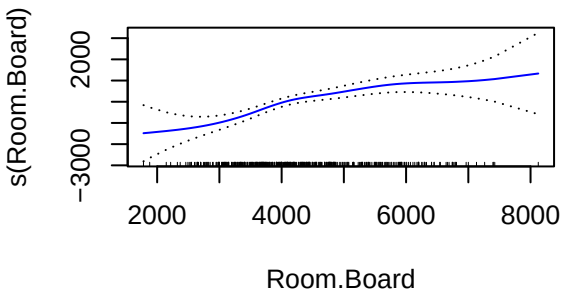
```
par(mfrow = c(2, 2))
gam.mod = gam(Outstate ~ Private + s(Accept) + s(Enroll) + s(Top25perc)
              + s(Room.Board) + s(Personal) + s(Terminal) + s(perc.alumni)
              + s(Expend) + s(Grad.Rate),  data = college_train)
plot(gam.mod, se = TRUE, col = "blue")
```

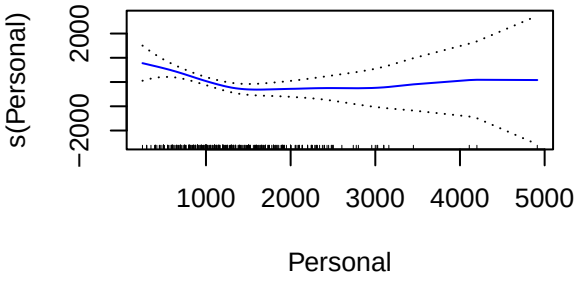

No
Yes
partial for Private
0
−2000
Private

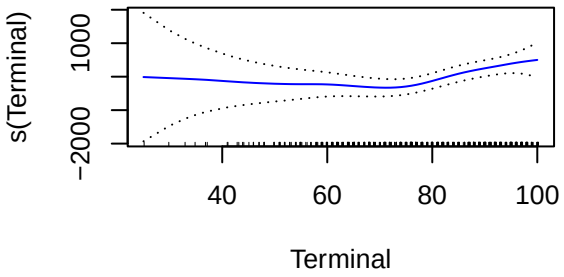

s(Accept)
0
10000
0
5000
10000
15000
Accept

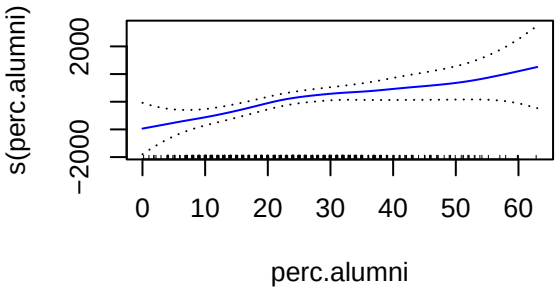

s(Enroll)
0
−10000
0
1000
3000
5000
Enroll

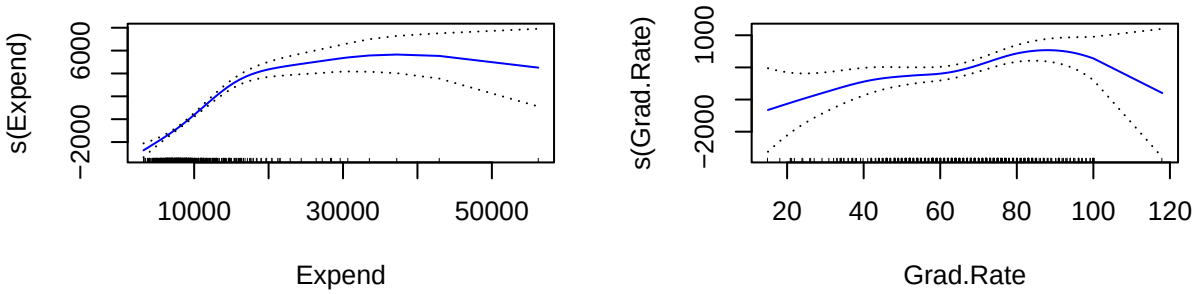

s(Top25perc)
0
−1500
20
40
60
80
100
Top25perc


s(Room.Board)
2000
−3000
2000
4000
6000
8000
Room.Board


s(Personal)
2000
−2000
1000
2000
3000
4000
5000
Personal


s(Terminal)
1000
−2000
40
60
80
100
Terminal


s(perc.alumni)
2000
−2000
0
10
20
30
40
50
60
perc.alumni



We can observe from each of the plots which of the selected predictors may have a nonlinear relationship with out-of-state tuition (mostly just `Expend` and `Grad.Rate`; it is less so for `Terminal` and `Personal`). Additionally, there is a large gap between the bars in the plot for `Private`, which may indicate a significant relationship between `Private` and `Outstate`.

(c) Evaluate the model obtained on the test set, and explain the results obtained.

```
college_preds = predict(gam.mod, newdata = college_test)

# Train and test MSE
mean((gam.mod$fitted.values - college_train$Outstate)^2)
```

```
## [1] 2773788
```

```
mean((college_preds - college_test$Outstate)^2)
```

```
## [1] 4098898
```

Exercise 11

In Section 7.7, it was mentioned that GAMs are generally fit using a backfitting approach. The idea behind backfitting is actually quite simple. We will now explore backfitting in the context of multiple linear regression. Suppose that we would like to perform multiple linear regression, but we do not have software to do so. Instead, we only have software to perform simple linear regression. Therefore, we take the

following iterative approach: we repeatedly hold all but one coefficient estimate fixed at its current value, and update only that coefficient estimate using a simple linear regression. The process is continued until convergence—that is, until the coefficient estimates stop changing. We now try this out on a toy example.

(a) Generate a response `Y` and two predictors X_1 and X_2, with `n = 100`.

```
n <- 100
x1 <- rnorm(n)
x2 <- rnorm(n)
y <- 2*x1 + 3*x2 + rnorm(n, mean = 0, sd = 1)
generated_df <- data.frame(y, x1, x2)
```

(b) Initialize $\hat{\beta}_1$ to take on a value of your choice. It does not matter what value you choose.

```
beta1 = 5
```

(c) Keeping $\hat{\beta}_1$ fixed, fit the model $Y - \hat{\beta}_1 X_1 = \beta_0 + \hat{\beta}_2 X_2 + \epsilon$.

```
a = y - beta1 * x1
beta2 = lm(a ~ x2)$coef[2]
```

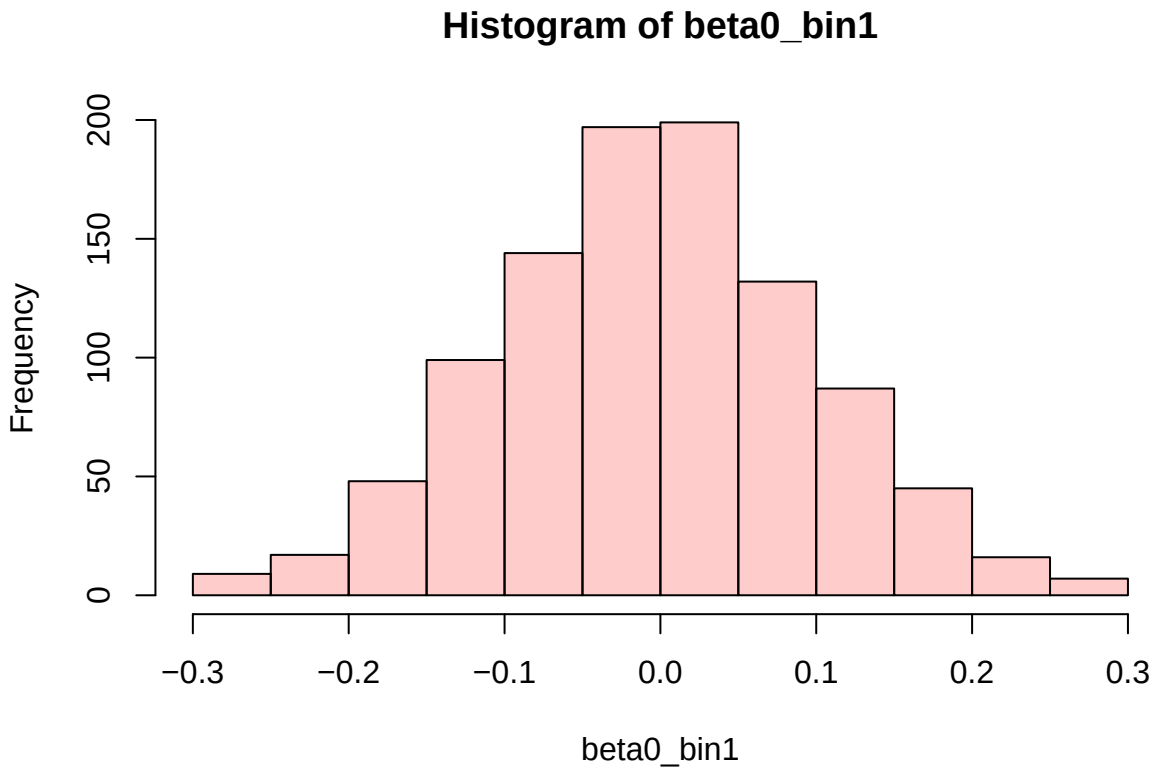
(d) Keeping $\hat{\beta}_2$ fixed, fit the model $Y - \hat{\beta}_2 X_2 = \beta_0 + \hat{\beta}_1 X_1 + \epsilon$.

```
a = y - beta2 * x2
beta1 = lm(a ~ x1)$coef[2]
```

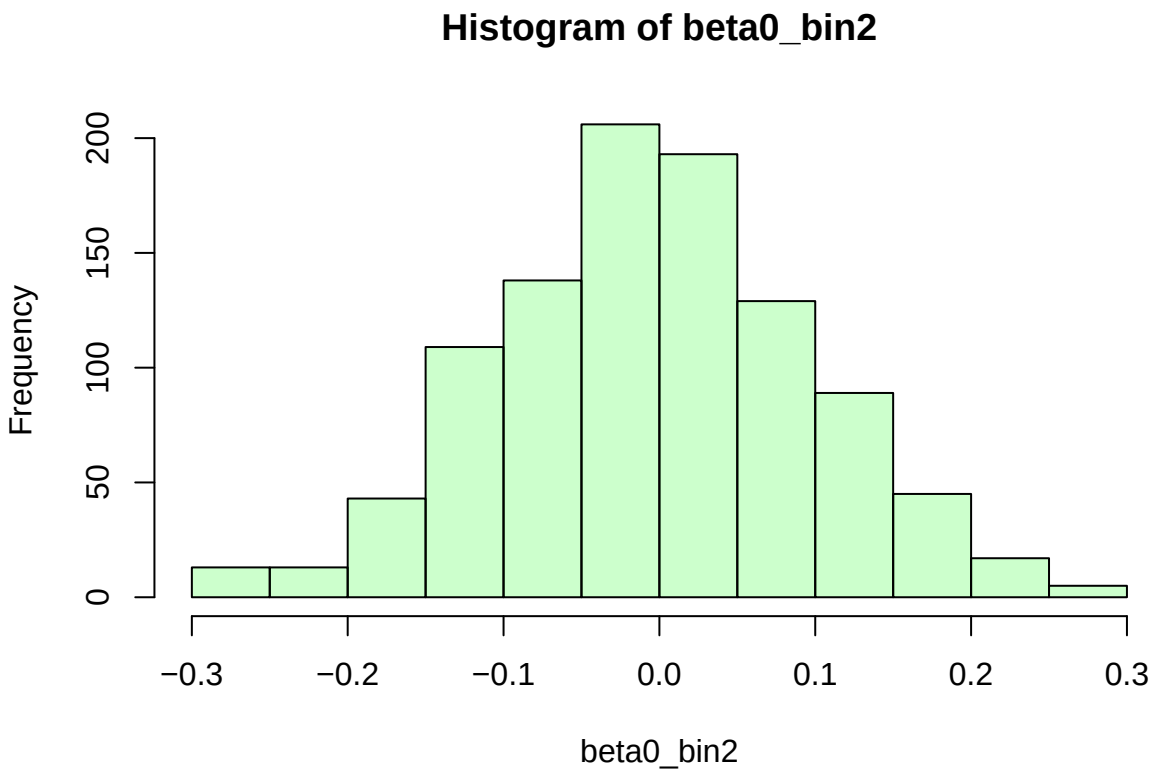
(e) Write a for loop to repeat (c) and (d) 1,000 times. Report the estimates of $\hat{\beta}_0$, $\hat{\beta}_1$, and $\hat{\beta}_2$ at each iteration of the for loop. Create a plot in which each of these values is displayed, with $\hat{\beta}_0$, $\hat{\beta}_1$, and $\hat{\beta}_2$ each shown in a different color.

```
# containers for the values
beta0_bin1 = rep(0, 1000)
beta0_bin2 = rep(0, 1000)
beta1_bin = rep(0, 1000)
beta2_bin = rep(0, 1000)

for (i in 1:1000) {
  n <- 100
  x1 <- rnorm(n)
  x2 <- rnorm(n)
  y <- 2*x1 + 3*x2 + rnorm(n, mean = 0, sd = 1)
  generated_df <- data.frame(y, x1, x2)

  a = y - beta1 * x1
  beta01 = lm(a ~ x2)$coef[1]
  beta2 = lm(a ~ x2)$coef[2]
  beta2_bin[i] = beta2
  beta0_bin1[i] = beta01

  a = y - beta2 * x2
  beta02 = lm(a ~ x1)$coef[1]
```

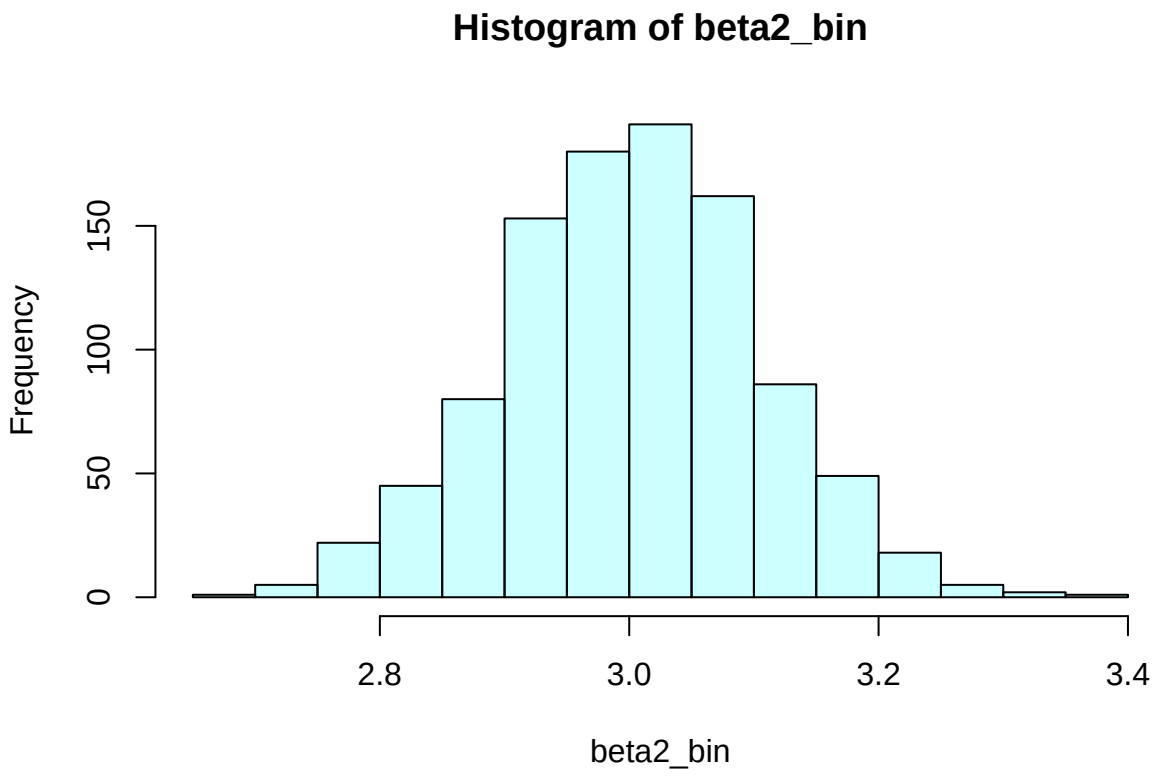
```
  beta1 = lm(a ~ x1)$coef[2]
  beta1_bin[i] = beta1
  beta0_bin2[i] = beta02
}

hist(beta0_bin1, col = rgb(1, 0, 0, 0.2))
```



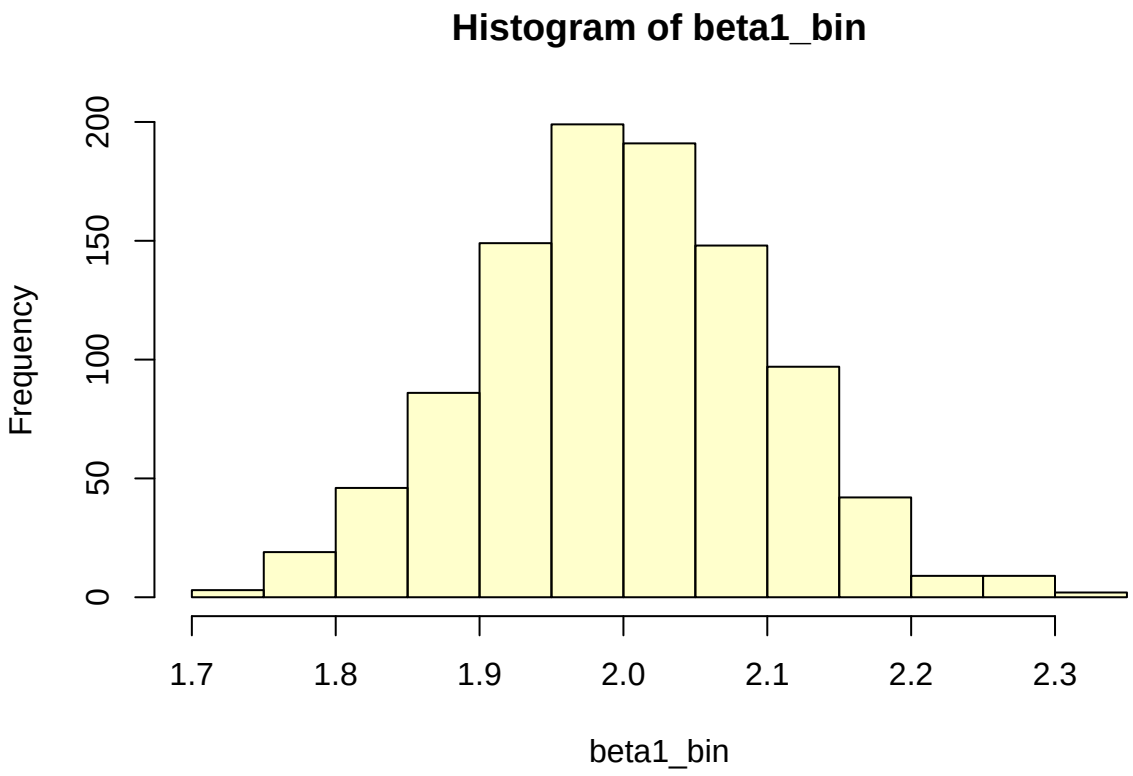

```
hist(beta0_bin2, col = rgb(0, 1, 0, 0.2))
```




```r
hist(beta2_bin, col = rgb(0, 1, 1, 0.2))
```




```
hist(beta1_bin, col = rgb(1, 1, 0, 0.2))
```



(f) Compare your answer in (e) to the results of simply performing multiple linear regression to predict `Y` using X_1 and X_2. Use the `abline()` function to overlay those multiple linear regression coefficient estimates on the plot obtained in (e).

```
mod_f = lm(y ~ ., data = generated_df)
mod_f$coefficients
```

```
## (Intercept)          x1          x2
##  -0.1039766   1.8408353   3.1020719
```

The coefficients of the estimated model correspond to the means of each histogram for each coefficient.

(g) On this data set, how many backfitting iterations were required in order to obtain a "good" approximation to the multiple regression coefficient estimates?

At least 30 iterations were sufficient to show that the coefficients were normally distributed around the mean, which is equivalent to the coefficients of the multiple linear regression model. Performing it 1000 times in part (e) was more than sufficient.