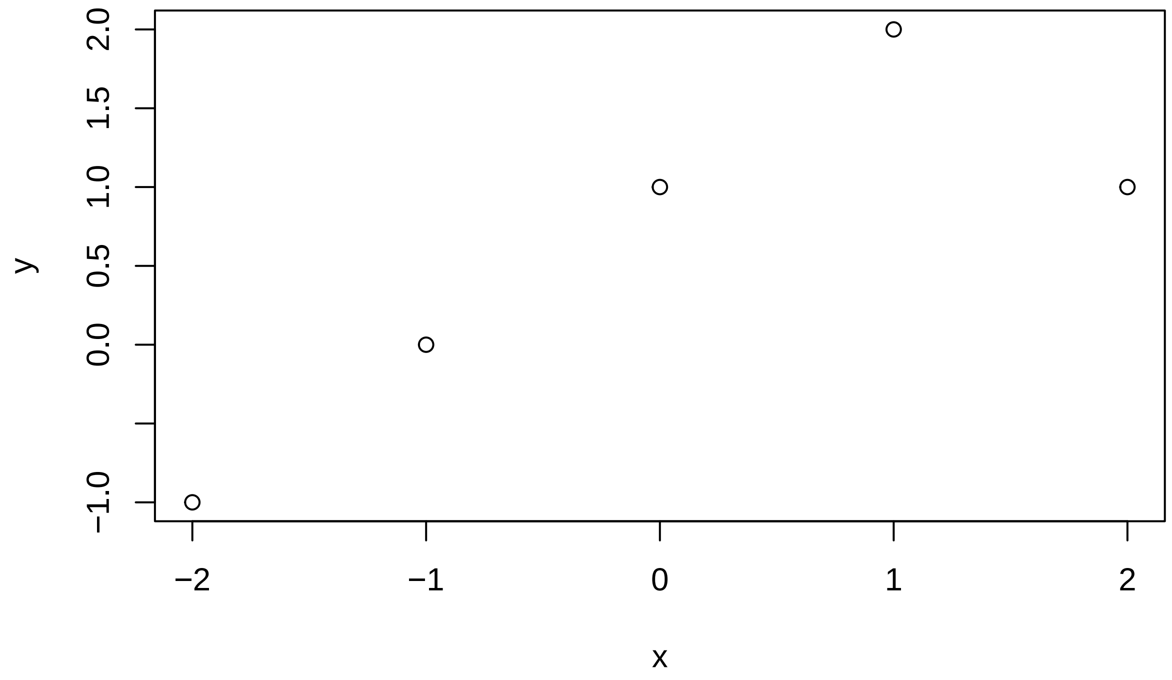
ch7exercises

Lauren Temple

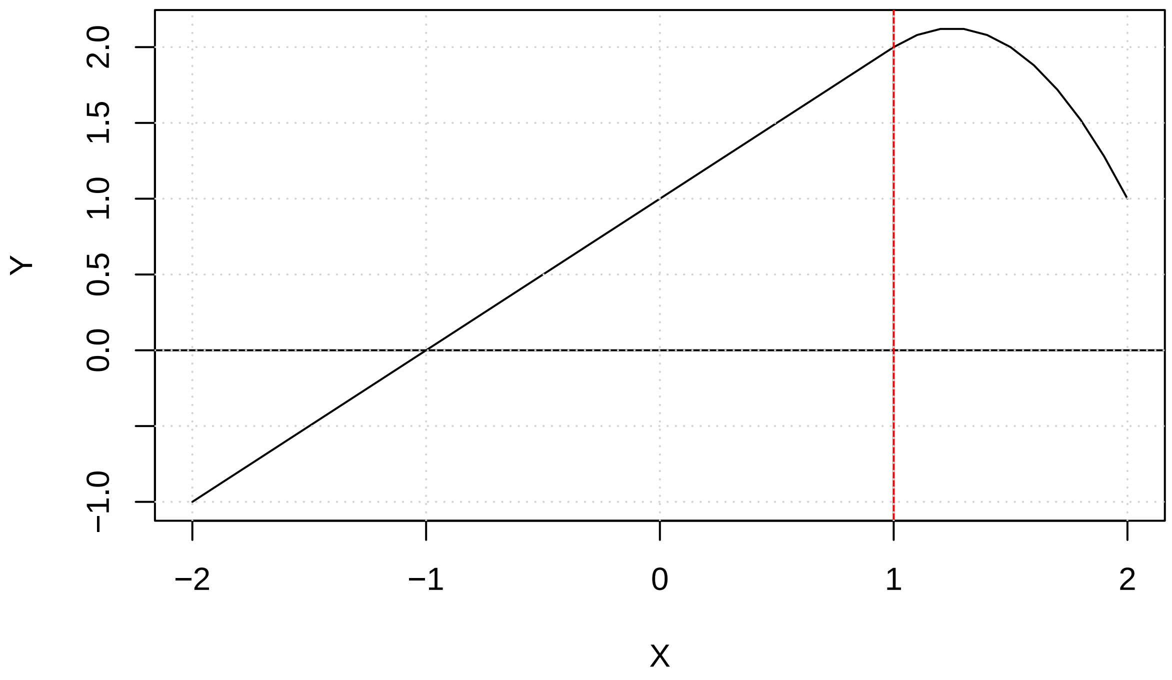
2/15/2022

7.3

```
x <- -2:2
y <- 1 + x + -2 * I(x>1)
plot(x,y)
```



```
X <- seq(-2,2,0.1)
Y <- rep(NA, length(X))

for(i in 1:length(X)){
  if(X[i]<1){
    Y[i] = 1 + 1*X[i]
  }
  else{
    Y[i] = 1 + 1*X[i] - 2 * (X[i]-1)^2
  }
}

plot(X, Y, type= 'l')
abline(h= 0); abline(v= 1, col= "red")
grid()
```



The line has a slope of 1 before X>1 and is linear. After X>1 the line has a quadratic shape and the slope changes.

7.9

```
library(ISLR2)
```

```
## Warning: package 'ISLR2' was built under R version 4.1.2
```

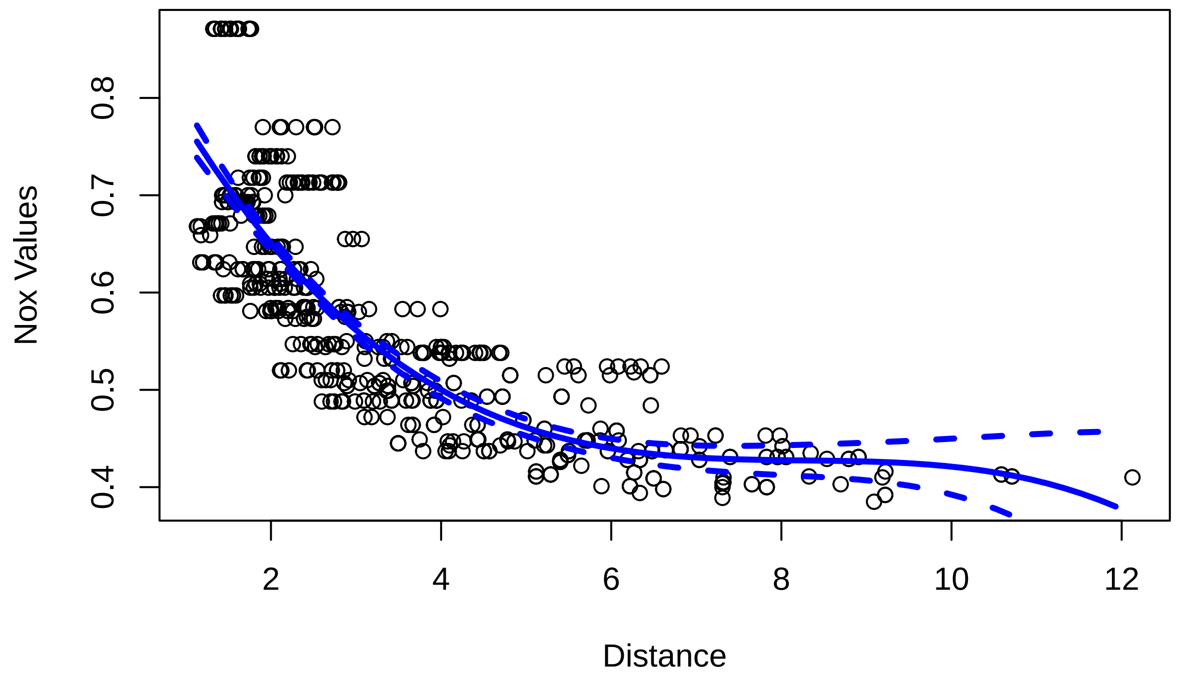
```
data(Boston)
Boston <- Boston
```

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.

```r
plot(Boston$dis, Boston$nox, xlab= "Distance", ylab= "Nox Values")

model1 <- glm(nox~ poly(dis, 3), data= Boston)
summary(model1)
```

```
##
## Call:
## glm(formula = nox ~ poly(dis, 3), data = Boston)
##
## Deviance 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
##
## (Dispersion parameter for gaussian family taken to be 0.003852802)
##
##     Null deviance: 6.7810  on 505  degrees of freedom
## Residual deviance: 1.9341  on 502  degrees of freedom
## AIC: -1370.9
##
## Number of Fisher Scoring iterations: 2
```

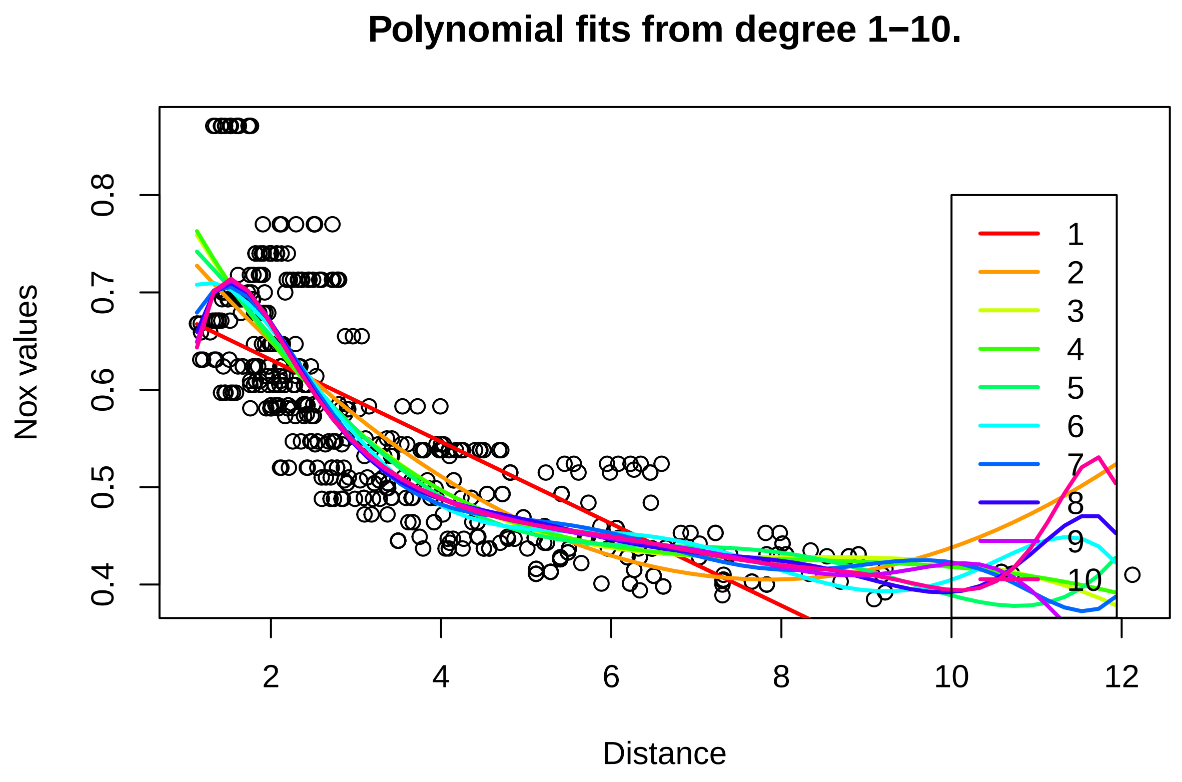
```r
dis.grid <- seq(from= min(Boston$dis), to= max(Boston$dis), 0.2)
preds <- predict(model1, newdata= list(dis= dis.grid), se= T)

lines(dis.grid, preds$fit, col= "blue", lwd= 3)
lines(dis.grid, preds$fit + 2*preds$se, col= "blue", lwd= 3, lty= 2)
lines(dis.grid, preds$fit - 2*preds$se, col="blue", lwd=3, lty=2)
```



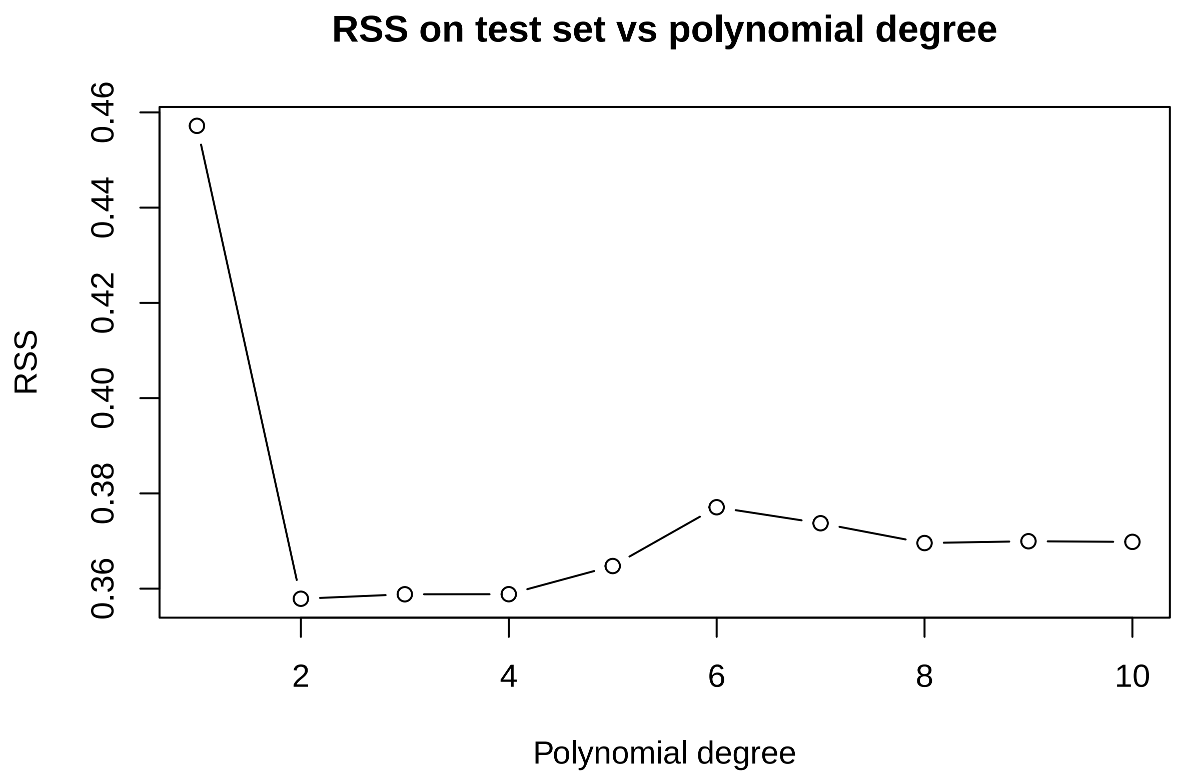
b. Plot the polynomial fits for a range of different polynomial degrees (say, from 1 to 10), and report the associated residual sum of squares.

```
library(caTools)
set.seed(42)
boston_sample <- sample.split(Boston$dis, SplitRatio = 0.80)
boston_train <- subset(Boston, boston_sample == TRUE)
boston_test <- subset(Boston, boston_sample == FALSE)
```

```
rss <- rep(0,10)
colors <- rainbow(10)
plot(Boston$dis, Boston$nox, xlab= "Distance", ylab= "Nox values", main= "Polynomial fits from degree 1
for (i in 1:10){
  model = glm(nox~poly(dis,i), data= boston_train)
  rss[i] = sum((boston_test$nox - predict(model, newdata= list(dis= boston_test$dis)))^2)
  preds = predict(model, newdata= list(dis= dis.grid))
  lines(dis.grid, preds, col= colors[i],  lwd=2, lty=1)
}
legend(10, 0.8, legend= 1:10, col= colors[1:10], lty=1, lwd=2)
```



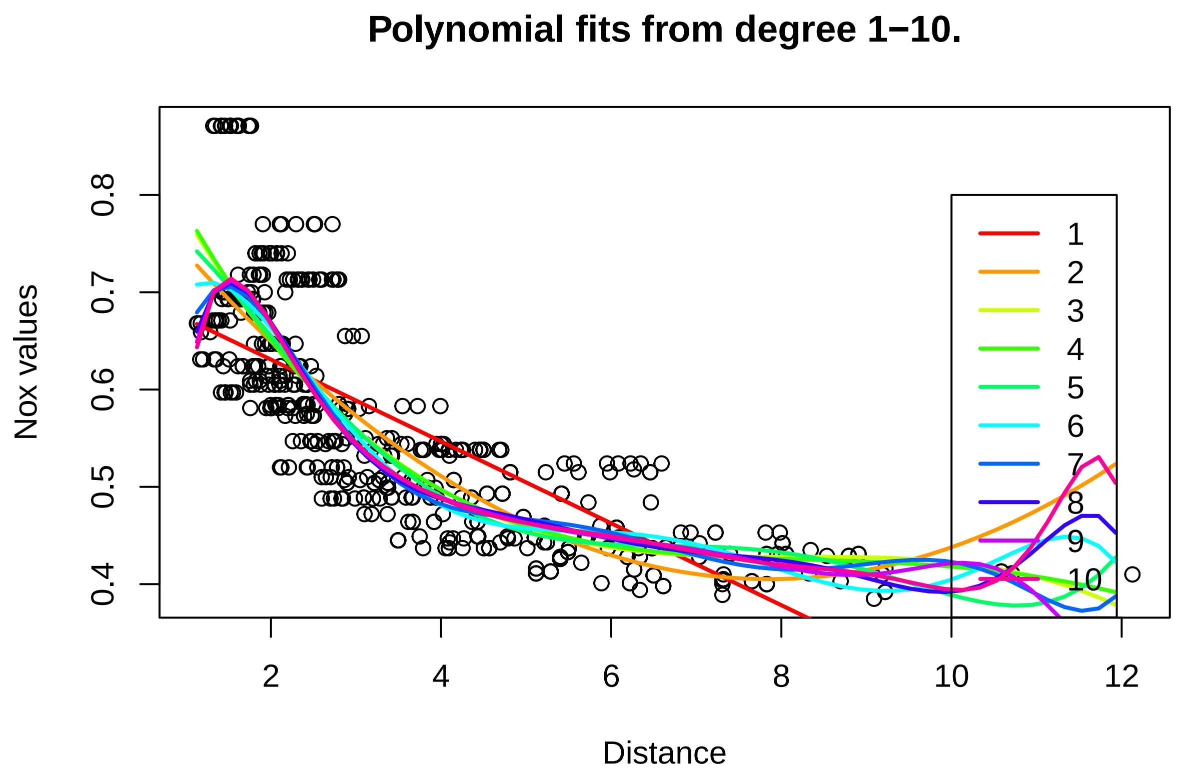
```
plot(1:10,rss,xlab="Polynomial degree", ylab="RSS", main="RSS on test set vs polynomial degree", type='
```



The minimum RSS value occurs at the degree 3 polynomial

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

```r
rss <- rep(0,10)
colors <- rainbow(10)
plot(Boston$dis, Boston$nox, xlab= "Distance", ylab= "Nox values", main= "Polynomial fits from degree 1
for (i in 1:10){
  model = glm(nox~poly(dis,i), data= boston_train)
  rss[i] = sum((boston_test$nox - predict(model, newdata= list(dis= boston_test$dis)))^2)
  preds = predict(model, newdata= list(dis= dis.grid))
  lines(dis.grid, preds, col= colors[i],  lwd=2, lty=1)
}
legend(10, 0.8, legend= 1:10, col= colors[1:10], lty=1, lwd=2)
```



```
rss
```

```
##  [1] 0.4571687 0.3578813 0.3588182 0.3588348 0.3647507 0.3770923 0.3737337
##  [8] 0.3695757 0.3699556 0.3698208
```

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.

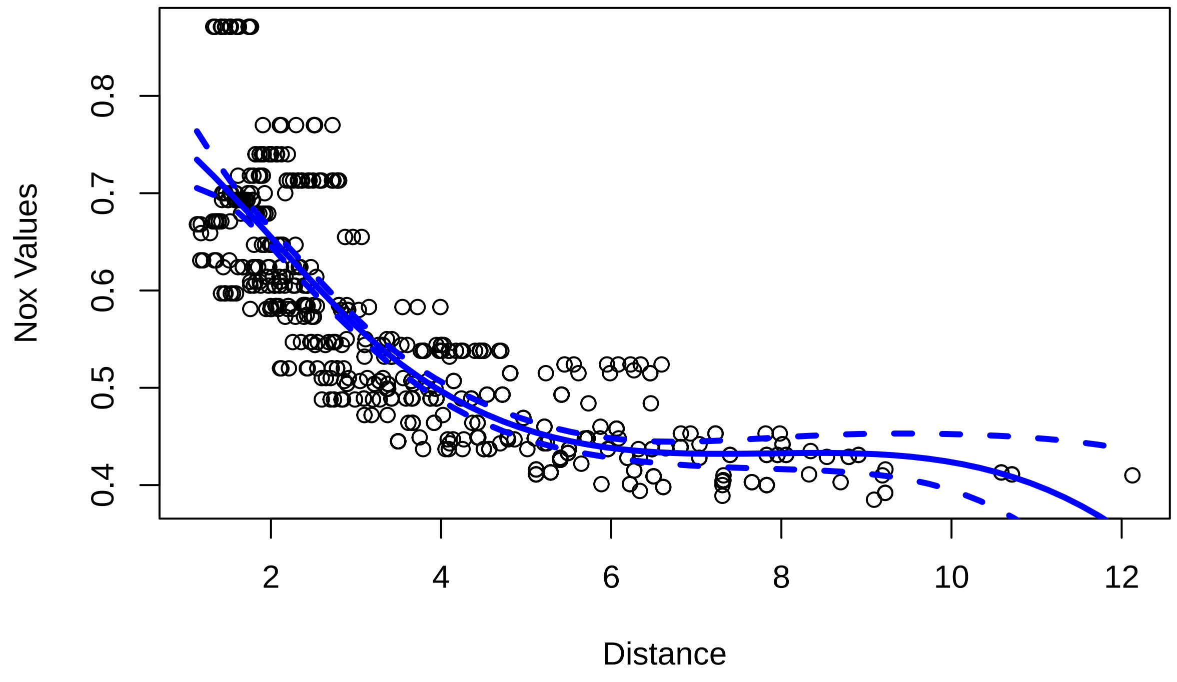
```
library(splines)
spline.fit <- lm(nox ~ bs(dis, df= 4), data= Boston)
summary(spline.fit)
```

```
##
## Call:
## lm(formula = nox ~ bs(dis, df = 4), data = Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.124622 -0.039259 -0.008514  0.020850  0.193891
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       0.73447    0.01460  50.306  < 2e-16 ***
## bs(dis, df = 4)1 -0.05810    0.02186  -2.658  0.00812 **
## bs(dis, df = 4)2 -0.46356    0.02366 -19.596  < 2e-16 ***
## bs(dis, df = 4)3 -0.19979    0.04311  -4.634 4.58e-06 ***
## bs(dis, df = 4)4 -0.38881    0.04551  -8.544  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06195 on 501 degrees of freedom
## Multiple R-squared:  0.7164, Adjusted R-squared:  0.7142
## F-statistic: 316.5 on 4 and 501 DF,  p-value: < 2.2e-16
```

```
attr(bs(Boston$dis, df= 4), "knots")
```

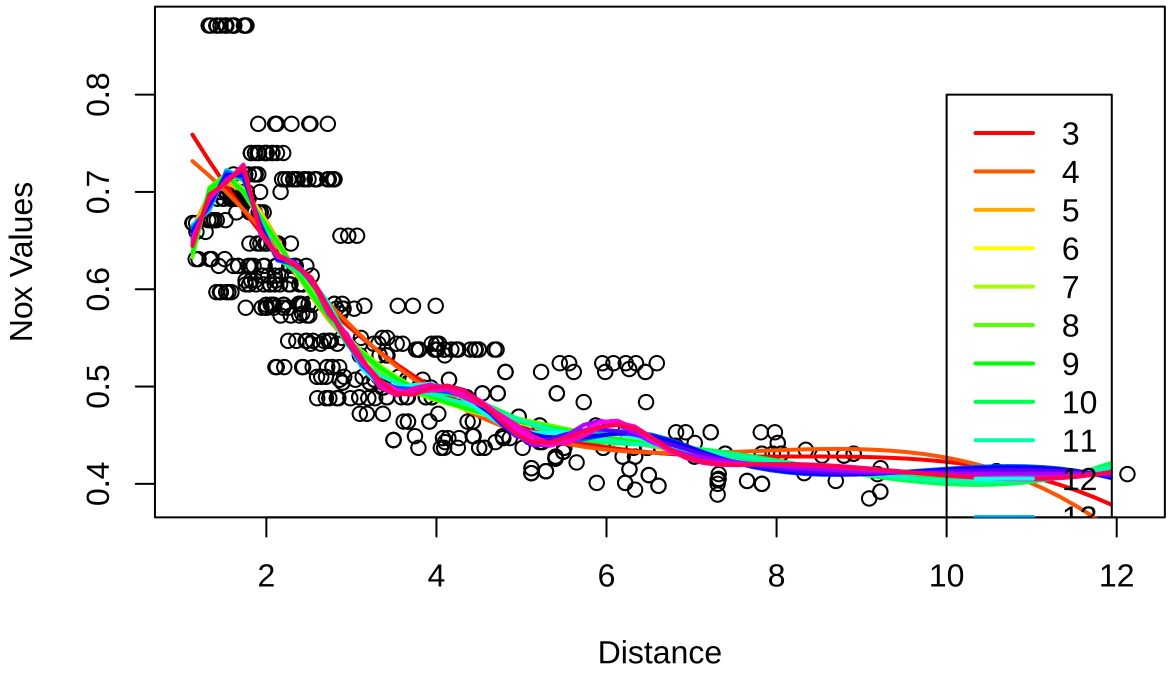
```
##     50%
## 3.20745
```

```
plot(Boston$dis, Boston$nox, xlab= "Distance", ylab= "Nox Values")
preds <- predict(spline.fit, newdata= list(dis= dis.grid), se= T)

lines(dis.grid, preds$fit, col= "blue", lwd= 3)
lines(dis.grid, preds$fit + 2*preds$se, col= "blue", lwd= 3, lty= 2)
lines(dis.grid, preds$fit - 2*preds$se, col="blue", lwd=3, lty=2)
```



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
rss <- rep(0,18)
colors <- rainbow(18)
plot(Boston$dis, Boston$nox, xlab= "Distance", ylab= "Nox Values")

for(i in 3:20){
  spline.model = lm(nox~bs(dis, df=i), data= boston_train)
  rss[i-2] = sum((boston_test$nox - predict(spline.model, newdata= list(dis= boston_test$dis)))^2)
  preds= predict(spline.model, newdata= list(dis= dis.grid))
  lines(dis.grid, preds, col= colors[i-2], lwd=2, lty=1)
}
legend(10, 0.8, legend= 3:20, col= colors[1:18],lty= 1, lwd= 2)
```



```r
which.min(rss)+2
```

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

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.

```r
library(boot)
set.seed(42)
cv.err = rep(0,18)

for(j in 3:20){
    fit= glm(nox~bs(dis, df= j), data= Boston)
    cv.err[j-2] = cv.glm(Boston, fit, K=10)$delta[1]
}
which.min(cv.err)+2
```

```
## [1] 10
```

7.10

```
data("College")
College <- College
```

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.

```
set.seed(42)
college_sample <- sample.split(College$Outstate, SplitRatio = 0.80)
college_train <- subset(College, college_sample == TRUE)
college_test <- subset(College, college_sample == FALSE)
```

```
library(leaps)
fit.fwd <- regsubsets(Outstate ~ ., data= college_train, nvmax= 17, method= "forward")
fit.summary <- summary(fit.fwd)
```

```
which.min(fit.summary$cp)
```

```
## [1] 13
```

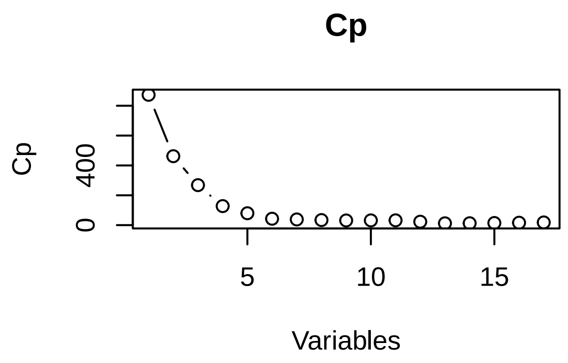
```
which.min(fit.summary$bic)
```

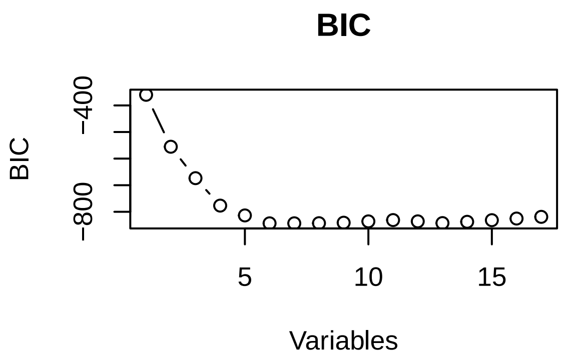
```
## [1] 6
```

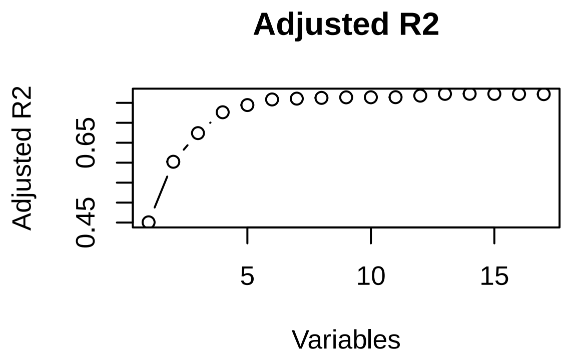
```
which.min(fit.summary$adjr2)
```

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

```
par(mfrow= c(2,2))
plot(1:17, fit.summary$cp,xlab="Variables",ylab="Cp",main="Cp", type='b')
plot(1:17, fit.summary$bic,xlab="Variables",ylab="BIC",main="BIC", type='b')
plot(1:17, fit.summary$adjr2,xlab="Variables",ylab="Adjusted R2",main="Adjusted R2", type='b')
```







All three show that after a model with 6 variables there is little improvement

```
coef(fit.fwd, 6)
```

```
##   (Intercept)    PrivateYes    Room.Board      Terminal   perc.alumni 
## -4505.5888871  2812.1267534     1.0024386    41.0213109    44.2822780 
##        Expend     Grad.Rate 
##     0.2020913    35.6639247
```

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.

```
library(akima)
```

```
## Warning: package 'akima' was built under R version 4.1.2
```

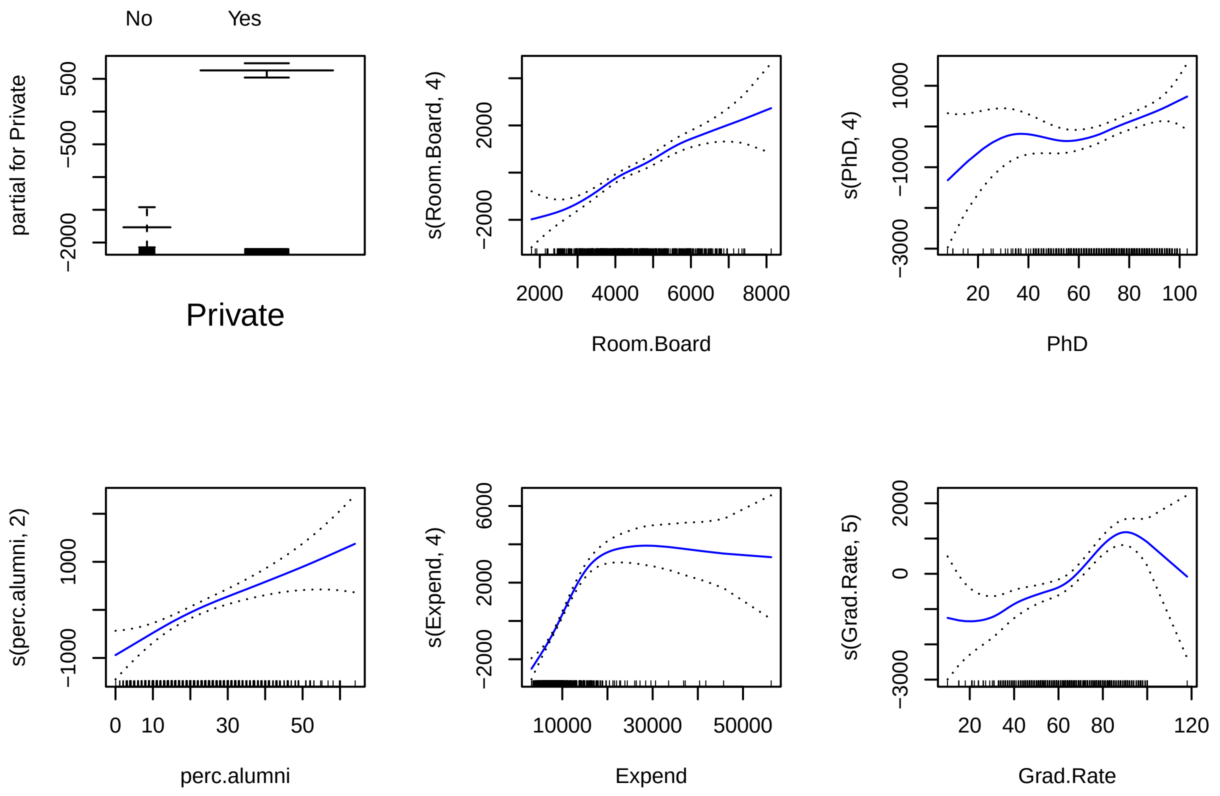
```
library(gam)
```

```
## Warning: package 'gam' was built under R version 4.1.2
```

```
## Loading required package: foreach
```

```
## Loaded gam 1.20
```

```
gam.model1 <- gam(Outstate ~ Private +
                    s(Room.Board, 4)+
                    s(PhD, 4)+
                    s(perc.alumni, 2)+
                    s(Expend, 4)+
                    s(Grad.Rate, 5), data= college_train)

par(mfrow= c(2,3))
plot(gam.model1, col= "blue", se= T)
```



Holding all the other variables fixed, out of state tuition increases as room and board and perc.alumni increases.

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

```
pred <- predict(gam.model1, newdata= college_test)
mse <- mean((college_test$Outstate - pred)^2); mse
```

```
## [1] 4219784
```

d. For which variables, if any, is there evidence of a non-linear relationship with the response?

```
gam.model2 <- gam(Outstate~ Private+
                    s(Room.Board,4)+
                    s(PhD,4)+
                    s(perc.alumni,2)+
                    s(Expend,4), data= college_train)
gam.model3 <- gam(Outstate~ Private+
                    s(Room.Board,4)+
                    s(PhD,4)+
                    s(perc.alumni,2)+
                    s(Expend,4)+
                    Grad.Rate, data=college_train)
gam.model4 <- gam(Outstate~ Private+
                    s(Room.Board,4)+
                    s(PhD,4)+
                    s(perc.alumni,2)+
                    s(Expend,4)+
                    s(Grad.Rate,4), data= college_train)

anova(gam.model2, gam.model3, gam.model4, gam.model1, test= "F")
```

```
## Analysis of Deviance Table
##
## Model 1: Outstate ~ Private + s(Room.Board, 4) + s(PhD, 4) + s(perc.alumni,
##     2) + s(Expend, 4)
## Model 2: Outstate ~ Private + s(Room.Board, 4) + s(PhD, 4) + s(perc.alumni,
##     2) + s(Expend, 4) + Grad.Rate
## Model 3: Outstate ~ Private + s(Room.Board, 4) + s(PhD, 4) + s(perc.alumni,
##     2) + s(Expend, 4) + s(Grad.Rate, 4)
## Model 4: Outstate ~ Private + s(Room.Board, 4) + s(PhD, 4) + s(perc.alumni,
##     2) + s(Expend, 4) + s(Grad.Rate, 5)
##   Resid. Df Resid. Dev      Df  Deviance       F   Pr(>F)
## 1       605 2183327757
## 2       604 2026037645 1.00000 157290112 47.5850 1.34e-11 ***
## 3       601 1992222360 2.99986  33815284  3.4102  0.01731 *
## 4       600 1983272677 0.99988   8949684  2.7079  0.10038
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

A GAM that includes Grad.Rate shows evidence of a non-linear function

7.11 a. Generate a response Y and two predictors X1 and X2, with n = 100.

```
x1 <- rnorm(100, sd=2)
x2 <- rnorm(100, sd= sqrt(2))
eps <- rnorm(100, sd= 1)
b0 <- 5
b1 <- 2.5
b2 <- 11.5
y <- b0 + b1*x1 + b2*x2 +eps
```

b. Initialize ˆ1 to take on a value of your choice. It does not matter what value you choose.

```
beta1 <- 0.4
```

c. Keeping ˆ1 fixed, fit the model Y − ˆ1X1 = 0 + 2X2 + .

```r
z <- y - beta1 *x1
beta2 <- lm(z~ x2)$coef[2]
beta2
```

```
##       x2
## 11.35575
```

d. Keeping ˆ2 fixed, fit the model Y − ˆ2X2 = 0 + 1X1 + .

```r
z <- y - beta2*x2
beta1 <- lm(z ~ x1)$coef[2]
beta1
```

```
##       x1
## 2.508157
```

e. Write a for loop to repeat (c) and (d) 1,000 times. Report the estimates of ˆ0, ˆ1, and ˆ2 at each iteration of the for loop. Create a plot in which each of these values is displayed, with ˆ0, ˆ1, and ˆ2 each shown in a different color.

```r
beta.df <- data.frame("beat0"= rep(0, 1000), "beta1"= rep(0,1000), "beta2"= rep(0,1000))
beta1 <- 0.4

for(i in 1:1000){
  z= y - beta1*x1
  model= lm(z~ x2)
  beta2= model$coef[2]
  beta.df$beta1[i]= beta2

  z= y- beta2*x2
  model = lm(z ~ x1)
  beta1 = model$coef[2]
  beta.df$beta1[i]= beta1

  beta.df$beta0[i]= model$coef[1]
}
```

```r
beta.df$beta0[5]
```

```
## [1] 5.024999
```

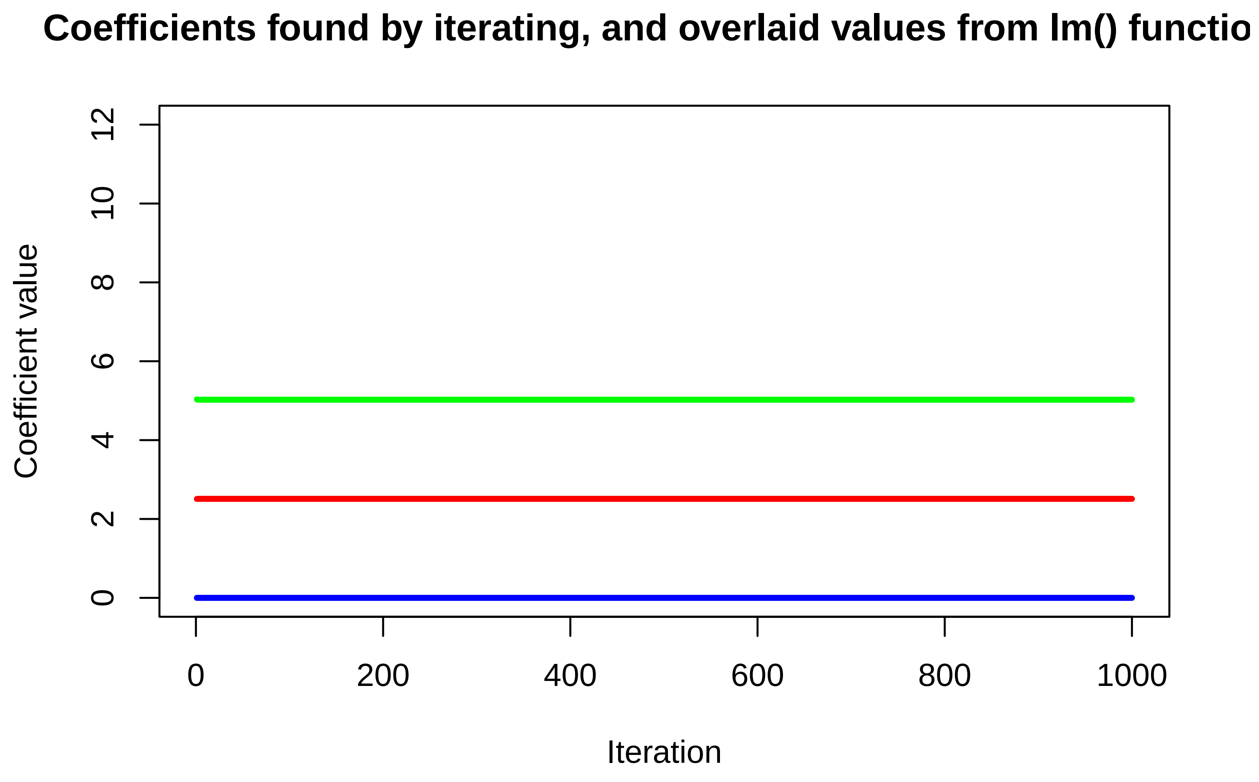
```r
beta.df$beta1[5]
```

```
## [1] 2.510918
```

```r
beta.df$beta2[5]
```

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

```
plot(1:1000, beta.df$beta2, ylim=range(0:12), type='l', lwd="3", col= "blue", xlab= "Iteration", ylab=
title("Coefficients found by iterating, and overlaid values from lm() function.")
lines(1:1000, beta.df$beta1,  col= "red", lwd=3)
lines(1:1000, beta.df$beta0, col= "green", lwd=3)
```

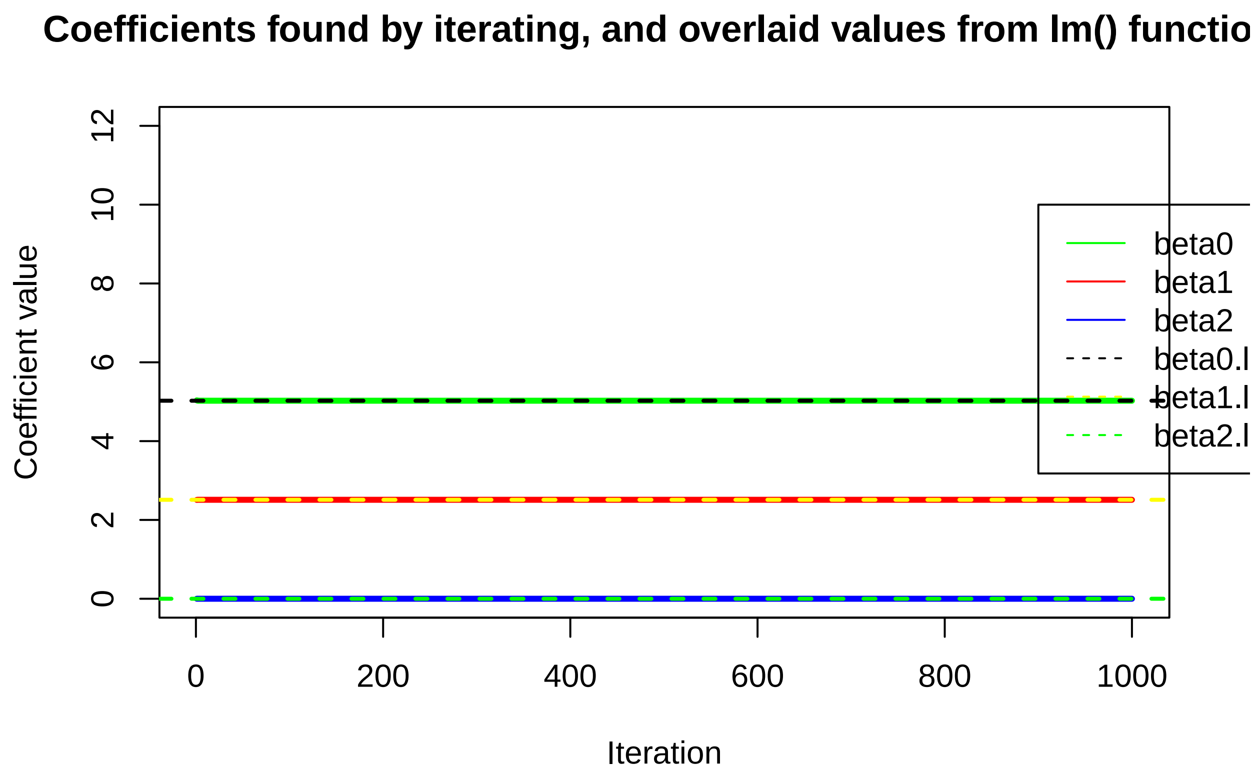


f. Compare your answer in (e) to the results of simply performing multiple linear regression to predict Y using X1 and X2. Use the abline() function to overlay those multiple linear regression coefficient estimates on the plot obtained in (e).

```
lm.fit = lm(y~ x1+x2)
coef(lm.fit)
```

```
## (Intercept)          x1          x2
##    5.024999    2.510918   11.463610
```

```
plot(1:1000, beta.df$beta2, ylim=range(0:12), type='l', lwd="3", col= "blue", xlab= "Iteration", ylab=
title("Coefficients found by iterating, and overlaid values from lm() function.")
lines(1:1000, beta.df$beta1,  col= "red", lwd=3)
lines(1:1000, beta.df$beta0, col= "green", lwd=3)

abline(h= 5.024999, lty= 2, lwd=2)
abline(h= 2.510918, lty= 2, lwd=2, col="yellow")
abline(h=0, lty=2, lwd=2, col="green")
legend(900,10, legend=c("beta0", "beta1", "beta2", "beta0.lm", "beta1.lm", "beta2.lm"),
       col=c("green","red","blue","black","yellow","green"), lty = c(1,1,1,2,2,2), xpd=T)
```



The coefficients found by multiple linear regression match the ones found by iteration.

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