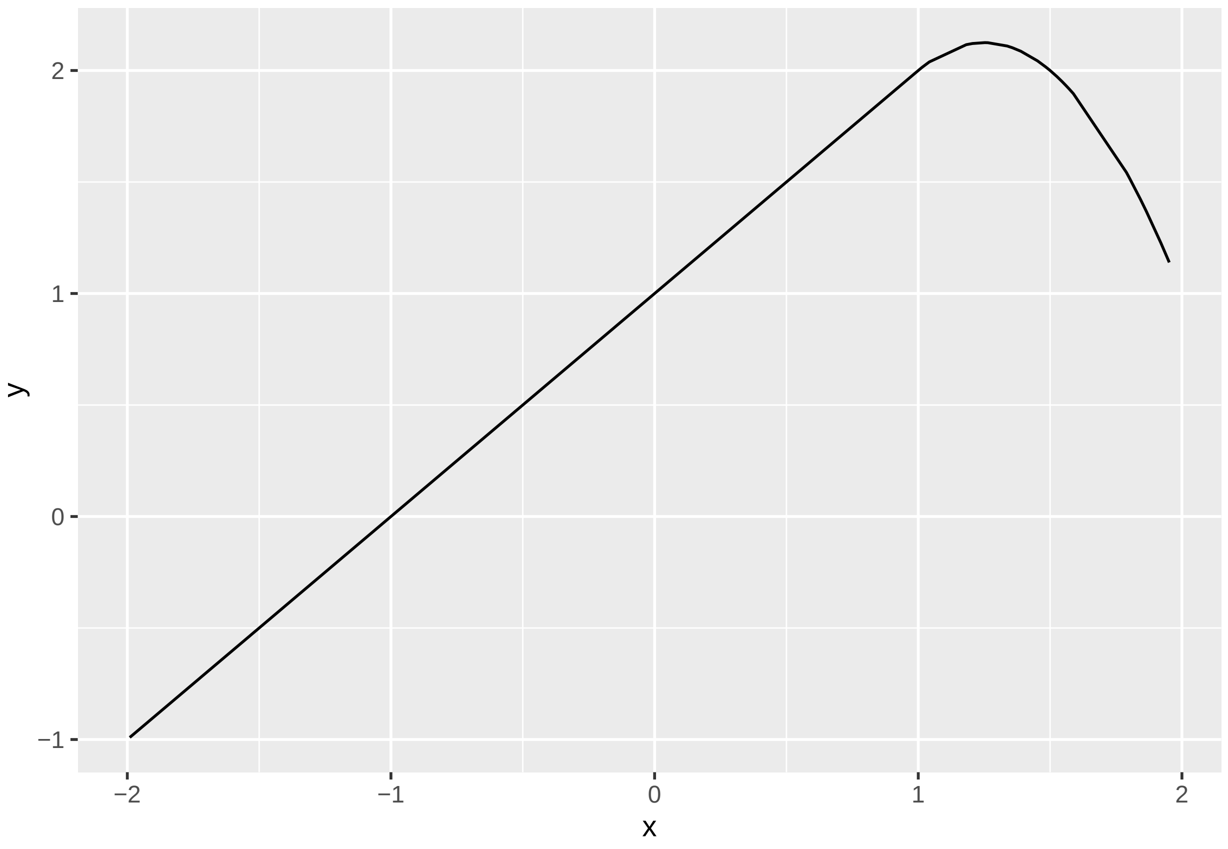
HW4_GAM

Shuting

2/25/2022

7.3

```
x <- runif(100,-2,2)
y <- 1+x-2*(x-1)^2*I(x>=1)
ggplot(aes(x=x,y=y),data = data.frame(x,y))+
  geom_line()
```



Answer:

7.9

```
library(MASS)
attach(Boston)
```

(a)

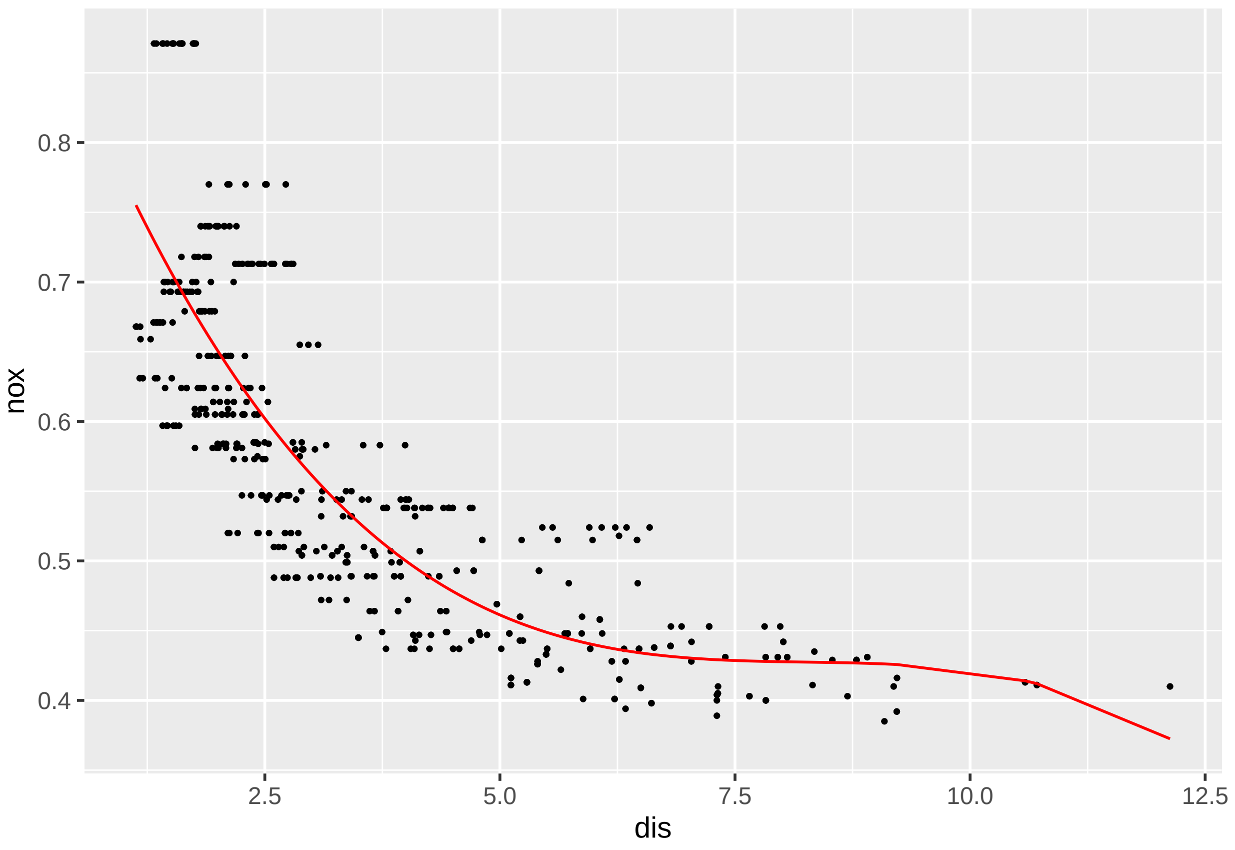
Use the poly() function to fit a cubic polynomial regression to predict nox using dis. Report the regression output, and plot the resulting data and polynomial fits.

```
fit7.9.1 <- glm(nox~poly(dis,3),data = Boston)
summary(fit7.9.1)
```

Answer:

```
##
## 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
```

```
ggplot(data = Boston)+
  geom_point(aes(x=dis,y=nox),lwd=0.5)+
  geom_line(aes(x=dis,y=fitted(fit7.9.1)),col="red")
```



(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.

```
rss <- NULL
for (i in 1:10){
  fit <- glm(nox~poly(dis,i),data = Boston)
  rss[i] <- sum((nox-fitted(fit))^2)
}
rss
```

Answer:

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

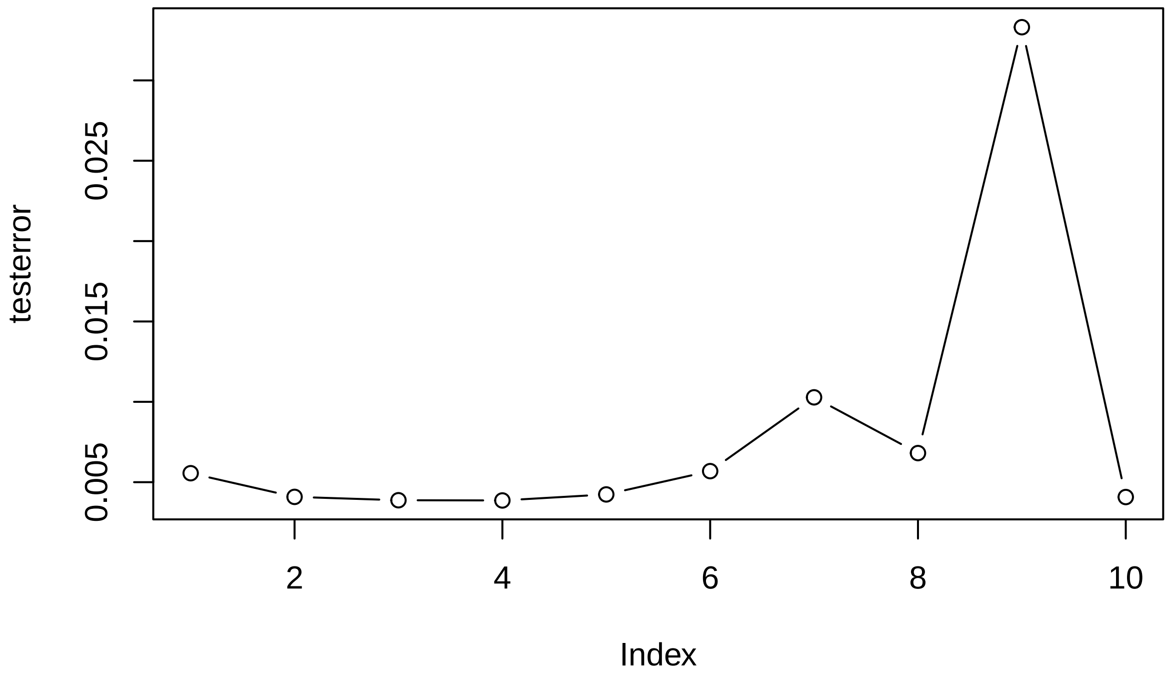
rss decreases when polynomial degrees increase.

(c)

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

```
library(boot)
testerror <- NULL
```

```r
set.seed(1)
for (i in 1:10){
   fit <- glm(nox~poly(dis,i),data = Boston)
   testerror[i] <- cv.glm(fit,K=10,data = Boston)$delta[1]
}
plot(testerror,type = "b")
```



Answer:
The best degree for the polynomial is 4 which has lowest test error.

(d)

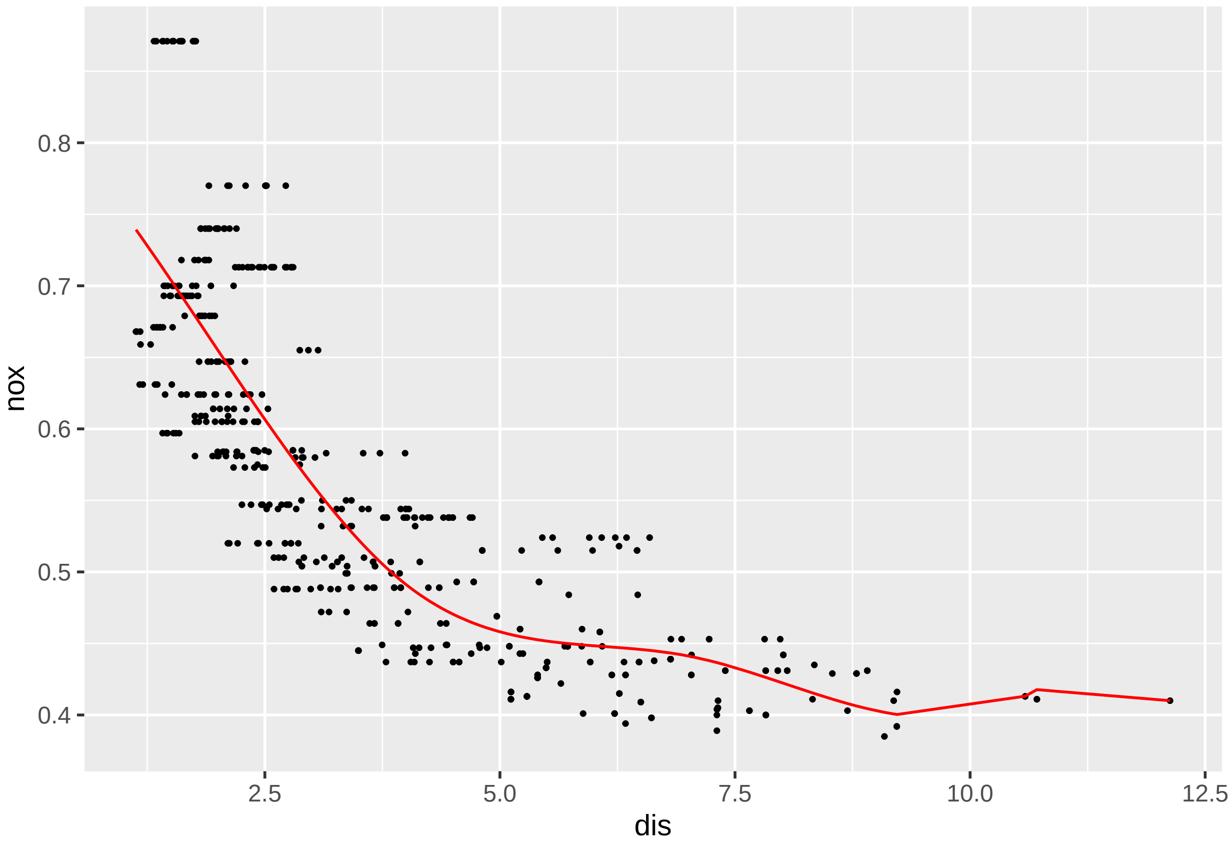
Use the bs() function to fit a regression spline to predict nox using dis. Report the output for the fit using four degrees of freedom. How did you choose the knots? Plot the resulting fit.

```r
library(splines)
fit.spine = glm(nox ~ bs(dis, df = 4, knots = c(4, 7, 11)), data = Boston)
summary(fit.spine)
```

Answer:

```
##
## Call:
## glm(formula = nox ~ bs(dis, df = 4, knots = c(4, 7, 11)), data = Boston)
##
## Deviance Residuals:
##       Min          1Q     Median          3Q         Max
## -0.124567  -0.040355  -0.008702   0.024740   0.192920
##
## Coefficients:
##                                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)                              0.73926    0.01331  55.537  < 2e-16 ***
## bs(dis, df = 4, knots = c(4, 7, 11))1 -0.08861    0.02504  -3.539  0.00044 ***
## bs(dis, df = 4, knots = c(4, 7, 11))2 -0.31341    0.01680 -18.658  < 2e-16 ***
## bs(dis, df = 4, knots = c(4, 7, 11))3 -0.26618    0.03147  -8.459 3.00e-16 ***
## bs(dis, df = 4, knots = c(4, 7, 11))4 -0.39802    0.04647  -8.565  < 2e-16 ***
## bs(dis, df = 4, knots = c(4, 7, 11))5 -0.25681    0.09001  -2.853  0.00451 **
## bs(dis, df = 4, knots = c(4, 7, 11))6 -0.32926    0.06327  -5.204 2.85e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.003825571)
##
##     Null deviance: 6.781  on 505  degrees of freedom
## Residual deviance: 1.909  on 499  degrees of freedom
## AIC: -1371.5
##
## Number of Fisher Scoring iterations: 2
```

```
ggplot(data = Boston)+
  geom_point(aes(x=dis,y=nox),lwd=0.5)+
  geom_line(aes(x=dis,y=fitted(fit.spine)),col="red")
```

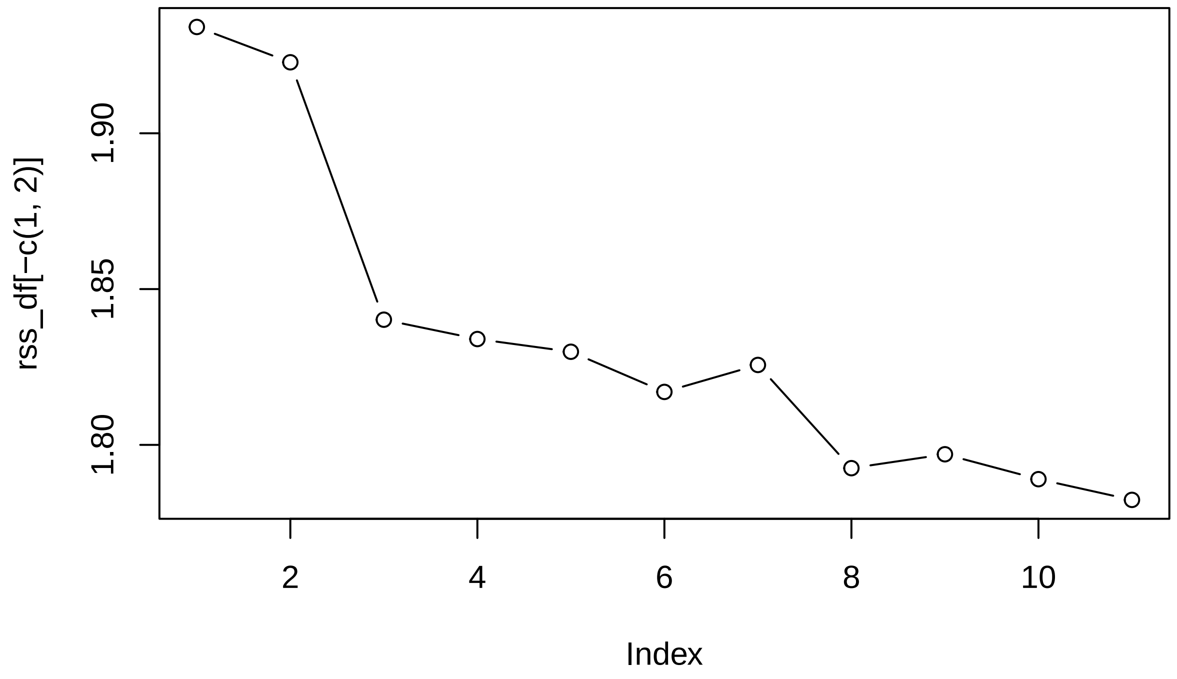


(e)

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.

```
rss_df <- NULL
for (i in 3:13){
```

```r
  fit <- glm(nox ~ bs(dis, df = i), data = Boston)
  rss_df[i] <- sum((nox-fitted(fit))^2)
}
plot(rss_df[-c(1,2)],type = "b")
```

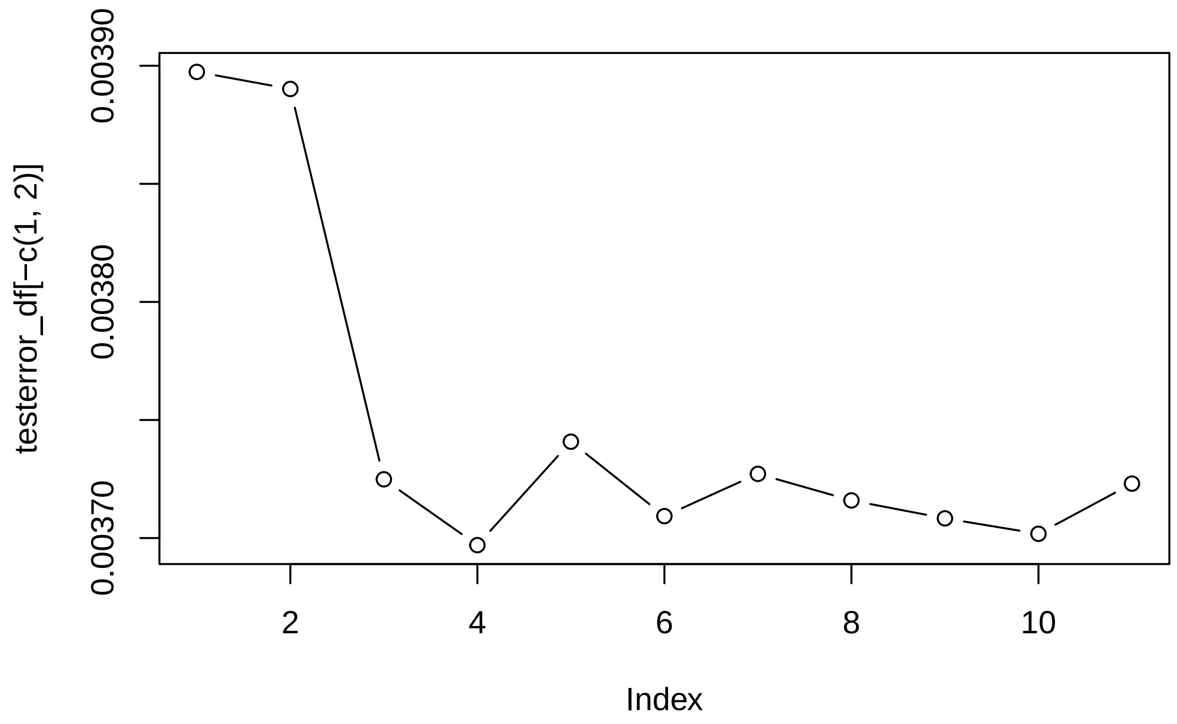


Answer:

(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
testerror_df <- NULL
set.seed(1)
for (i in 3:13){
   fit <- glm(nox ~ bs(dis, df = i), data = Boston)
   testerror_df[i] <- cv.glm(fit,K=10,data = Boston)$delta[1]
}
plot(testerror_df[-c(1,2)],type = "b")
```



Answer:
The best degree of freedom is 4.

7.10

```
library(ISLR2)
```

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

```
##
## Attaching package: 'ISLR2'
```

```
## The following object is masked from 'package:MASS':
##
##     Boston
```

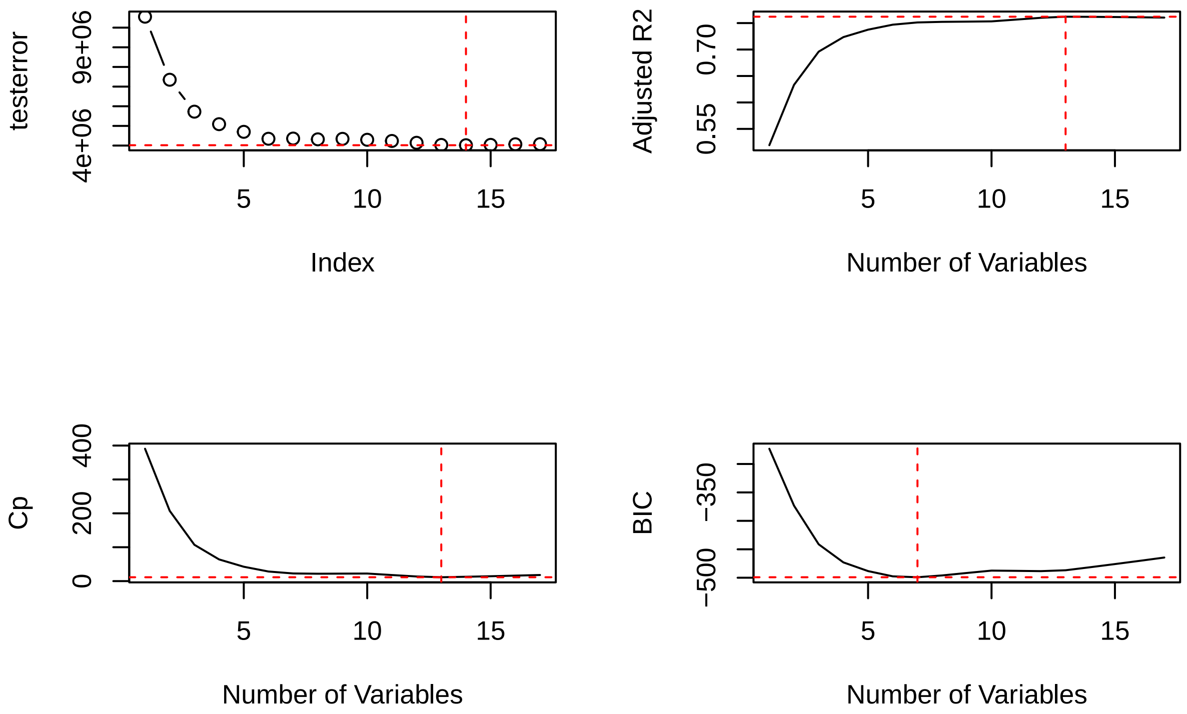
```
library(leaps)
attach(College)
```

(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(2)
train = sample(length(Outstate), length(Outstate)/2)

reg.fit = regsubsets(Outstate ~ ., data = College[train,], nvmax = 17, method = "forward")
reg.summary = summary(reg.fit)
par(mfrow = c(2, 2))
testerror <- NULL
test.mat <- model.matrix(Outstate ~ ., data = College[-train,])
```

```r
for (i in 1:17){
  coefi <- coef(reg.fit,id=i)
  pred <- test.mat[,names(coefi)]%*%coefi
  testerror[i] <- mean((Outstate[-train]-pred)^2)
}
plot(testerror,type = "b")
abline(v=which.min(testerror),h=min(testerror),col="red",lty=2)

plot(reg.summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted R2", type = "l")
abline(v=which.max(reg.summary$adjr2),h=max(reg.summary$adjr2),col="red",lty=2)

plot(reg.summary$cp, xlab = "Number of Variables", ylab = "Cp", type = "l")
abline(v=which.min(reg.summary$cp),h=min(reg.summary$cp),col="red",lty=2)

plot(reg.summary$bic, xlab = "Number of Variables", ylab = "BIC", type = "l")
abline(v=which.min(reg.summary$bic),h=min(reg.summary$bic),col="red",lty=2)
```



Answer:
I would like to choose 6 as the optimal predictor size, because when model has 6 predictors, its BIC is optimal. What's more, although Cp and adjusted R^2 are optimal when predictor size is 14, the test error of 14-size model is not much better than 6-size model, so for better interpretation, I would like to choose 6-size model.

(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.

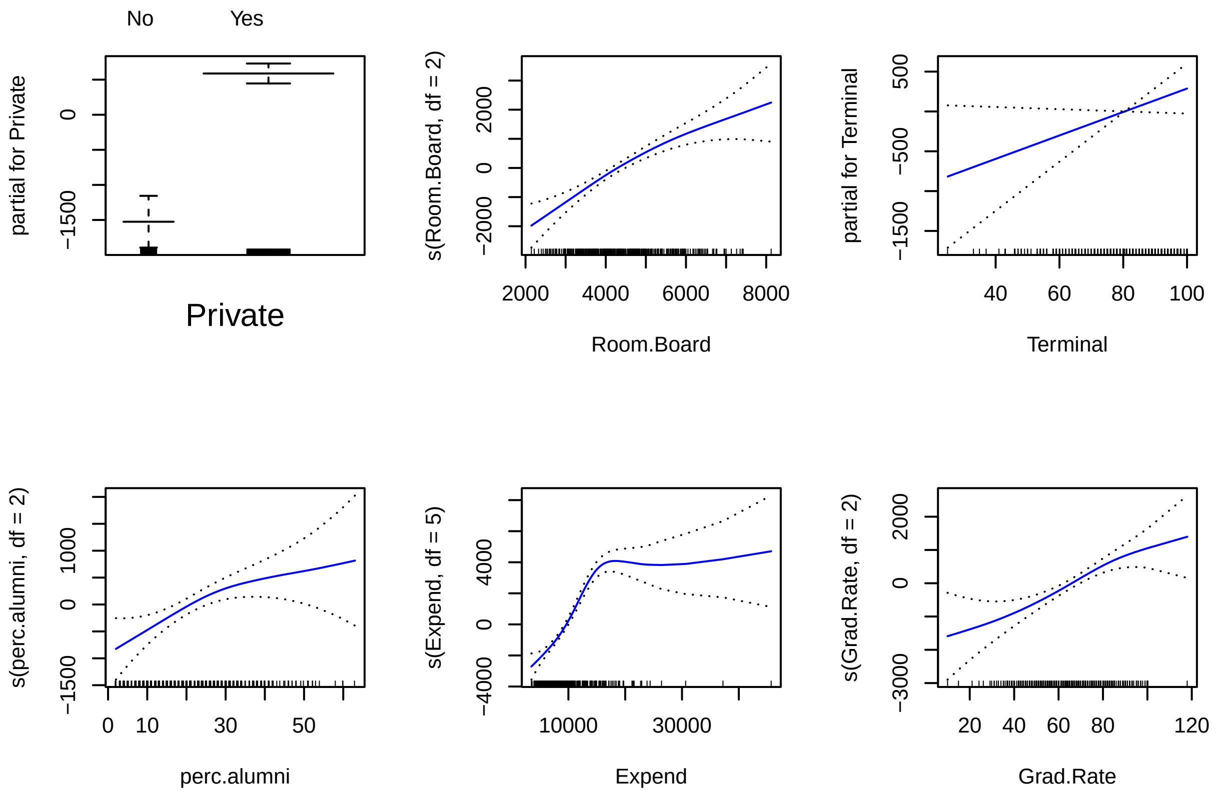
```r
coef6 <- coef(reg.fit,id=6)

library(gam)
```

Answer:

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

```
## Loaded gam 1.20
```

```
gam.fit = gam(Outstate ~ Private + s(Room.Board, df = 2) + Terminal +
    s(perc.alumni, df = 2) + s(Expend, df = 5) + s(Grad.Rate, df = 2), data = College[train,])

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



(c)

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

```
gam.pred = predict(gam.fit, College[-train,])
gam.rmse = mean((College[-train,]$Outstate - gam.pred)^2)
gam.tss = mean((College[-train,]$Outstate - mean(College[-train,]$Outstate))^2)
test.r2 = 1 - gam.rmse/gam.tss
test.r2
```

Answer:

```
## [1] 0.7749275
```

The model obtained by gam can explain 77% information in total.

(d)

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

```
summary(gam.fit)
```

Answer:

```
##
## Call: gam(formula = Outstate ~ Private + s(Room.Board, df = 2) + Terminal +
##     s(perc.alumni, df = 2) + s(Expend, df = 5) + s(Grad.Rate,
##     df = 2), data = College[train, ])
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -6565.30  -984.55   -23.25  1276.96  7154.45
##
## (Dispersion Parameter for gaussian family taken to be 3361806)
##
##     Null Deviance: 6294538288 on 387 degrees of freedom
## Residual Deviance: 1257314068 on 373.9996 degrees of freedom
## AIC: 6947.698
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##                         Df     Sum Sq    Mean Sq F value    Pr(>F)
## Private                  1 1587890516 1587890516 472.333 < 2.2e-16 ***
## s(Room.Board, df = 2)    1 1387894367 1387894367 412.842 < 2.2e-16 ***
## Terminal                 1  392604970  392604970 116.784 < 2.2e-16 ***
## s(perc.alumni, df = 2)   1  231587068  231587068  68.888 1.920e-15 ***
## s(Expend, df = 5)        1  671991923  671991923 199.890 < 2.2e-16 ***
## s(Grad.Rate, df = 2)     1   87734144   87734144  26.097 5.186e-07 ***
## Residuals              374 1257314068    3361806
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                        Npar Df  Npar F     Pr(F)
## (Intercept)
## Private
## s(Room.Board, df = 2)        1  2.7414   0.09862 .
## Terminal
## s(perc.alumni, df = 2)       1  2.2520   0.13429
## s(Expend, df = 5)            4 21.1785 8.882e-16 ***
## s(Grad.Rate, df = 2)         1  2.4440   0.11881
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Anova test shows that there is a non-linear relationship between response and Expend.

7.11

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 esti- mate fixed at its current value, and update only that coefficient estimate using a simple linear regression. The process is continued un- til convergence—that is, until the coefficient estimates stop changing. We now try this out on a toy example.

(a)

```
set.seed(1)
x1 <- rnorm(100,mean = 10)
x2 <- rnorm(100,mean = 5)
eps <- rnorm(100)
y <- 1 + 2 * x1 + 3 * x2 + eps
```

Answer:

(b)

```
set.seed(1)
beta1 <- 1
```

Answer:

(c)

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

Answer:

(d)

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

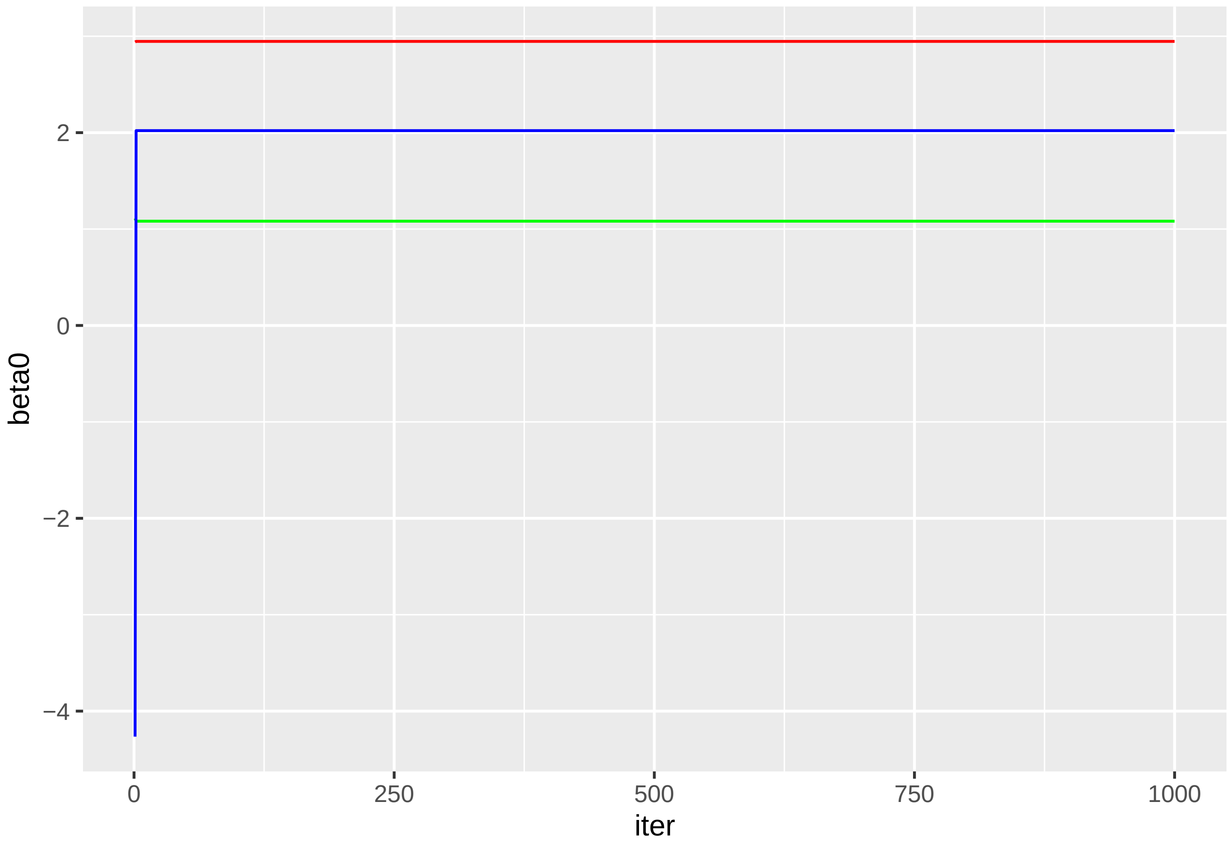
Answer:

(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.

```
beta1 <- NULL
beta2 <- NULL
beta0 <- NULL
set.seed(1)
beta1[1] <- rnorm(1,2,10)
for (i in 1:1000){
  a <- y - beta1[i] * x1
  beta2[i] <- lm(a~x2)$coef[2]
  a <- y - beta2[i] * x2
  if(i<1000){
    beta1[i+1] <- lm(a~x1)$coef[2] ##
```

```r
  }
  beta0[i] <- lm(a~x1)$coef[1]
}
data <- data.frame(iter=1:1000,beta0,beta1,beta2)
ggplot(data)+
  geom_line(aes(x=iter,y=beta0),col="green")+
  geom_line(aes(x=iter,y=beta1),col="blue")+
  geom_line(aes(x=iter,y=beta2),col="red")
```

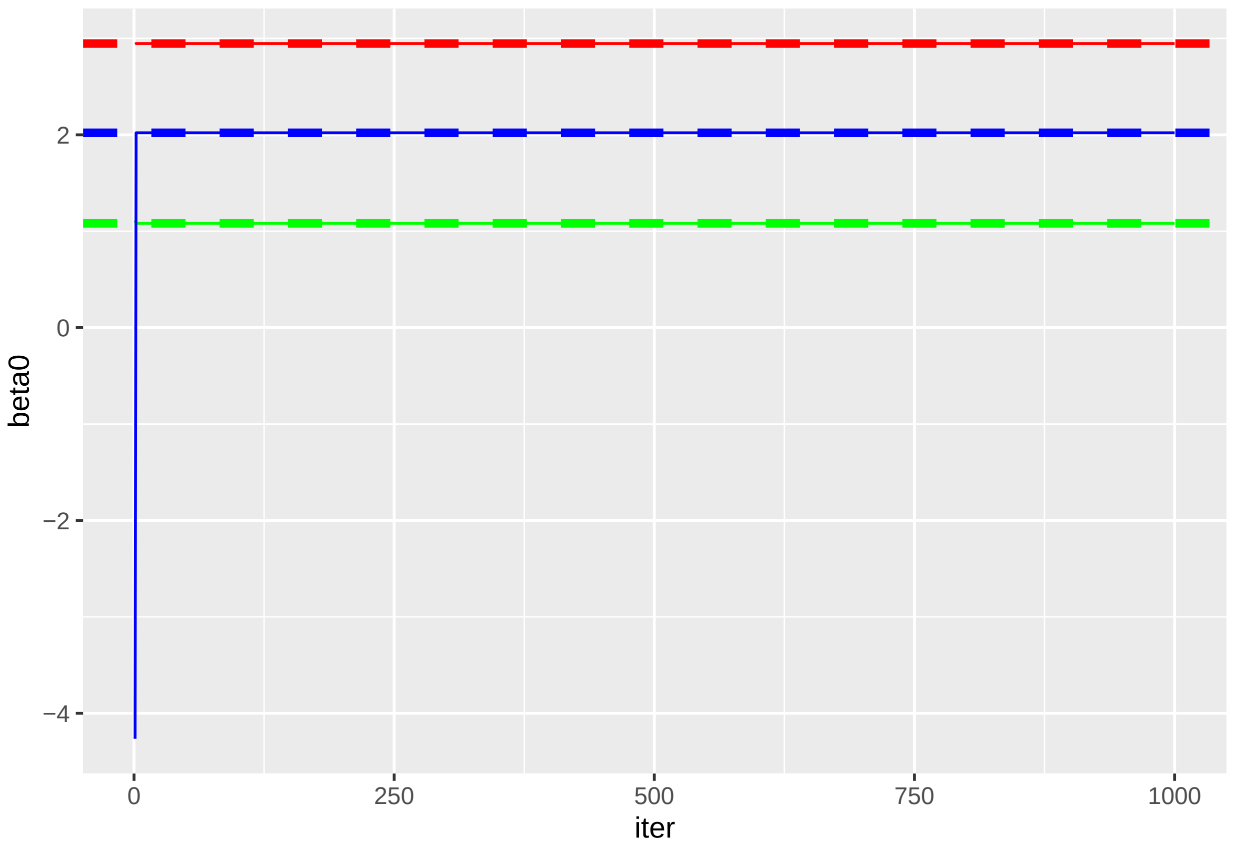


Answer:
The coefficients quickly keep constant.

(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).

```r
multifit <- lm(y~x1+x2)
ggplot(data)+
  geom_line(aes(x=iter,y=beta0),col="green")+
  geom_line(aes(x=iter,y=beta1),col="blue")+
  geom_line(aes(x=iter,y=beta2),col="red")+
  geom_hline(yintercept=multifit$coefficients[1],col="green",lty="dashed",lwd=1.5)+
  geom_hline(yintercept=multifit$coefficients[2],col="blue",lty="dashed",lwd=1.5)+
  geom_hline(yintercept=multifit$coefficients[3],col="red",lty="dashed",lwd=1.5)
```



Answer:

(g)

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

Answer: One iteration can get a good approximation to the linear multiple regression coefficient estimates.