Chapter 7 Moving Beyond Linearity

2023-02-05

```
library(splines)
library(ISLR2)
library(boot)
library(gam)
```

```
## Loading required package: foreach

## Loaded gam 1.22-1
```

```
library(ggplot2)
library(gridExtra)
library(splines)
library(leaps)
library(gam)
```

Exercise 1

a

$$f_1(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$

Here

$$a_1, b_1, c_1, d_1$$

are

$$\beta_0, \beta_1, \beta_2, \beta_3$$

respectively. And,

$$\beta_4 = 0$$

b

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x^3 - 3x^2\xi + 3x\xi^2 - \xi^3) = \beta_0 - \beta_4 \xi^3 + (\beta_1 + 3\xi^2\beta_4)x + (\beta_2 - 3\xi\beta_4)x^2 + (\beta_3 + \beta_4)x^3$$

$$f_2(x) = \beta_0 - \beta_4 \xi^3 + (\beta_1 + 3\xi^2\beta_4)x + (\beta_2 - 3\xi\beta_4)x^2 + (\beta_3 + \beta_4)x^3$$

Here

$$a_2, b_2, c_2, d_2$$

are

$$(\beta_0 - \beta_4\xi^3), (\beta_1 + 3\xi^2\beta_4), (\beta_2 - 3\xi\beta_4), (\beta_3 + \beta_4)$$

respectively.

c

$$f_1(\xi) = \beta_0 + \beta_1\xi + \beta_2\xi^2 + \beta_3\xi^3$$

$$f_2(\xi) = \beta_0 - \beta_4\xi^3 + (\beta_1 + 3\xi^2\beta_4)\xi + (\beta_2 - 3\xi\beta_4)\xi^2 + (\beta_3 + \beta_4)\xi^3 = \beta_0 + \beta_1\xi + \beta_2\xi^2 + \beta_3\xi^3 = f_1(\xi)$$

Therefore,

$$f(x)$$

id continuous at

$$\xi$$

d

$$f_1'(x) = \beta_1 + 2\beta_2 x + 3\beta_3 x^2$$

$$\Rightarrow f_1'(\xi) = \beta_1 + 2\beta_2\xi + 3\beta_3\xi^2$$

$$f_2'(x) = \beta_1 + 3\beta_4\xi^2 + 2(\beta_2 - 3\xi\beta_4)x + 3(\beta_3 + \beta_4)x^2$$

$$\Rightarrow f_2'(\xi) = \beta_1 + 3\xi^2\beta_4 + 2(\beta_2 - 3\xi\beta_4)\xi + 3(\beta_3 + \beta_4)\xi^2 = \beta_1 + 2\beta_2\xi + 3\beta_3\xi^2$$

Hence,

$$f_1'(\xi) = f_2'(\xi) = \beta_1 + 2\beta_2\xi + 3\beta_3\xi^2$$

In other words,

$$f'(x)$$

is continuous at

$$\xi$$

e

$$f_1''(x) = 2\beta_2 + 6\beta_3 x$$

$$\Rightarrow f_1''(\xi) = 2\beta_2 + 6\beta_3\xi$$

$$f_2''(x) = 2(\beta_2 - 3\xi\beta_4) + 6(\beta_3 + \beta_4)x$$

$$\Rightarrow f_2''(\xi) = 2(\beta_2 - 3\xi\beta_4) + 6(\beta_3 + \beta_4)\xi = 2\beta_2 + 6\beta_3\xi = f_1''(\xi)$$

That is,

$$f''(x)$$

is continuous at

$$\xi$$

Exercise 2

$$\hat{g} = \underset{g}{\operatorname{argmin}} \left(\sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int \left[g^{(m)}(x) \right]^2 dx \right)$$

a

$$\lambda = \infty, m = 0$$

In order for

$$\hat{g}$$

to be minimised,

$$g(x)$$

must be zero otherwise the second term

$$\lambda \int \left[g^{(m)}(x) \right]^2 dx$$

becomes very large. Therefore,

$$\hat{g}(x) = 0$$

b

$$\lambda = \infty, m = 1$$

In order for

$$\hat{g}$$

to be minimised,

$$g'(x)$$

must be zero otherwise the second term

$$\lambda \int \left[g^{(1)}(x) \right]^2 dx$$

becomes very large.

$$g'(x) = 0 \Leftrightarrow g(x) = c$$

Here

$$c$$

is a constant number.

$$\hat{g}(x)$$

is a horizontal line.

c

$$\lambda = \infty, m = 2$$

Using the idea from (b),

$$g''(x) = 0 \Leftrightarrow g(x) = ax + b$$

In this case,

$$\hat{g}(x)$$

is a straight line.

d

$$\lambda = \infty, m = 3$$

$$g'''(x) = 0 \Leftrightarrow g(x) = ax^2 + bx + c$$

And

$$\hat{g}(x)$$

is a quadratic line in this scenario.

e

$$\lambda = 0, m = 3$$

Now,

$$\hat{g}$$

can be written as

$$\hat{g} = \underset{g}{\operatorname{argmin}} \left(\sum_{i=1}^{n} (y_i - g(x_i))^2 \right)$$

Just overfit the data as much as possible, till

$$y_i = g_i \quad \forall i$$

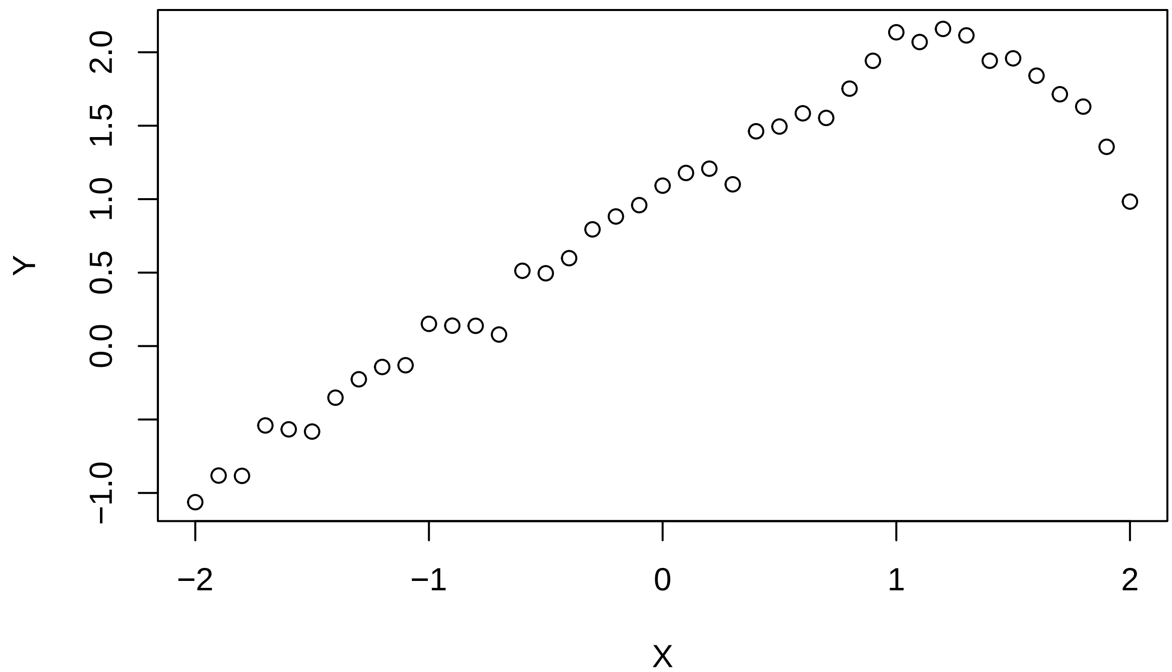
In other words,

$$\hat{g}$$

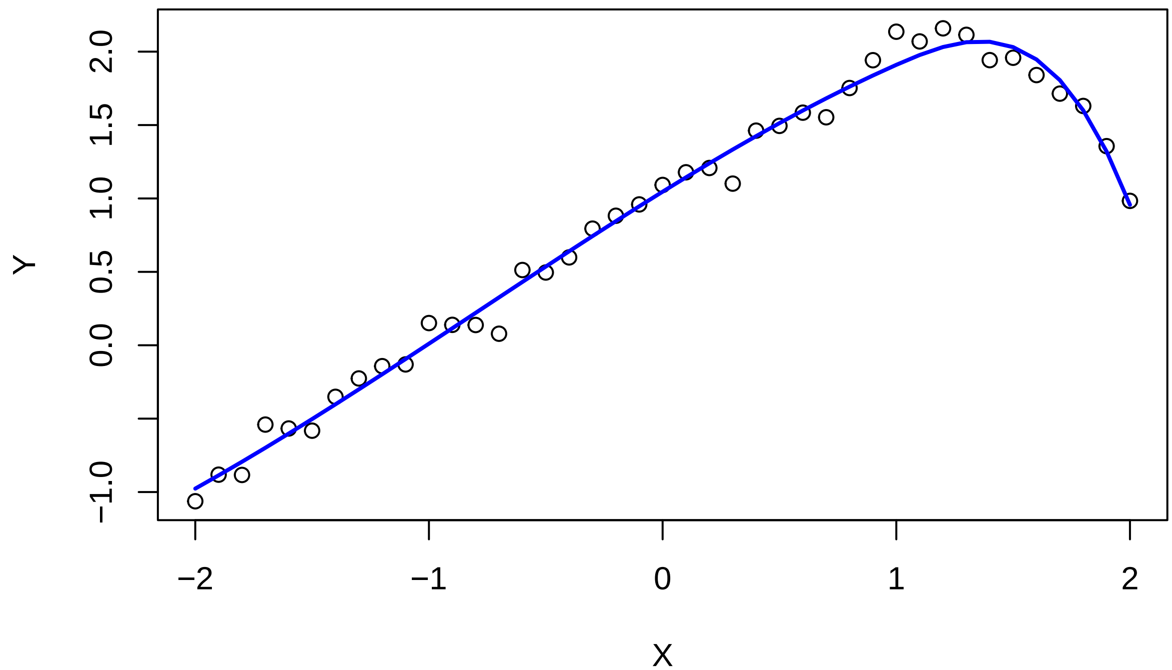
is now a curve that interpolates all observations.

Exercise 3

```
set.seed(1)

X = seq(-2, 2, by = 0.1)
e = rnorm(41, 0, sd = 0.1)
Y = 1 + 1 * X - 2 * (X - 1)^2 * I(X >= 1) + e
plot(X, Y)
```

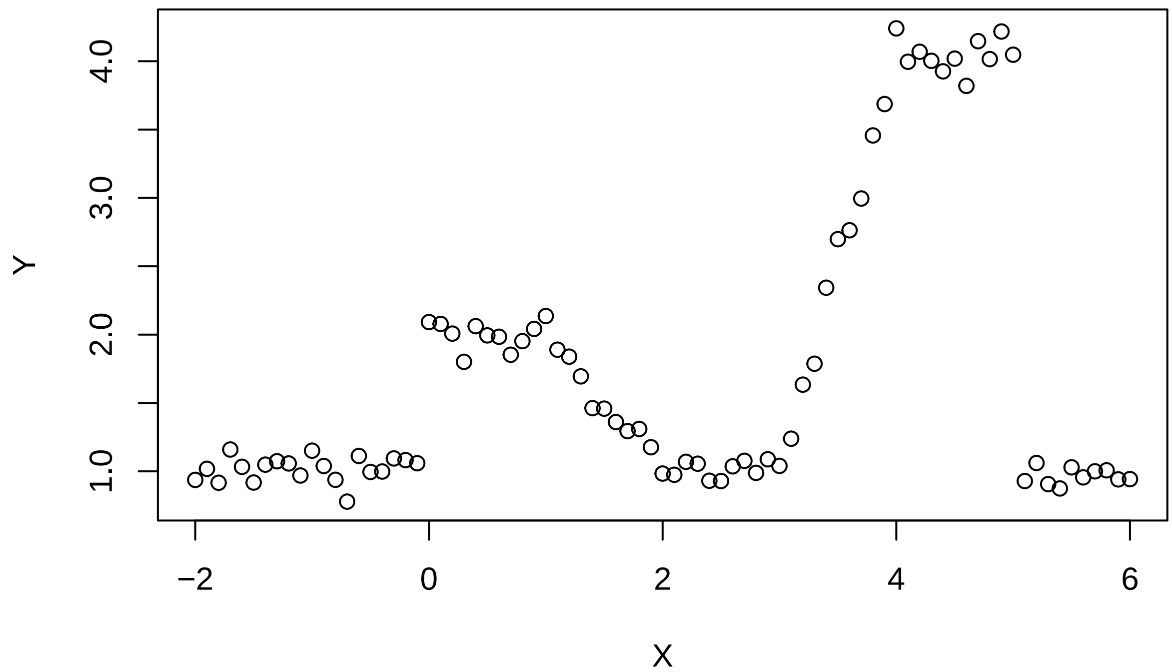



```
dat = data.frame(X, Y)
fit = lm(Y ~ bs(X, knots = 1), data = dat)
pred = predict(fit, newdata = list(X = X))
plot(X, Y)
lines(X, pred, lwd = 2, col = 'blue')
```

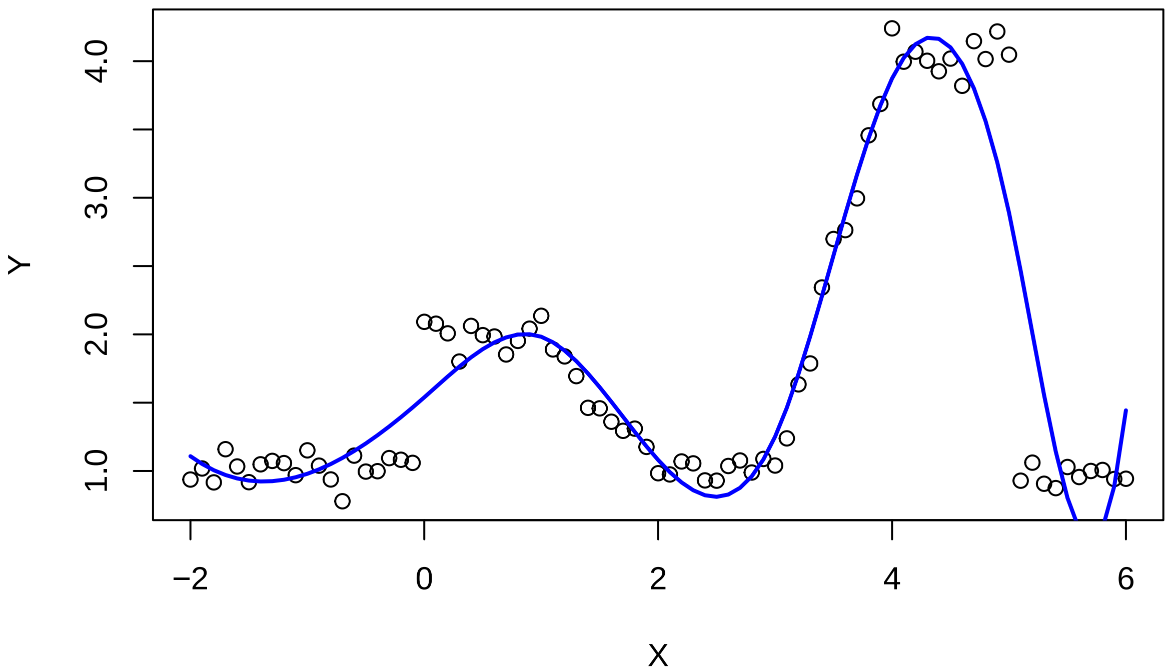


Exercise 4

```
set.seed(1)

X = seq(-2, 6, by = 0.1)
b1 = I(X >= 0 & X <= 2) - (X - 1) * I(X >= 1 & X <= 2)
b2 = (X - 3) * I(X >= 3 & X <= 4) + I(X > 4 & X <= 5)
e = rnorm(81, 0, sd = 0.1)
Y = 1 + 1 * b1 + 3 * b2 + e
plot(X, Y)
```




```
dat = data.frame(X, Y)
fit = lm(Y ~ bs(X, knots = c(0, 1, 2, 3, 4, 5)), data = dat)
pred = predict(fit, newdata = list(X = X))
plot(X, Y)
lines(X, pred, lwd = 2, col = 'blue')
```



Exercise 5

$$\hat{g}_1 = \underset{g}{\text{argmin}} \left(\sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int \left[g^{(3)}(x) \right]^2 dx \right)$$

$$\hat{g}_2 = \underset{g}{\text{argmin}} \left(\sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int \left[g^{(4)}(x) \right]^2 dx \right)$$

From exercise 2, we know that when

$$\lambda$$

approaches infinity, the two curve functions above can be minimised by setting

$$g^{(m)} = 0$$

$$g^{(3)} = 0 \Leftrightarrow g_3(x) = ax^2 + bx + c$$

$$g^{(4)} = 0 \Leftrightarrow g_4(x) = ax^4 + bx^3 + cx + d$$

a

$$\hat{g}_4(x)$$

has the smaller training RSS as it is more flexible.

b

We can't say for sure when it comes to the test RSS, it depends on the true relationship between X and Y.

c

For

$$\lambda = 0$$

, the both two curves interpolate all its observations therefore there they will have the same training RSS which is zero. As for the test RSS, we can't say for sure also since it depends on the true relationship between X and Y.

Exercise 6

```
head(Wage)
```

```
##        year age            maritl     race       education             region
## 231655 2006  18 1. Never Married 1. White    1. < HS Grad 2. Middle Atlantic
## 86582  2004  24 1. Never Married 1. White 4. College Grad 2. Middle Atlantic
## 161300 2003  45       2. Married 1. White 3. Some College 2. Middle Atlantic
## 155159 2003  43       2. Married 3. Asian 4. College Grad 2. Middle Atlantic
## 11443  2005  50      4. Divorced 1. White      2. HS Grad 2. Middle Atlantic
## 376662 2008  54       2. Married 1. White 4. College Grad 2. Middle Atlantic
##              jobclass         health health_ins  logwage      wage
## 231655  1. Industrial      1. <=Good      2. No 4.318063  75.04315
## 86582  2. Information 2. >=Very Good      2. No 4.255273  70.47602
## 161300  1. Industrial      1. <=Good     1. Yes 4.875061 130.98218
## 155159 2. Information 2. >=Very Good     1. Yes 5.041393 154.68529
## 11443  2. Information      1. <=Good     1. Yes 4.318063  75.04315
## 376662 2. Information 2. >=Very Good     1. Yes 4.845098 127.11574
```

```
colnames(Wage)
```

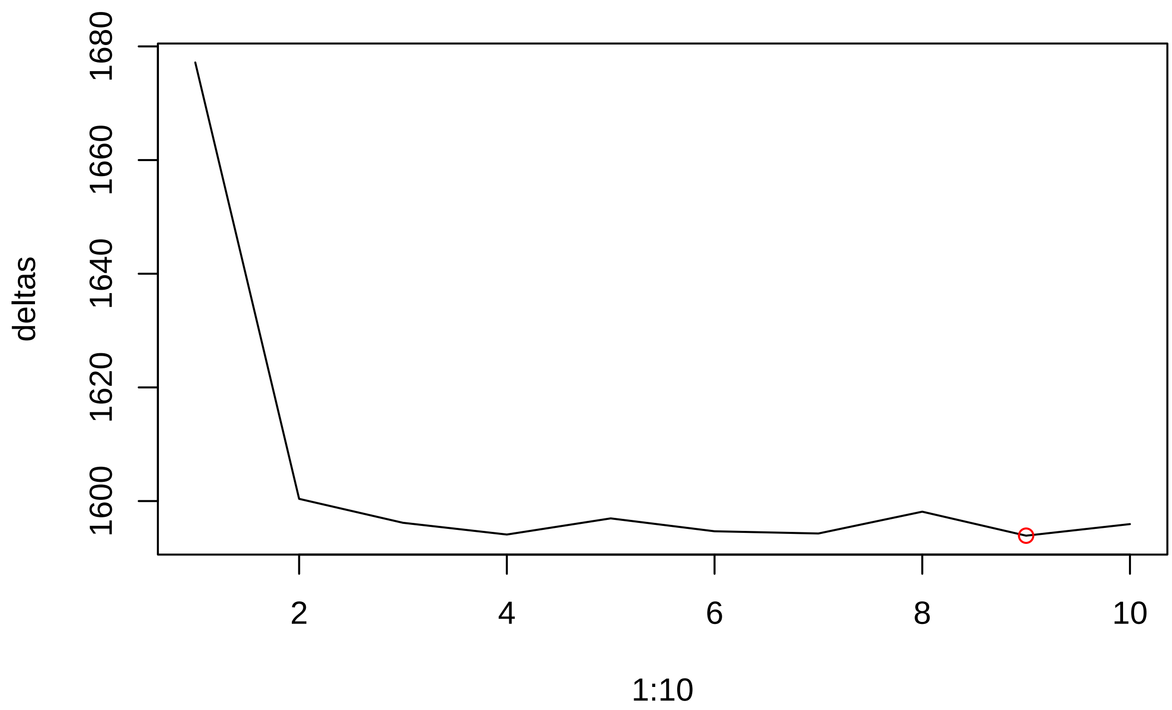
```
##  [1] "year"       "age"        "maritl"     "race"       "education"
##  [6] "region"     "jobclass"   "health"     "health_ins" "logwage"
## [11] "wage"
```

a

The optimal degree d = 9 was chosen the most often. However, using anova, models with 4 degrees or 9 degrees were chosen. The results from the two approaches somewhat agree with each other.

CV

```r
deltas = rep(NA, 10)
for (i in 1:10){
  model = glm(wage ~ poly(age, i), data = Wage)
  cv_model = cv.glm(model, data = Wage, K = 10)
  deltas[i] = cv_model$delta[1]
}

plot(1:10, deltas, type = 'l')
points(which.min(deltas), deltas[which.min(deltas)], col = 'red')
```



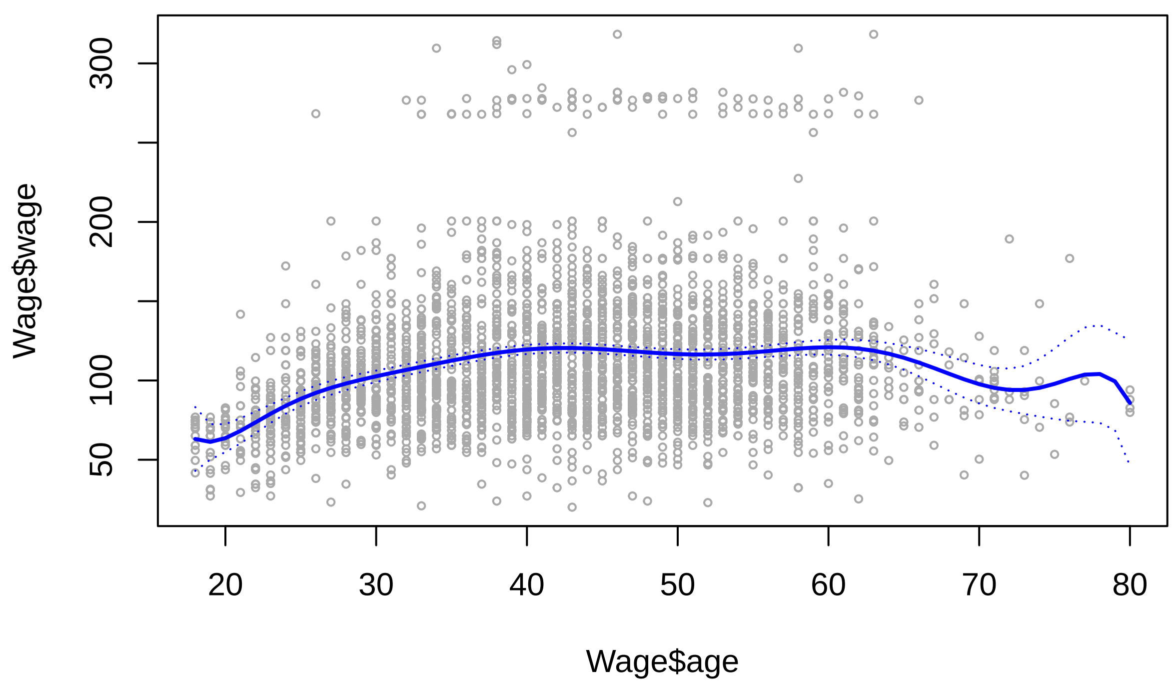

```r
optimal_degrees = rep(NA, 3)

for (i in 1:100){

  deltas = rep(NA, 10)

  for (j in 1:10){

    model = glm(wage ~ poly(age, j), data = Wage)
    cv_model = cv.glm(model, data = Wage, K = 10)
    deltas[j] = cv_model$delta[1]
  }
```

```
  optimal_degrees[i] = which.min(deltas)
}
```

```
table(optimal_degrees)
```

```
## optimal_degrees
##  4  5  6  7  8  9 10
##  6  4 10 17  7 35 21
```

```
agelims = range(Wage$age)
age_grid = seq(from = agelims[1], to = agelims[2])
preds = predict(glm(wage ~ poly(age, 9), data = Wage),
                newdata = list(age = age_grid), se = T)
se_bands = cbind(preds$fit + 2 * preds$se.fit, preds$fit - 2 * preds$se.fit)

plot(Wage$age, Wage$wage, xlim = agelims, cex = .5, col = 'darkgrey')
lines(age_grid, preds$fit, lwd = 2, col = 'blue')
matlines(age_grid, se_bands, lwd = 1, col = 'blue', lty = 3)
```

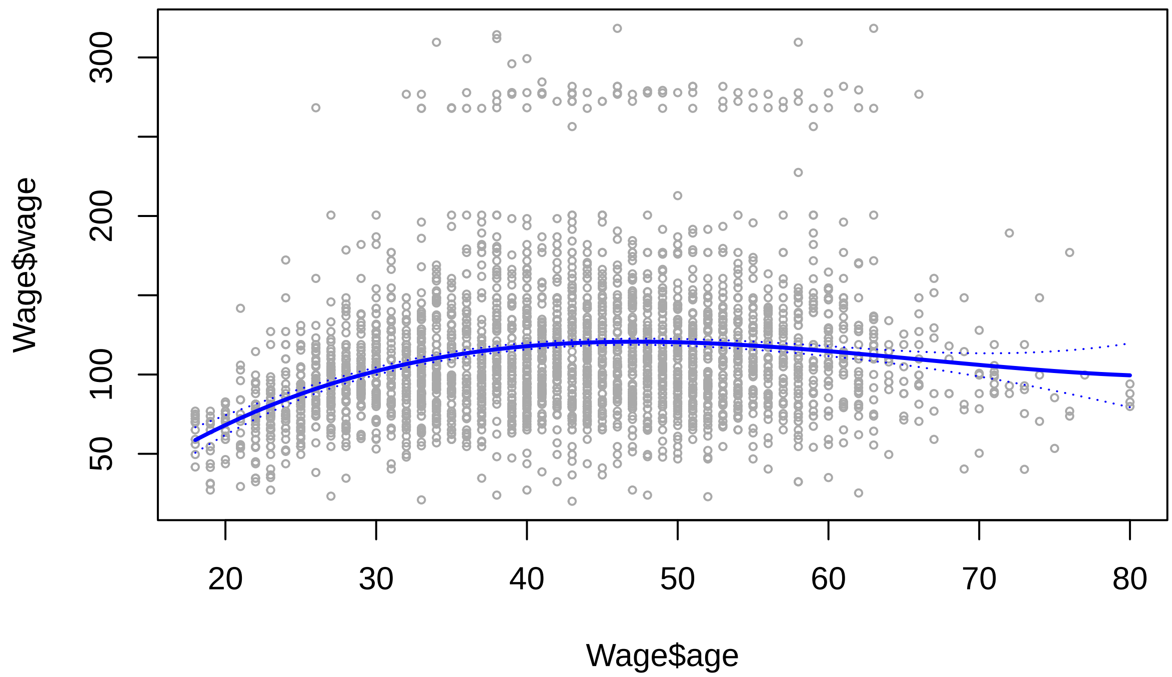


ANOVA

```r
fit1 = lm(wage ~ age, data = Wage)
fit2 = lm(wage ~ poly(age, 2), data = Wage)
fit3 = lm(wage ~ poly(age, 3), data = Wage)
fit4 = lm(wage ~ poly(age, 4), data = Wage)
fit5 = lm(wage ~ poly(age, 5), data = Wage)
fit6 = lm(wage ~ poly(age, 6), data = Wage)
fit7 = lm(wage ~ poly(age, 7), data = Wage)
fit8 = lm(wage ~ poly(age, 8), data = Wage)
fit9 = lm(wage ~ poly(age, 9), data = Wage)
fit10 = lm(wage ~ poly(age, 10), data = Wage)

anova(fit1, fit2, fit3, fit4, fit5, fit6, fit7, fit8, fit9, fit10)
```

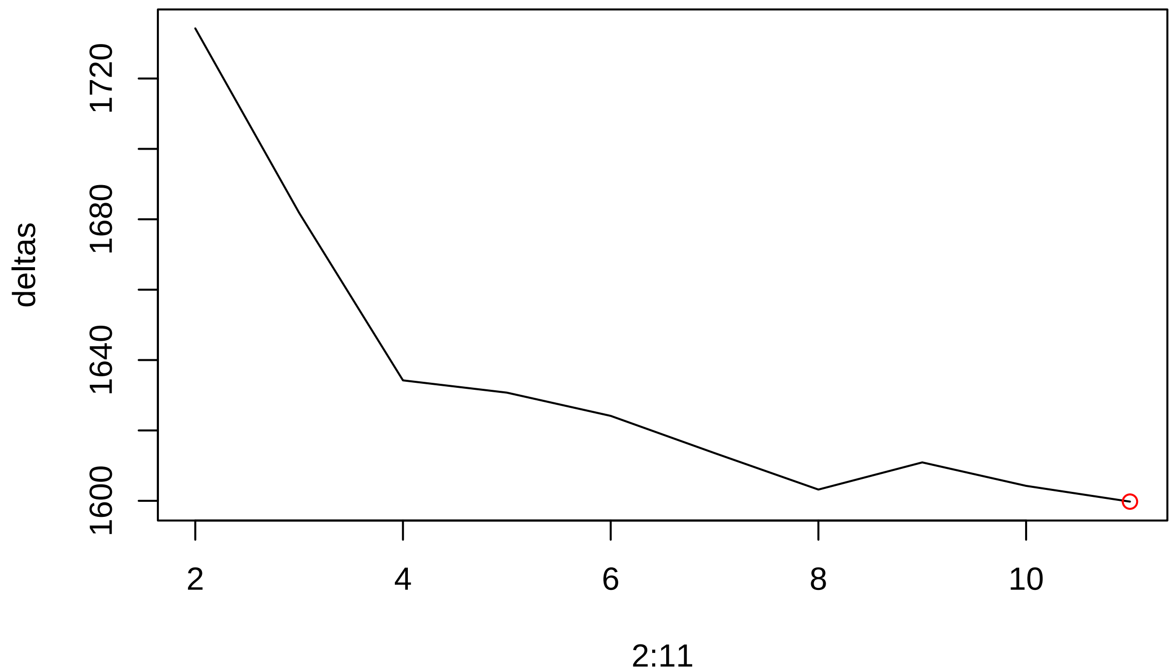
```
## Analysis of Variance Table
##
## Model  1: wage ~ age
## Model  2: wage ~ poly(age, 2)
## Model  3: wage ~ poly(age, 3)
## Model  4: wage ~ poly(age, 4)
## Model  5: wage ~ poly(age, 5)
## Model  6: wage ~ poly(age, 6)
## Model  7: wage ~ poly(age, 7)
## Model  8: wage ~ poly(age, 8)
## Model  9: wage ~ poly(age, 9)
## Model 10: wage ~ poly(age, 10)
##    Res.Df     RSS Df Sum of Sq        F    Pr(>F)
## 1    2998 5022216
## 2    2997 4793430  1    228786 143.7638 < 2.2e-16 ***
## 3    2996 4777674  1     15756   9.9005  0.001669 **
## 4    2995 4771604  1      6070   3.8143  0.050909 .
## 5    2994 4770322  1      1283   0.8059  0.369398
## 6    2993 4766389  1      3932   2.4709  0.116074
## 7    2992 4763834  1      2555   1.6057  0.205199
## 8    2991 4763707  1       127   0.0796  0.777865
## 9    2990 4756703  1      7004   4.4014  0.035994 *
## 10   2989 4756701  1         3   0.0017  0.967529
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
preds = predict(glm(wage ~ poly(age, 3), data = Wage),
                newdata = list(age = age_grid), se = T)
se_bands = cbind(preds$fit + 2 * preds$se.fit, preds$fit - 2 * preds$se.fit)

plot(Wage$age, Wage$wage, xlim = agelims, cex = .5, col = 'darkgrey')
lines(age_grid, preds$fit, lwd = 2, col = 'blue')
matlines(age_grid, se_bands, lwd = 1, col = 'blue', lty = 3)
```



b

```
deltas = rep(NA, 10)

for (i in 2:11){
  Wage$age_cut = cut(Wage$age, i)
  model = glm(wage ~ age_cut, data = Wage)
  cv_model = cv.glm(Wage, model, K = 10)
  deltas[i-1] = cv_model$delta[1]
}

plot(2:11, deltas, type = 'l')
min_index = which.min(deltas)
points(min_index + 1, deltas[min_index], col = 'red')
```




```
optimal_cuts = rep(NA, 10)

for (i in 1:100){

  set.seed(i)
  deltas = rep(NA, 10)

  for (j in 2:11){
    Wage$age_cut = cut(Wage$age, j)
    model = glm(wage ~ age_cut, data = Wage)
    cv_model = cv.glm(Wage, model, K = 10)
    deltas[j-1] = cv_model$delta[1]
  }

  optimal_cuts[i] = which.min(deltas) + 1

}

table(optimal_cuts)
```

```
## optimal_cuts
##  8 11
## 22 78
```

Exercise 7

Exploring

Here I use four predictors, they are martial status, education, race and health. In terms of

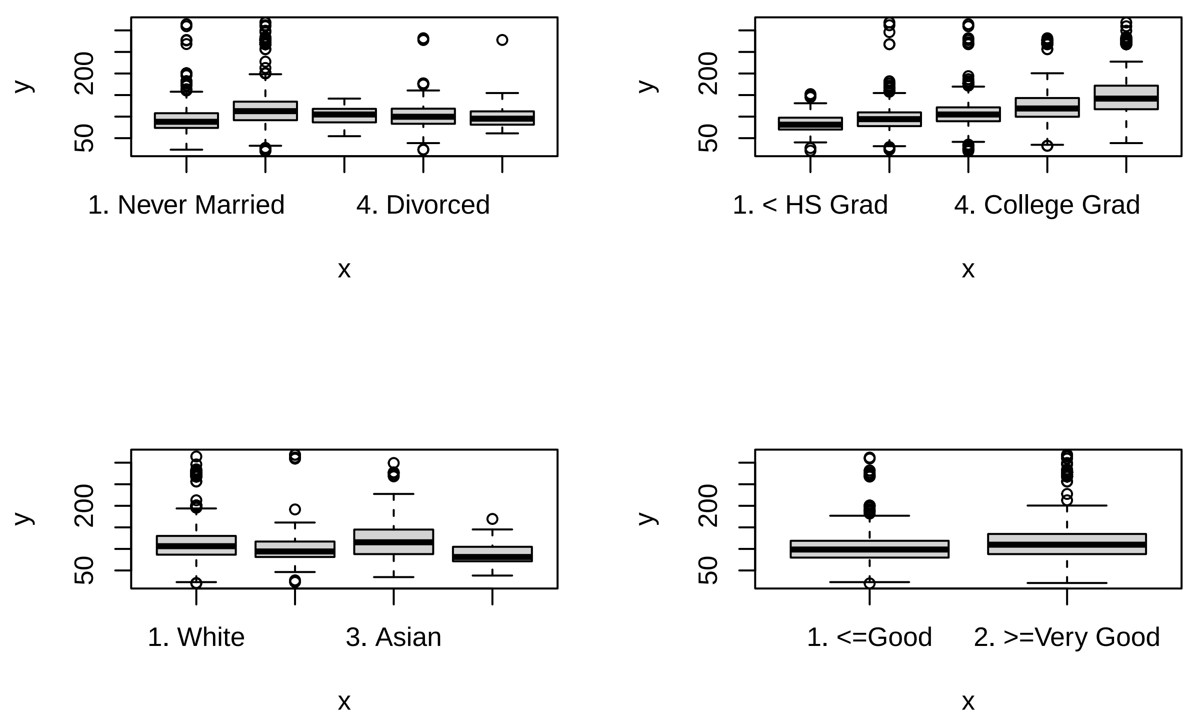
Martial status: doesn't play have a strong relationship with wage. Married workers tend to have higher income on average. Most people whose income really high (higher than $200.000) are in the never married and married group.

Education: on average, the higher in education level, the higher in wage.

Race: white and Asian people tend to have higher income on average. Most people whose income really high (higher than $200.000) are in the white group.

Health: seems to be not a significant predictor. People with good health condition tend to have higher income on average.

```
par(mfrow = c(2, 2))
plot(Wage$maritl, Wage$wage)
plot(Wage$education, Wage$wage)
plot(Wage$race, Wage$wage)
plot(Wage$health, Wage$wage)
```

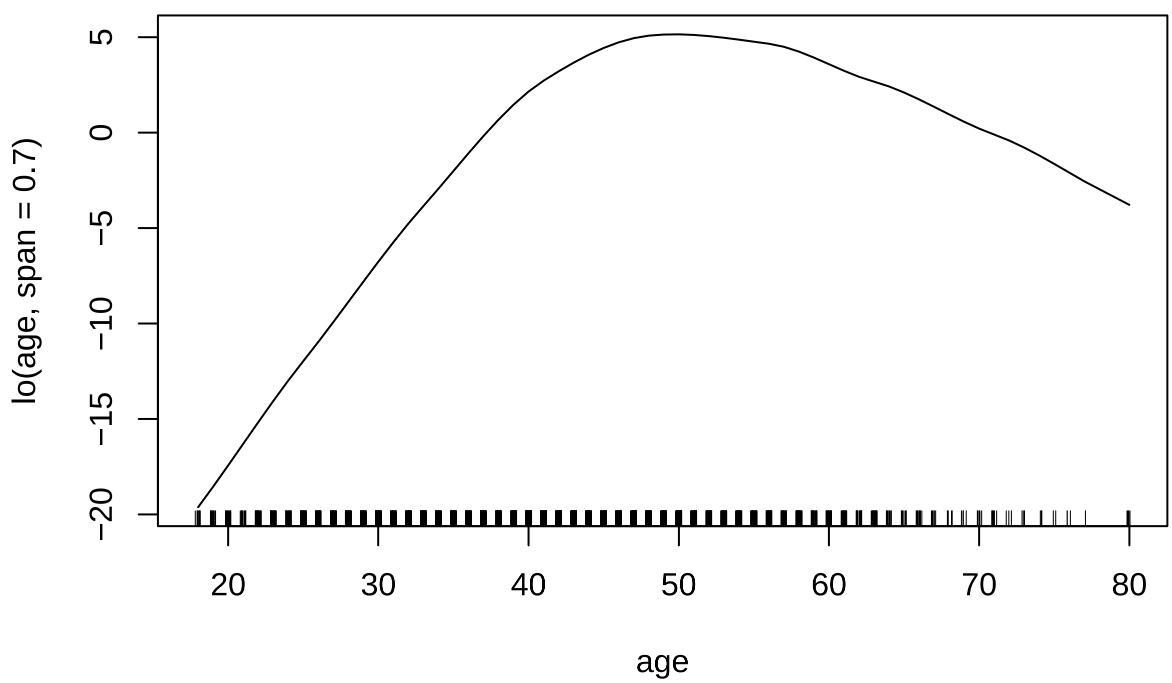


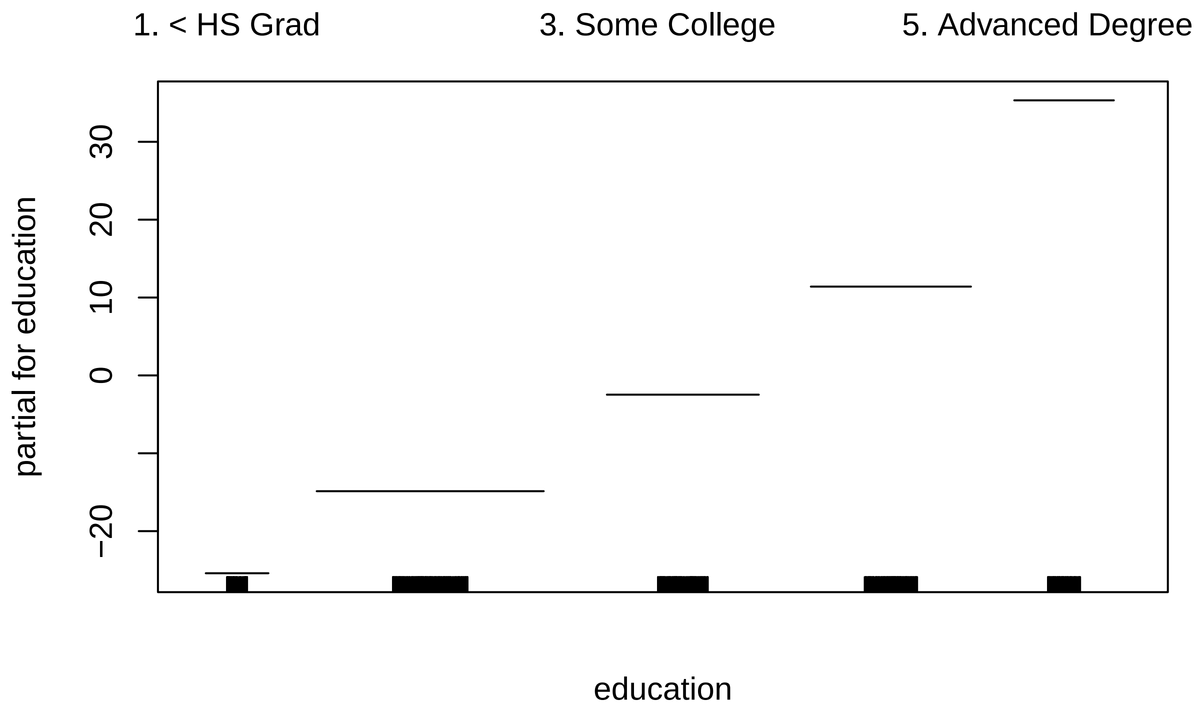
Modelling

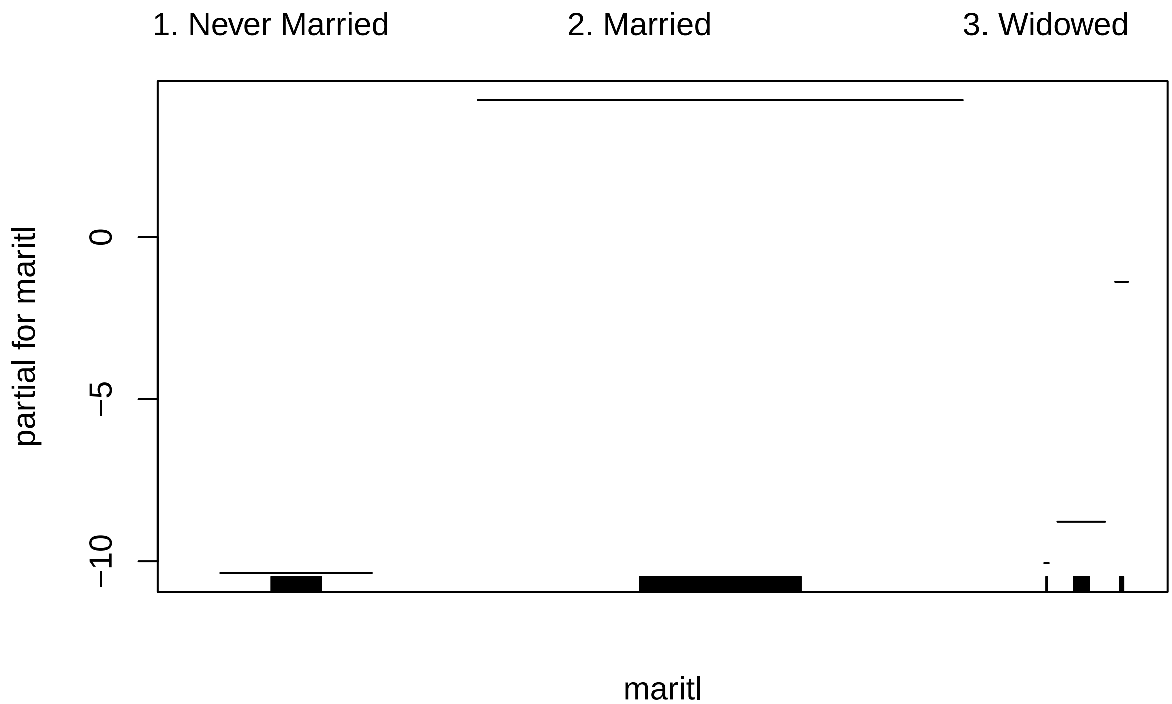
```r
library(gam)
model_1 = gam(wage ~ lo(age, span = 0.7) + education, data = Wage)
model_2 = gam(wage ~ lo(age, span = 0.7) + education + maritl, data = Wage)
model_3 = gam(wage ~ lo(age, span = 0.7) + education + maritl + race, data = Wage)
model_4 = gam(wage ~ lo(age, span = 0.7) + education + maritl + race + health, data = Wage)

anova(model_1, model_2, model_3, model_4)
```

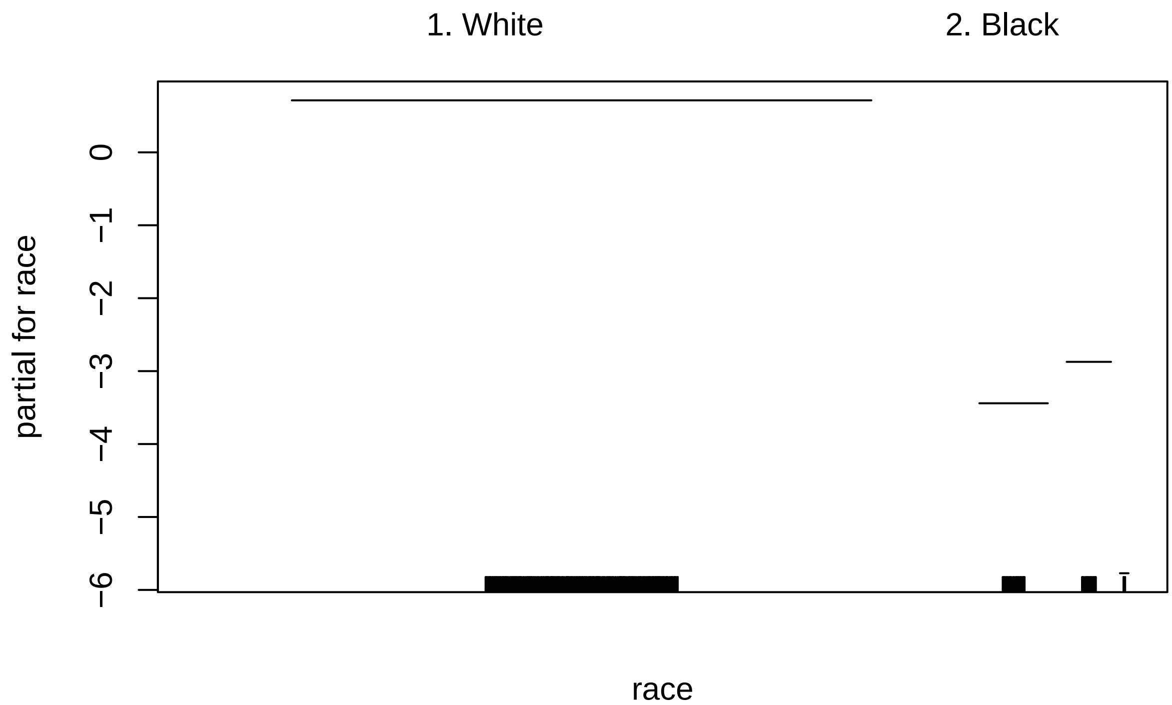
```
## Analysis of Deviance Table
##
## Model 1: wage ~ lo(age, span = 0.7) + education
## Model 2: wage ~ lo(age, span = 0.7) + education + maritl
## Model 3: wage ~ lo(age, span = 0.7) + education + maritl + race
## Model 4: wage ~ lo(age, span = 0.7) + education + maritl + race + health
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1    2992.8    3737372
## 2    2988.8    3634221  4   103150 < 2.2e-16 ***
## 3    2985.8    3626320  3     7901   0.08732 .
## 4    2984.8    3594978  1    31342 3.375e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

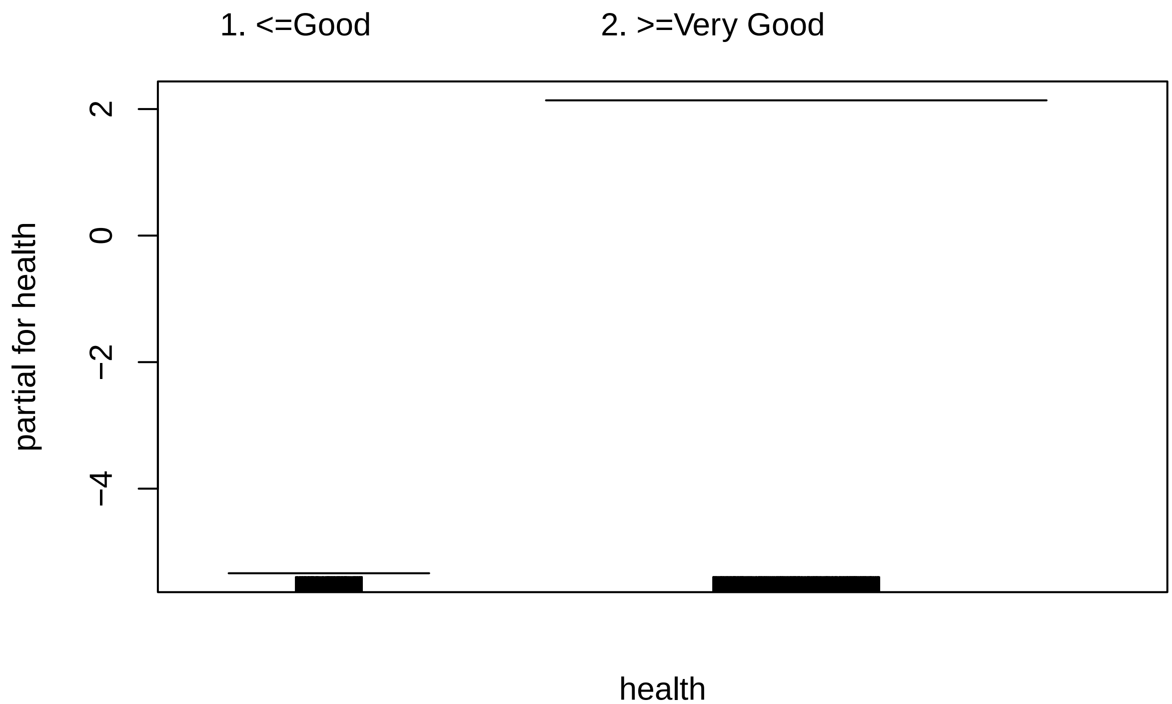
```r
plot(model_4)
```




1. < HS Grad
3. Some College
5. Advanced Degree
partial for education
30
20
10
0
−20
education


1. Never Married
2. Married
3. Widowed
partial for maritl
0
−5
−10
maritl


1. White
2. Black
0
−1
−2
−3
−4
−5
−6
partial for race
race



Exercise 8

```
head(Auto)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                        name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

```
colnames(Auto)
```

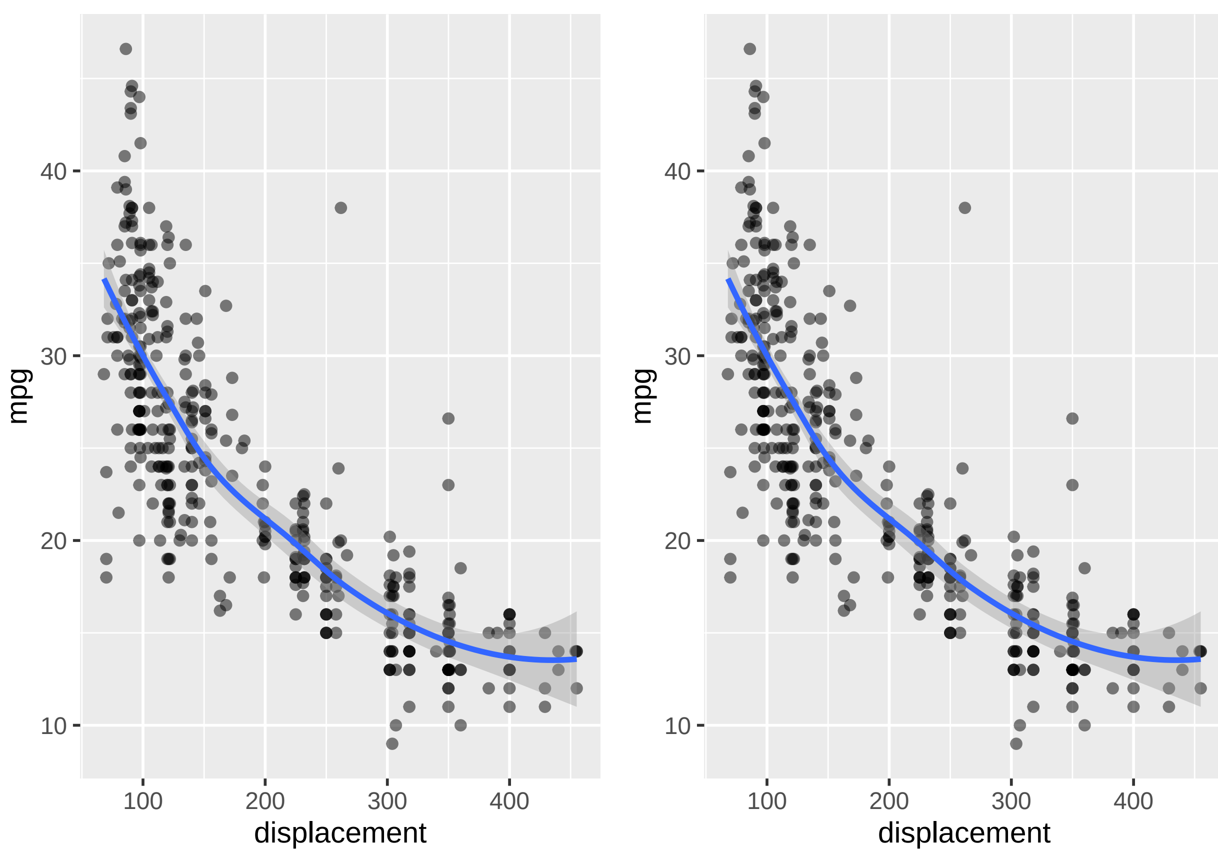
```
## [1] "mpg"          "cylinders"    "displacement" "horsepower"   "weight"
## [6] "acceleration" "year"         "origin"       "name"
```

There is absolutely non-linear relationship between some predictors and the response (here is mpg as we did in the previous chapters). We can plot the relationship, perform polynomial model or step function to ensure there is indeed a non-linear relationship.

Plots

```
plot1 = ggplot(Auto, aes(x = displacement, y = mpg)) +
  geom_point(alpha = 0.5) +
  geom_smooth()

plot2 = ggplot(Auto, aes(x = displacement, y = mpg)) +
  geom_point(alpha = 0.5) +
  geom_smooth()

grid.arrange(plot1, plot2, ncol = 2)
```

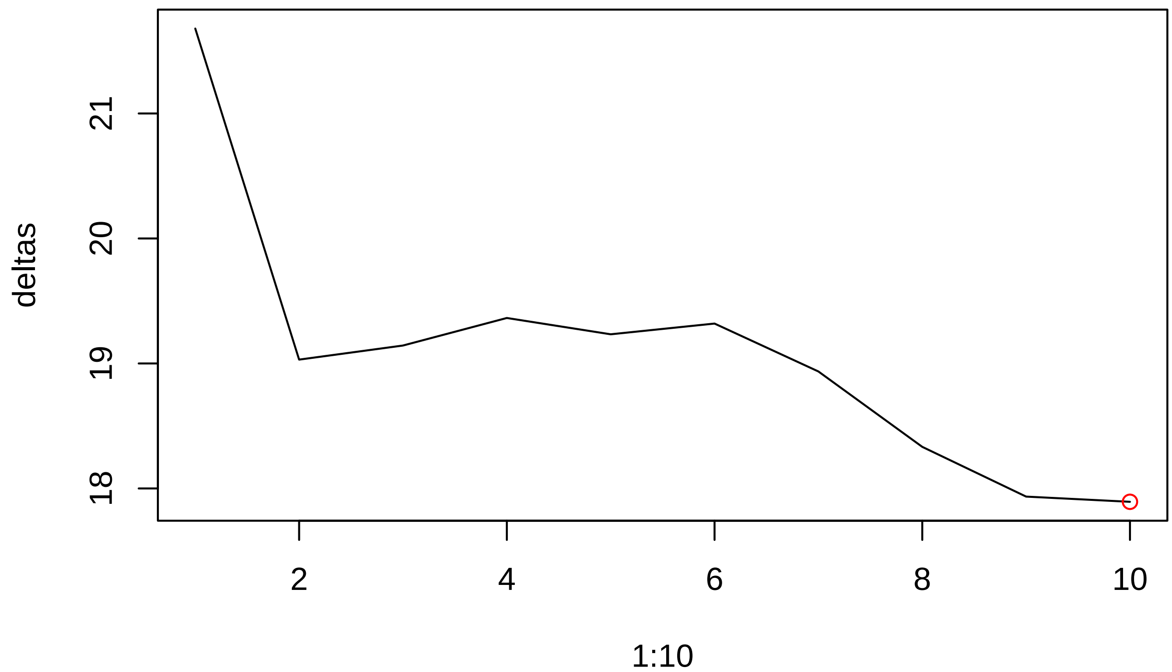
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



CV

Here we see that the optimal degree is 10, significantly better than 1.

```r
deltas = rep(NA, 10)
for (i in 1:10){
  model = glm(mpg ~ poly(displacement, i), data = Auto)
  cv_model = cv.glm(model, data = Auto, K = 10)
  deltas[i] = cv_model$delta[1]
}

plot(1:10, deltas, type = 'l')
points(which.min(deltas), deltas[which.min(deltas)], col = 'red')
```

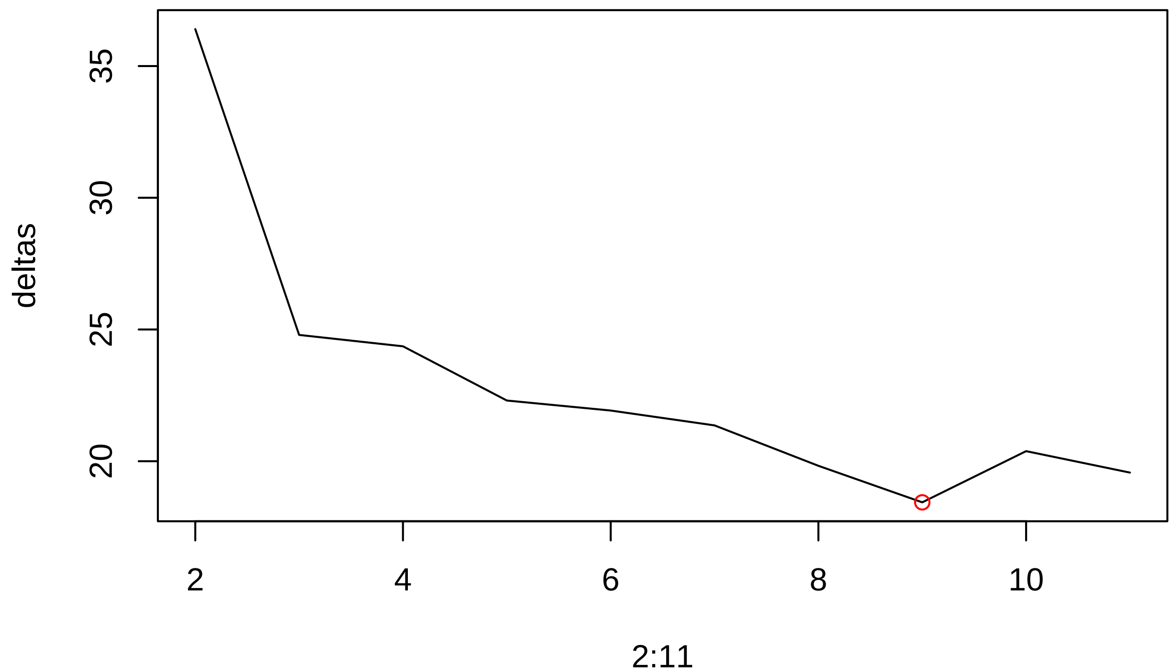


Cuts

It is suggested to have 9 cuts. (If the relationship is linear, the number of cuts should be small).

```r
deltas = rep(NA, 10)

for (i in 2:11){
  Auto$dis_cut = cut(Auto$displacement, i)
  model = glm(mpg ~ dis_cut, data = Auto)
  cv_model = cv.glm(Auto, model, K = 10)
  deltas[i-1] = cv_model$delta[1]
}

plot(2:11, deltas, type = 'l')
```

```r
min_index = which.min(deltas)
points(min_index + 1, deltas[min_index], col = 'red')
```



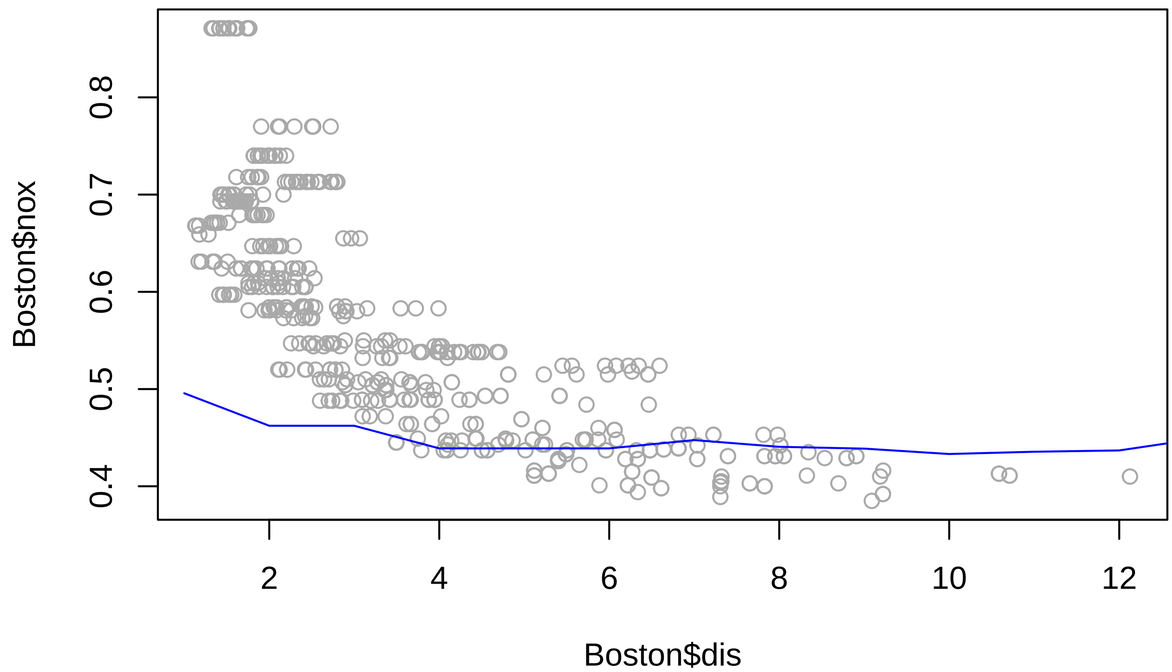
Exercise 9

```r
head(Boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio lstat medv
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3  4.98 24.0
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8  9.14 21.6
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8  4.03 34.7
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7  2.94 33.4
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7  5.33 36.2
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7  5.21 28.7
```

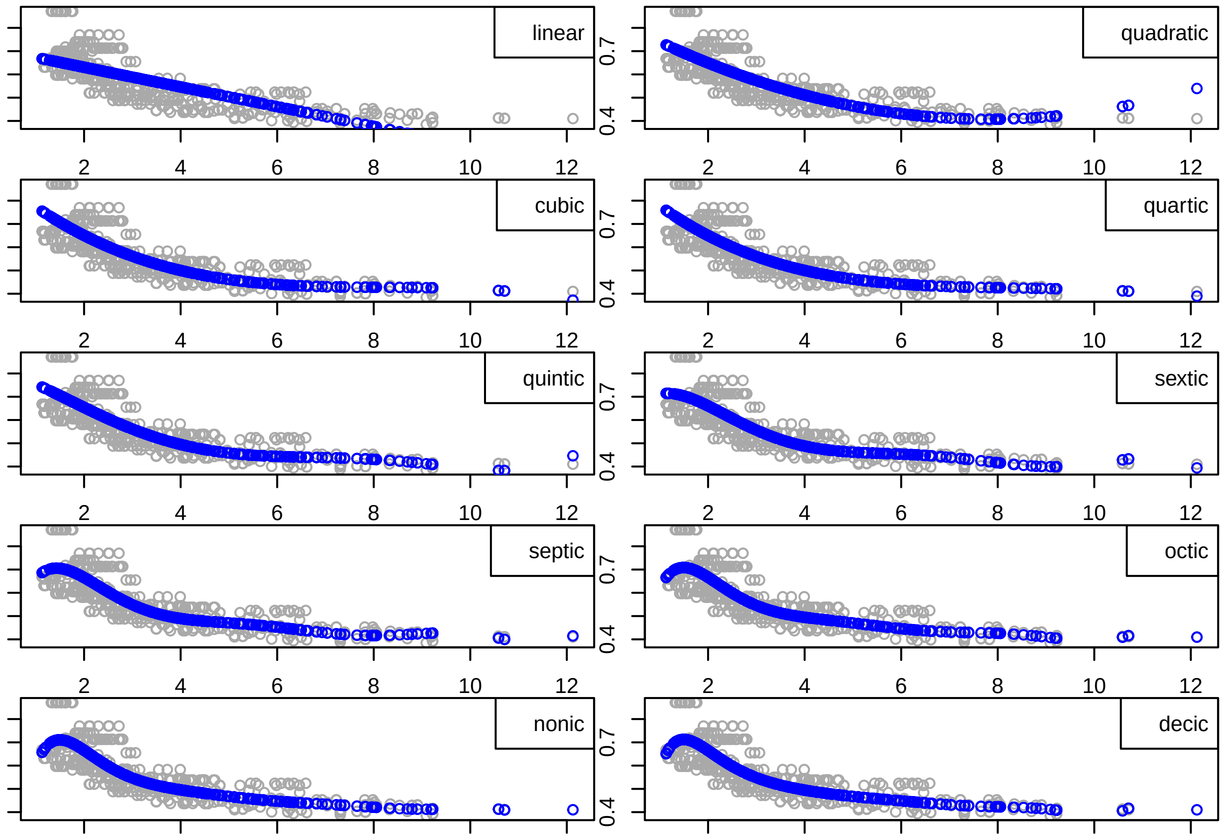
a

```r
cubic_model = lm(nox ~ poly(dis, 3), data = Boston)
plot(Boston$dis, Boston$nox, col = 'darkgrey')
points(cubic_model$fitted.values, type = 'l', col = 'blue')
```

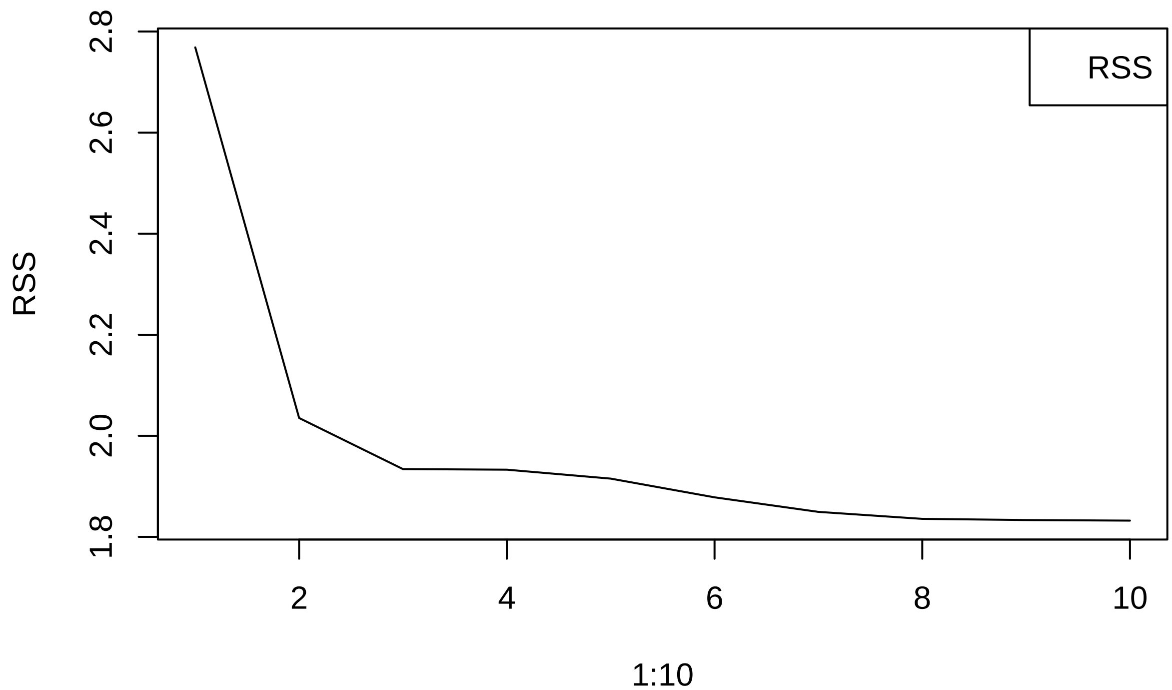


b

```
par(mfrow = c(5, 2))
par(mar = c(1, 1, 1, 1))
polynomials = c('linear', 'quadratic', 'cubic', 'quartic', 'quintic',
                'sextic', 'septic', 'octic', 'nonic', 'decic')
RSS = rep(0, 10)

for (i in 1:10){
  model = lm(nox ~ poly(dis, i), data = Boston)
  plot(Boston$dis, Boston$nox, col = 'darkgrey')
  points(Boston$dis, model$fitted.values, col = 'blue')
  legend('topright', legend = polynomials[i])
  RSS[i] = sum(model$residuals^2)
}
```

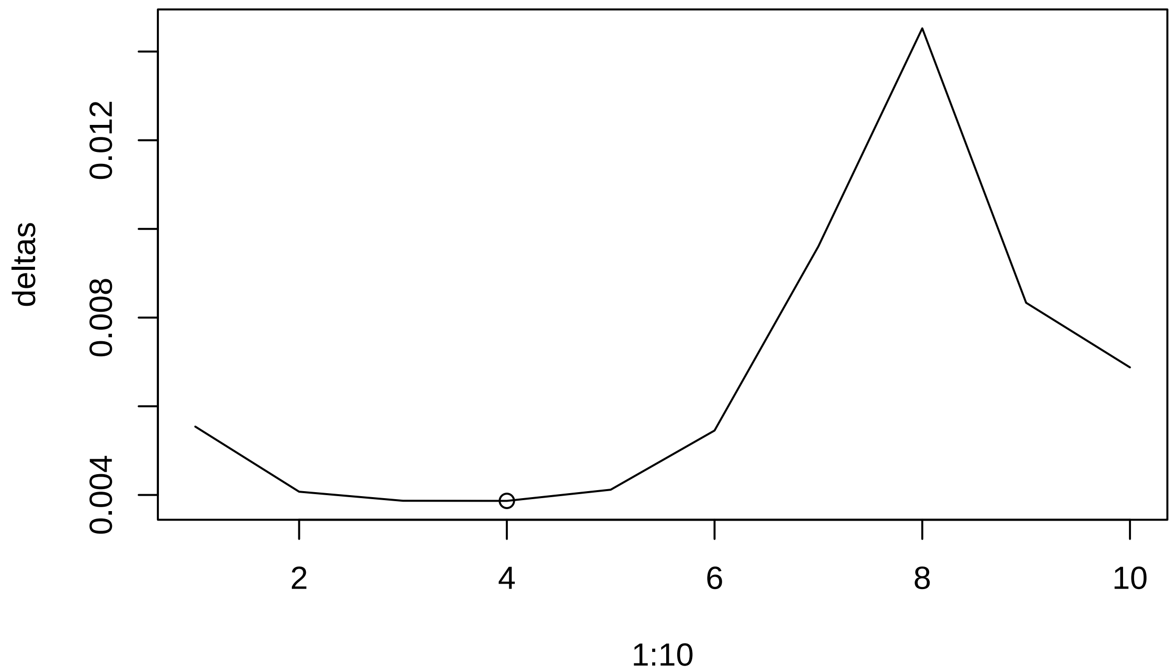



```
plot(1:10, RSS, type = 'l')
legend('topright', legend = c('RSS'))
```



c

```
deltas = rep(0, 10)

for (i in 1:10){
  model = glm(nox ~ poly(dis, i), data = Boston)
  delta = cv.glm(Boston, model, K = 10)$delta[1]
  deltas[i] = delta
}

plot(1:10, deltas, type = 'l')
points(which.min(deltas), deltas[which.min(deltas)])
```



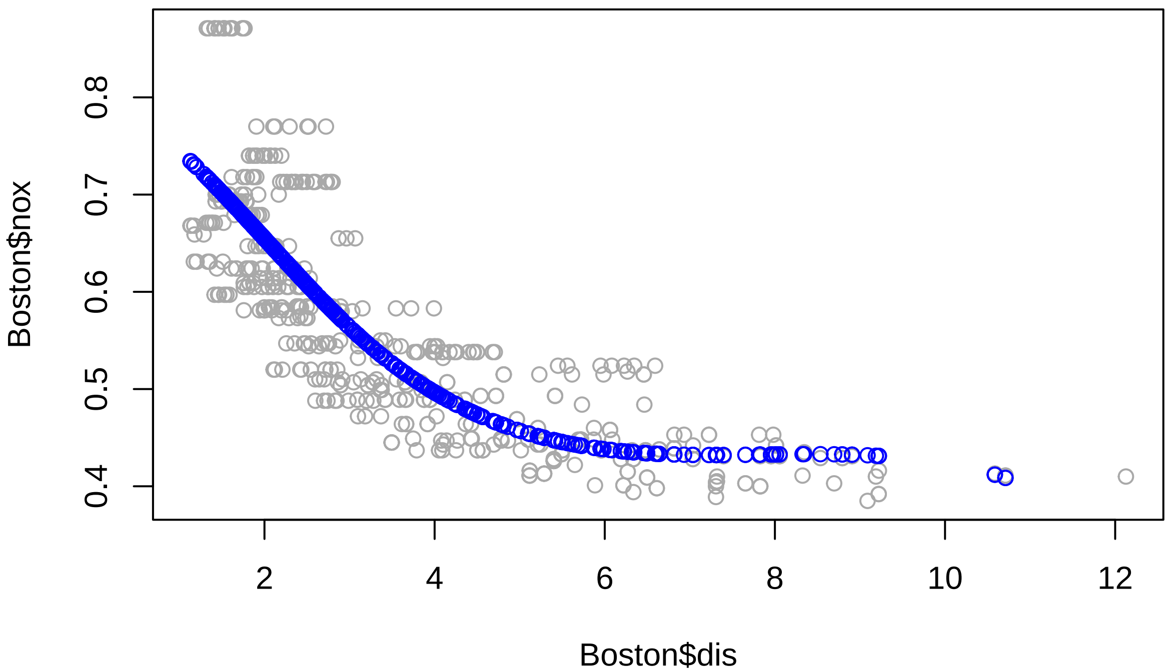
The optimal degree is four and this suggests that there is a non-linear relationship between the response the the predictor. From (b), we see that a decic model overfits the data. You might notice that a nonic model is also good enough but it is not the parsimonious model.

d

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

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

```
spline_model = lm(nox ~ bs(dis, df = 4), data = Boston)
spline_preds = predict(spline_model)

plot(Boston$dis, Boston$nox, col = 'darkgrey')
points(Boston$dis, spline_preds, col = 'blue')
```



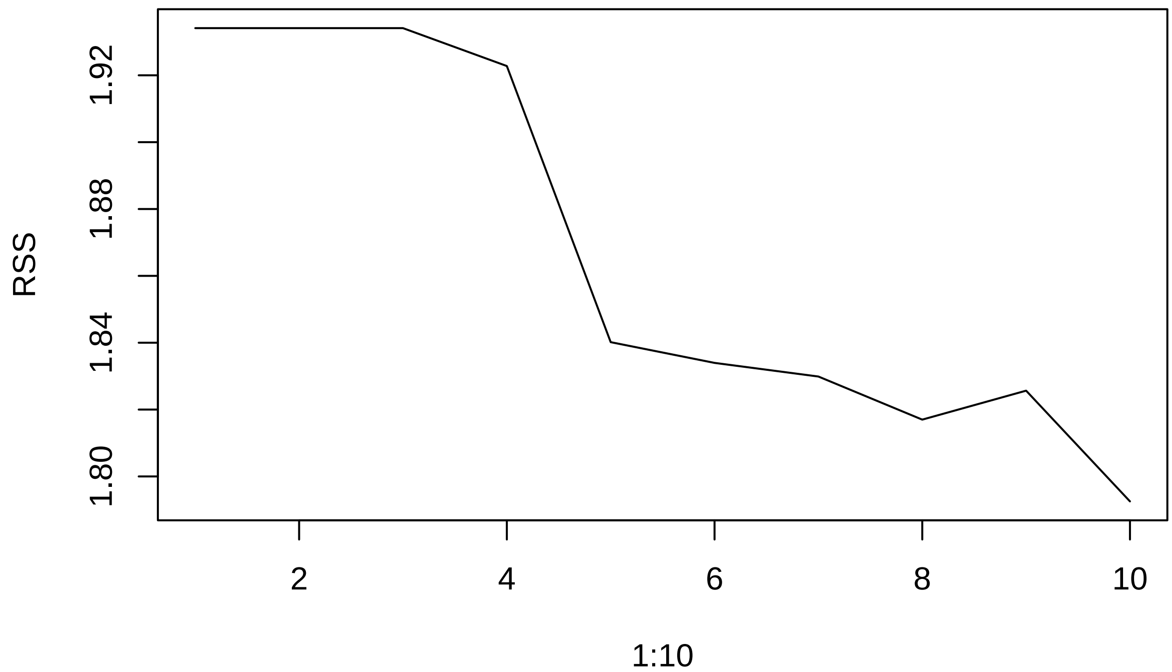
e

Using the whole dataset and RSS as a metric, 10 degrees of freedom turns out to be the best number. This is predictable since our model tried to overfit the data.

```
RSS = rep(NA, 10)

for (i in 1:10){
  model = lm(nox ~ bs(dis, df = i), data = Boston)
  RSS[i] = sum(model$residuals^2)
}
```

```
## Warning in bs(dis, df = i): 'df' was too small; have used 3

## Warning in bs(dis, df = i): 'df' was too small; have used 3
```

```
plot(1:10, RSS, type = 'l')
```



f

CV

```
set.seed(1)
deltas = rep(NA, 16)

for (i in 3:18){
  model = glm(nox ~ bs(dis, df = i), data = Boston)
  cv_model = cv.glm(data = Boston, model, K = 10)
  deltas[i-2] = cv_model$delta[1]
}
```

```
## Warning in bs(dis, degree = 3L, knots = numeric(0), Boundary.knots = c(1.137, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = numeric(0), Boundary.knots = c(1.137, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = numeric(0), Boundary.knots = c(1.1296, :
```

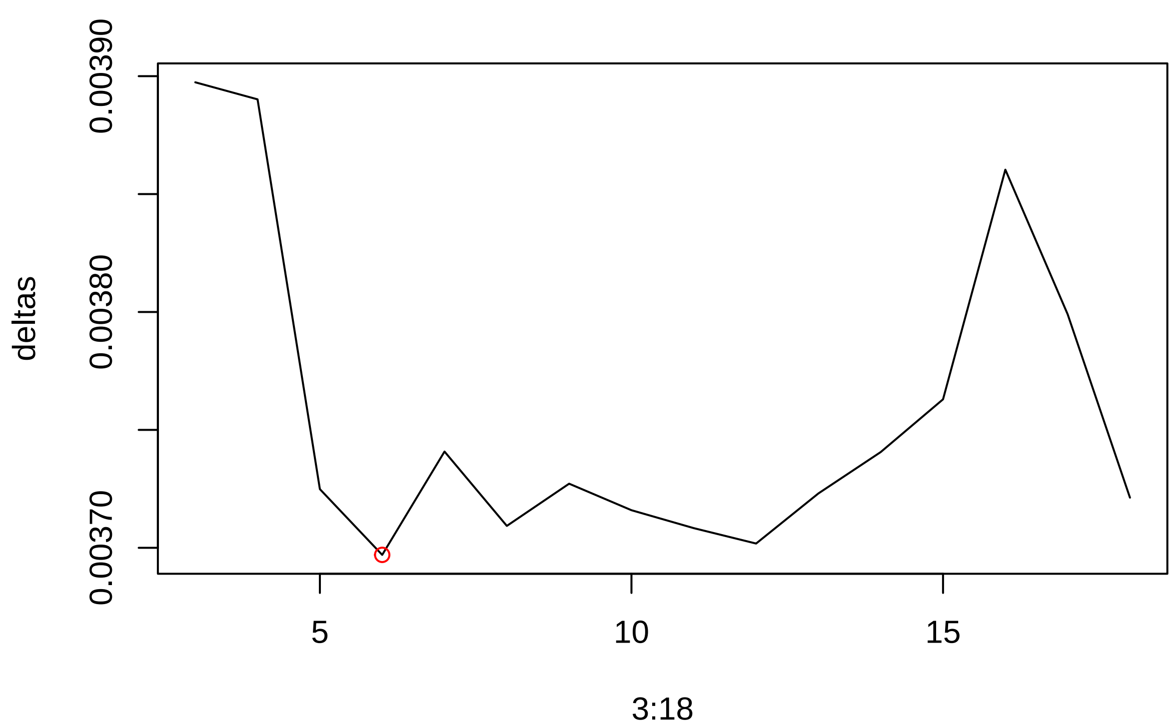
```
## some ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = numeric(0), Boundary.knots = c(1.1296, :
## some ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘50%‘ = 3.0993), Boundary.knots =
## c(1.137, : some ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘50%‘ = 3.0993), Boundary.knots =
## c(1.137, : some ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘50%‘ = 3.3603), Boundary.knots =
## c(1.1296, : some ’x’ values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = c(‘50%‘ = 3.3603), Boundary.knots =
## c(1.1296, : some ’x’ values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = c(‘33.33333%‘ = 2.38876666666667, : some
## ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘33.33333%‘ = 2.38876666666667, : some
## ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘33.33333%‘ = 2.3088, ‘66.66667%‘
## = 4.09726666666667: some ’x’ values beyond boundary knots may cause
## ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘33.33333%‘ = 2.3088, ‘66.66667%‘
## = 4.09726666666667: some ’x’ values beyond boundary knots may cause
## ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘25%‘ = 2.087875, ‘50%‘ = 3.19095, :
## some ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘25%‘ = 2.087875, ‘50%‘ = 3.19095, :
## some ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘20%‘ = 1.92404, ‘40%‘ = 2.55946, :
## some ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘20%‘ = 1.92404, ‘40%‘ = 2.55946, :
## some ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘20%‘ = 1.94984, ‘40%‘ = 2.59774, :
## some ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘20%‘ = 1.94984, ‘40%‘ = 2.59774, :
## some ’x’ values beyond boundary knots may cause ill-conditioned bases
```

```
## Warning in bs(dis, degree = 3L, knots = c(`16.66667%` = 1.86636666666667, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`16.66667%` = 1.86636666666667, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`16.66667%` = 1.82085, `33.33333%`
## = 2.36386666666667, : some 'x' values beyond boundary knots may cause
## ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`16.66667%` = 1.82085, `33.33333%`
## = 2.36386666666667, : some 'x' values beyond boundary knots may cause
## ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`14.28571%` = 1.79078571428571, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`14.28571%` = 1.79078571428571, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`14.28571%` = 1.7912, `28.57143%` =
## 2.1705, : some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`14.28571%` = 1.7912, `28.57143%` =
## 2.1705, : some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`12.5%` = 1.757275, `25%` = 2.1084, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`12.5%` = 1.757275, `25%` = 2.1084, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`12.5%` = 1.76375, `25%` = 2.10525, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`12.5%` = 1.76375, `25%` = 2.10525, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`11.11111%` = 1.69124444444444, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`11.11111%` = 1.69124444444444, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`11.11111%` = 1.71297777777778, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`11.11111%` = 1.71297777777778, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`10%` = 1.66236, `20%` = 1.98518, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases
```

```
## Warning in bs(dis, degree = 3L, knots = c(`10%` = 1.66236, `20%` = 1.98518, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`10%` = 1.6624, `20%` = 1.9769, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`10%` = 1.6624, `20%` = 1.9769, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`9.090909%` = 1.59007272727273, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`9.090909%` = 1.59007272727273, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`9.090909%` = 1.61941818181818, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`9.090909%` = 1.61941818181818, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`8.333333%` = 1.58948333333333, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`8.333333%` = 1.58948333333333, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`8.333333%` = 1.5874, `16.66667%` =
## 1.8651, : some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`8.333333%` = 1.5874, `16.66667%` =
## 1.8651, : some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`7.692308%` = 1.57254615384615, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`7.692308%` = 1.57254615384615, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`7.692308%` = 1.57254615384615, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`7.692308%` = 1.57254615384615, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`7.142857%` = 1.53135, `14.28571%` =
## 1.7821, : some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`7.142857%` = 1.53135, `14.28571%` =
## 1.7821, : some 'x' values beyond boundary knots may cause ill-conditioned bases
```

```
## Warning in bs(dis, degree = 3L, knots = c(`7.142857%` = 1.54498571428571, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`7.142857%` = 1.54498571428571, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`6.666667%` = 1.52093333333333, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`6.666667%` = 1.52093333333333, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`6.666667%` = 1.5218, `13.33333%` =
## 1.75478, : some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`6.666667%` = 1.5218, `13.33333%` =
## 1.75478, : some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`6.25%` = 1.5187, `12.5%` = 1.74615, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`6.25%` = 1.5187, `12.5%` = 1.74615, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases
```

```r
plot(3:18, deltas, type = 'l')
points(which.min(deltas) + 2, deltas[which.min(deltas)], col = 'red')
```



ANOVA

```r
fit1 = lm(nox ~ bs(dis, 3), data = Boston)
fit2 = lm(nox ~ bs(dis, 4), data = Boston)
fit3 = lm(nox ~ bs(dis, 5), data = Boston)
fit4 = lm(nox ~ bs(dis, 6), data = Boston)
fit5 = lm(nox ~ bs(dis, 7), data = Boston)
fit6 = lm(nox ~ bs(dis, 8), data = Boston)
fit7 = lm(nox ~ bs(dis, 9), data = Boston)
fit8 = lm(nox ~ bs(dis, 10), data = Boston)
fit9 = lm(nox ~ bs(dis, 11), data = Boston)
fit10 = lm(nox ~ bs(dis, 12), data = Boston)
fit11 = lm(nox ~ bs(dis, 13), data = Boston)
fit12 = lm(nox ~ bs(dis, 14), data = Boston)
fit13 = lm(nox ~ bs(dis, 15), data = Boston)
fit14 = lm(nox ~ bs(dis, 16), data = Boston)
fit15 = lm(nox ~ bs(dis, 17), data = Boston)
fit16 = lm(nox ~ bs(dis, 18), data = Boston)

anova(fit1, fit2, fit3, fit4, fit5, fit6, fit7, fit8, fit9,
      fit10,fit11, fit12, fit13, fit14, fit15, fit16)
```

```
## Analysis of Variance Table
##
## Model  1: nox ~ bs(dis, 3)
## Model  2: nox ~ bs(dis, 4)
## Model  3: nox ~ bs(dis, 5)
## Model  4: nox ~ bs(dis, 6)
## Model  5: nox ~ bs(dis, 7)
## Model  6: nox ~ bs(dis, 8)
## Model  7: nox ~ bs(dis, 9)
## Model  8: nox ~ bs(dis, 10)
## Model  9: nox ~ bs(dis, 11)
## Model 10: nox ~ bs(dis, 12)
## Model 11: nox ~ bs(dis, 13)
## Model 12: nox ~ bs(dis, 14)
## Model 13: nox ~ bs(dis, 15)
## Model 14: nox ~ bs(dis, 16)
## Model 15: nox ~ bs(dis, 17)
## Model 16: nox ~ bs(dis, 18)
##    Res.Df    RSS Df Sum of Sq       F    Pr(>F)
## 1     502 1.9341
## 2     501 1.9228  1  0.011332  3.1076  0.078556 .
## 3     500 1.8402  1  0.082602 22.6525 2.563e-06 ***
## 4     499 1.8340  1  0.006207  1.7022  0.192622
## 5     498 1.8299  1  0.004081  1.1193  0.290597
## 6     497 1.8170  1  0.012889  3.5347  0.060692 .
## 7     496 1.8256  1 -0.008657
## 8     495 1.7925  1  0.033118  9.0821  0.002716 **
## 9     494 1.7970  1 -0.004457
## 10    493 1.7890  1  0.007993  2.1919  0.139386
## 11    492 1.7824  1  0.006649  1.8233  0.177546
## 12    491 1.7818  1  0.000512  0.1405  0.707937
```

```
## 13    490 1.7828  1 -0.000960
## 14    489 1.7835  1 -0.000748
## 15    488 1.7798  1  0.003757  1.0303  0.310585
## 16    487 1.7758  1  0.003950  1.0833  0.298478
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In general, using cross-validation or ANOVAm the results from polynomial regression and spline regression seem to agree with each other. An optimal degree varies from 4 to 6 is likely to yield a parsimonious model.

Exercise 10

```
head(College)
```

```
##                              Private Apps Accept Enroll Top10perc Top25perc
## Abilene Christian University     Yes 1660   1232    721        23        52
## Adelphi University               Yes 2186   1924    512        16        29
## Adrian College                   Yes 1428   1097    336        22        50
## Agnes Scott College              Yes  417    349    137        60        89
## Alaska Pacific University        Yes  193    146     55        16        44
## Albertson College                Yes  587    479    158        38        62
##                              F.Undergrad P.Undergrad Outstate Room.Board Books
## Abilene Christian University        2885         537     7440       3300   450
## Adelphi University                  2683        1227    12280       6450   750
## Adrian College                      1036          99    11250       3750   400
## Agnes Scott College                  510          63    12960       5450   450
## Alaska Pacific University            249         869     7560       4120   800
## Albertson College                    678          41    13500       3335   500
##                              Personal PhD Terminal S.F.Ratio perc.alumni Expend
## Abilene Christian University     2200  70       78      18.1          12   7041
## Adelphi University               1500  29       30      12.2          16  10527
## Adrian College                   1165  53       66      12.9          30   8735
## Agnes Scott College               875  92       97       7.7          37  19016
## Alaska Pacific University        1500  76       72      11.9           2  10922
## Albertson College                 675  67       73       9.4          11   9727
##                              Grad.Rate
## Abilene Christian University        60
## Adelphi University                  56
## Adrian College                      54
## Agnes Scott College                 59
## Alaska Pacific University           15
## Albertson College                   55
```

```
dim(na.omit(College))
```

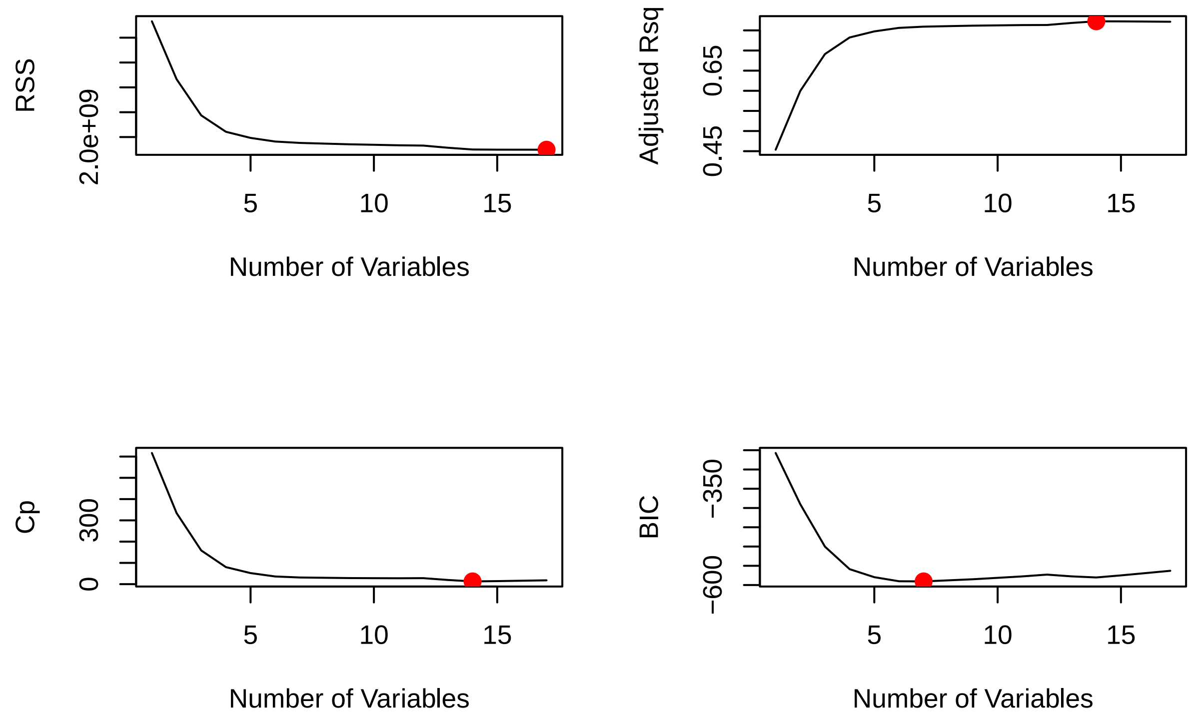
```
## [1] 777  18
```

a

```r
set.seed(1)
train_indices = sample(777, 444)
train_set = College[train_indices, ]
test_set = College[-train_indices, ]
```

```r
reg_model = regsubsets(Outstate ~ ., data = train_set,
                       nvmax = 17, method = 'forward')
reg_summary = summary(reg_model)
```

```r
par(mfrow = c(2, 2))

plot(reg_summary$rss, xlab = 'Number of Variables', ylab = 'RSS', type = 'l')
rss_min = which.min(reg_summary$rss)
points(rss_min, reg_summary$rss[rss_min], col = 'red', cex = 2, pch = 20)

plot(reg_summary$adjr2, xlab = 'Number of Variables', ylab = 'Adjusted Rsq', type = 'l')
adjr2_max = which.max(reg_summary$adjr2)
points(adjr2_max, reg_summary$adjr2[adjr2_max], col = 'red', cex = 2, pch = 20)

plot(reg_summary$cp, xlab = 'Number of Variables', ylab = 'Cp', type = 'l')
cp_min = which.min(reg_summary$cp)
points(cp_min, reg_summary$cp[cp_min], col = 'red', cex = 2, pch = 20)

plot(reg_summary$bic, xlab = 'Number of Variables', ylab = 'BIC', type = 'l')
bic_min = which.min(reg_summary$bic)
points(bic_min, reg_summary$bic[bic_min], col = 'red', cex = 2, pch = 20)
```



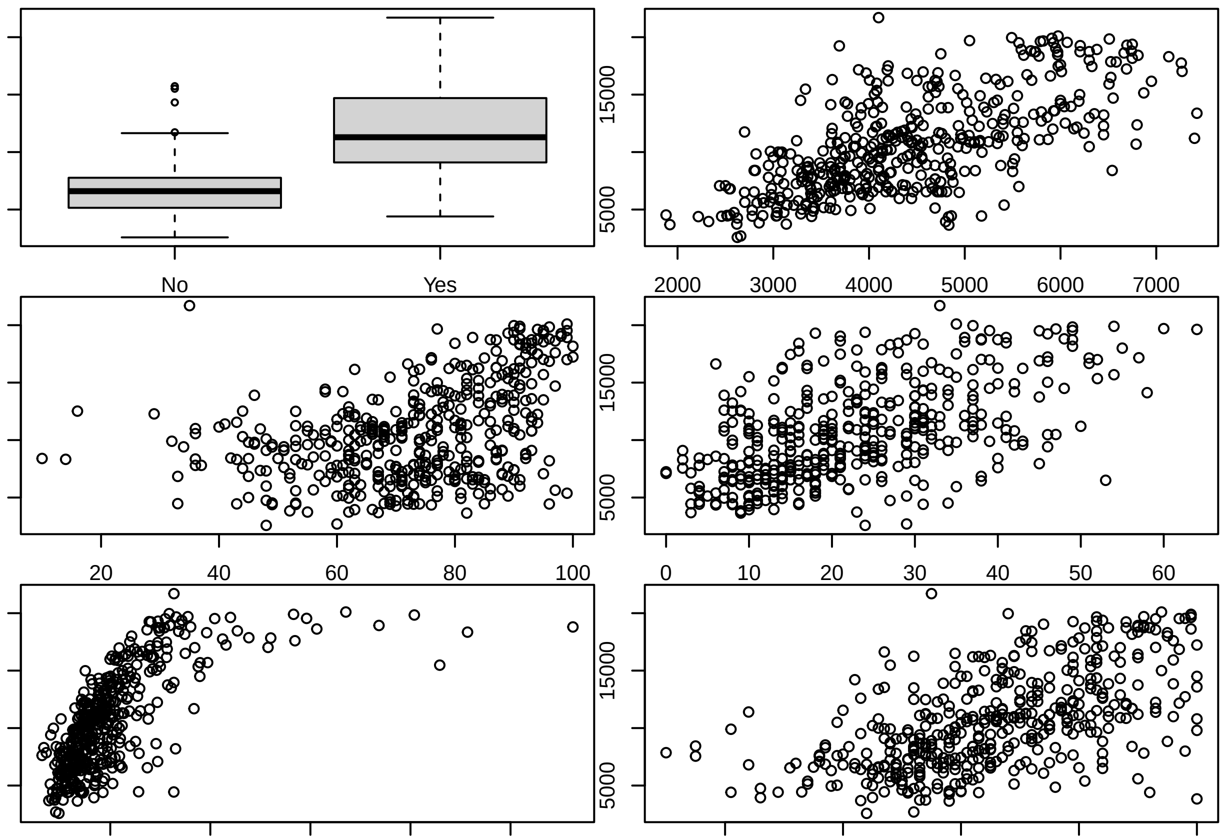
Here we would choose 6 as the optimal number of variables. It is understandable that using 15 or even all 17 variables results in small RSS or Adjusted R-squared since the model overfits the data. As you might notice, there is no significant difference in using from 6 to 17 variables. We want a parsimonious model.

```
coef(reg_model, 6)
```

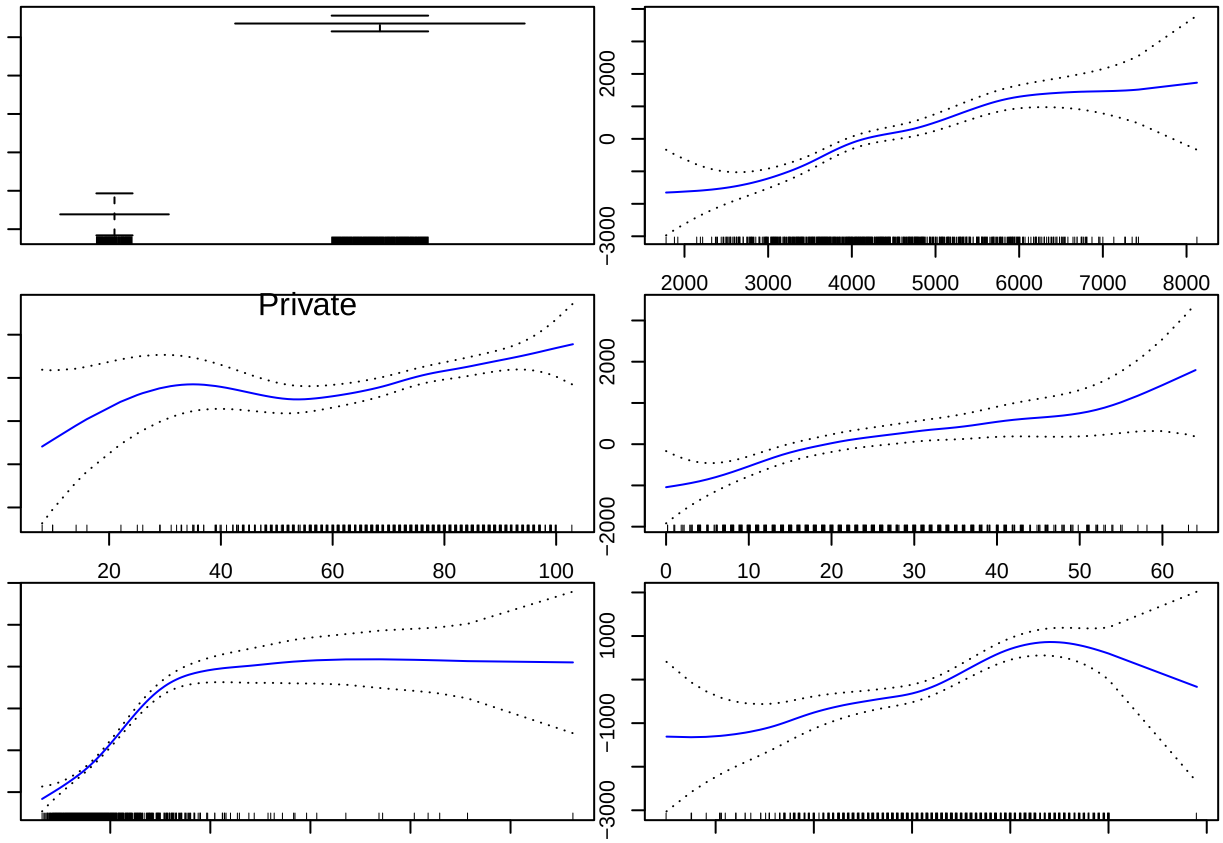
```
##   (Intercept)    PrivateYes    Room.Board           PhD   perc.alumni
## -4065.9665184  2788.5353954     1.0799406    35.5263418    57.7080695
##        Expend     Grad.Rate
##     0.2013602    29.5272517
```

b

```
par(mfrow = c(3, 2))
par(mar = c(1, 1, 1, 1))
plot(train_set$Private, train_set$Outstate)
plot(train_set$Room.Board, train_set$Outstate)
plot(train_set$PhD, train_set$Outstate)
plot(train_set$perc.alumni, train_set$Outstate)
plot(train_set$Expend, train_set$Outstate)
plot(train_set$Grad.Rate, train_set$Outstate)
```




```r
gam_model = gam(Outstate ~ Private + s(Room.Board, 5) + s(PhD, 5)
                + s(perc.alumni, 5) + s(Expend, 5) + s(Grad.Rate, 5),
                data = College)

par(mfrow = c(3, 2))
par(mar = c(1, 1, 1, 1))
plot(gam_model, se = T, col = 'blue')
```



c

```
linear_model = gam(Outstate ~ Private + Room.Board + PhD + perc.alumni
                   + Expend + Grad.Rate, data = College)
linear_preds = predict(linear_model, newdata = test_set)

err = mean((linear_preds - test_set$Outstate)^2)
tss = mean((mean(test_set$Outstate) - test_set$Outstate)^2)
1 - err / tss
```

```
## [1] 0.7268812
```

```
err
```

```
## [1] 3761153
```

```
gam_preds = predict(gam_model, newdata = test_set)
err = mean((gam_preds - test_set$Outstate)^2)
1 - err / tss
```

```
## [1] 0.7753438
```

```
err
```

```
## [1] 3093770
```

d

The plots and results from (c) suggest that there is a non-linear relationship between some predictors and the response. Specifically, they are Room.Board, PhD, Expend and Grad.Rate

```
summary(gam_model)
```

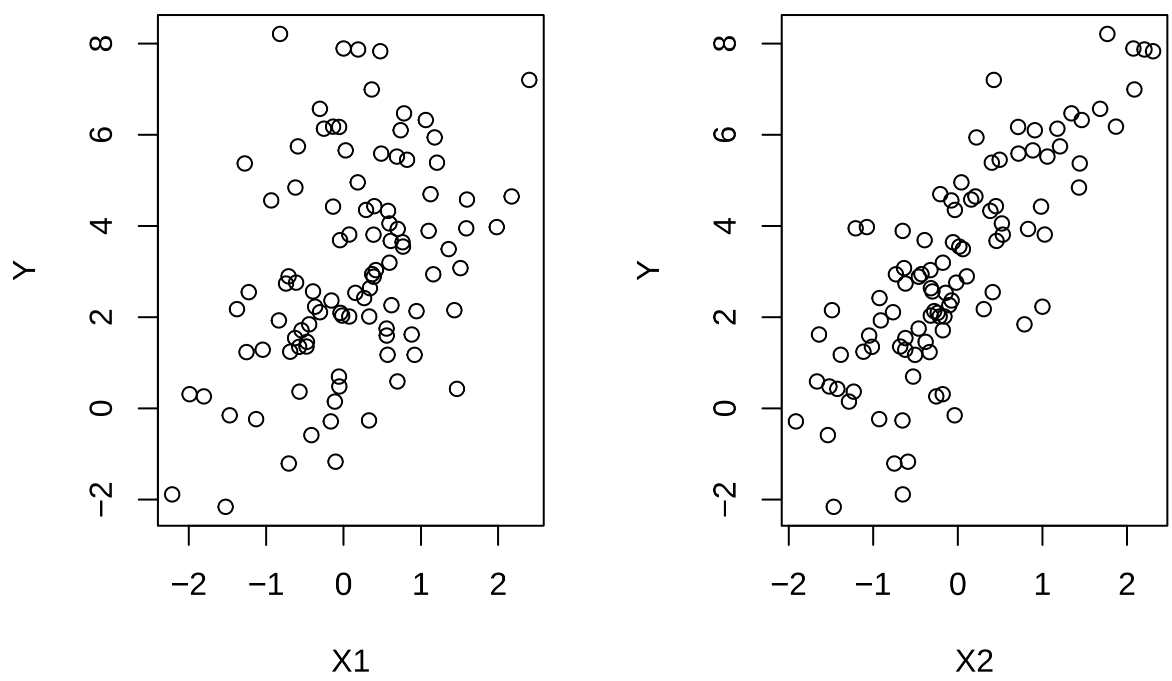
```
##
## Call: gam(formula = Outstate ~ Private + s(Room.Board, 5) + s(PhD,
##     5) + s(perc.alumni, 5) + s(Expend, 5) + s(Grad.Rate, 5),
##     data = College)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -7576.49 -1102.48    47.67  1287.40  7492.43
##
## (Dispersion Parameter for gaussian family taken to be 3425681)
##
##     Null Deviance: 12559297426 on 776 degrees of freedom
## Residual Deviance: 2569258985 on 749.9994 degrees of freedom
## AIC: 13924.92
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##                    Df     Sum Sq    Mean Sq F value    Pr(>F)
## Private             1 3370516566 3370516566 983.897 < 2.2e-16 ***
## s(Room.Board, 5)    1 2484278051 2484278051 725.192 < 2.2e-16 ***
## s(PhD, 5)           1  818616768  818616768 238.965 < 2.2e-16 ***
## s(perc.alumni, 5)   1  494085299  494085299 144.230 < 2.2e-16 ***
## s(Expend, 5)        1 1015946099 1015946099 296.568 < 2.2e-16 ***
## s(Grad.Rate, 5)     1  148600665  148600665  43.378 8.486e-11 ***
## Residuals         750 2569258985    3425681
## ---
## 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, 5)        4  2.547 0.03824 *
## s(PhD, 5)               4  2.083 0.08126 .
## s(perc.alumni, 5)       4  1.144 0.33444
## s(Expend, 5)            4 32.545 < 2e-16 ***
## s(Grad.Rate, 5)         4  2.670 0.03121 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Exercise 11

a

```
set.seed(1)

X1 = rnorm(100)
X2 = rnorm(100)
epsilon = rnorm(100)
Y = 1*X1 + 2*X2 + 3 + epsilon
```

```
par(mfrow = c(1, 2))
plot(X1, Y)
plot(X2, Y)
```



b

```
B1_hat = 100
```

c

```r
a = Y - B1_hat * X1
B2_hat = lm(a ~ X2)$coef[2]
B2_hat
```

```
##       X2
## 2.038818
```

d

```r
a = Y - B2_hat * X2
B1_hat = lm(a ~ X1)$coef[2]
B1_hat
```

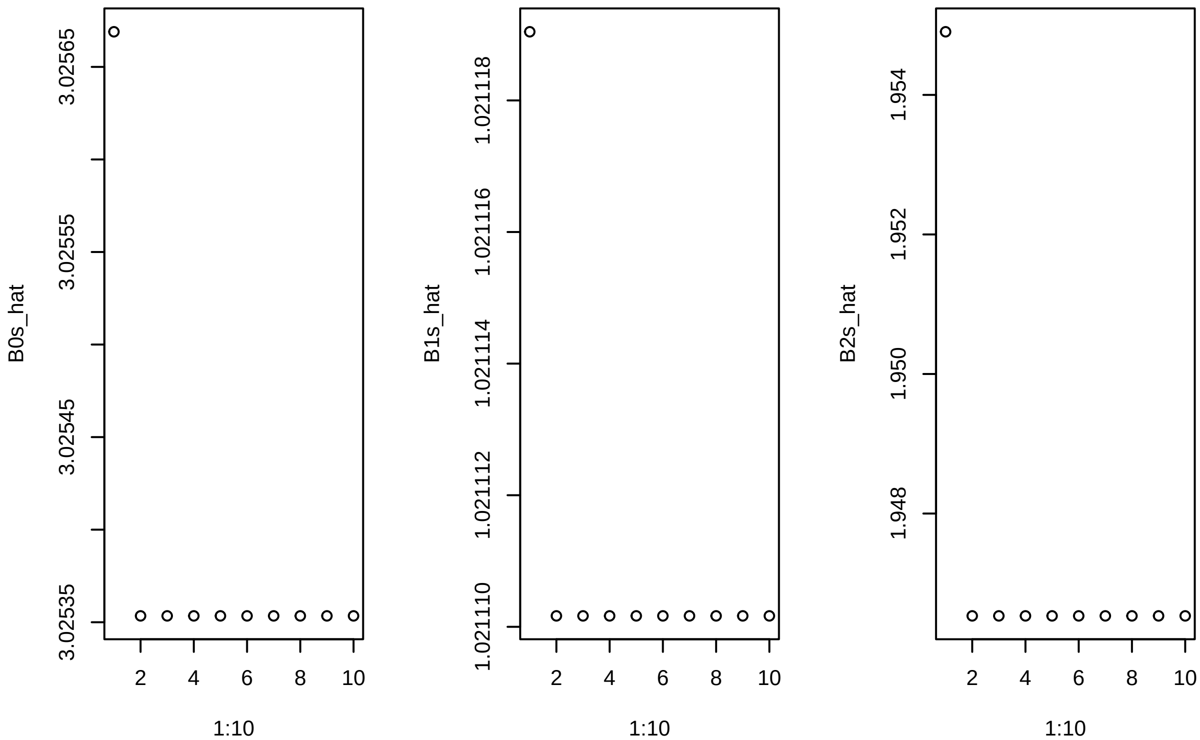
```
##       X1
## 1.021208
```

e

Here I'll only iterate the process for 10 times since we got very close to the actual coefficients just after 2 tries.

```r
B1_hat = 10

B0s_hat = rep(NA, 10)
B1s_hat = rep(NA, 10)
B2s_hat = rep(NA, 10)

for (i in 1:10){

  a = Y - B1_hat * X1
  B2_hat = lm(a ~ X2)$coef[2]
  B2s_hat[i] = B2_hat

  a = Y - B2_hat * X2
  model = lm(a ~ X1)
  B1_hat = model$coef[2]
  B1s_hat[i] = B1_hat

  B0s_hat[i] = model$coef[1]

}
```

```r
par(mfrow = c(1, 3))
plot(1:10, B0s_hat)
plot(1:10, B1s_hat)
plot(1:10, B2s_hat)
```

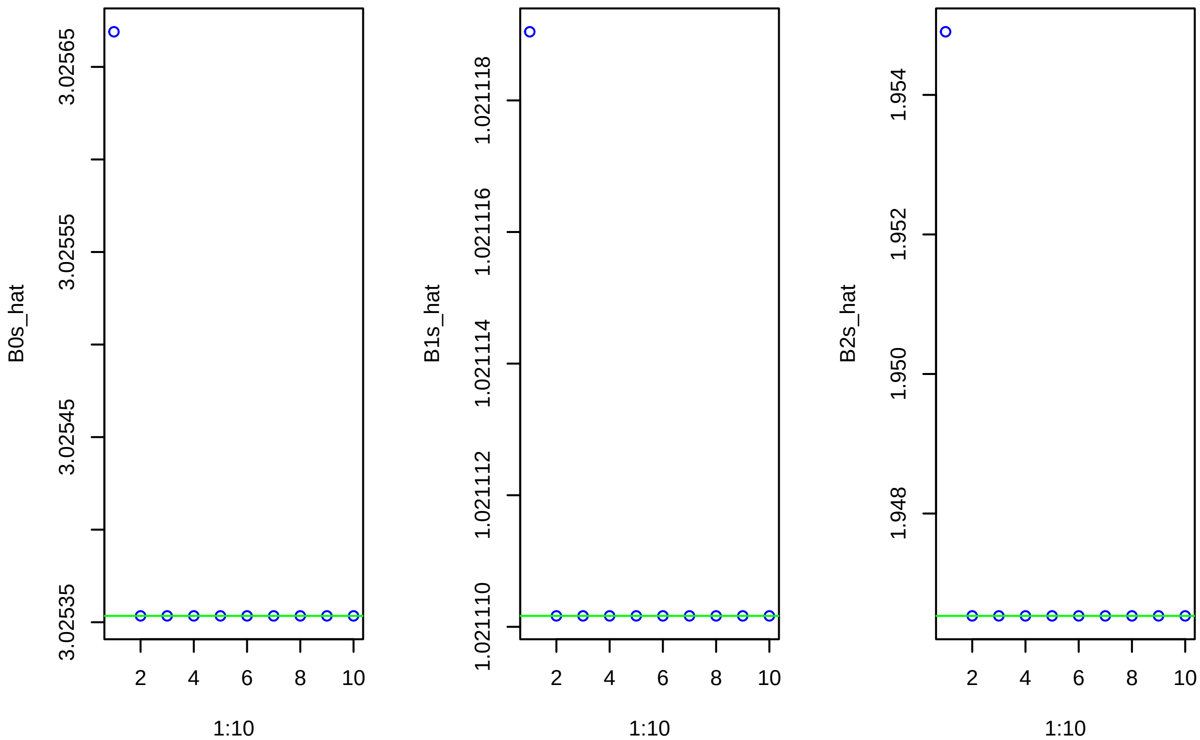


f

```
multi_model = lm(Y ~ X1 + X2)
coeff = multi_model$coef
coeff
```

```
## (Intercept)          X1          X2
##    3.025353    1.021110    1.946533
```

```
par(mfrow = c(1, 3))

plot(1:10, B0s_hat, col = 'blue')
abline(h = coeff[1], col = 'green')
plot(1:10, B1s_hat, col = 'blue')
abline(h = coeff[2], col = 'green')
plot(1:10, B2s_hat, col = 'blue')
abline(h = coeff[3], col = 'green')
```



g

We only need a single backfitting iteration to obtain a "good" approximation to the multiple regression coefficient estimates.

Exercise 12

```
set.seed(1)

B = c(1:100)
intercept = 1
X = matrix(rnorm(1000 * 100), nrow = 1000, ncol = 100)
e = rnorm(1000)

Y = intercept + X %*% B + e
```

```
set.seed(1)

B_hat = rnorm(100)
iters = 30
B_his = matrix(NA, nrow = iters, ncol = 101)
```

```r
for (i in 1:iters) {

  for (p in 1:100) {

    a = Y - X[ ,-p] %*% B_hat[-p]
    coeff = lm(a ~ X[ ,p])$coef
    B_hat[p]  = coeff[2]

    B_his[i, p+1] = B_hat[p]
  }

  beta0 = coeff[1]
  B_his[i, 1] = beta0
}


B_hat[1:5]
```

```
## [1] 0.9727505 2.0606211 2.9936625 3.9582157 4.9714572
```

```r
B[1:5]
```

```
## [1] 1 2 3 4 5
```

```r
B_his[1:20, 1:6]
```

```
##             [,1]       [,2]     [,3]      [,4]       [,5]       [,6]
##  [1,] -0.9928798 -2.1347165 1.080987  1.800493 -11.531064 -29.444173
##  [2,]  0.7535299  5.2524427 4.971155 -5.542070   6.152879   1.103353
##  [3,]  0.8613899  2.0157122 2.894328  2.137096   2.890126   3.784716
##  [4,]  0.9998247  1.0441617 1.934883  3.017297   3.712200   4.802683
##  [5,]  1.0189743  0.9563286 1.975577  3.061277   3.911303   4.935899
##  [6,]  1.0201238  0.9620477 2.021010  3.024469   3.945772   4.965000
##  [7,]  1.0191109  0.9694400 2.046744  3.003595   3.954409   4.970677
##  [8,]  1.0185902  0.9718723 2.056407  2.996406   3.957068   4.971361
##  [9,]  1.0184064  0.9725178 2.059460  2.994357   3.957918   4.971422
## [10,]  1.0183484  0.9726867 2.060324  2.993826   3.958154   4.971439
## [11,]  1.0183311  0.9727330 2.060550  2.993698   3.958208   4.971449
## [12,]  1.0183262  0.9727459 2.060605  2.993669   3.958217   4.971454
## [13,]  1.0183249  0.9727494 2.060618  2.993663   3.958217   4.971456
## [14,]  1.0183245  0.9727502 2.060621  2.993662   3.958216   4.971457
## [15,]  1.0183244  0.9727504 2.060621  2.993662   3.958216   4.971457
## [16,]  1.0183244  0.9727505 2.060621  2.993662   3.958216   4.971457
## [17,]  1.0183244  0.9727505 2.060621  2.993662   3.958216   4.971457
## [18,]  1.0183244  0.9727505 2.060621  2.993663   3.958216   4.971457
## [19,]  1.0183244  0.9727505 2.060621  2.993663   3.958216   4.971457
## [20,]  1.0183244  0.9727505 2.060621  2.993663   3.958216   4.971457
```

The algorithm obtained “good” approximation to coefficients just after around 5 iterations. Cool :)