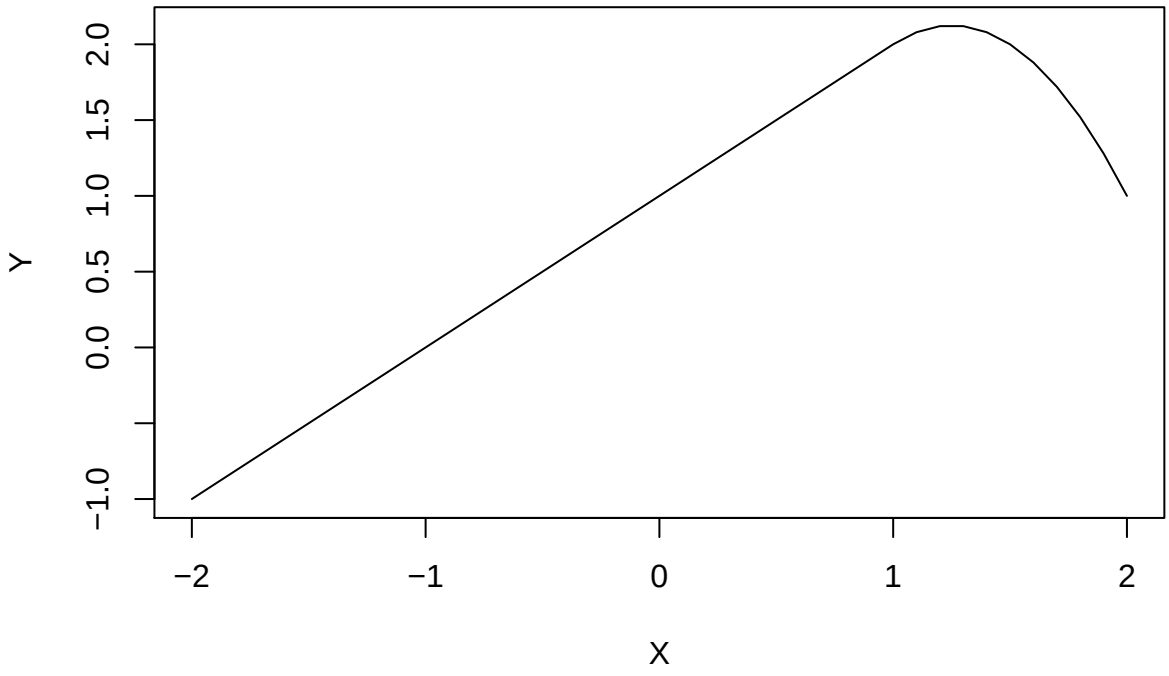
Smoothing HW

Megha Pandit

February 26, 2019

3. Suppose we fit a curve with basis functions b1(X) = X, b2(X) = (X - 1)^2I(X >= 1). (Note that I(X >= 1) equals 1 for X >= 1 and 0 otherwise.) We fit the linear regression model $Y = \beta_0+\beta_1 b_1(X)+\beta_2 b_2(X)+\epsilon$, and obtain coefficient estimates $\hat{\beta}_0 = 1$, $\hat{\beta}_1 = 1$, $\hat{\beta}_2 = -2$. Sketch the estimated curve between X = -2 and X = 2. Note the intercepts, slopes, and other relevant information.

```
X <- seq(-2,2,0.1)
Y = 1 + 1*X - 2*((X - 1)^2)*I(X >= 1)
plot(X, Y, type = "l")
```



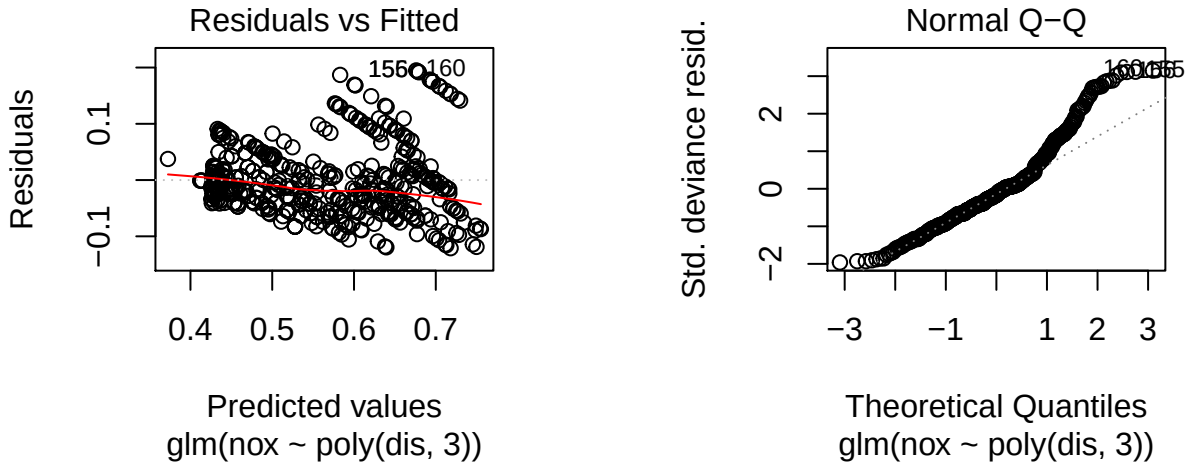
9. This question uses the variables dis (the weighted mean of distances to five Boston employment centers) and nox (nitrogen oxides concentration in parts per 10 million) from the Boston data. We will treat dis as the predictor and nox as the response.

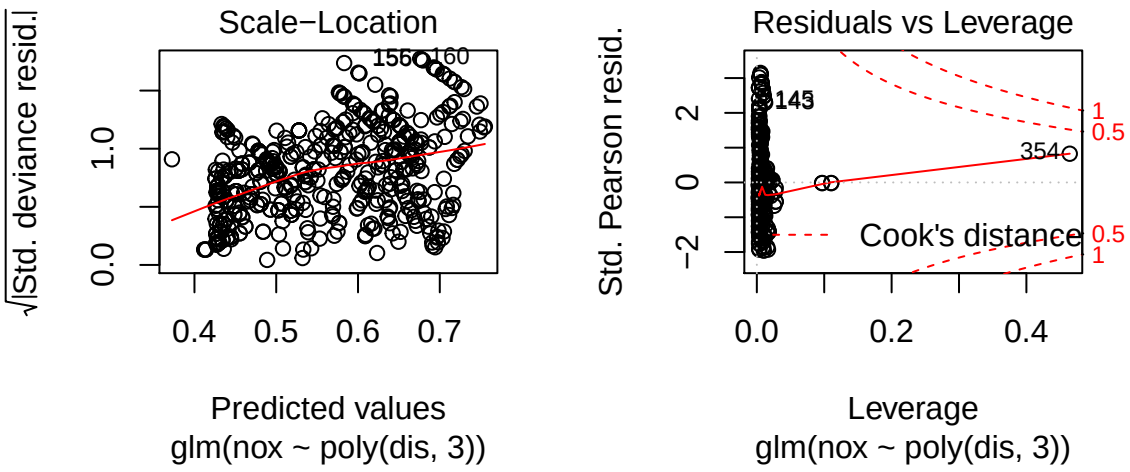
(a) Use the poly() function to fit a cubic polynomial regression to predict nox using dis. Report the regression output, and plot the resulting data and polynomial fits.

```
library(MASS)
data("Boston")

fit1 <- glm(nox ~ poly(dis, 3), data = Boston)
summary(fit1)
```

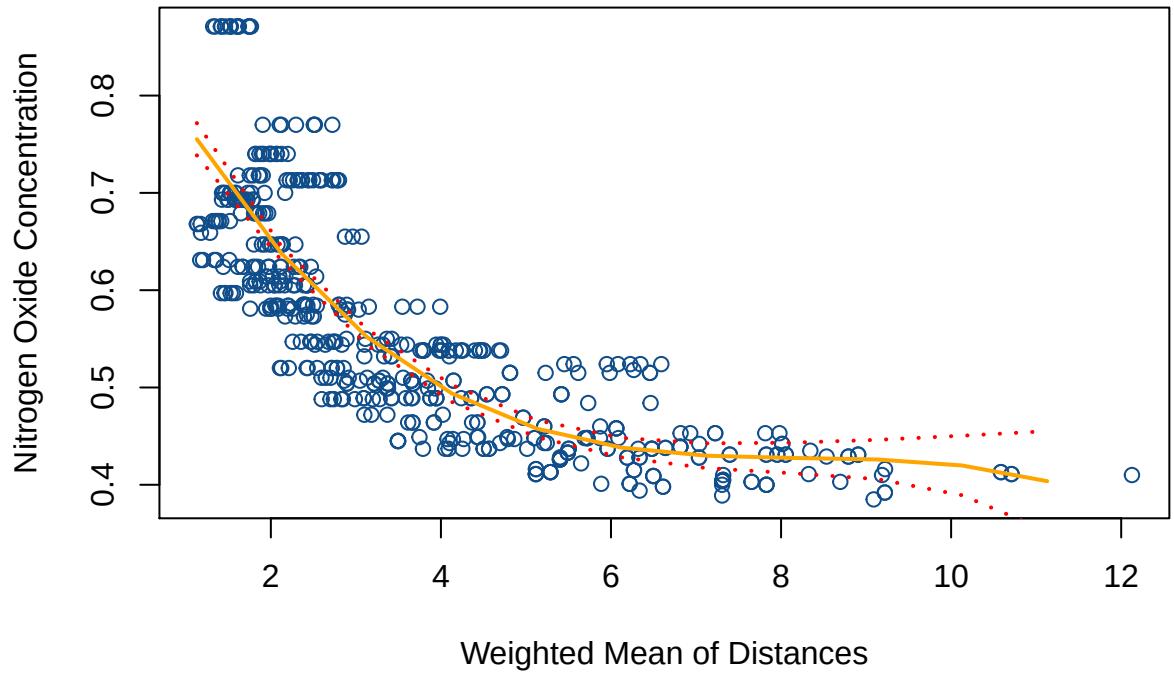
```
##
## Call:
## glm(formula = nox ~ poly(dis, 3), data = Boston)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -0.121130  -0.040619  -0.009738   0.023385   0.194904
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   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

```
plot(fit1)
```



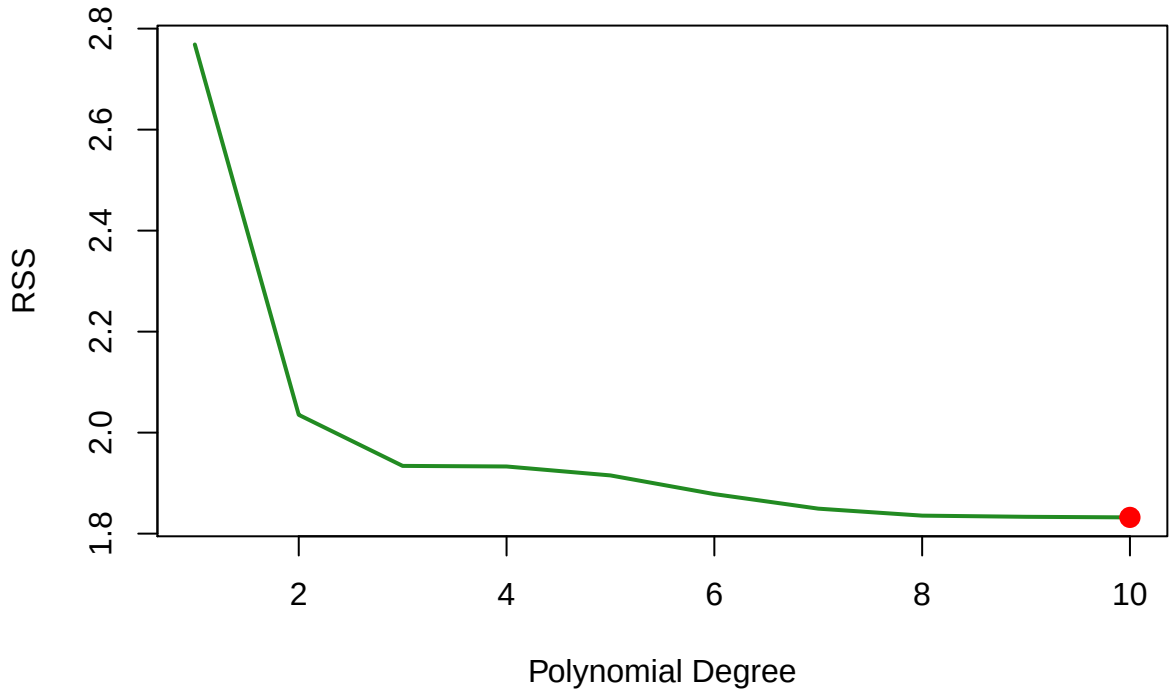



```r
lims_dis <- range(Boston$dis)
grid_dis <- seq(lims_dis[1], lims_dis[2])
pred1 <- predict(fit1, list(dis = grid_dis), se = TRUE)
se_lines <- cbind(pred1$fit + 2*pred1$se.fit, pred1$fit - 2*pred1$se.fit)
plot(Boston$dis, Boston$nox, xlab = "Weighted Mean of Distances", ylab = "Nitrogen Oxide Concentration"
lines(grid_dis, pred1$fit, col = "orange", lwd = 2)
matlines(grid_dis, se_lines, lwd = 2, col = "red", lty = 3)
```



(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.

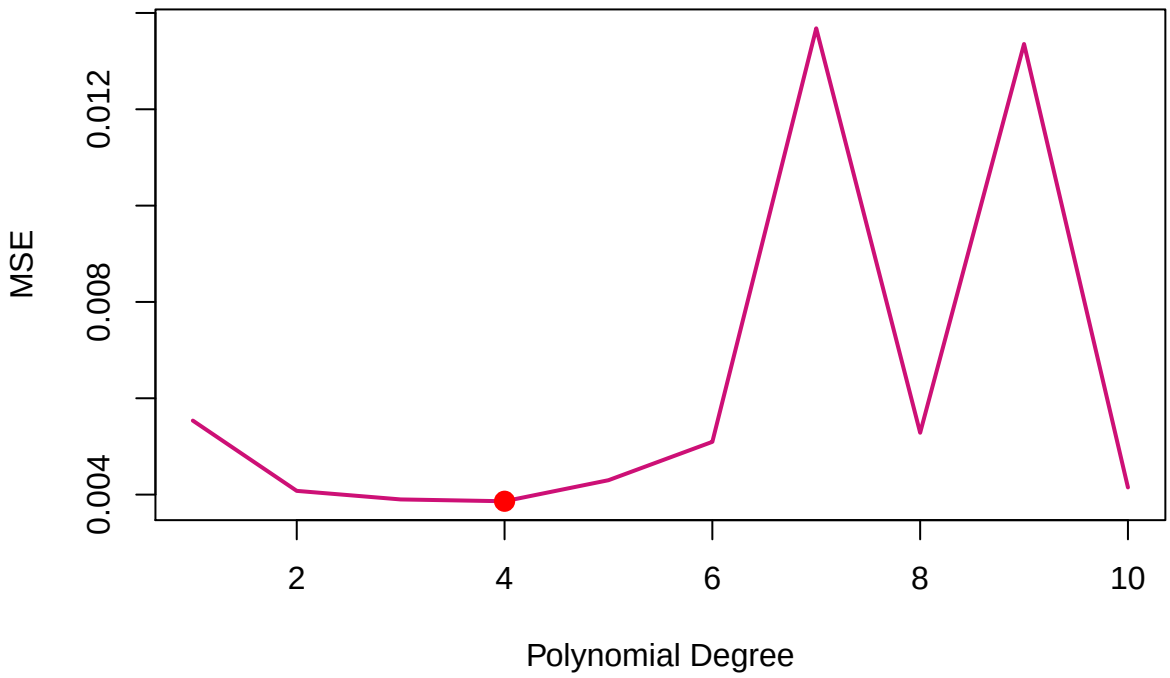
```
set.seed(1)
rss <- rep(NA, 10)
for (i in 1:10){
  fit <- glm(nox ~ poly(dis, i), data = Boston)
  rss[i] <- sum(fit$residuals^2)
}

plot(1:10, rss, xlab = "Polynomial Degree", ylab = "RSS", type = "l", col = "forestgreen", lwd = 2)
points(which.min(rss), rss[which.min(rss)], col='red',pch=20,cex=2)
```



The minimum RSS is for a polynomial degree of 10.

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

```
err <- rep(NA, 10)
for (i in 1:10) {
  fit <- glm(nox ~ poly(dis, i), data = Boston)
  err[i] <- cv.glm(Boston, fit, K = 10)$delta[1]
}
plot(1:10, err, xlab = "Polynomial Degree", ylab = "MSE", type = "l", col = "deeppink3", lwd = 2)
points(which.min(err), err[which.min(err)], col='red',pch=20,cex=2)
```



MSE is the smallest for the polynomial with degree 3.

(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.

```
range(Boston$dis)
```

```
## [1]  1.1296 12.1265
```

```
summary(Boston$dis)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.130   2.100   3.207   3.795   5.188  12.127
```

```
fit2 <- lm(nox ~ bs(dis, df = 4), data = Boston)
summary(fit2)
```

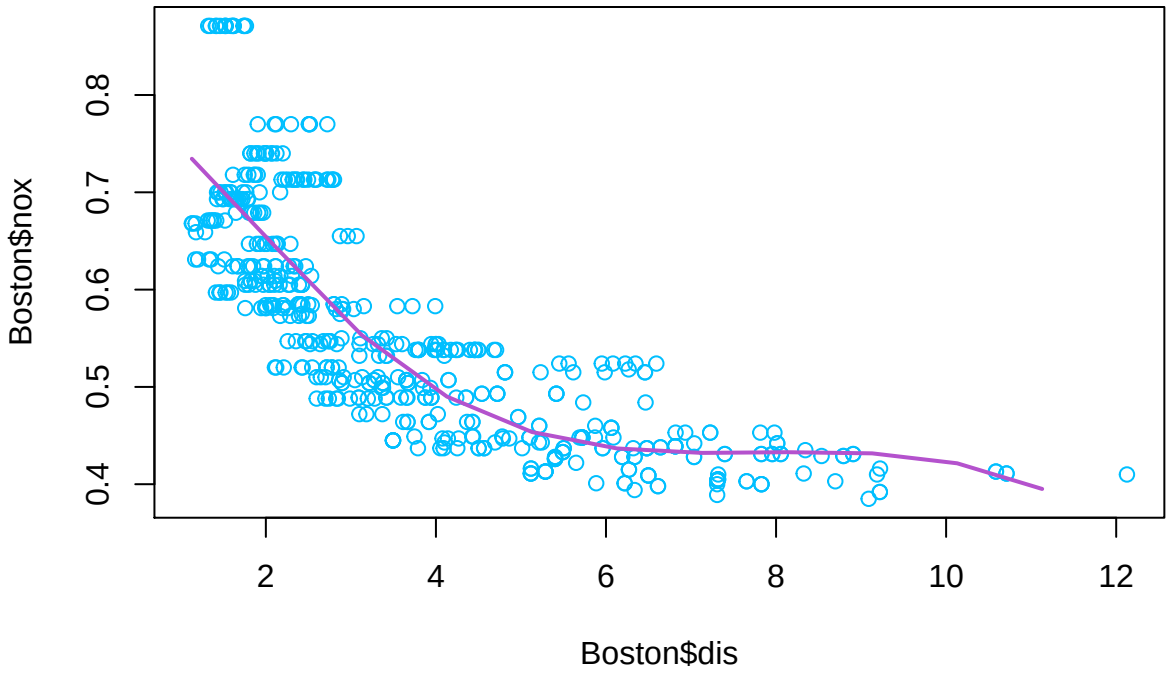
```
##
## Call:
## lm(formula = nox ~ bs(dis, df = 4), data = Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.124622 -0.039259 -0.008514  0.020850  0.193891
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
```

```
## (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
```

```
x <- seq(min(Boston$dis), max(Boston$dis))
y <- predict(fit2, data.frame(dis = x))
plot(Boston$dis, Boston$nox, col = "deepskyblue")
lines(x, y, col = "mediumorchid3", lwd = 2)
```

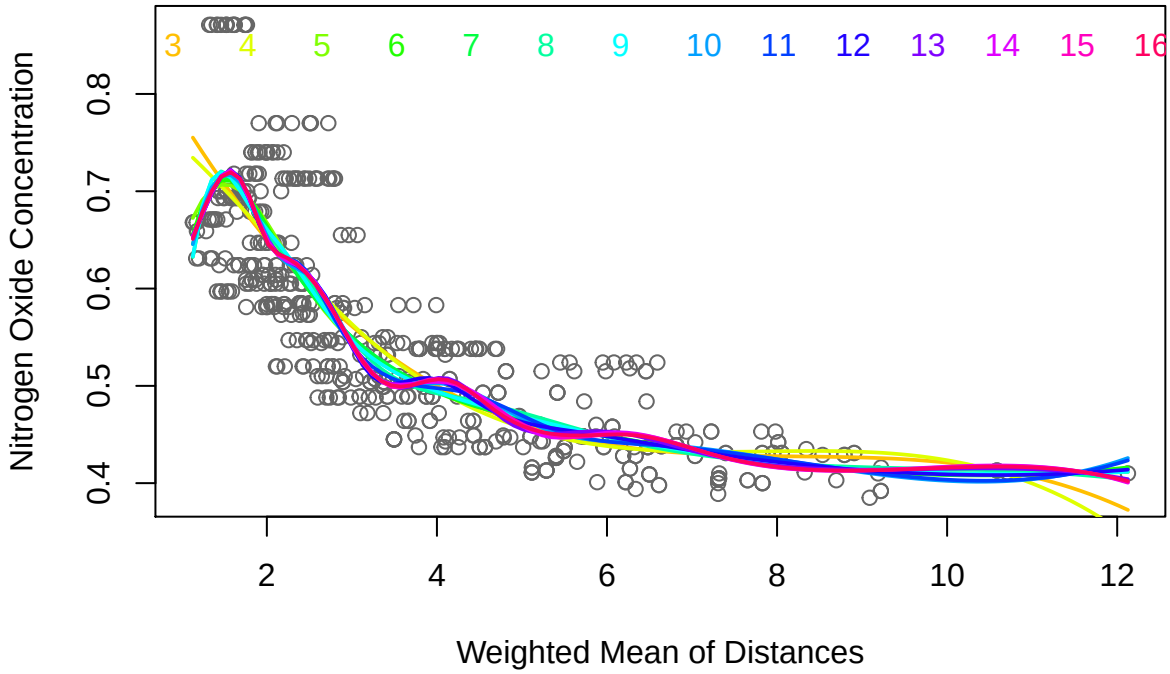


In this case, since we put it the degrees of freedom, R chose the knot at 3.207, which corresponds to the 50th percentile of the weighted mean of distances. This also coincides with the fact that the knot is chosen at the median of the variable dis.

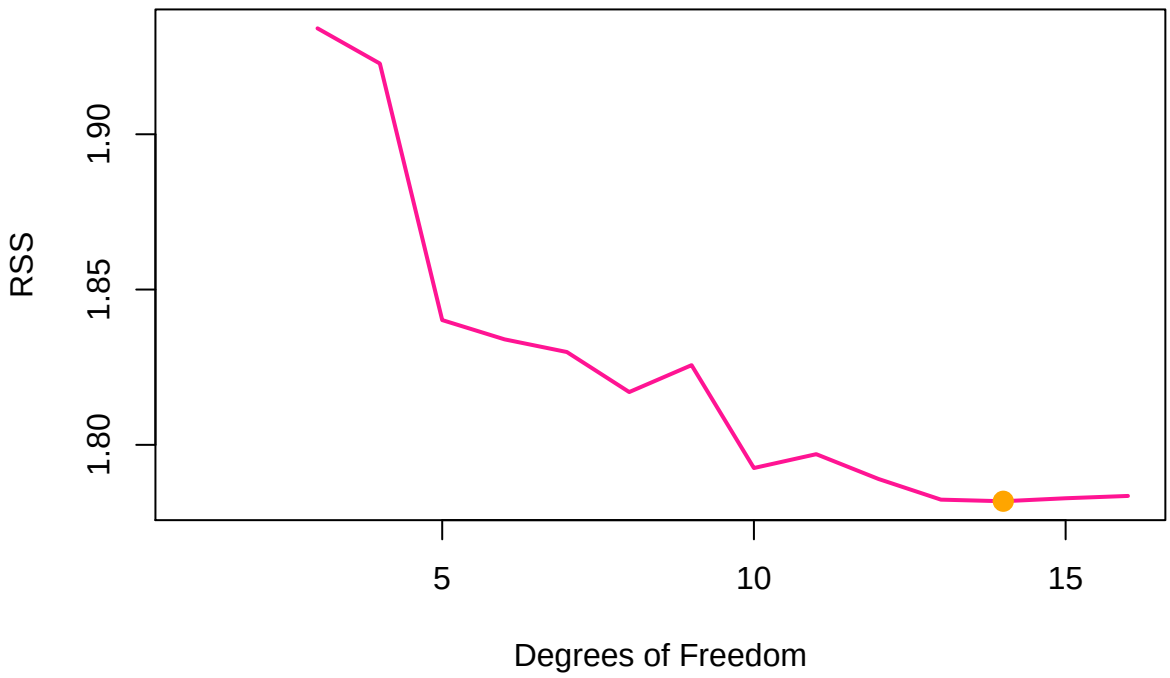
(e) Now fit a regression spline for a range of degrees of freedom, and plot the resulting fits and report the

resulting RSS. Describe the results obtained.

```
plot(Boston$dis, Boston$nox, xlab = "Weighted Mean of Distances", ylab = "Nitrogen Oxide Concentration"
clrs <- rainbow(16)
legend(x = "topright", legend = 3:16, text.col = clrs[3:16], text.width = 0.2, bty = "n", horiz = T)
x <- seq(min(Boston$dis), max(Boston$dis), length.out = 100)
rss_df <- c()
for (i in 3:16) {
  fit <- lm(nox ~ bs(dis, df = i), data = Boston)
  pred <- predict(fit, data.frame(dis = x))
  lines(x, pred, col = clrs[i], lwd = 1.85)
  rss_df[i] <- sum(fit$residuals^2)
}
```



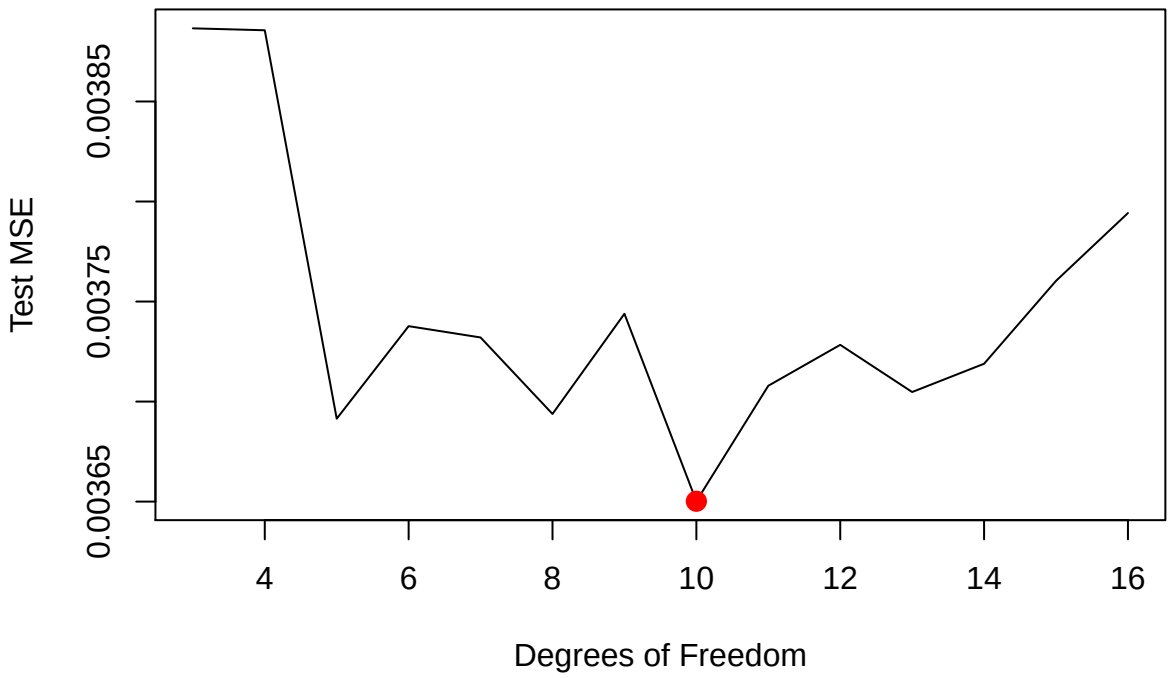

```
plot(1:16, rss_df, xlab = "Degrees of Freedom", ylab = "RSS", type = "l", col = "deeppink1", lwd = 2)
points(which.min(rss_df), rss_df[which.min(rss_df)], col='orange',pch=20,cex=2)
```



The smallest RSS is for 14 degrees of freedom.

(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.

```
set.seed(9)
cv <- rep(NA, 16)
for (i in 3:16) {
  fit <- glm(nox ~ bs(dis, df = i), data = Boston)
  cv[i] <- cv.glm(Boston, fit, K = 10)$delta[1]
}

plot(3:16, cv[3:16], xlab = "Degrees of Freedom", ylab = "Test MSE", type = "l")
points(which.min(cv), cv[which.min(cv)], col = "red", pch = 20, cex = 2)
```



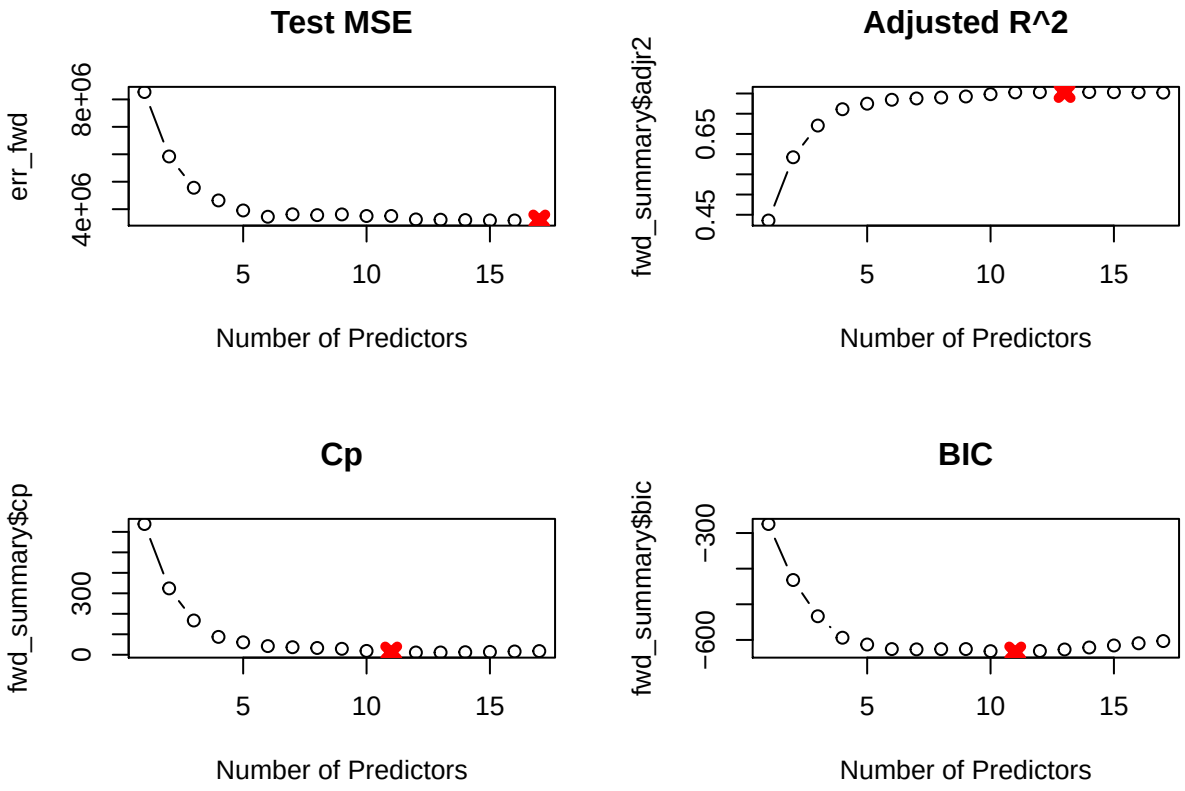
Cross-validation shows the smallest MSE for 10 degrees of freedom.

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.

```
data("College")
set.seed(1)
train_id <- sample(1:nrow(College), 500)
train <- College[train_id,]
test <- College[-train_id,]

fit_fwd <- regsubsets(Outstate ~ ., train, nvmax = ncol(College)-1, method = "forward")
fwd_summary <- summary(fit_fwd)

test_mat <- model.matrix(Outstate ~ ., test, nvmax = ncol(College)-1)

err_fwd <- rep(NA, ncol(College)-1)
for (i in 1:(ncol(College)-1)) {
  coeff <- coef(fit_fwd, id = i)
  pred_fwd <- test_mat[, names(coeff)] %*% coeff
  err_fwd[i] <- mean((test$Outstate - pred_fwd)^2)
}
par(mfrow = c(2,2))
```

```r
plot(err_fwd, type = "b", main = "Test MSE", xlab = "Number of Predictors")
min_mse <- which.min(err_fwd)
points(min_mse, err_fwd[min_mse], col = "red", pch = 4, lwd = 5)
plot(fwd_summary$adjr2, type = "b", main = "Adjusted R^2", xlab = "Number of Predictors")
max_adjr <- which.max(fwd_summary$adjr2)
points(max_adjr, fwd_summary$adjr2[max_adjr], col = "red", pch = 4, lwd = 5)
plot(fwd_summary$cp, type = "b", main = "Cp", xlab = "Number of Predictors")
min_cp <- which.min(fwd_summary$cp)
points(min_cp, fwd_summary$cp[min_cp], col = "red", pch = 4, lwd = 5)
plot(fwd_summary$bic, type = "b", main = "BIC", xlab = "Number of Predictors")
min_bic <- which.min(fwd_summary$bic)
points(min_bic, fwd_summary$bic[min_bic], col = "red", pch = 4, lwd = 5)
```



The model metrics do not seem to improve much after 10 predictors.

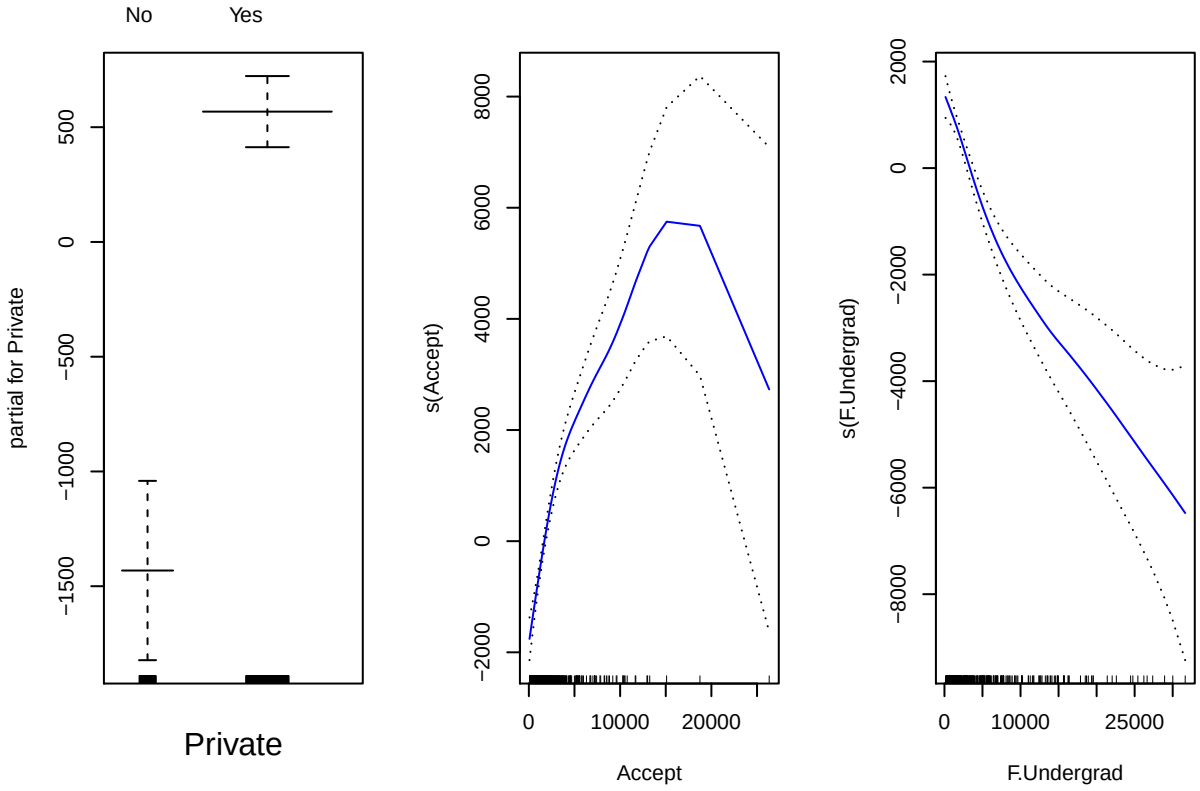
```r
coef(fit_fwd, 10)
```

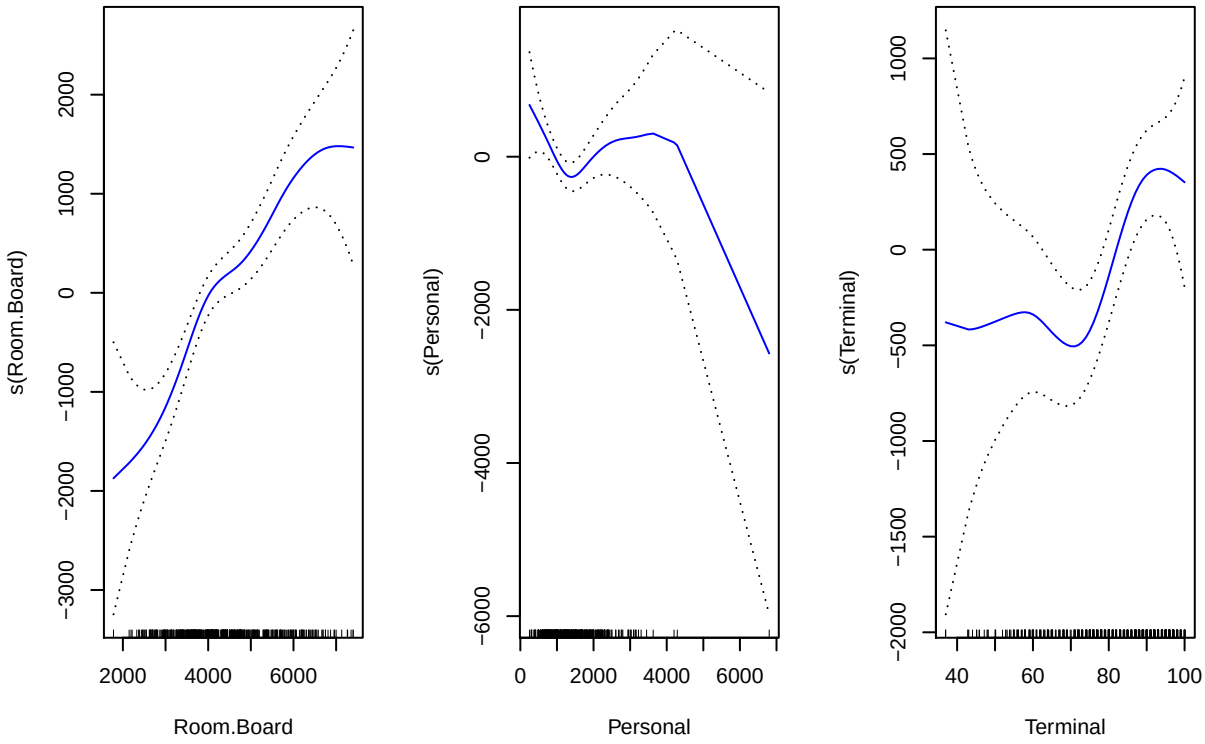
```
##   (Intercept)    PrivateYes        Accept   F.Undergrad    Room.Board
## -1319.5952439  2321.3669112     0.3176508    -0.1453921     0.8786746
##      Personal      Terminal     S.F.Ratio   perc.alumni        Expend
##    -0.2720261    42.2477269   -97.5030888    49.6682035     0.1775606
##     Grad.Rate
##    25.7422224
```

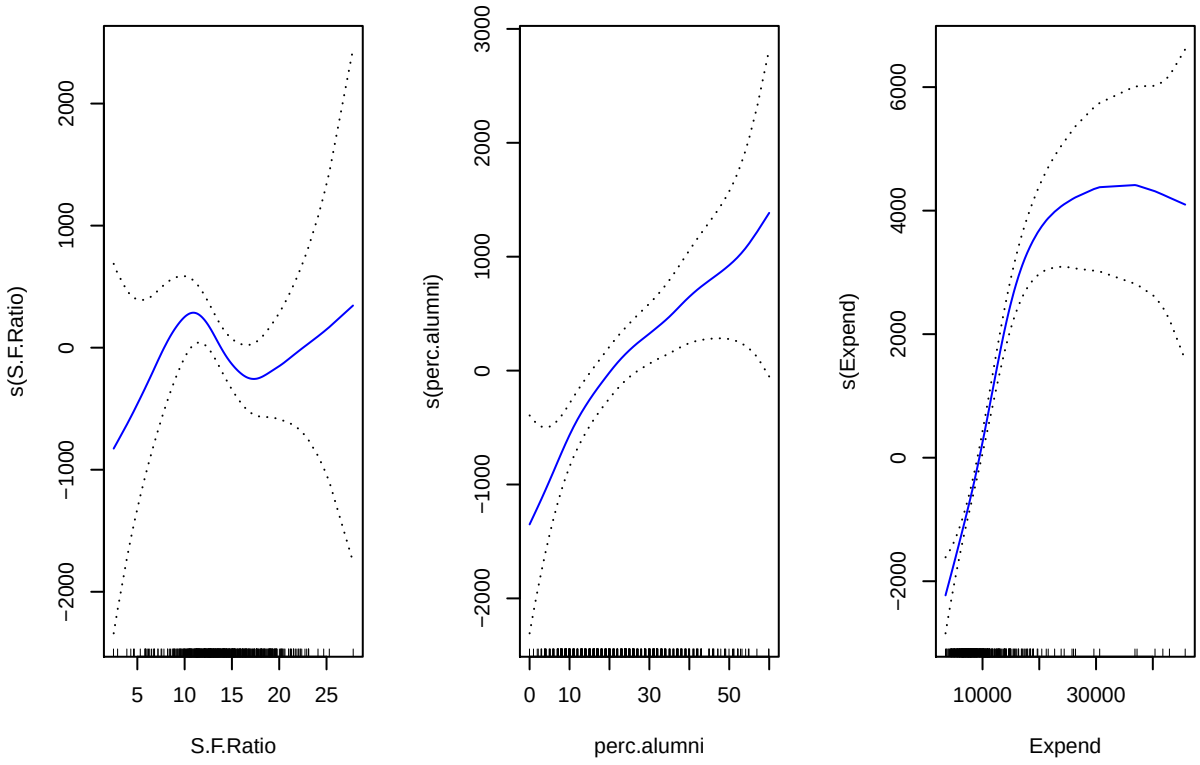
However, the coefficient for Private has an astronomical value compared to the other coefficients.

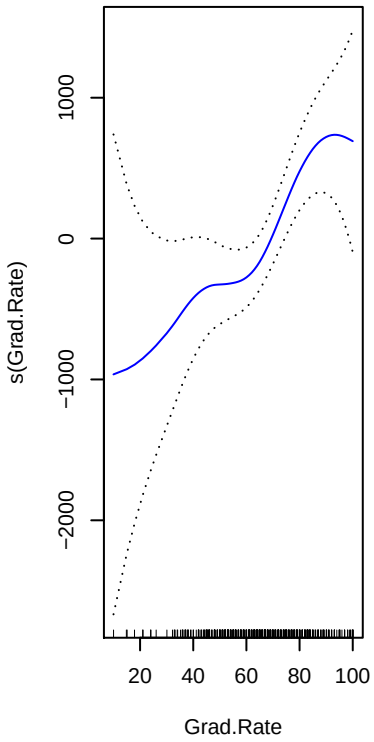
(b) Fit a GAM on the training data, using out-of-state tuition as the response and the features selected in the previous step as the predictors. Plot the results, and explain your findings.

```
library(gam)
gam1 <- gam(Outstate ~ Private + s(Accept) + s(F.Undergrad) + s(Room.Board) + s(Personal) + s(Terminal)

par(mfrow = c(1,3))
plot(gam1, se = TRUE, col = "blue")
```




s(Room.Board)
2000
1000
0
−1000
−2000
−3000
2000
4000
6000
Room.Board
s(Personal)
0
−2000
−4000
−6000
0
2000
4000
6000
Personal
s(Terminal)
1000
500
0
−500
−1000
−1500
−2000
40
60
80
100
Terminal


s(S.F.Ratio)
2000
1000
0
−1000
−2000
5
10
15
20
25
S.F.Ratio
s(perc.alumni)
3000
2000
1000
0
−1000
−2000
0
10
30
50
perc.alumni
s(Expend)
6000
4000
2000
0
−2000
10000
30000
Expend



The out-of-state tuition seems non-monotonic and has a non-linear relationship with almost all of the predictors included in the model. Most of them have large standard errors for higher values, except for Grad.Rate which has large standard error for lower values.

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

```
pred_gam <- predict(gam1, test)
err_gam <- mean((test$Outstate - pred_gam)^2)
err_gam
```

```
## [1] 3136625
```

```
tss_gam <- mean(((test$Outstate) - mean(test$Outstate))^2)
rss_gam <- 1 - err_gam/tss_gam
rss_gam
```

```
## [1] 0.8031944
```

The R squared for the GAM model with 10 predictors is 0.8031 or 80.31% of the variation in the model is explained by the predictors chosen. This seems to be quite a good model.

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

```
summary(gam1)
```

```
##
## Call: gam(formula = Outstate ~ Private + s(Accept) + s(F.Undergrad) +
```

```
##     s(Room.Board) + s(Personal) + s(Terminal) + s(S.F.Ratio) +
##     s(perc.alumni) + s(Expend) + s(Grad.Rate), data = train)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -5679.22 -1128.60    86.39  1160.81  7500.53
##
## (Dispersion Parameter for gaussian family taken to be 3183850)
##
##     Null Deviance: 8144554092 on 499 degrees of freedom
## Residual Deviance: 1470940414 on 462.0005 degrees of freedom
## AIC: 8944.217
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                  Df     Sum Sq    Mean Sq F value    Pr(>F)
## Private           1 2348042860 2348042860 737.485 < 2.2e-16 ***
## s(Accept)         1  491007042  491007042 154.218 < 2.2e-16 ***
## s(F.Undergrad)    1  293475928  293475928  92.176 < 2.2e-16 ***
## s(Room.Board)     1 1049419434 1049419434 329.607 < 2.2e-16 ***
## s(Personal)       1   36155125   36155125  11.356 0.0008152 ***
## s(Terminal)       1  387538671  387538671 121.720 < 2.2e-16 ***
## s(S.F.Ratio)      1  261331649  261331649  82.080 < 2.2e-16 ***
## s(perc.alumni)    1  227603251  227603251  71.487 3.687e-16 ***
## s(Expend)         1  443134830  443134830 139.182 < 2.2e-16 ***
## s(Grad.Rate)      1   46274635   46274635  14.534 0.0001563 ***
## Residuals       462 1470940414    3183850
## ---
## 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(Accept)             3 18.8140 1.578e-11 ***
## s(F.Undergrad)        3  4.0011  0.007868 **
## s(Room.Board)         3  3.1799  0.023840 *
## s(Personal)           3  4.7060  0.003014 **
## s(Terminal)           3  3.3145  0.019896 *
## s(S.F.Ratio)          3  3.0775  0.027347 *
## s(perc.alumni)        3  0.7941  0.497580
## s(Expend)             3 18.1856 3.601e-11 ***
## s(Grad.Rate)          3  1.8764  0.132713
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

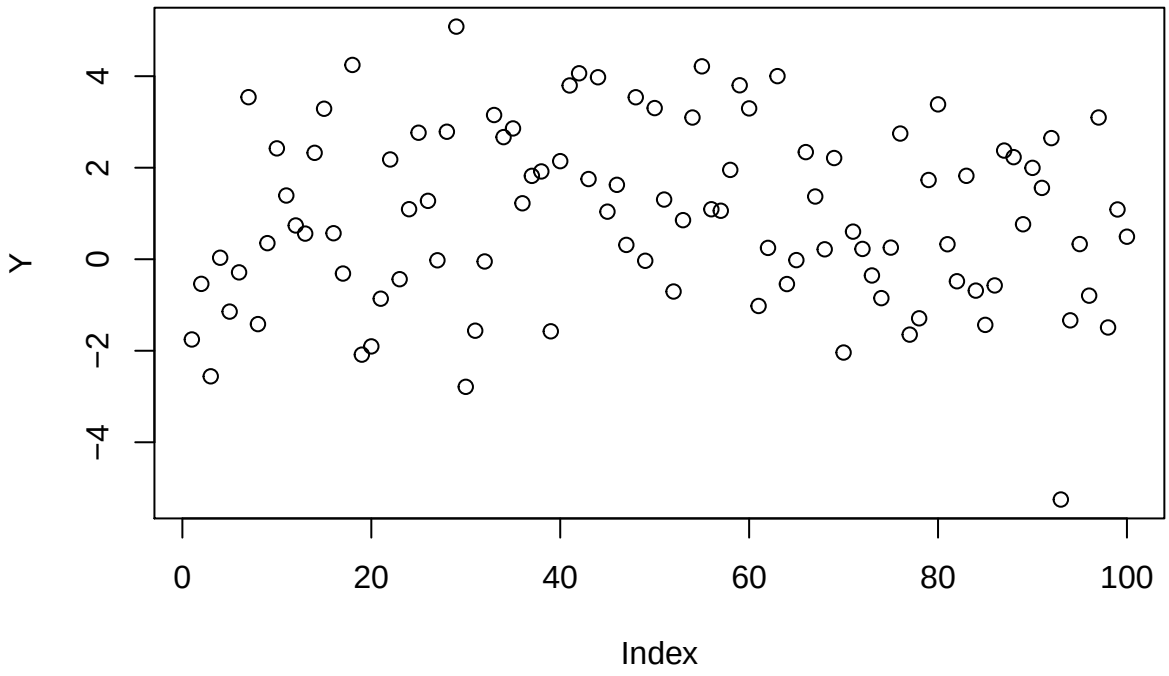
perc.alumni and Grad.Rate do not have significant non-parametric effects. They could probably be converted to linear parametric terms. Accept and Expend have strong non-linear relationships with out-of-state tuition, the response variable.

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 X1 and X2, with n = 100.

```
set.seed(99)
n <- 100
X1 <- rnorm(100)
X2 <- rnorm(100)
eps <- rnorm(1:100, sd = 1)

b_0 <- 0.9
b_1 <- -1.5
b_2 <- 1
Y = b_0 + b_1*X1 + b_2*X2 +eps

plot(Y)
```



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

```
b_h1 <- 1
```

(c) Keeping $\hat{\beta}_1$ fixed, fit the model Y - $\hat{\beta}_1 X_1 = \beta_0 + \beta_2 X_2 + \epsilon$. You can do this as follows:

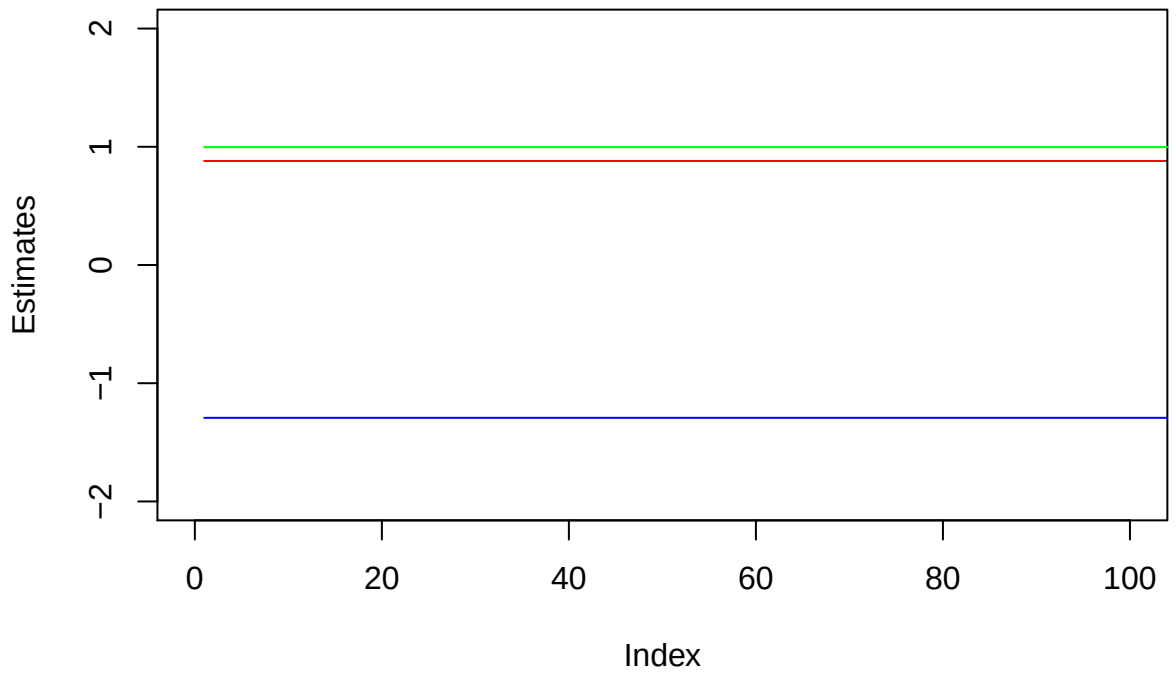
```
a=Y-b_h1 *X1
b_h2=lm(a~X2)$coef [2]
```

(d) Keeping $\hat{\beta}_2$ fixed, fit the model Y - $\hat{\beta}_2 X_2 = \beta_0 + \beta_1 X_1 + \epsilon$. You can do this as follows:

```
a=Y-b_h2 *X2
b_h1=lm(a~X1)$coef [2]
```

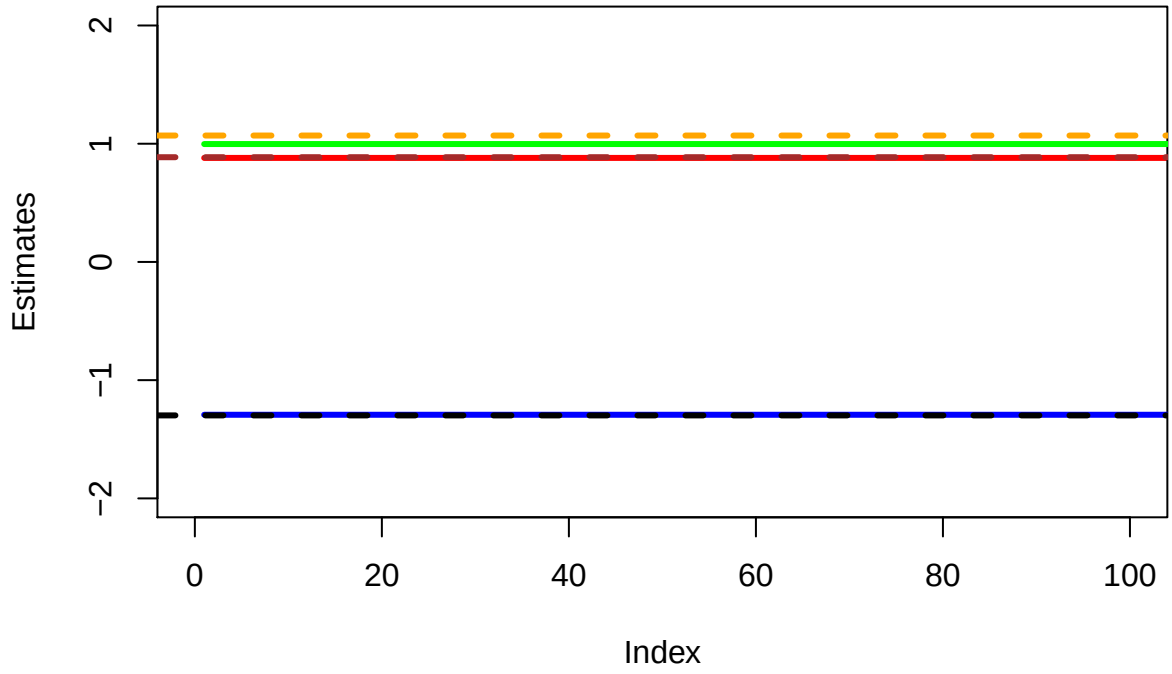
(e) Write a for loop to repeat (c) and (d) 1,000 times. Report the estimates of $\hat{\beta}_0$, $\hat{\beta}_1$, and $\hat{\beta}_2$ at each iteration of the for loop. Create a plot in which each of these values is displayed, with $\hat{\beta}_0$, $\hat{\beta}_1$, and $\hat{\beta}_2$ each shown in a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

```
b_hat0 <- rep(0,1000)
b_hat1 <- rep(0,1000)
b_hat2 <- rep(0,1000)

for (i in 1:1000) {
  a <- Y - b_hat1[i]*X1
  b_hat2[i] <- lm(a ~ X2)$coef[2]
  a <- Y - b_hat2[i]*X2
  b_hat1[i] <- lm(a ~ X1)$coef[2]
  b_hat0[i] <- lm(a ~ X1)$coef[1]
}

plot(b_hat0, ylab = "Estimates", type = "l", col = "red", ylim = c(-2,2), xlim = c(0,100))
lines(b_hat1, col = "blue")
lines(b_hat2, col = "green")
```



(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).

```
fit3 <- lm(Y ~ X1 + X2)

plot(b_hat0, ylab = "Estimates", type = "l", col = "red", ylim = c(-2,2), xlim = c(0,100), lwd = 3)
lines(b_hat1, col = "blue", lwd = 3)
lines(b_hat2, col = "green", lwd = 3)
abline(h = coef(fit3)[1], lty = "dashed", col = "brown", lwd = 3)
abline(h = coef(fit3)[2], lty = "dashed", col = "black", lwd = 3)
abline(h = coef(fit3)[3], lty = "dashed", col = "orange", lwd = 3)
```



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

```
b <- data.frame(b_hat0, b_hat1, b_hat2)
head(b)
```

```
##      b_hat0    b_hat1    b_hat2
## 1 0.8799804 -1.292655 0.9972832
## 2 0.8799804 -1.292655 0.9972832
## 3 0.8799804 -1.292655 0.9972832
## 4 0.8799804 -1.292655 0.9972832
## 5 0.8799804 -1.292655 0.9972832
```

```
## 6 0.8799804 -1.292655 0.9972832
```

One iteration was enough to obtain a good approximation to the multiple regression coefficients.