Homework #9

Justin Robinette

March 26, 2019

No collaborators for any problem

Question 7.9.6, pg 299: In this exercise, you will further analyze the **Wage** data set considered throughout this chapter.

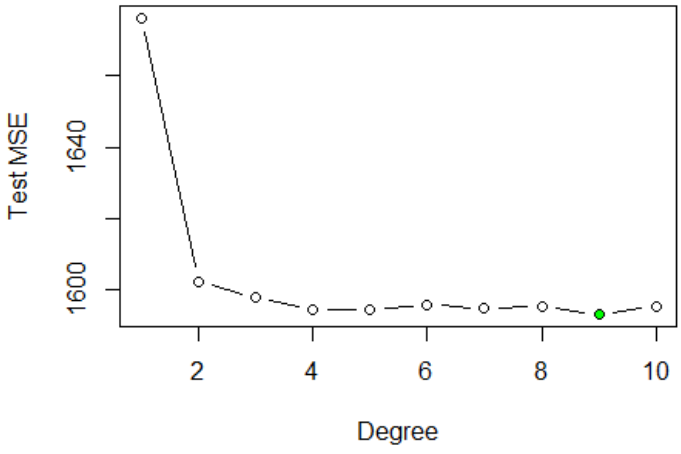
Part A: Perform polynomial regression to predict *wage* using *age*. Use cross-validation to select the optimal degree *d* for the polynomial. What degree was chosen and how does this compare to the results of hypothesis testing using ANOVA? Make a plot of the resulting polynomial fit to the data.

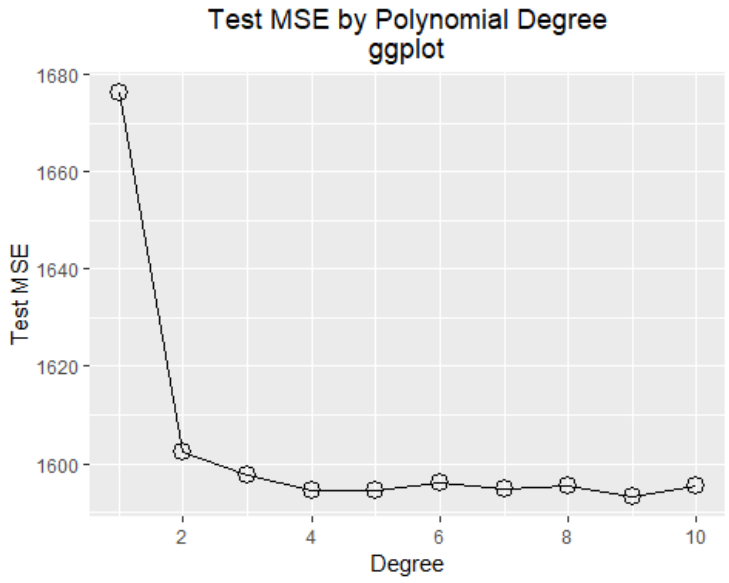
Results: Here I first used cross-validation to select the optimal *d* for the polynomial of the predictor *age*. As we can see below from the plot, the optimal degree was determined to be 9 (denoted by the green point). I made a plot of the resulting polynomial fit to the data using this 9th degree polynomial. Aside from the high wage outliers, the line fits the data pretty well.

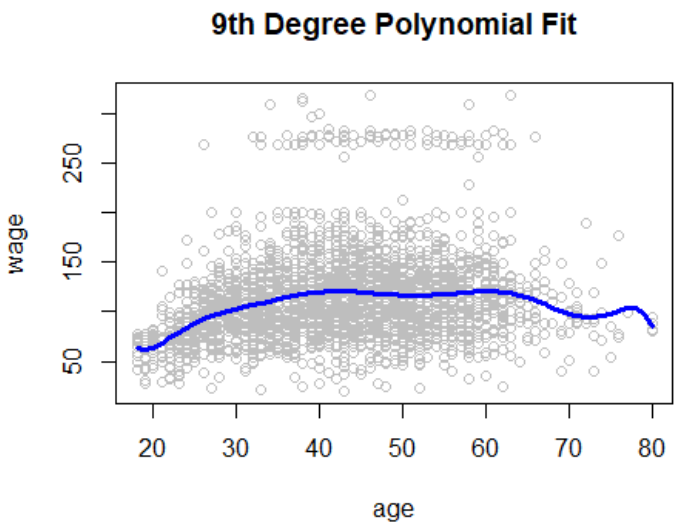
Next, I used ANOVA to determine the optimal degree *d* to see if it matches the results from the cross-validation. As we can see from the table below, the 2nd, 3rd and 9th degree polynomials are best according to the p-values. Because the p-value of the 2nd degree polynomial is the lowest, I will use this *d* for the plot.

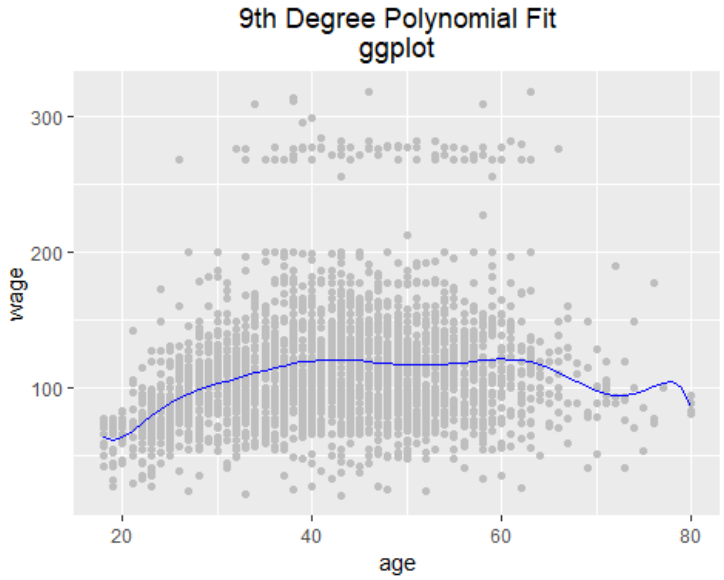
Lastly, per the instructions, I made a plot of the resulting polynomial fit to the data. The blue line represents the prediction, by age, of wage. As we can see, aside from some high wage outliers, the line appears to fit the data ok considering the wide range at each age. Analogous ggplots are included per homework instructions.

```
## [1] "The optimal degree (d) for the polynomial, from cross-validation, is: 9"
```



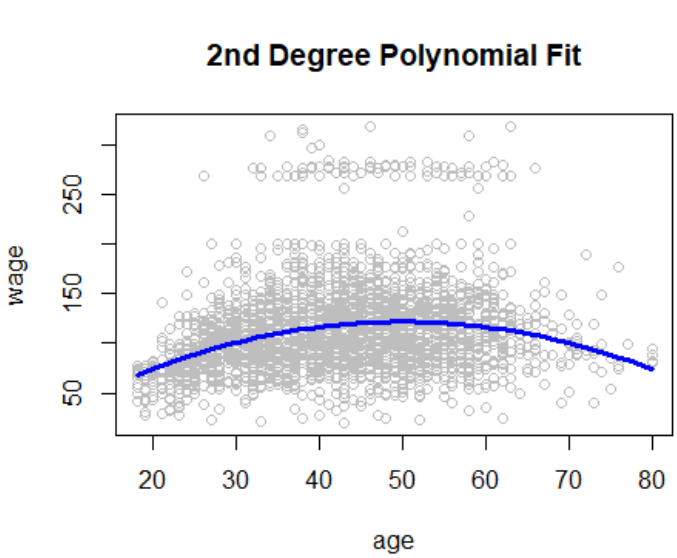


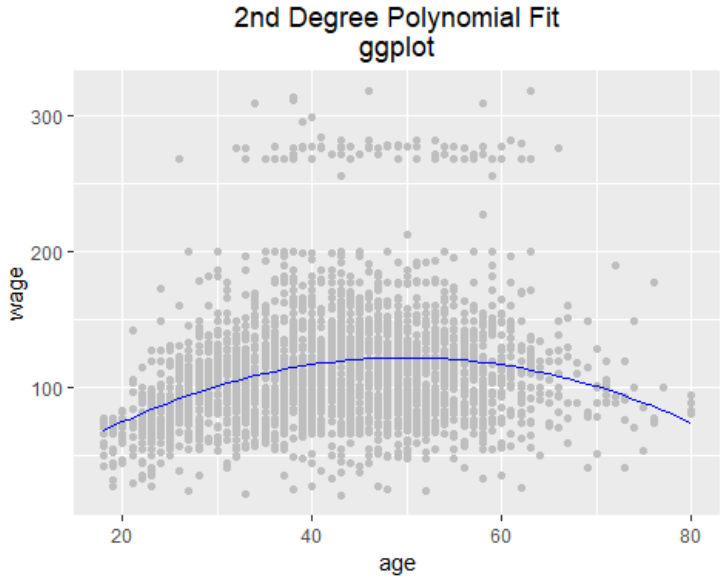




P-Value by Polynomial Degree from ANOVA

Degree	P-Value
1	NA
2	2.18733010887147e-32
3	0.00166858259572121
4	0.050908739564463
5	0.369397769981588
6	0.116074445054321
7	0.205198937139046
8	0.777865390322466
9	0.0359942486414226
10	0.967529190782386



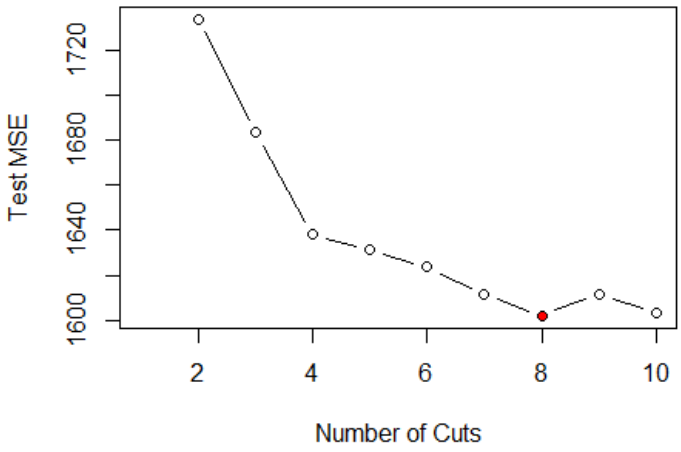


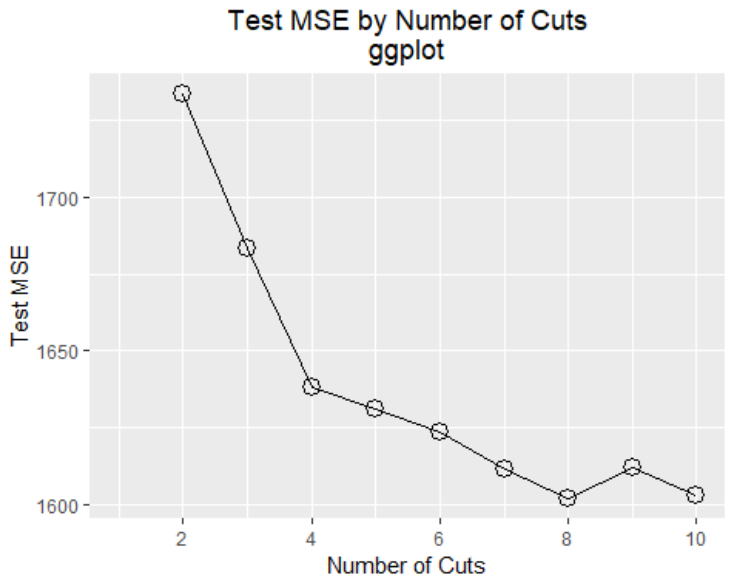
Part B: Fit a step function to predict *wage* using *age* and perform cross-validation to choose the optimal number of cuts.

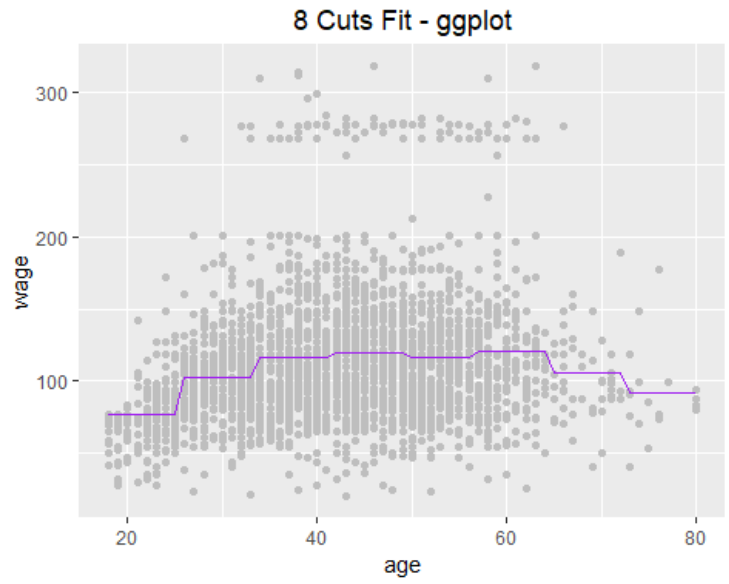
Results: From the plot below, we can see that the number of cuts that minimizes the MSE is 8 (denoted by the red point).

Using cuts = 8, I fit a glm with *wage* as the response and *age* as the predictor. Again, I plotted the points of wage by age and added a line that is the prediction from the glm with cuts = 8. Similar to in part A, we see that the model does a resonably decent job of predicting most of the wage values considering the wide range of wages at any given age. A comparison ggplot is added.

```
## [1] "The optimal number of cuts, from cross-validation, is 8"
```



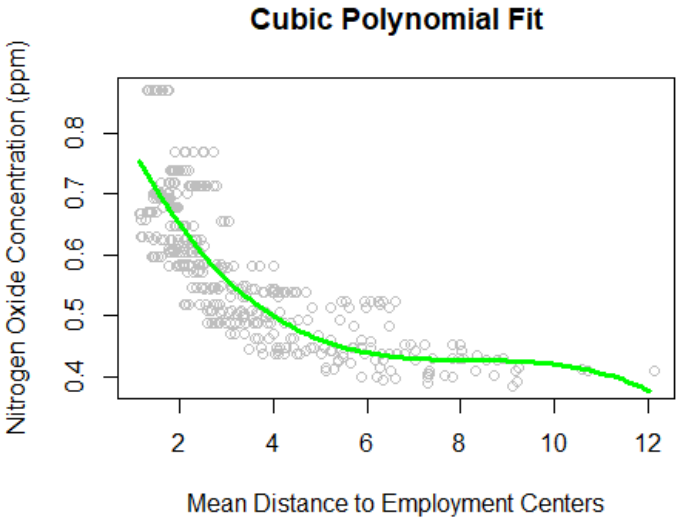


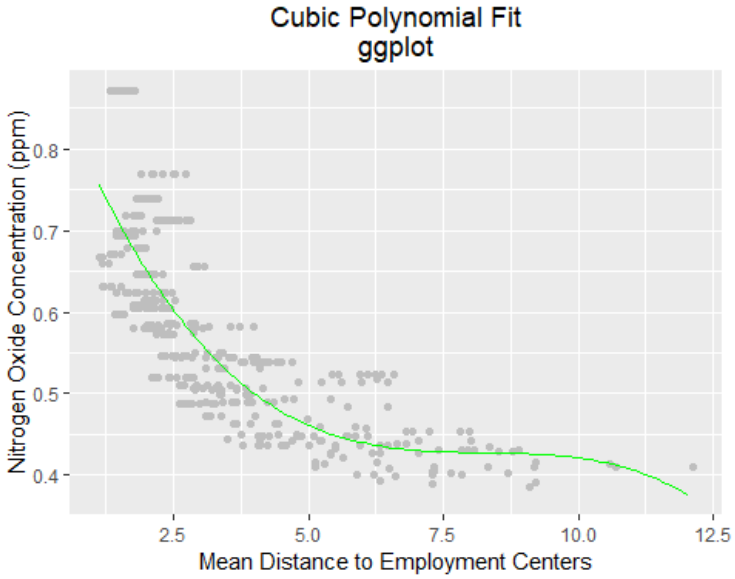


Question 7.9.9, pg 299: This question uses the variable *dis* (weighted mean of distances to five Boston employment centers) and *nox* (nitrogen oxides concentration in parts per 10 million) from the **Boston** data set. We will treat *dis* as the predictor and *nox* as the response.

Part 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.

Results: Below we see the plot of the resulting fit. As we can see from the reasonably good fit of the plot and the summary, the polynomial terms appear to be significant as predictors of *nox*. An analogous ggplot is added.

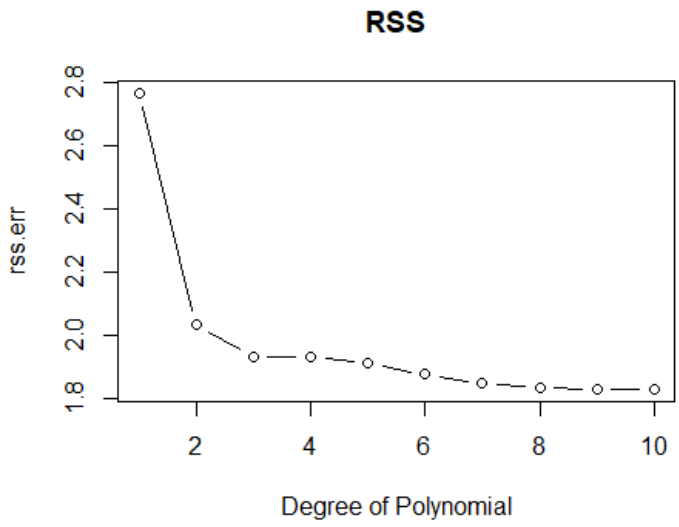


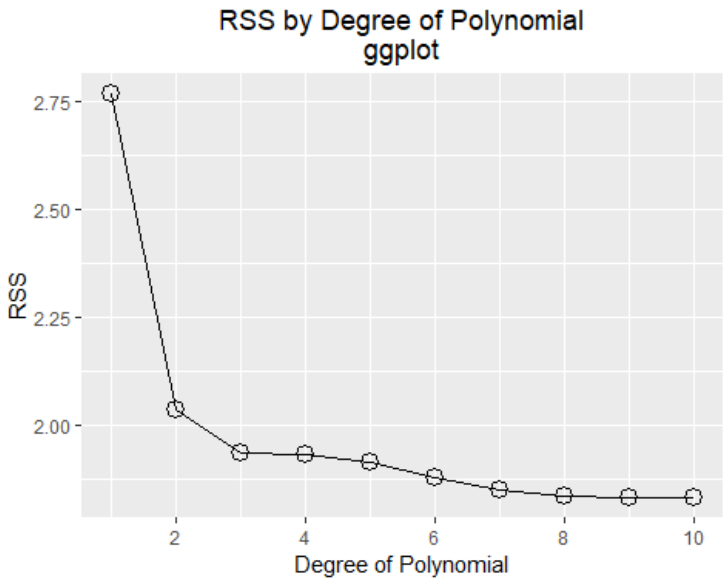



```
##
## Call:
## lm(formula = nox ~ poly(dis, 3), data = Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.121130 -0.040619 -0.009738  0.023385  0.194904
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.554695   0.002759 201.021  < 2e-16 ***
## poly(dis, 3)1 -2.003096   0.062071 -32.271  < 2e-16 ***
## poly(dis, 3)2  0.856330   0.062071  13.796  < 2e-16 ***
## poly(dis, 3)3 -0.318049   0.062071  -5.124 4.27e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06207 on 502 degrees of freedom
## Multiple R-squared:  0.7148, Adjusted R-squared:  0.7131
## F-statistic: 419.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

Part B: Plot the polynomial fits for a range of different polynomial degrees (1 - 10), and report the associated residual sum of squares.

Results: The RSS decreases as the degree of the polynomial increases, with the exception of the increase from degree 3 to 4. The minimum RSS is seen with a polynomial with degree = 10.

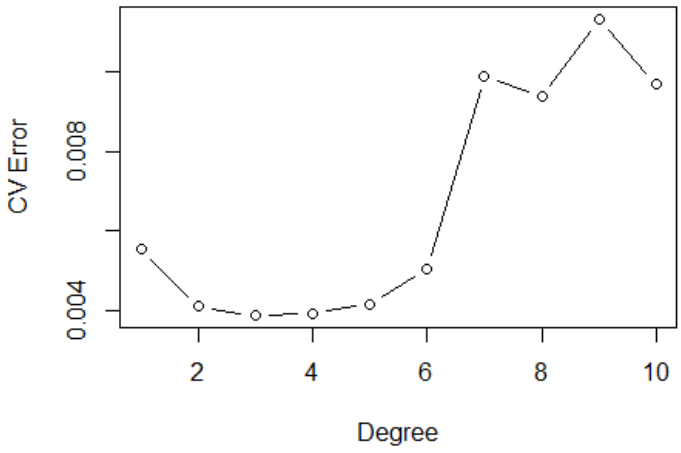


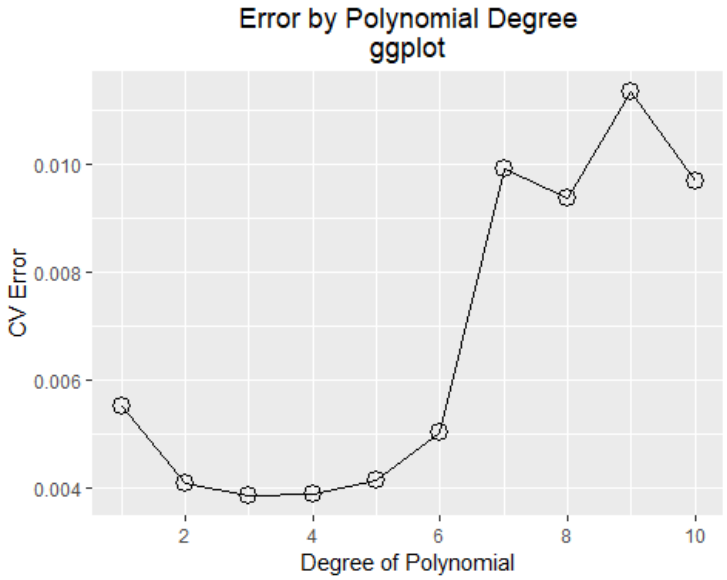



```
## [1] "The RSS is minimized when the Degree of the Polynomial is: 10"
```

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

Results: As we can see from the plot and statement below, the optimal degree for the polynomial using cross-validation is 3. Analogous ggplot is added. The chart below also confirms that the lowest CV Error is found when the polynomial is 3.





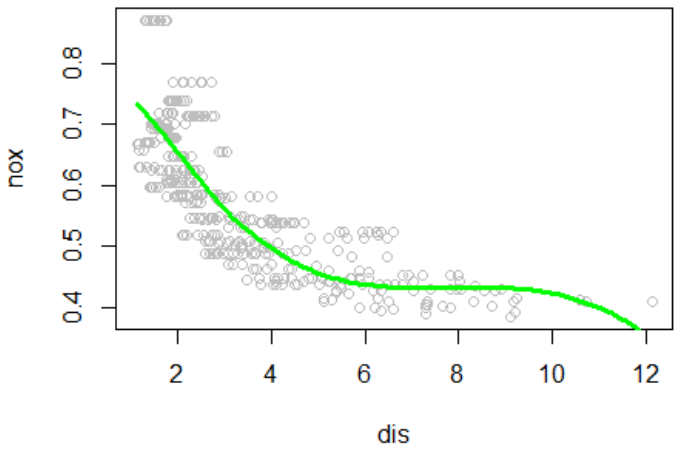
CV Error by Degree of Polynomial

Degree	Error
1	0.0055235
2	0.0040808
3	0.0038698
4	0.0039042
5	0.0041563
6	0.0050423
7	0.0099105
8	0.0093794
9	0.0113391
10	0.0096904

```
## [1] "The optimal degree of polynomial from CV is: 3"
```

Part 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.

Results: First, I fit a regression spline to predict *nox* using *dis*. Then I plot the resulting fit and printed the summary which shows that all terms in the spline fit are significant. As we can see from the plot, the spline fit does a reasonable job of predicting the response. Lastly, the table shows that one knot is chosen at the 50th percentile.



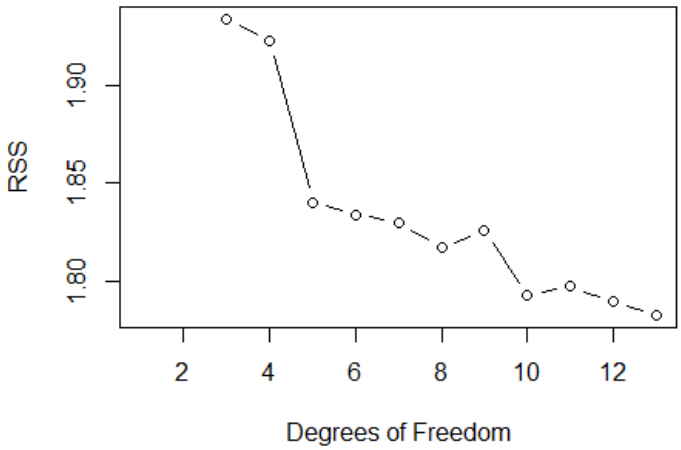

```
##
## Call:
## lm(formula = nox ~ bs(dis, df = 4), data = Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.124622 -0.039259 -0.008514  0.020850  0.193891
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       0.73447    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
```

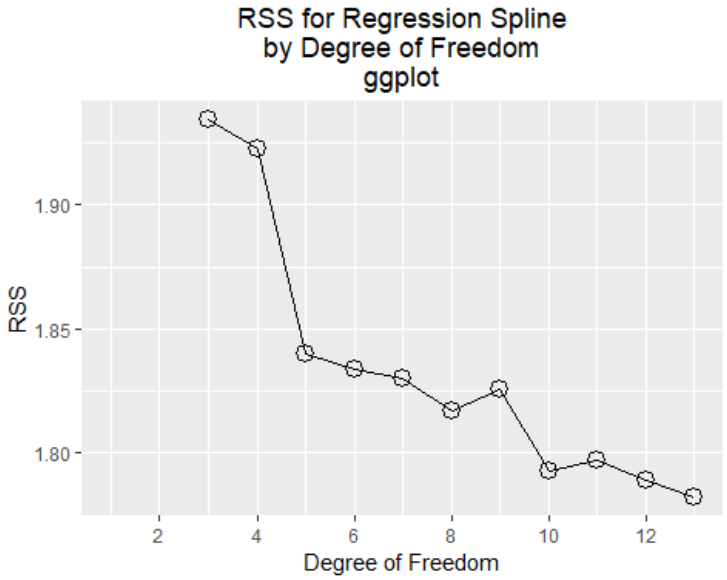
Knots Chosen

	Knot
50%	3.20745

Part E: Now fit a regression spline for a range of degrees of freedom and plot the resulting fits and report the resulting RSS. Descrive the results obtained.

Results: We can see that the RSS decreases (with the exception of when df goes from 8 to 9). The sharpest decrease in RSS takes place between df = 4 and df = 5. When df = 13, the RSS is minimized.





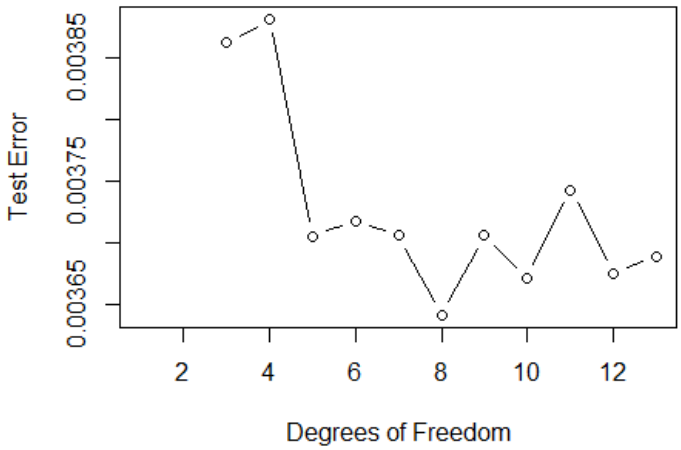

```
## [1] "The best degrees of freedom, as selected by RSS, is: 13"
```

RSS by Degree of Freedom of Regression Spline

Degree	RSS
3	1.934107
4	1.922775
5	1.840173
6	1.833966
7	1.829884
8	1.816995
9	1.825652
10	1.792535
11	1.796992
12	1.788999
13	1.782350

Part F: Perform cross-validation or another approach in order to select the best degrees of freedom for a regression spline on the data. Describe your results.

Results: Test Error is minimized when the degrees of freedom is 8. When df = 13, which was chosen in the above exercise, test error is also low relative to most of the df options in the plot.




```
## [1] "The best degrees of freedom, as chosen by CV, is: 8"
```

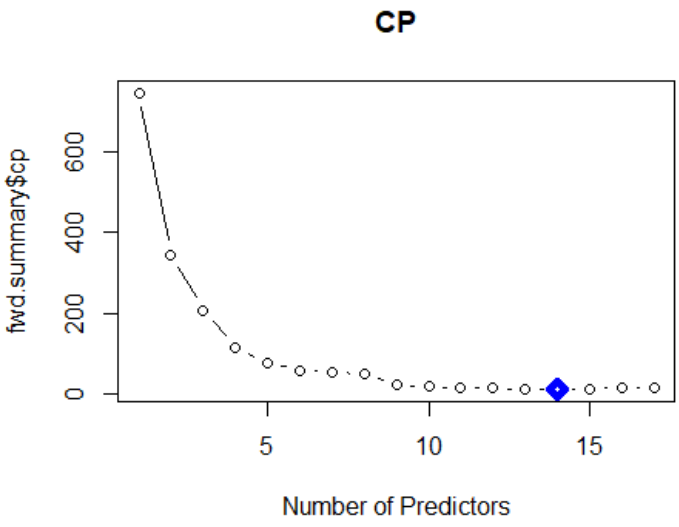
Test Error by Degree of Freedom of Regression Spline

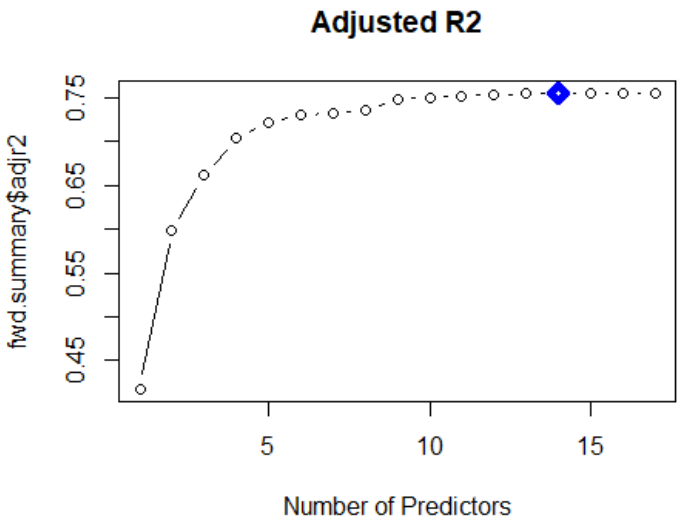
Degree	Test Error
3	0.0038630
4	0.0038821
5	0.0037056
6	0.0037180
7	0.0037062
8	0.0036414
9	0.0037059
10	0.0036714
11	0.0037433
12	0.0036757
13	0.0036890

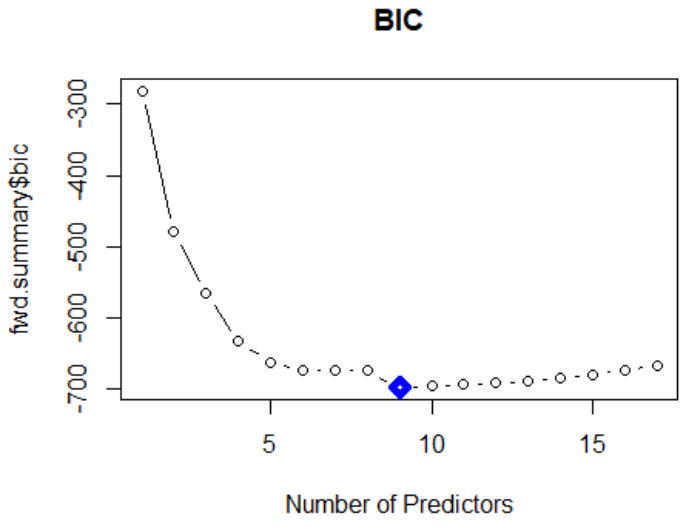
Question 7.9.10, pg 300: This question relates to the **College** data set.

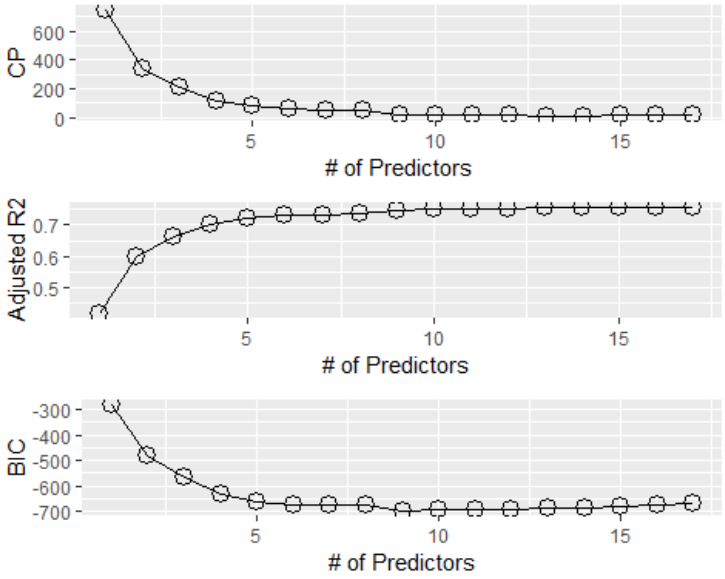
Part A: Split the data into a training set and a test set. Using out-of-state tuition as the response and the other variables as the predictors, perform forward stepwise selection on the training set in order to identify a satisfactory model that uses just a subset of the predictors.

Results: After loading the dataset and splitting into train and test subsets, I performed forward stepwise selection on the training set. As we can see from the plots below, there doesn't seem to be much improvement after 9 predictors. I included the 9 predictors that we will use going forward in the table below.







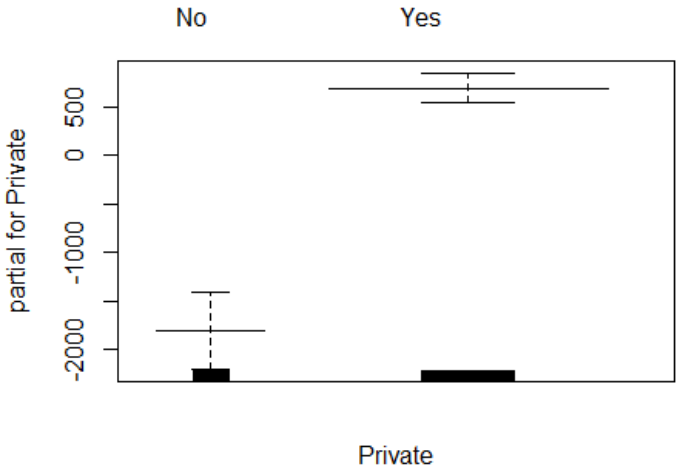


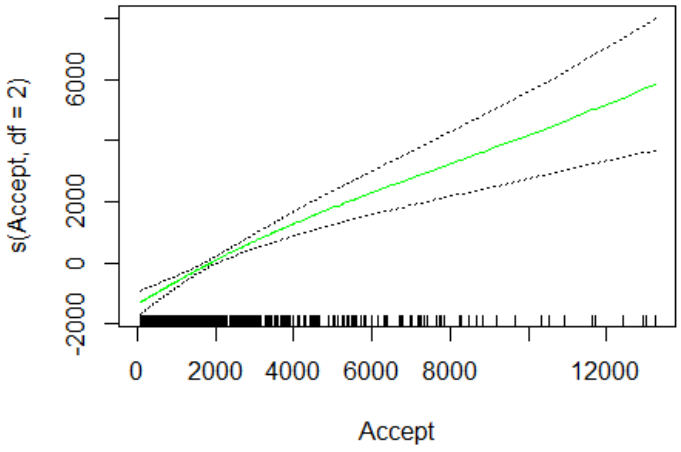
Optimal Predictors by Statistical Measure

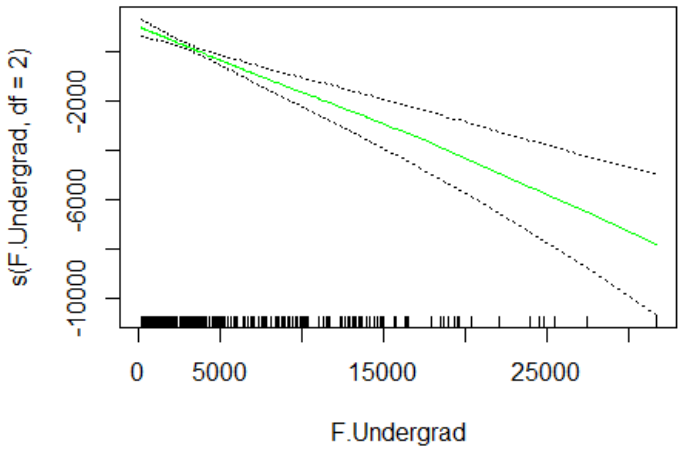
CP	R2	BIC
14	14	9
Predictors		
(Intercept)	-1820.5433159	
PrivateYes	2697.0169129	
Accept	0.5623539	
F.Undergrad	-0.2554521	
Room.Board	0.7608560	
Personal	-0.2344181	
PhD	34.4907907	
perc.alumni	59.9908105	
Expend	0.2093306	
Grad.Rate	18.7129802	

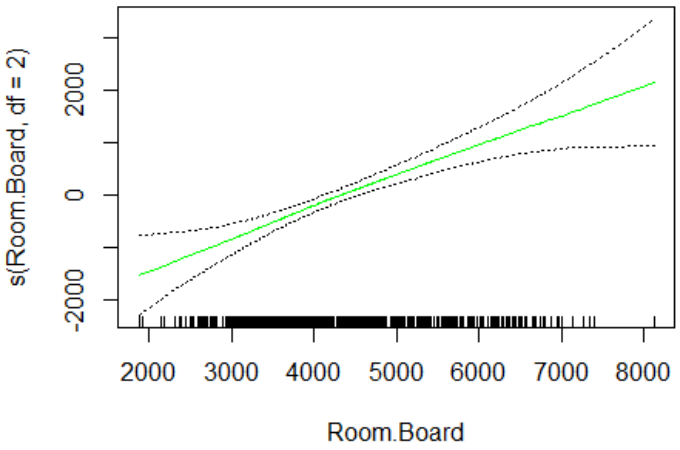
Part B: Fit a GAM on the training data, using out-of-state tuition as the response and the predictors selected in the previous step. Plot the results, and explain your findings.

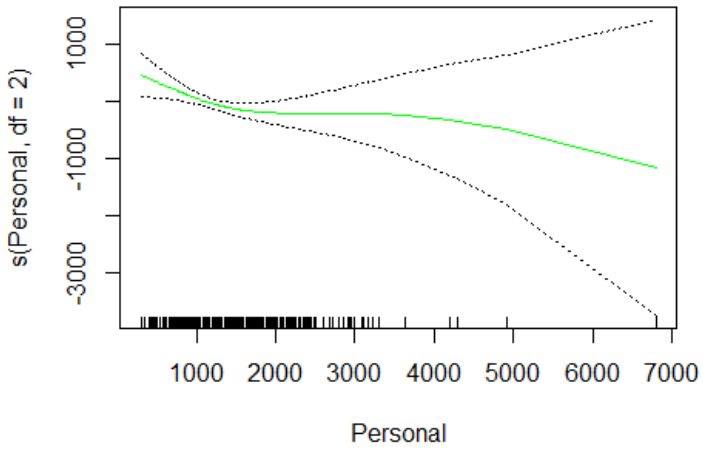
Results: Below I fit a GAM model on the training data using the 8 predictors outlined above. As we see from the plots below, most of the predictors appear to have a linear relationship with *Outstate*. We will examine this further in the next step of this exercise.

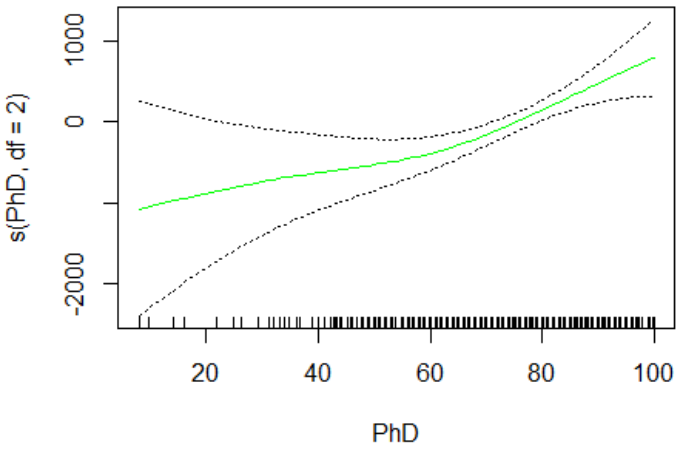


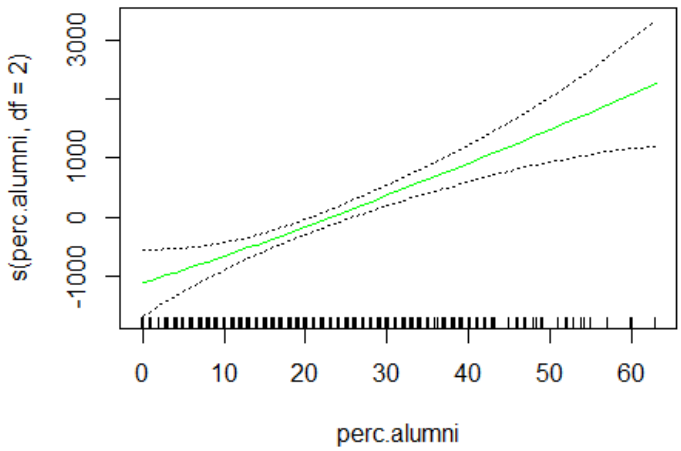


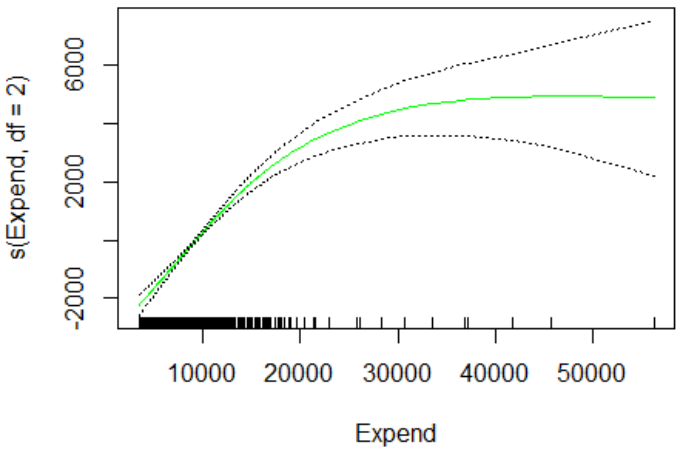

s(F.Undergrad, df = 2)
-10000
-6000
-2000
0
5000
15000
25000
F.Undergrad

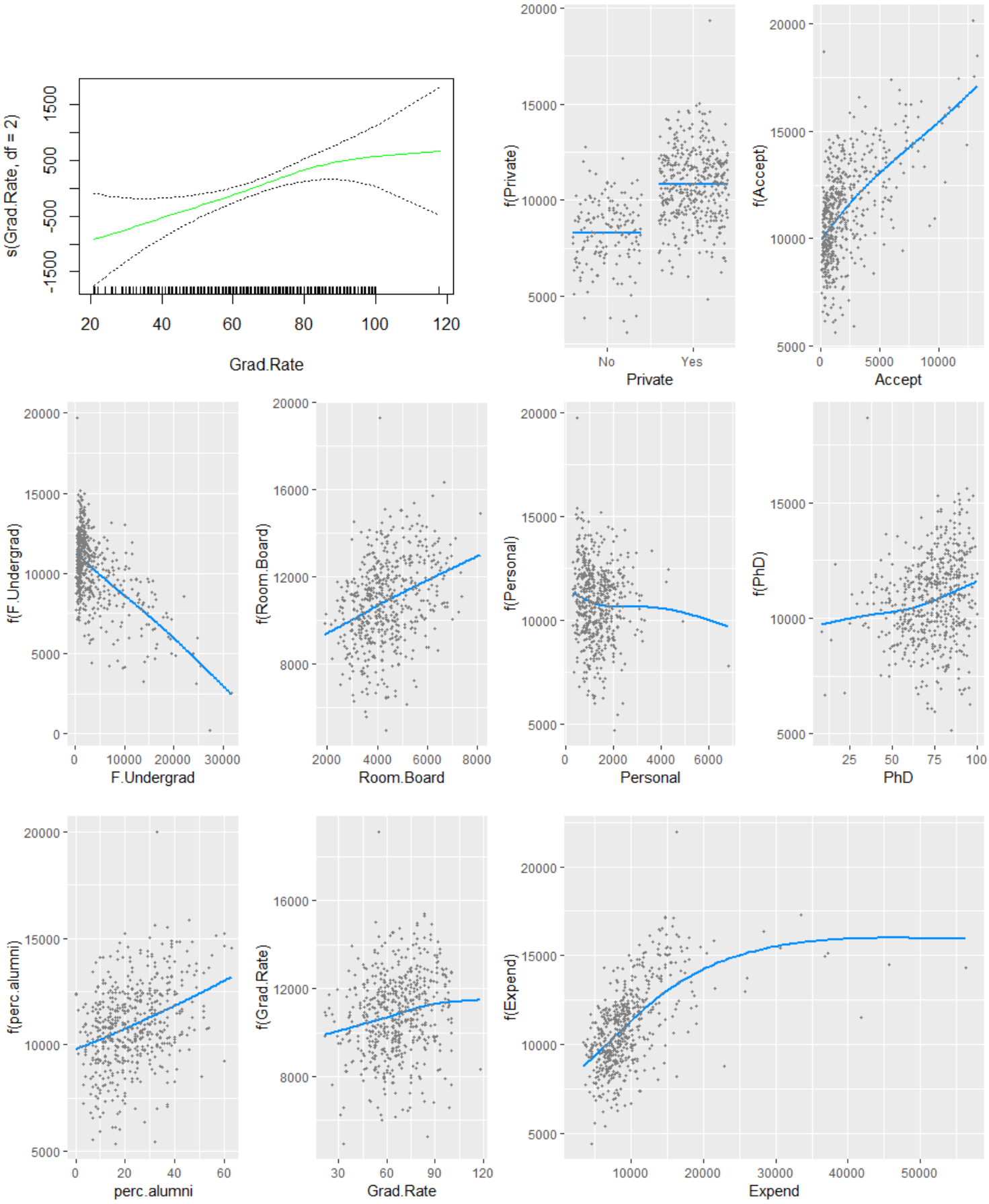

s(Room.Board, df = 2)
-2000
0
2000
2000
3000
4000
5000
6000
7000
8000
Room.Board


s(Personal, df = 2)
-3000
-1000
1000
1000
2000
3000
4000
5000
6000
7000
Personal


s(PhD, df = 2)
-2000
0
1000
20
40
60
80
100
PhD


s(perc.alumni, df = 2)
-1000
0
1000
3000
0
10
20
30
40
50
60
perc.alumni


s(Expend, df = 2)
-2000
2000
6000
10000
20000
30000
40000
50000
Expend


s(Grad.Rate, df = 2)
1500
500
-500
-1500
20
40
60
80
100
120
Grad.Rate
f(Private)
20000
15000
10000
5000
No
Yes
Private
f(Accept)
20000
15000
10000
5000
0
5000
10000
Accept
f(F.Undergrad)
20000
15000
10000
5000
0
0
10000
20000
30000
F.Undergrad
f(Room.Board)
20000
16000
12000
8000
2000
4000
6000
8000
Room.Board
f(Personal)
20000
15000
10000
5000
0
2000
4000
6000
Personal
f(PhD)
15000
10000
5000
25
50
75
100
PhD
f(perc.alumni)
20000
15000
10000
5000
0
20
40
60
perc.alumni
f(Grad.Rate)
16000
12000
8000
30
60
90
120
Grad.Rate
f(Expend)
20000
15000
10000
5000
10000
20000
30000
40000
50000
Expend

Part C: Evaluate the model obtained on the test set and explain the results obtained.

Results: The test error rate is better for the GAM model then for a simple linear model using the same 8 predictors.

```
## [1] "The test error rate of the gam model is: 3792841.20739412"

## [1] "The test error rate of the linear model is: 4227789.84574161"
```

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

Results: There is evidence of a non-linear relationship for the response variable with *Accept, Personal* and *Expend* at an alpha of 0.05.

```
##
## Call: gam(formula = Outstate ~ Private + s(Accept, df = 2) + s(F.Undergrad,
##     df = 2) + s(Room.Board, df = 2) + s(Personal, df = 2) + s(PhD,
##     df = 2) + s(perc.alumni, df = 2) + s(Expend, df = 2) + s(Grad.Rate,
##     df = 2), data = College.train)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -5941.65 -1206.87    51.45  1180.91  8559.38
##
## (Dispersion Parameter for gaussian family taken to be 3435052)
##
##     Null Deviance: 8709842030 on 542 degrees of freedom
## Residual Deviance: 1803403922 on 525.0004 degrees of freedom
## AIC: 9732.563
##
## Number of Local Scoring Iterations: 2
## Anova for Parametric Effects
##                          Df     Sum Sq    Mean Sq F value    Pr(>F)
## Private                   1 2654315615 2654315615 772.715 < 2.2e-16 ***
## s(Accept, df = 2)         1  606886172  606886172 176.674 < 2.2e-16 ***
## s(F.Undergrad, df = 2)    1  289344220  289344220  84.233 < 2.2e-16 ***
## s(Room.Board, df = 2)     1  931048329  931048329 271.043 < 2.2e-16 ***
## s(Personal, df = 2)       1   58859299   58859299  17.135 4.056e-05 ***
## s(PhD, df = 2)            1  510273304  510273304 148.549 < 2.2e-16 ***
## s(perc.alumni, df = 2)    1  373082653  373082653 108.611 < 2.2e-16 ***
## s(Expend, df = 2)         1  504662070  504662070 146.915 < 2.2e-16 ***
## s(Grad.Rate, df = 2)      1   37020563   37020563  10.777  0.001096 **
## Residuals               525 1803403922    3435052
## 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, df = 2)            1  6.355 0.01200 *
## s(F.Undergrad, df = 2)       1  1.056 0.30461
## s(Room.Board, df = 2)        1  0.463 0.49672
## s(Personal, df = 2)          1  6.659 0.01014 *
## s(PhD, df = 2)               1  3.442 0.06413 .
## s(perc.alumni, df = 2)       1  0.364 0.54653
## s(Expend, df = 2)            1 76.657 < 2e-16 ***
## s(Grad.Rate, df = 2)         1  1.633 0.20188
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

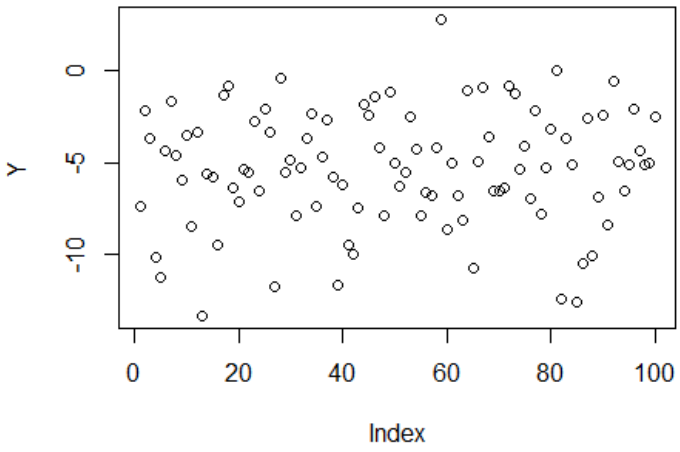
BONUS Question 7.9.11, pg 300: In section 7.7, it was mentioned that GAMs are generally fit using backfitting approach. The idea behind backfitting is actually quite simple. We will now explore backfitting in the context of multiple linear regression.

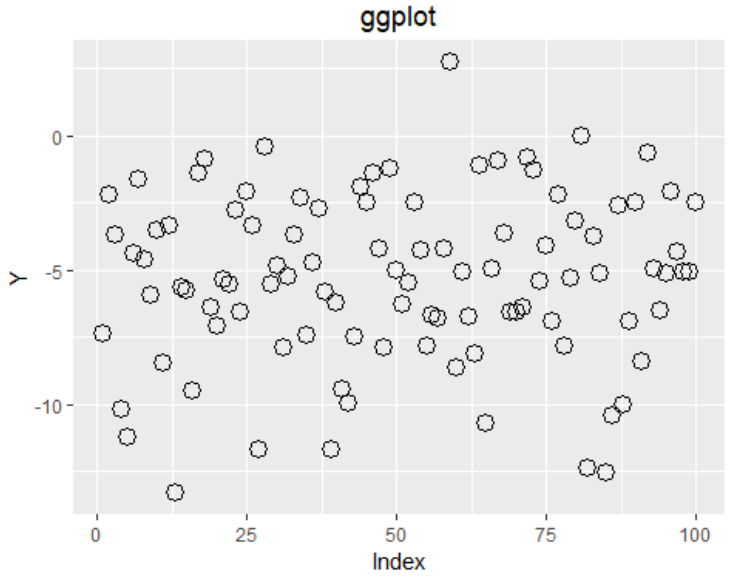
Suppose we would like to perform multiple linear regression, but we do not have the software to do so. Instead, we only have software to perform simple linear regression. Therefore, we take the following iterative approach: we repeatedly hold all but one coefficient estimate fixed at its current value, 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.

Part A: Generate a response **Y** and two predictors **X1** and **X2** with n = 100.

Results: First, I randomly generated the X values. Then I created β and ϵ values and set Y. Then I plotted Y to confirm a result was generated.





Part B: Initialize β_1 to take on a value of your choice. It does not matter what value you choose.

Results: I set betahat_1 equal to 1.

```
# set betahat_1 equal to 1
betahat_1 <- 1
```

Part C: Keeping β_1 fixed, fit the model $Y - \beta_1 X_1 = \beta_0 + \beta_2 X_2 + \epsilon$

Results: I used the recommended code from the text and printed the value of betahat_2.

```
## [1] "betahat_2 = -3.50895741989353"
```

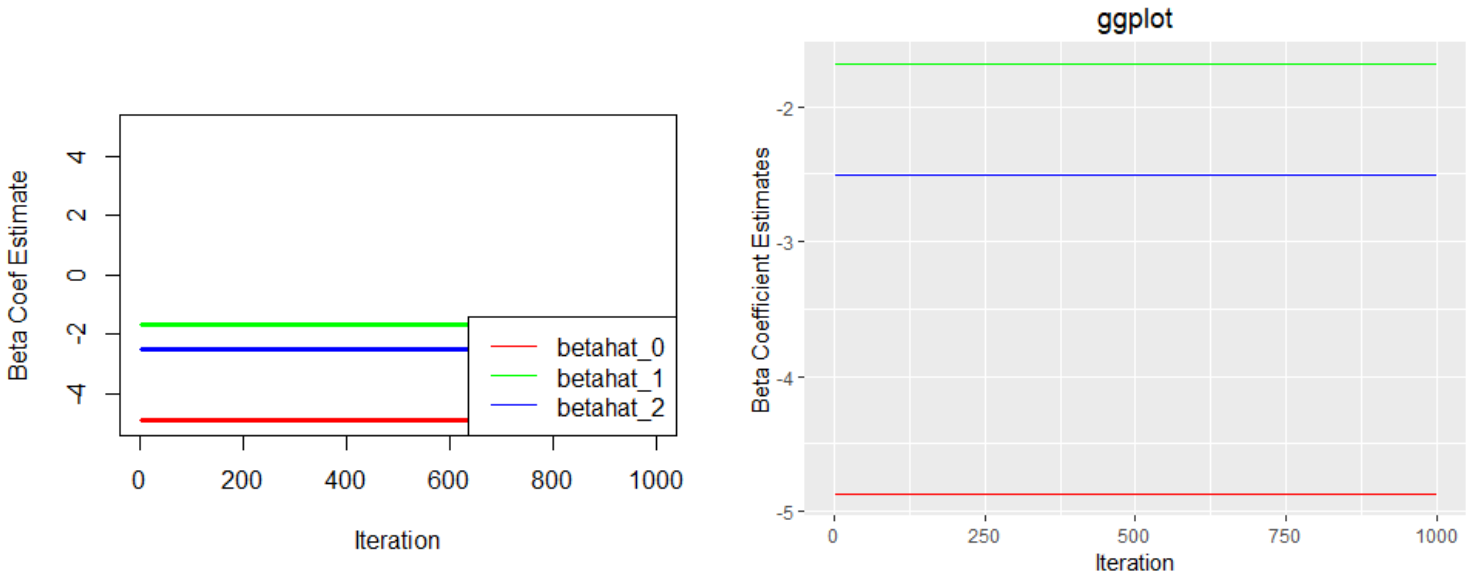
Part D: Keeping β_2 fixed, fit the model $Y - \beta_2 X_2 = \beta_0 + \beta_1 X_1 + \epsilon$

Results: I used the recommended code from the text and printed the updated value for betahat_1.

```
## [1] "betahat_1 = -1.74593491339584"
```

Part E: Write a loop to repeat (c) and (d) 1,000 times. Report the estimates of $\beta_0, \beta_1, \beta_2$ at each iteration of the for loop. Create a plot in which each of these values is displayed, with each beta shown in a different color.

Results: I created a loop that iterates 1000 times. Interestingly, the beta values converged on the first iteration of the loop. The plot and table below show the convergence and the values for each of the three beta coefficient estimates.



First 6 Beta Coefficient Estimates of 1000 Iterations

Iteration	betahat_0	betahat_1	betahat_2
1	-4.875034	-1.683967	-2.508957
2	-4.875034	-1.683967	-2.508957
3	-4.875034	-1.683967	-2.508957
4	-4.875034	-1.683967	-2.508957
5	-4.875034	-1.683967	-2.508957
6	-4.875034	-1.683967	-2.508957

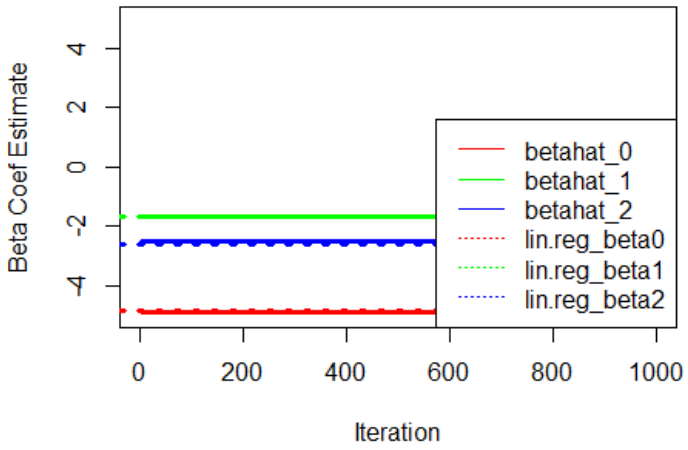
Part F: Compare your answer in (e) to the results of simply performing multiple linear regression to predict *Y* using *X1* and *X2*. Use the *abline()* function to overlay those multiple linear regression coefficient estimates on the plot obtained in (e).

Results: The table below shows the estimates from using multiple linear regression. As we can see, the values are reasonably close to the estimates received from the loop in the previous exercise. The plot further depicts the similarities between the two methods of extracting the beta coefficient estimates.

The dashed lines represent the coefficients as estimated by the lm funciton. The solid lines represent the coefficients as estimated by the loop in Part E. The red lines correspond to the betahat_0 estimates, the green lines correspond to the betahat_1 estimates, and the blue lines correspond to the betahat_2 estimates.

Estimates from Multiple Linear Regression

	Beta Coefficient Estimates
(Intercept)	-4.861481
X1	-1.689730
X2	-2.601960



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

Results: As we can see from the tables and calculations below, the 1st iteration was sufficient to obtain an estimate that is quite close to the estimate obtained through multiple linear regression in the exercise above.

Beta Coefficient Estimates from Iterations

Iteration	betahat_0	betahat_1	betahat_2
1	-4.875034	-1.683967	-2.508957
2	-4.875034	-1.683967	-2.508957
3	-4.875034	-1.683967	-2.508957
4	-4.875034	-1.683967	-2.508957
5	-4.875034	-1.683967	-2.508957
6	-4.875034	-1.683967	-2.508957

Beta Estimates from Multiple Linear Regression

	Beta Coefficient Estimates
(Intercept)	-4.861481
X1	-1.689730
X2	-2.601960

```
## [1] "The difference in estimates for beta 0 = -0.0135527936544211"


## [1] "The difference in estimates for beta 1 = -0.00576318903300654"


## [1] "The difference in estimates for beta 2 = -0.0930028398777365"
```