Data Science II Homework 2

Camille Okonkwo

Contents

```r
library(tidymodels)
library(splines)
library(caret)
```

Partition the dataset into two parts: training data (80%) and test data (20%) with `tidymodels`.

```r
college = read_csv("data/College.csv") |>
  drop_na() |>
  select(-College)

set.seed(2)

# create a random split of 80% training and 20% test data
data_split <- initial_split(data = college, prop = 0.8)

# partitioned datasets
training_data = training(data_split)
testing_data = testing(data_split)

head(training_data)
```

```
## # A tibble: 6 x 17
##    Apps Accept Enroll Top10perc Top25perc F.Undergrad P.Undergrad Outstate
##   <dbl>  <dbl>  <dbl>     <dbl>     <dbl>       <dbl>       <dbl>    <dbl>
## 1  1380    768    263        57        82        1000         105    19300
## 2   434    321    141        28        53         624         269    10950
## 3  2013   1053    212        33        61         912         158     5150
## 4  2324   1319    370        52        81        1686          35    16560
## 5  1709   1385    634        36        72        2281          50    14125
## 6   427    385    143        18        38         581         533    12700
## # i 9 more variables: Room.Board <dbl>, Books <dbl>, Personal <dbl>, PhD <dbl>,
## #   Terminal <dbl>, S.F.Ratio <dbl>, perc.alumni <dbl>, Expend <dbl>,
## #   Grad.Rate <dbl>
```

```r
head(testing_data)
```

```
## # A tibble: 6 x 17
##    Apps Accept Enroll Top10perc Top25perc F.Undergrad P.Undergrad Outstate
##   <dbl>  <dbl>  <dbl>     <dbl>     <dbl>       <dbl>       <dbl>    <dbl>
## 1  2186   1924    512        16        29        2683        1227    12280
## 2  1428   1097    336        22        50        1036          99    11250
## 3   193    146     55        16        44         249         869     7560
## 4   582    498    172        21        44         799          78    10468
## 5  1732   1425    472        37        75        1830         110    16548
## 6   494    313    157        23        46        1317        1235     8352
## # i 9 more variables: Room.Board <dbl>, Books <dbl>, Personal <dbl>, PhD <dbl>,
## #   Terminal <dbl>, S.F.Ratio <dbl>, perc.alumni <dbl>, Expend <dbl>,
## #   Grad.Rate <dbl>
```

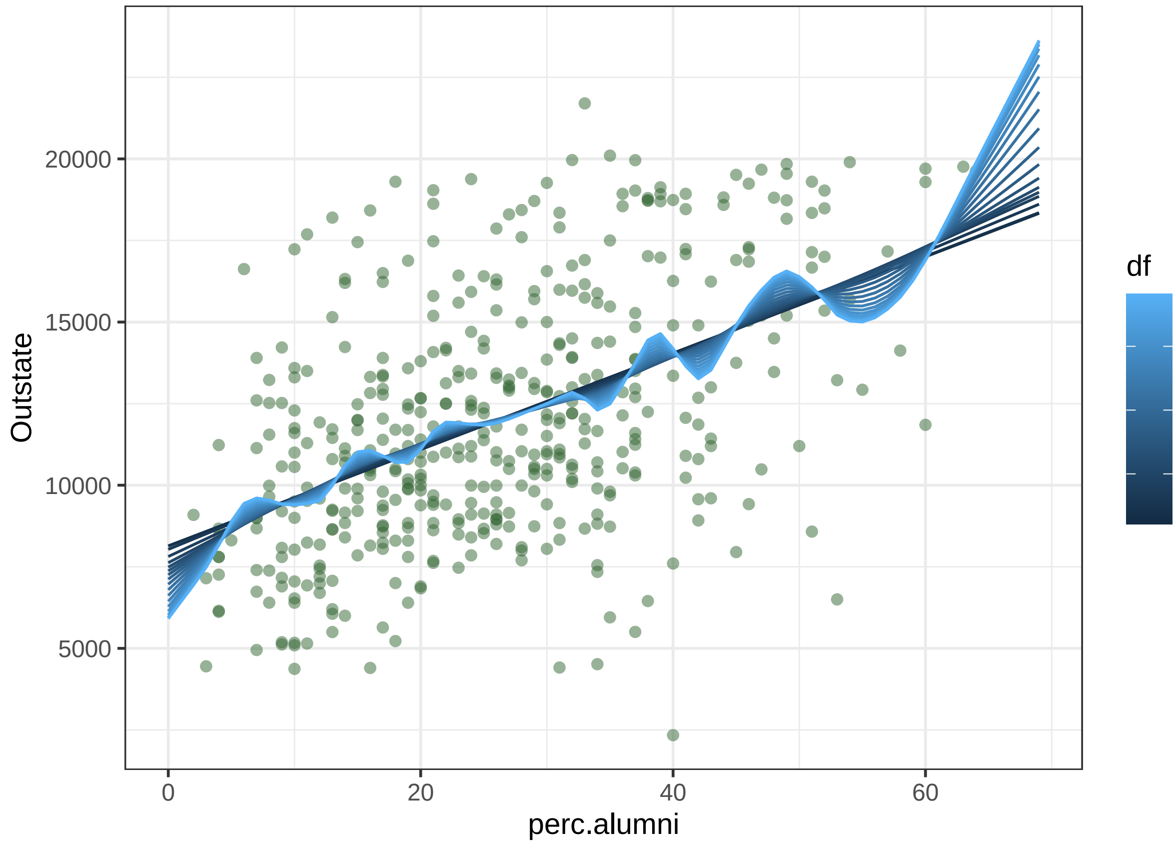
```r
# training data
x <- model.matrix(Outstate ~ ., training_data)[, -1] # matrix of predictors
head(x)
```

```
##   Apps Accept Enroll Top10perc Top25perc F.Undergrad P.Undergrad Room.Board
## 1 1380    768    263        57        82        1000         105       6694
## 2  434    321    141        28        53         624         269       4600
## 3 2013   1053    212        33        61         912         158       3036
## 4 2324   1319    370        52        81        1686          35       5140
## 5 1709   1385    634        36        72        2281          50       3600
## 6  427    385    143        18        38         581         533       5800
##   Books Personal PhD Terminal S.F.Ratio perc.alumni Expend Grad.Rate
## 1   600      700  89       93       6.1          18  14779        83
## 2   550      950  79       82      12.9          30   9264        81
## 3   500     1655  64       74      10.5          11   7547        59
## 4   558     1152  91       93      10.5          30  16196        79
## 5   400      700  79       89      12.5          58   9907        80
## 6   450      700  81       85      10.3          37  11758        84
```

```
y <- training_data$Outstate # vector of response

# testing data
x2 <- model.matrix(Outstate ~ .,testing_data)[, -1] # matrix of predictors
y2 <- testing_data$Outstate # vector of response
```

1a) Fit smoothing spline models to predict out-of-state tuition (Outstate) using the percentage of alumni who donate (perc.alumni) as the only predictor, across a range of degrees of freedom. Plot the model fits for each degree of freedom. Describe the observed patterns that emerge with varying degrees of freedom. Select an appropriate degree of freedom for the model and plot this optimal fit. Explain the criteria you used to determine the best choice of degree of freedom.

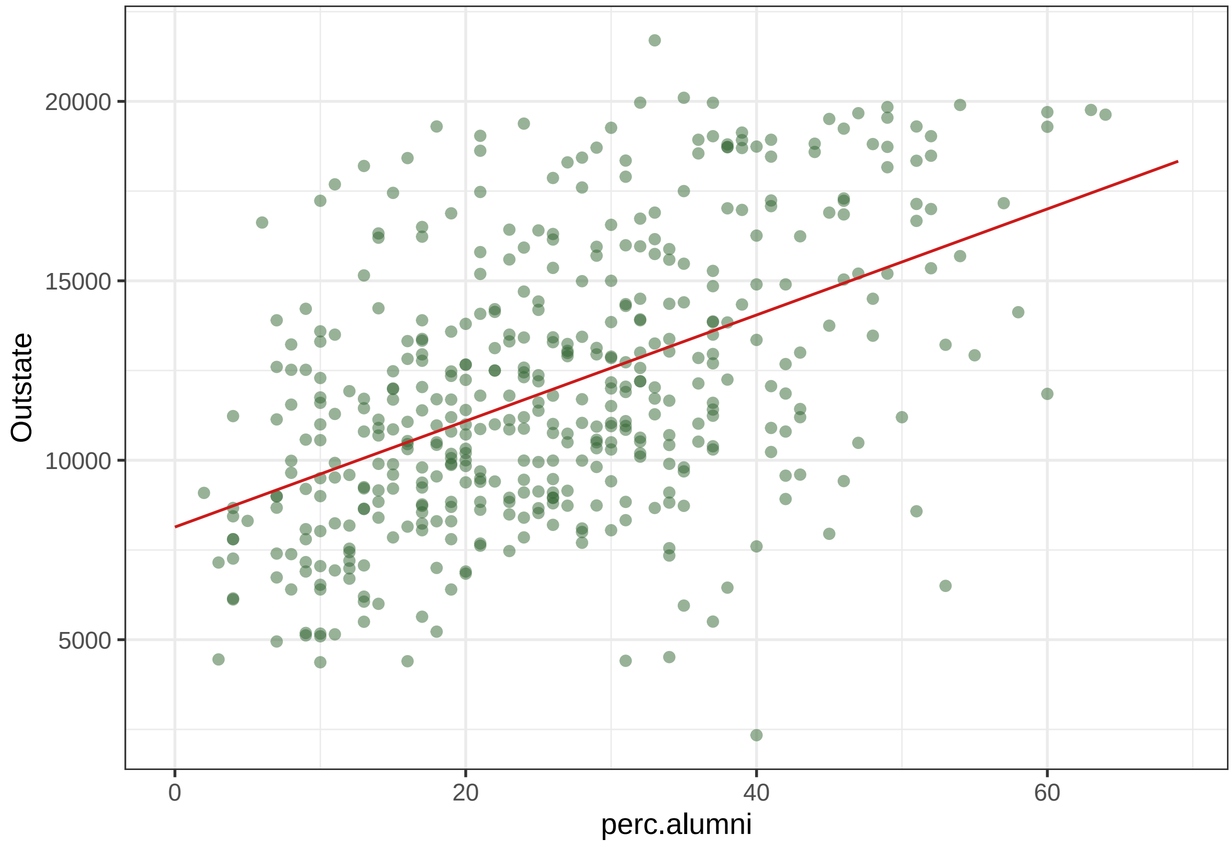
```
# create a grid for x
perc.alumni.grid <- seq(0, max(college$perc.alumni) + 5, by = 1)

# loop prep
fit.ss = list()
pred.ss = list()
pred.ss.df = list()
pred.ss.df.range = data.frame()

set.seed(2)
# loop for a range of degrees of freedom
for (i in 1.1:20) {
  fit.ss[[i]] = smooth.spline(training_data$perc.alumni, training_data$Outstate, df = i)
  pred.ss[[i]] = predict(fit.ss[[i]], x = perc.alumni.grid)
  pred.ss.df[[i]] = data.frame(pred = pred.ss[[i]]$y, perc.alumni = perc.alumni.grid, df = i)
  pred.ss.df.range = rbind(pred.ss.df[[i]], pred.ss.df.range)
}

# scatter plot
p <- ggplot(data = training_data, aes(x = perc.alumni, y = Outstate)) + geom_point(color = rgb(0.2, 0.4

# plot the model fits for each degree of freedom
p +
  geom_line(aes(x = perc.alumni, y = pred, group = df, color = df), data = pred.ss.df.range) + theme_bw
```




```r
set.seed(2)

# select an appropriate degree of freedom for the model
fit.ss.optimal = smooth.spline(training_data$perc.alumni, training_data$Outstate)

fit.ss.optimal$df
```

```
## [1] 2.000245
```

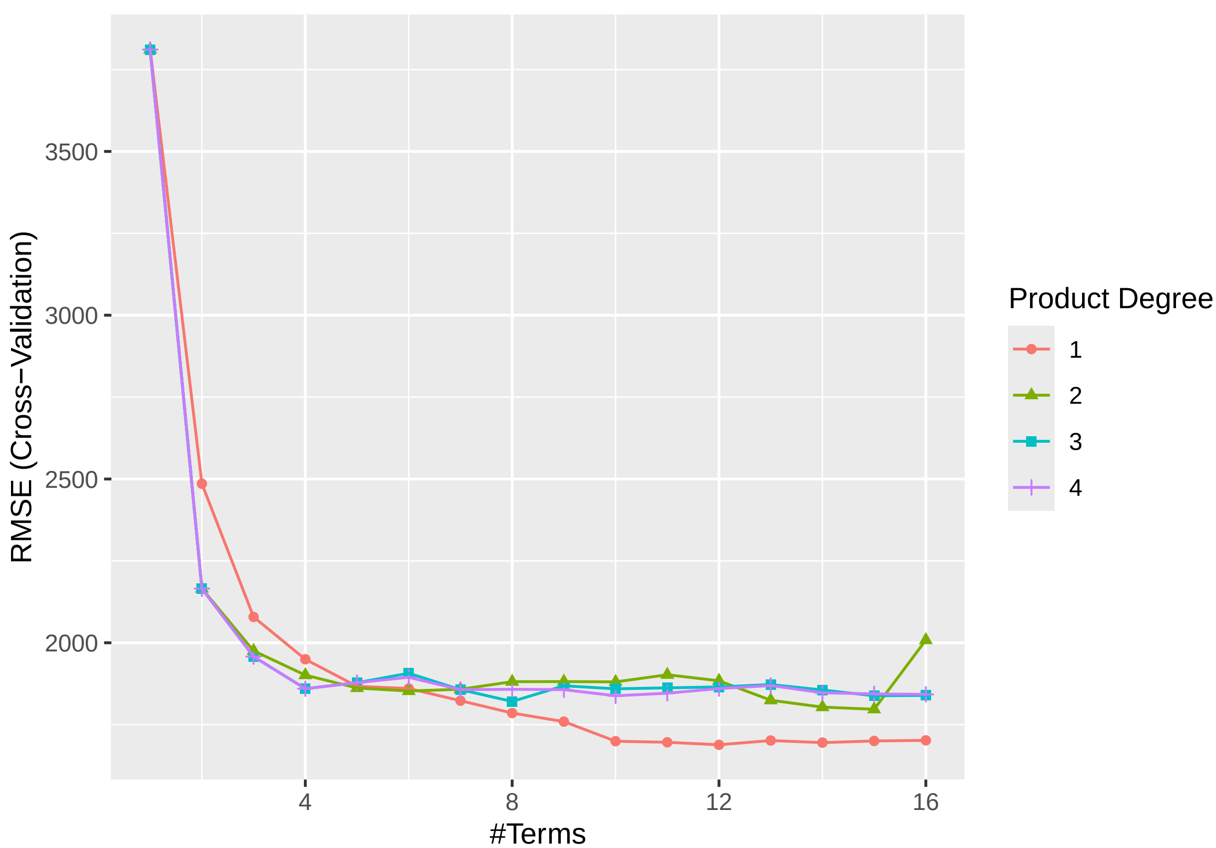
```r
# predicted values
pred.ss.optimal <- predict(fit.ss.optimal,
                    x = perc.alumni.grid)

pred.ss.optimal.df <- data.frame(pred = pred.ss.optimal$y,
                          perc.alumni = perc.alumni.grid)

# plot this optimal fit
p.optimal <- ggplot(data = training_data, aes(x = perc.alumni, y = Outstate)) + geom_point(color = rgb(

p.optimal +
  geom_line(aes(x = perc.alumni, y = pred), data = pred.ss.optimal.df,
            color = rgb(0.8, 0.1, 0.1, 1)) + theme_bw()
```



When the degrees of freedom is smaller, the model resembles a linear model. As the degrees of freedom increase, the model becomes more flexible and we can see that exemplified through the wavy lines. For the optimal smoothing splines model, the degrees of freedom = 2.0002451. To determine the best choice of degrees of freedom, we can use cross-validation to choose the degrees of freedom that result in the best predictive performance.

1b) Train a multivariate adaptive regression spline (MARS) model to predict the response variable. Report the regression function. Present the partial dependence plot of an arbitrary predictor in your model. Report the test error.

```
# 10-fold cross-validation
ctrl <- trainControl(method = "cv", number = 10)

# set grid
mars_grid <- expand.grid(degree = 1:4, nprune = 1:16)

set.seed(2)

# fit a MARS model
mars.fit <- train(x, y,
                  method = "earth",
                  tuneGrid = mars_grid,
                  trControl = ctrl)
# plot
ggplot(mars.fit)
```



```r
# best tuning parameters
mars.fit$bestTune
```

```
##    nprune degree
## 12     12      1
```

```r
# regression function
mars.fit$finalModel
```

```
## Selected 12 of 22 terms, and 9 of 16 predictors (nprune=12)
## Termination condition: RSq changed by less than 0.001 at 22 terms
## Importance: Expend, Grad.Rate, Room.Board, Accept, Enroll, F.Undergrad, ...
## Number of terms at each degree of interaction: 1 11 (additive model)
## GCV 2729781    RSS 1111485831    GRSq 0.8134662    RSq 0.8312207
```

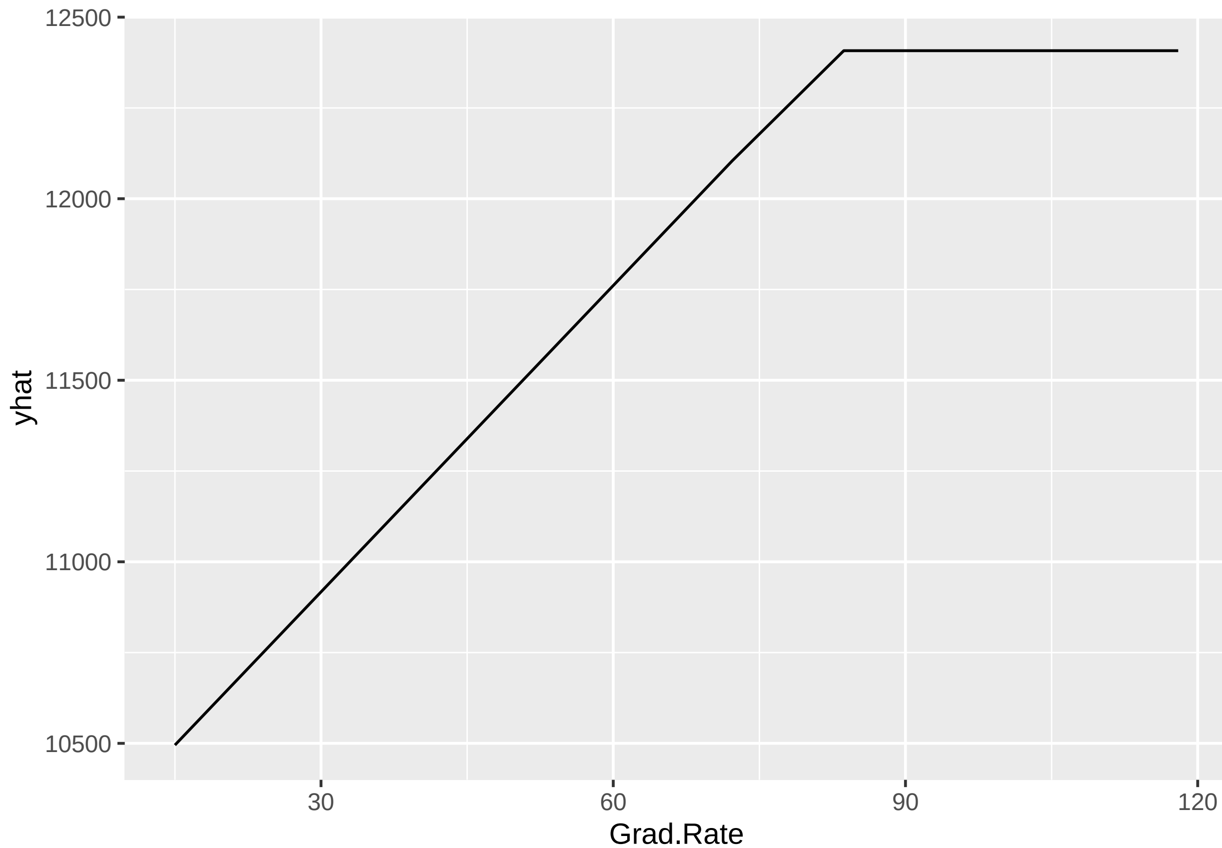
```r
# report the regression function
summary(mars.fit)
```

```
## Call: earth(x=matrix[452,16], y=c(19300,10950,5...), keepxy=TRUE, degree=1,
##             nprune=12)
##
##                     coefficients
## (Intercept)            9748.1791
## h(Apps-3646)              0.4632
## h(2279-Accept)           -1.8972
## h(913-Enroll)             6.3690
## h(Enroll-913)            -1.9849
## h(1363-F.Undergrad)      -1.9452
## h(5895-Room.Board)       -0.6939
## h(1230-Personal)          0.8542
## h(perc.alumni-6)         25.4428
## h(Expend-6864)            0.7506
## h(Expend-15387)          -0.7703
## h(83-Grad.Rate)         -28.1240
##
## Selected 12 of 22 terms, and 9 of 16 predictors (nprune=12)
## Termination condition: RSq changed by less than 0.001 at 22 terms
## Importance: Expend, Grad.Rate, Room.Board, Accept, Enroll, F.Undergrad, ...
## Number of terms at each degree of interaction: 1 11 (additive model)
## GCV 2729781    RSS 1111485831    GRSq 0.8134662    RSq 0.8312207
```

```r
coef(mars.fit$finalModel)
```

```
##         (Intercept)     h(Expend-15387)     h(83-Grad.Rate)  h(5895-Room.Board)
##        9748.1790945          -0.7702870         -28.1240387          -0.6939388
## h(1363-F.Undergrad)    h(1230-Personal)        h(Apps-3646)       h(Enroll-913)
##          -1.9451538           0.8542158           0.4631943          -1.9848536
##       h(913-Enroll)      h(2279-Accept)    h(perc.alumni-6)      h(Expend-6864)
##           6.3690181          -1.8971697          25.4427652           0.7506273
```

```r
# partial dependence plot on arbitrary predictor Grad.Rate
p1 <- pdp::partial(mars.fit, pred.var = c("Grad.Rate"), grid.resolution = 10) |>
  autoplot()

p1
```




```r
# test error
pred.mars <- predict(mars.fit, newdata = testing_data)

test.error.mars <- mean((pred.mars - y2)^2)
```

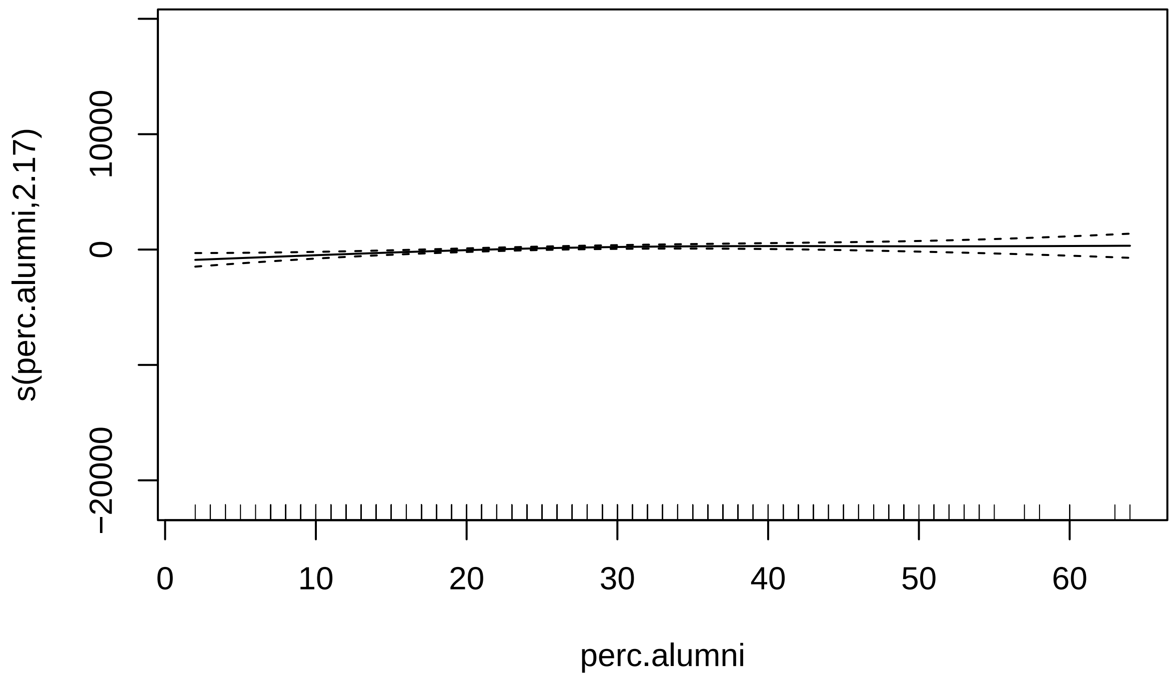
The regression function for the MARS model is **f(Outstate) = 9748.1791 + 0.4632 h(Apps−3646) − 1.8972 h(2279-Accept) + 6.3690 h(913-Enroll) − 1.9849 h(Enroll−913) − 1.9452 h(1363-F.Undergrad) − 0.6939 h(5895-Room.Board) + 0.8542 h(1230-Personal) + 25.4428 h(perc.alumni−6) + 0.7506 h(Expend−6864) − 0.7703 h(Expend−15387) − 28.1240 h(83-Grad.Rate)**. The test error is 3.4360573×10^6.

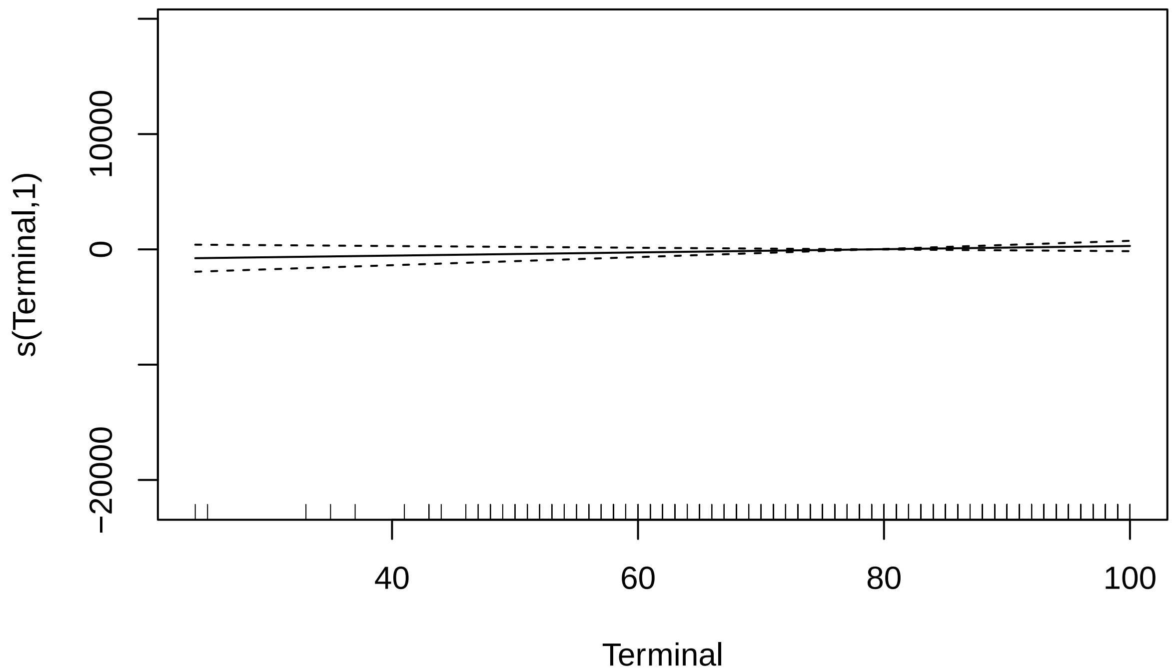
1c) Construct a generalized additive model (GAM) to predict the response variable. Does your GAM model include all the predictors? For the nonlinear terms included in your model, generate plots to visualize these relationships and discuss your observations. Report the test error.

```r
set.seed(2)

# fit a GAM model using 10-fold cross-validation
gam.fit <- train(x, y,
                 method = "gam",
                 tuneGrid = data.frame(method = "GCV.Cp", select = c(TRUE, FALSE)),
                 trControl = ctrl)

# for non-linear terms, generate plots to visualize relationships
gam.fit$finalModel
```

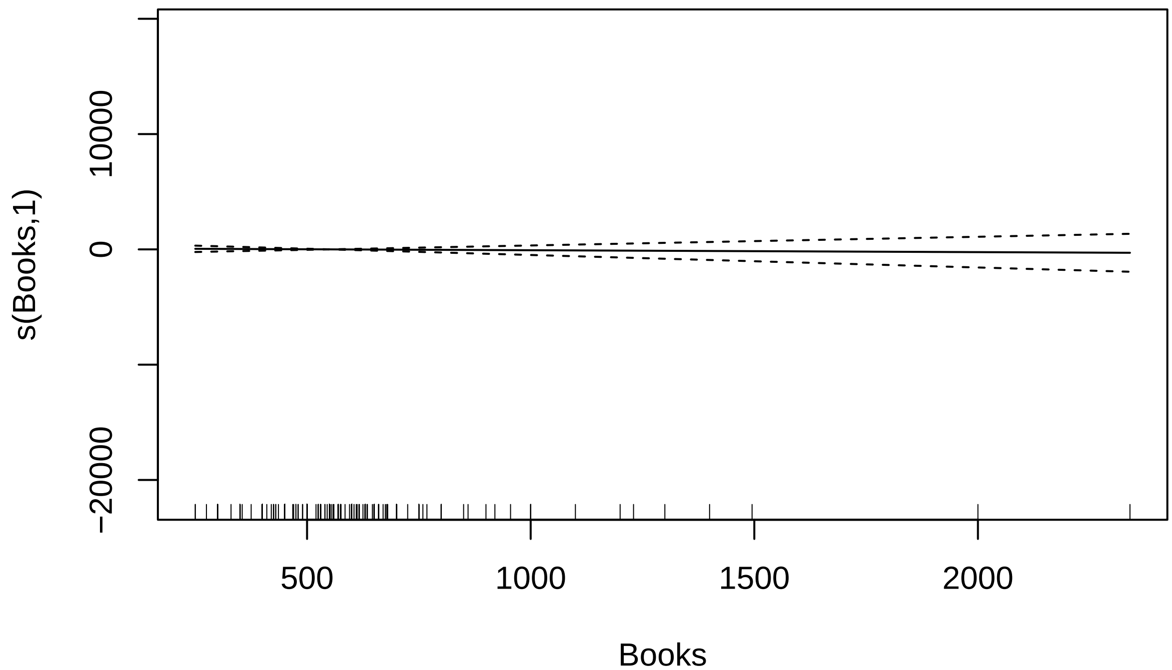
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ s(perc.alumni) + s(Terminal) + s(Books) + s(PhD) +
##     s(Grad.Rate) + s(Top10perc) + s(Top25perc) + s(S.F.Ratio) +
##     s(Personal) + s(P.Undergrad) + s(Room.Board) + s(Enroll) +
##     s(Accept) + s(F.Undergrad) + s(Apps) + s(Expend)
##
## Estimated degrees of freedom:
## 2.17 1.00 1.00 2.32 4.49 7.59 1.00
## 3.74 1.00 1.00 2.50 1.00 2.84 6.17
## 3.89 7.39  total = 50.1
##
## GCV score: 2689385
```

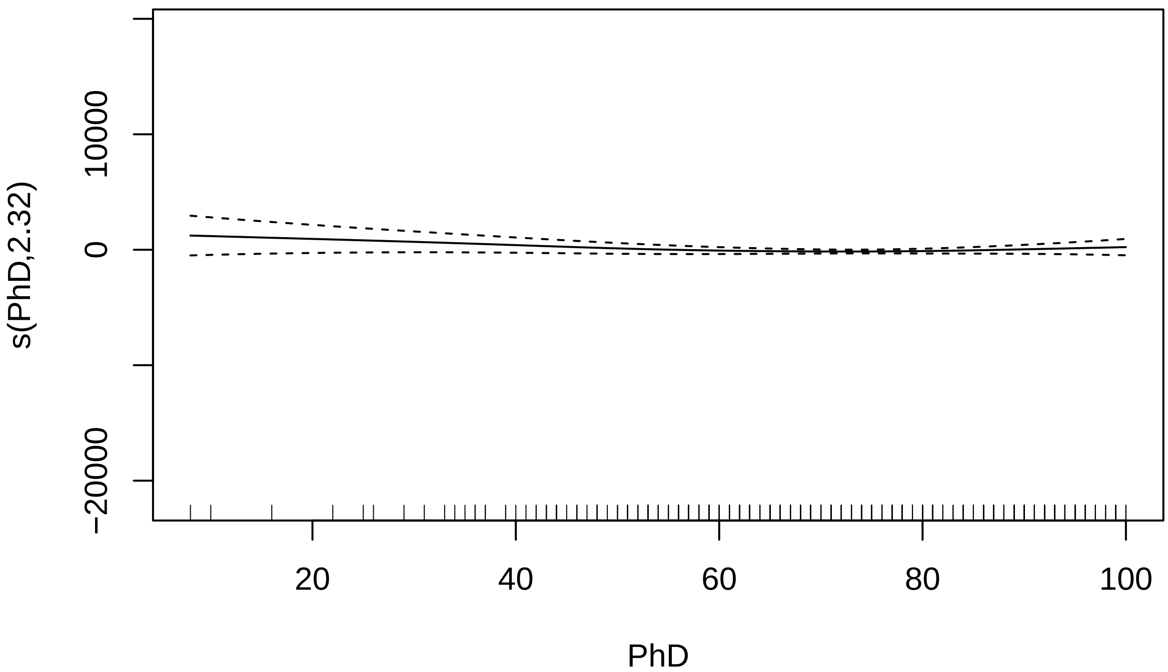
```r
plot(gam.fit$finalModel)
```


s(perc.alumni,2.17)
10000
0
−20000
0
10
20
30
40
50
60
perc.alumni

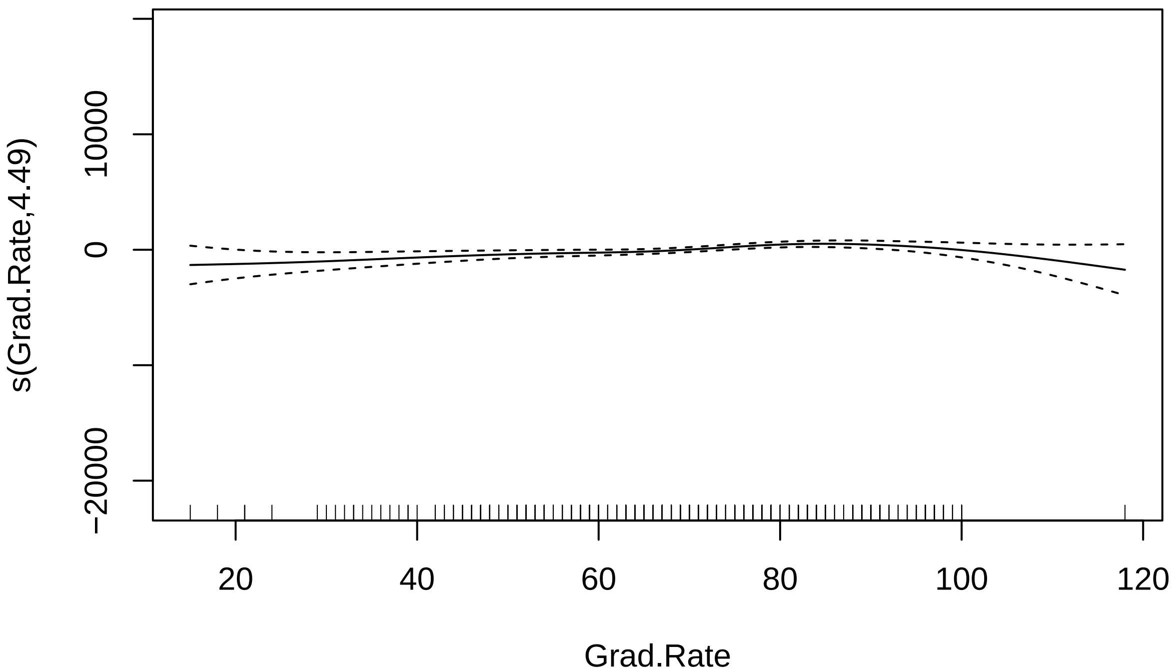


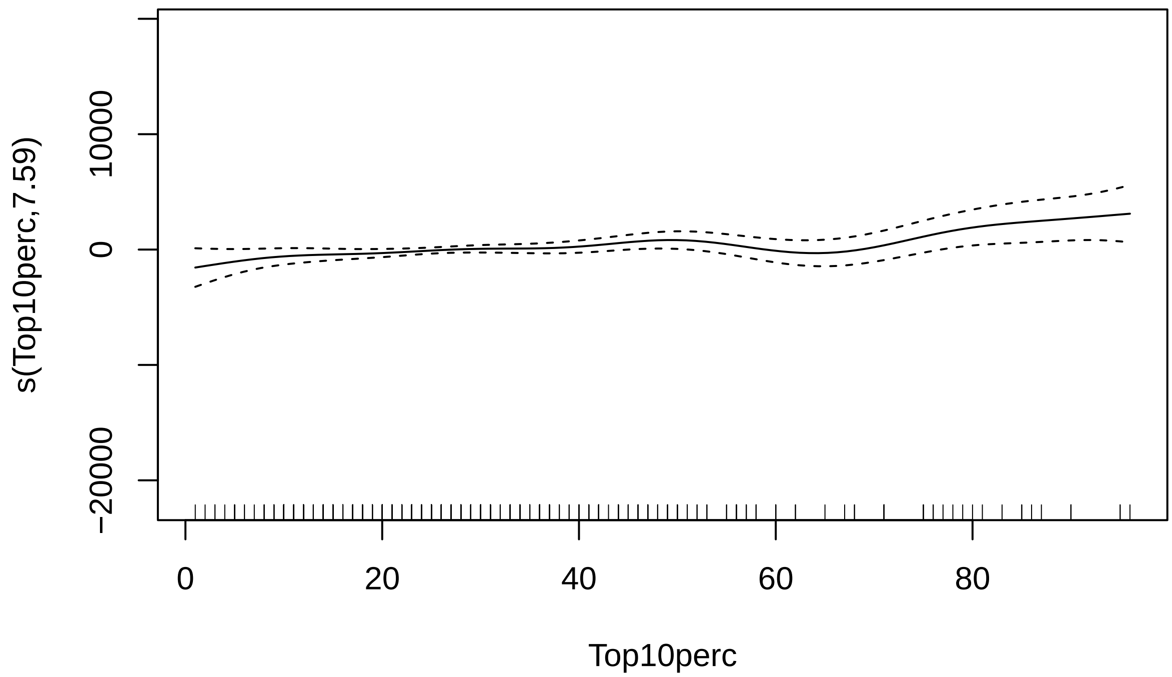
s(Terminal,1)
−20000
0
10000
40
60
80
100
Terminal


s(Books,1)
10000
0
−20000
500
1000
1500
2000
Books

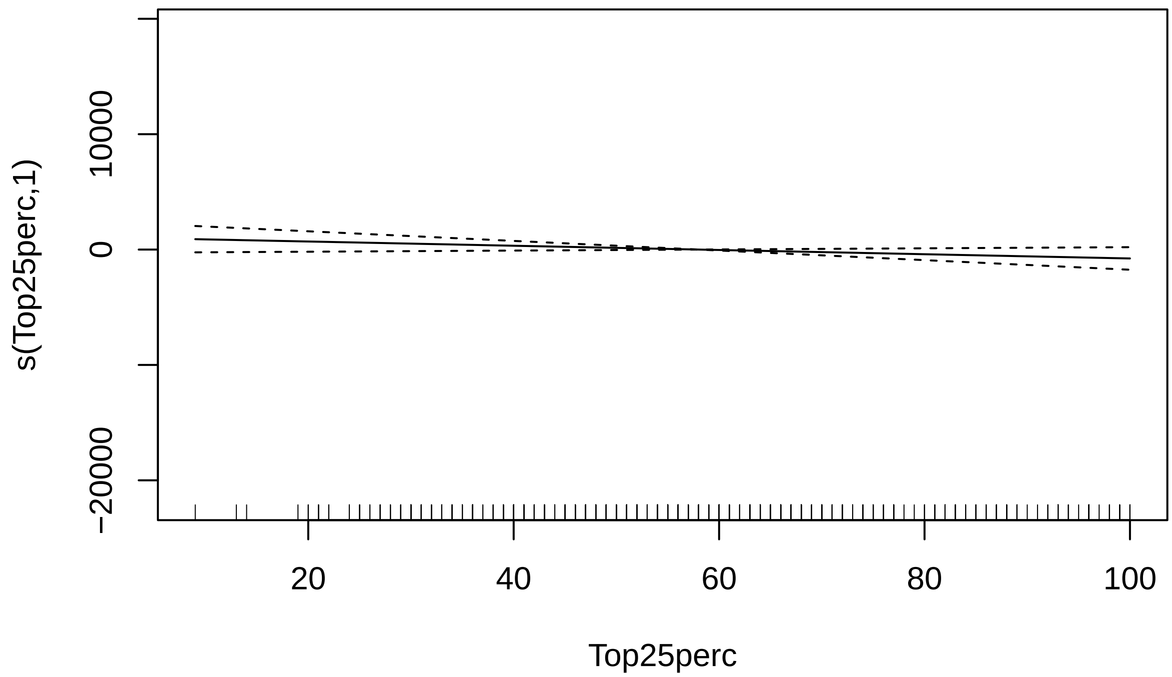


s(PhD,2.32)
10000
0
−20000
20
40
60
80
100
PhD

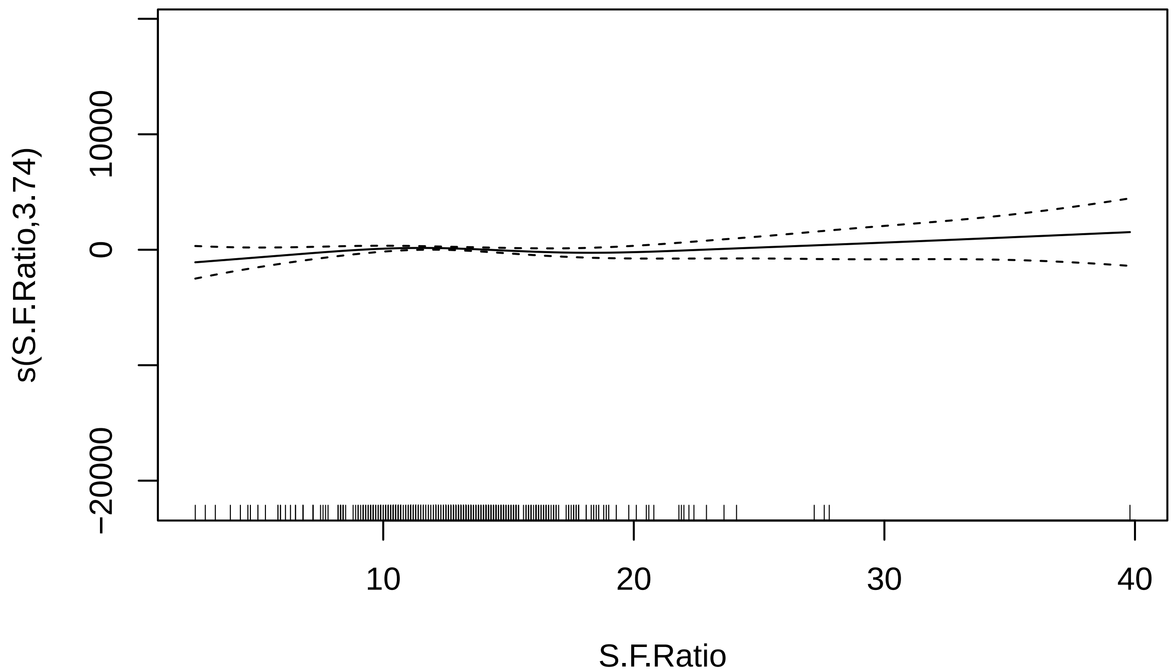

s(Grad.Rate,4.49)
10000
0
−20000
20
40
60
80
100
120
Grad.Rate



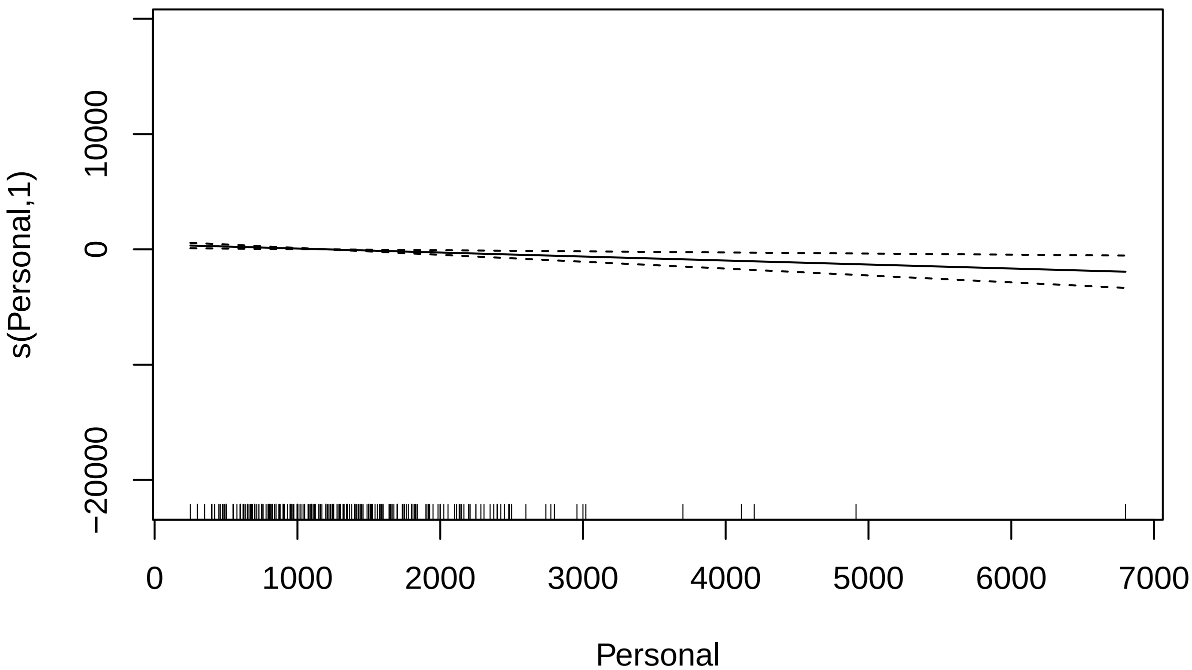
s(Top10perc,7.59)
10000
0
−20000
0
20
40
60
80
Top10perc

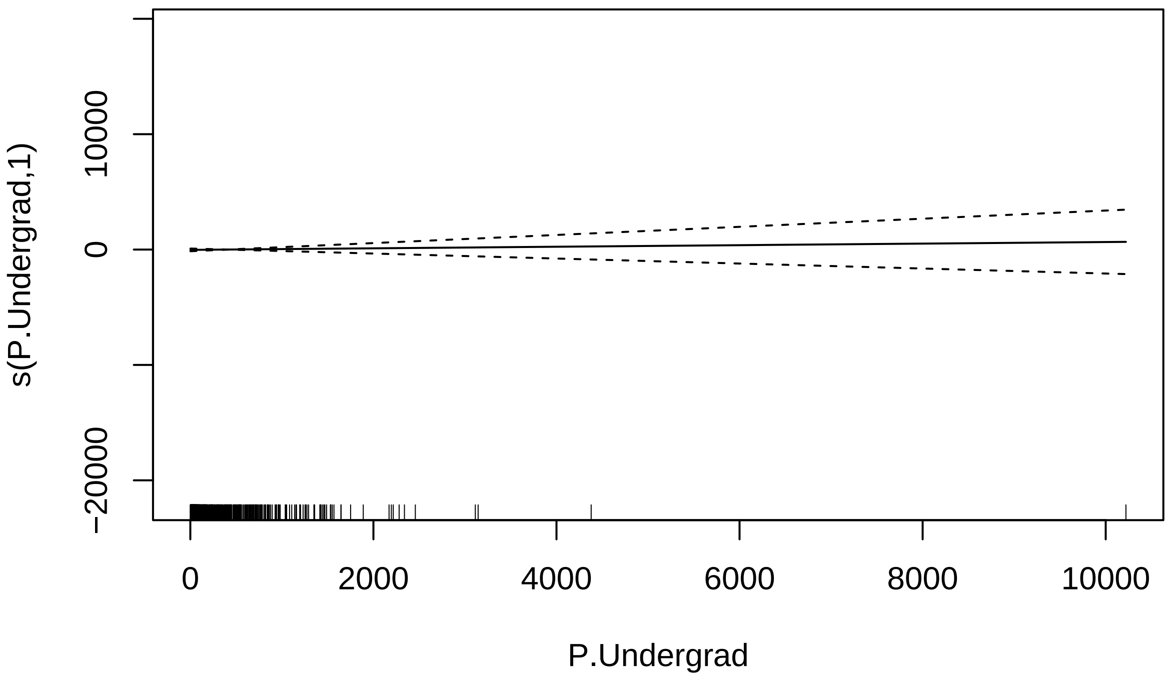


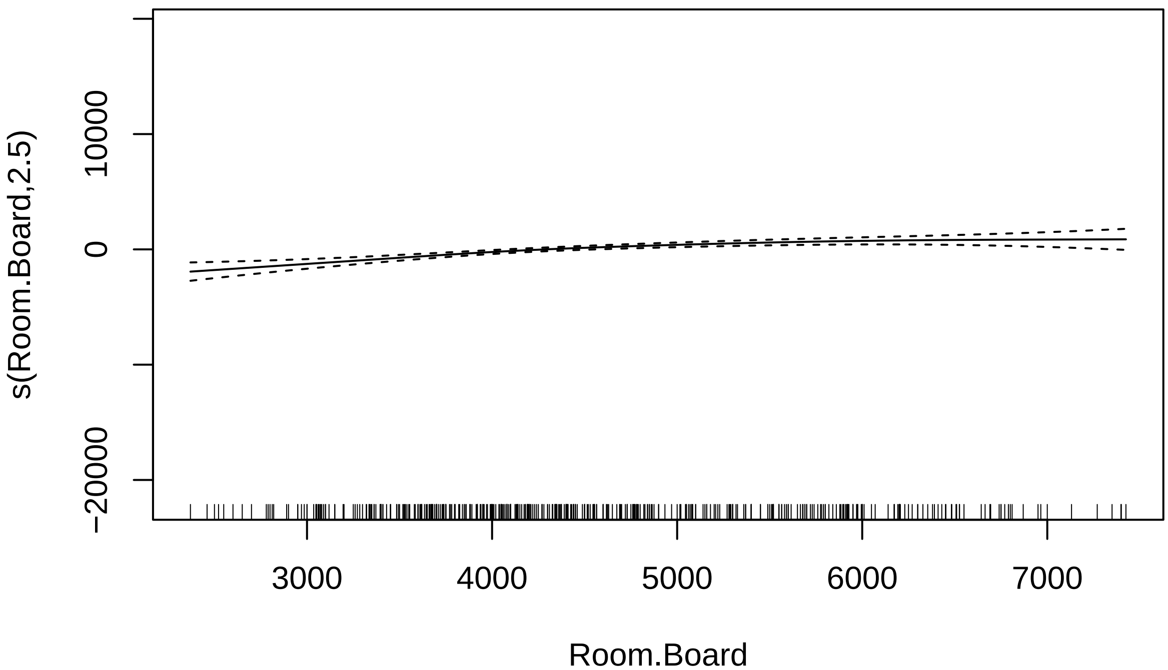
s(Top25perc,1)
10000
0
−20000
20
40
60
80
100
Top25perc

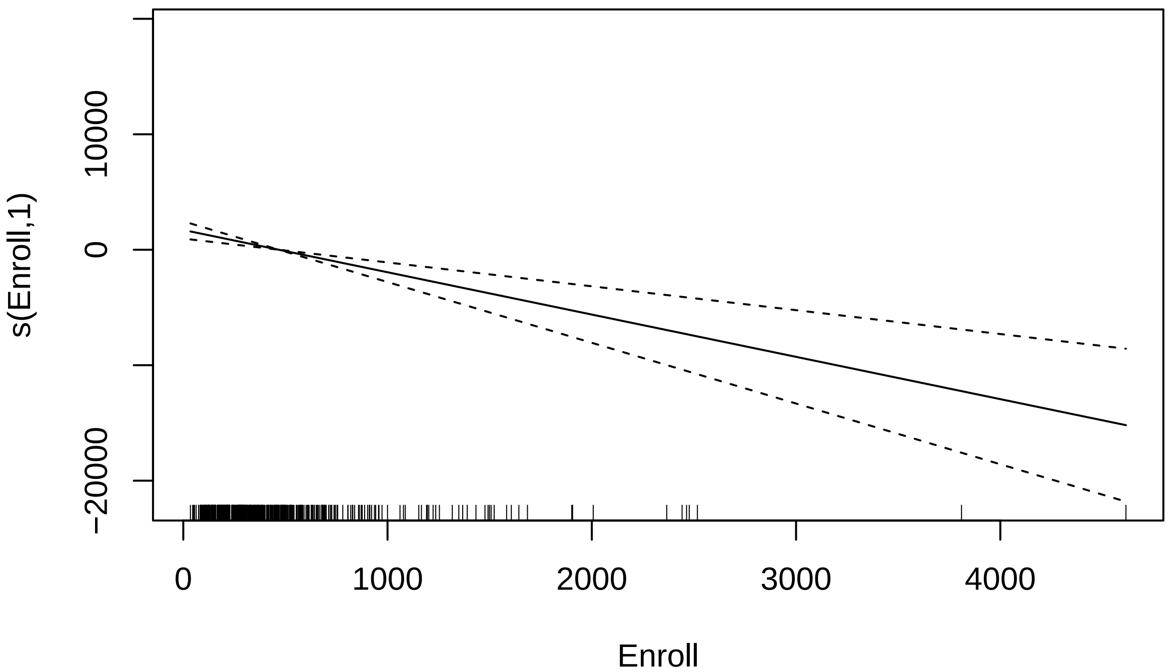


s(S.F.Ratio,3.74)
−20000
0
10000
10
20
30
40
S.F.Ratio

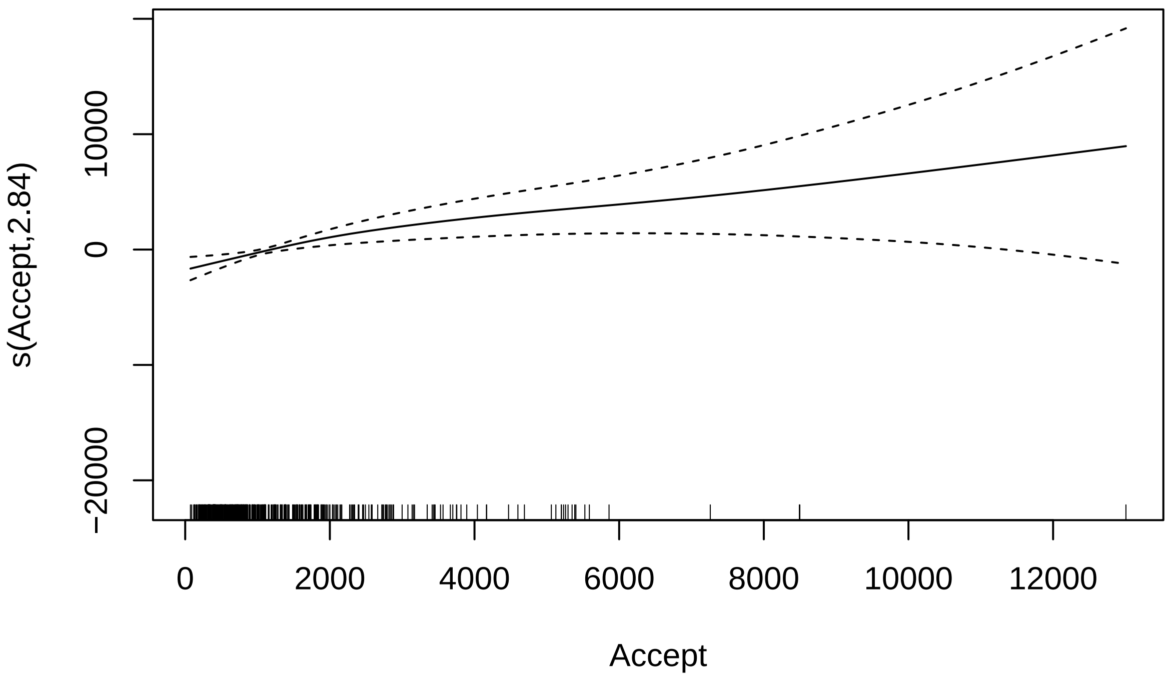

s(Personal,1)
10000
0
−20000
0
1000
2000
3000
4000
5000
6000
7000
Personal

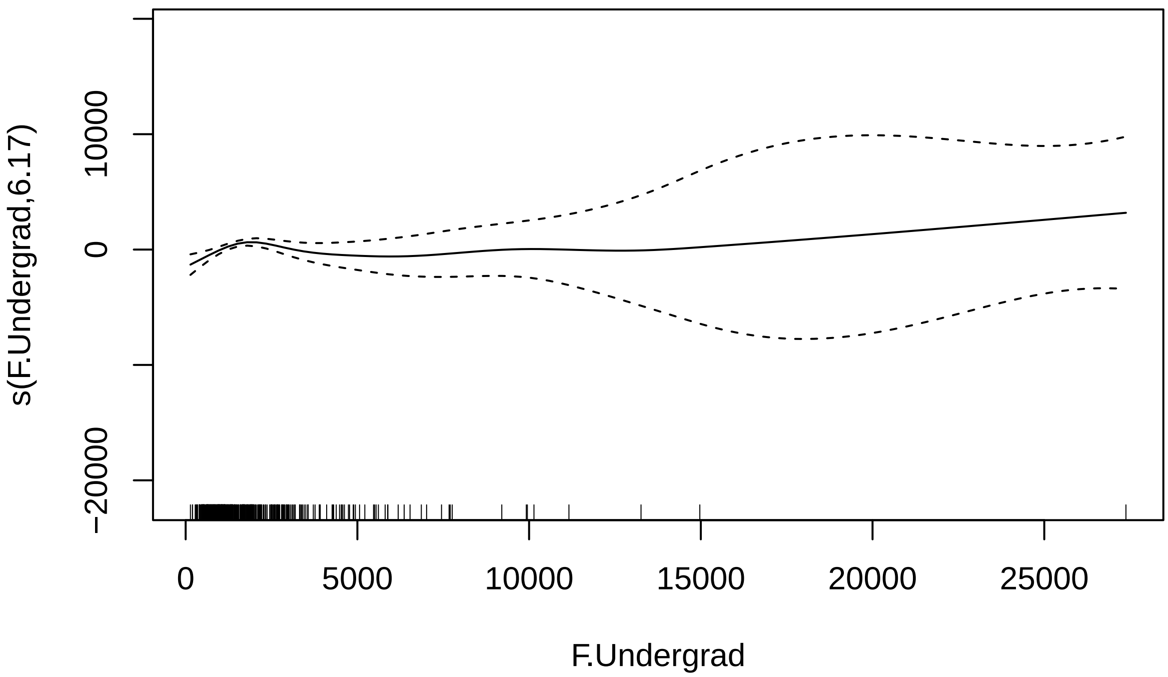

s(P.Undergrad,1)
10000
0
−20000
0
2000
4000
6000
8000
10000
P.Undergrad


s(Room.Board,2.5)
10000
0
−20000
3000
4000
5000
6000
7000
Room.Board

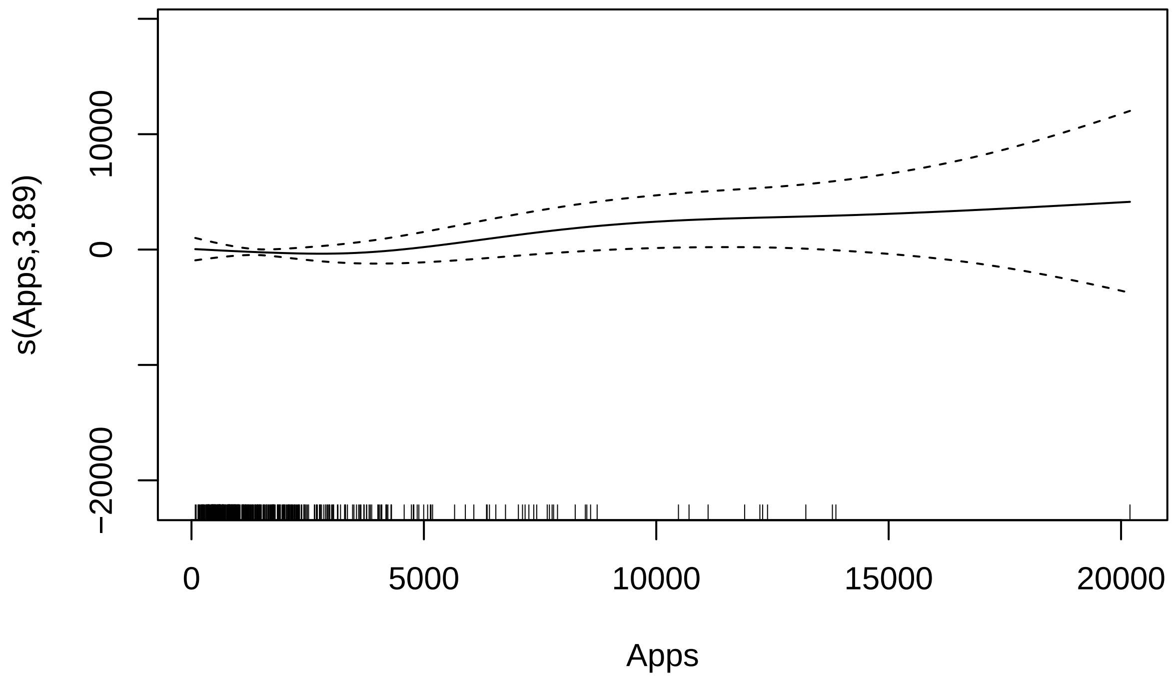


s(Enroll,1)
−20000
0
10000
0
1000
2000
3000
4000
Enroll

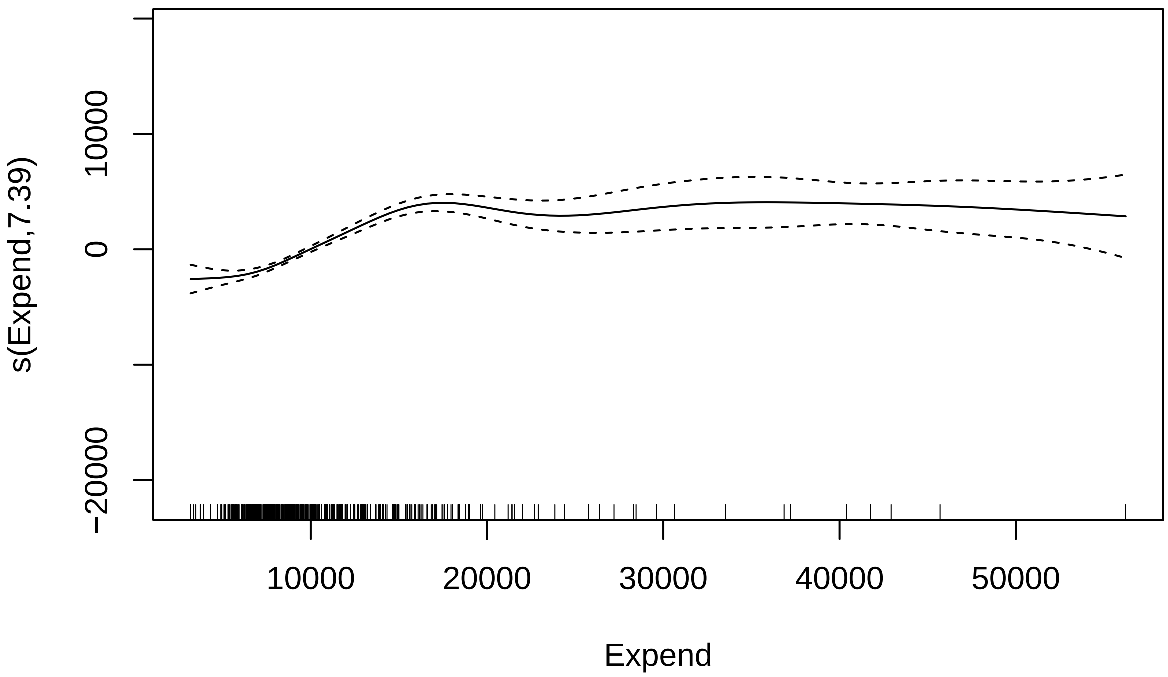

s(Accept,2.84)
10000
0
−20000
0
2000
4000
6000
8000
10000
12000
Accept



s(F.Undergrad,6.17)
−20000
0
10000
0
5000
10000
15000
20000
25000
F.Undergrad



s(Apps,3.89)
10000
0
−20000
0
5000
10000
15000
20000
Apps




```
# report test error
pred.gam <- predict(gam.fit, newdata = testing_data)

test.error.gam <- mean((pred.gam - y2)^2)
```

The best fit GAM model does include all predictors, as the selection method is **FALSE**. From the final models plots, we see that `Terminal`, `Books`, `Top25Perc`, `Personal`, and `Enroll` appear to be linear terms as their df = 1, but the all the predictors in the model have an "s", indicating the smoothing function was used and these terms are interpreted as non-linear. We can observe that some predictors are more non-linear than others as the degrees of freedom increase. For example, `Top10perc` has more wave in the graph compared to `S.F.Ratio`. The test error is 3.4345234×10^6.

1d) In this dataset, would you favor a MARS model over a linear model for predicting out-of-state tuition? If so, why? More broadly, in general applications, do you consider a MARS model to be superior to a linear model? Please share your reasoning.

```r
set.seed(2)

# fit a linear model using 10-fold cross-validation
lm.fit <- train(x, y,
                method = "lm",
                trControl = ctrl)

summary(lm.fit)
```

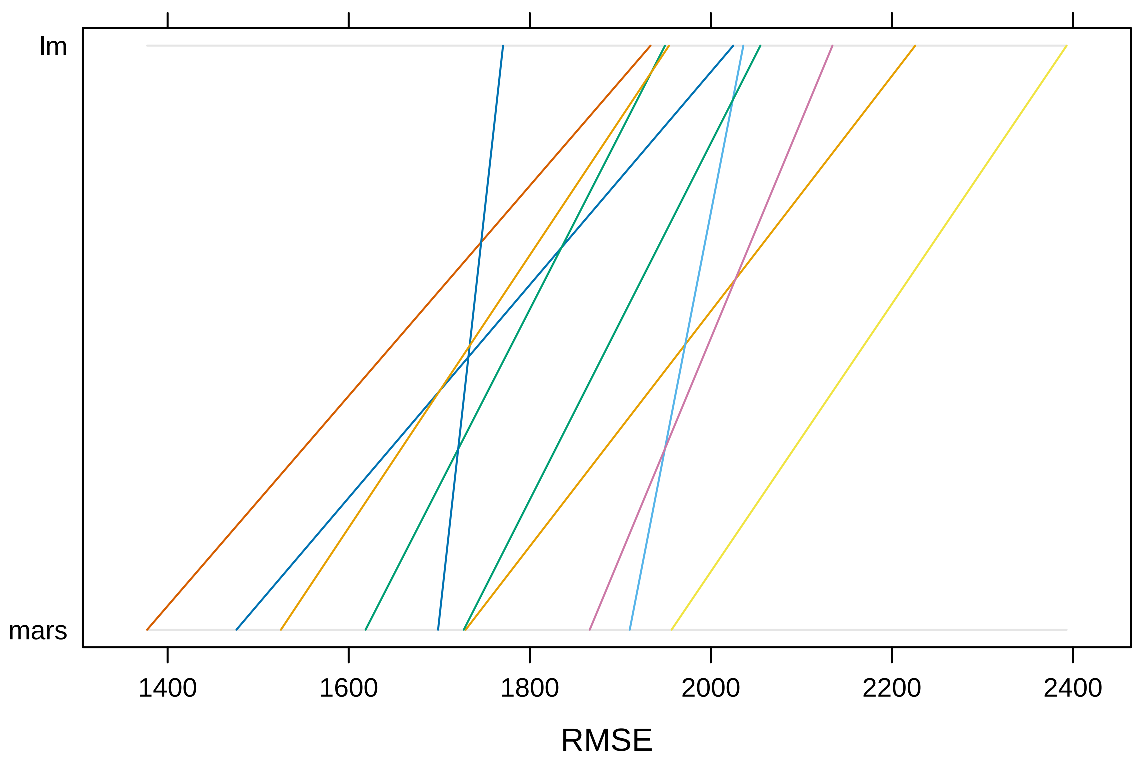
```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -6611  -1248     -7   1349   9810
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 752.97152  963.39650   0.782  0.43489
## Apps         -0.04677    0.11802  -0.396  0.69207
## Accept        1.33042    0.20335   6.542 1.70e-10 ***
## Enroll       -3.83350    0.91337  -4.197 3.28e-05 ***
## Top10perc    30.21342   15.52858   1.946  0.05234 .
## Top25perc     0.22821   12.58307   0.018  0.98554
## F.Undergrad   0.06437    0.14400   0.447  0.65508
## P.Undergrad  -0.19878    0.16435  -1.210  0.22711
## Room.Board    0.87381    0.11536   7.575 2.18e-13 ***
## Books         0.36615    0.54397   0.673  0.50123
## Personal     -0.48636    0.15441  -3.150  0.00175 **
## PhD           6.58415   11.93575   0.552  0.58148
## Terminal     32.39712   13.16108   2.462  0.01422 *
## S.F.Ratio   -37.67851   31.82476  -1.184  0.23708
## perc.alumni  39.30520    9.53167   4.124 4.47e-05 ***
## Expend        0.16060    0.02711   5.923 6.42e-09 ***
## Grad.Rate    20.39531    7.25323   2.812  0.00515 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2000 on 435 degrees of freedom
## Multiple R-squared:  0.7358, Adjusted R-squared:  0.7261
## F-statistic: 75.73 on 16 and 435 DF,  p-value: < 2.2e-16
```

```r
# compare models
resamp <- resamples(list(lm = lm.fit, mars = mars.fit))
```

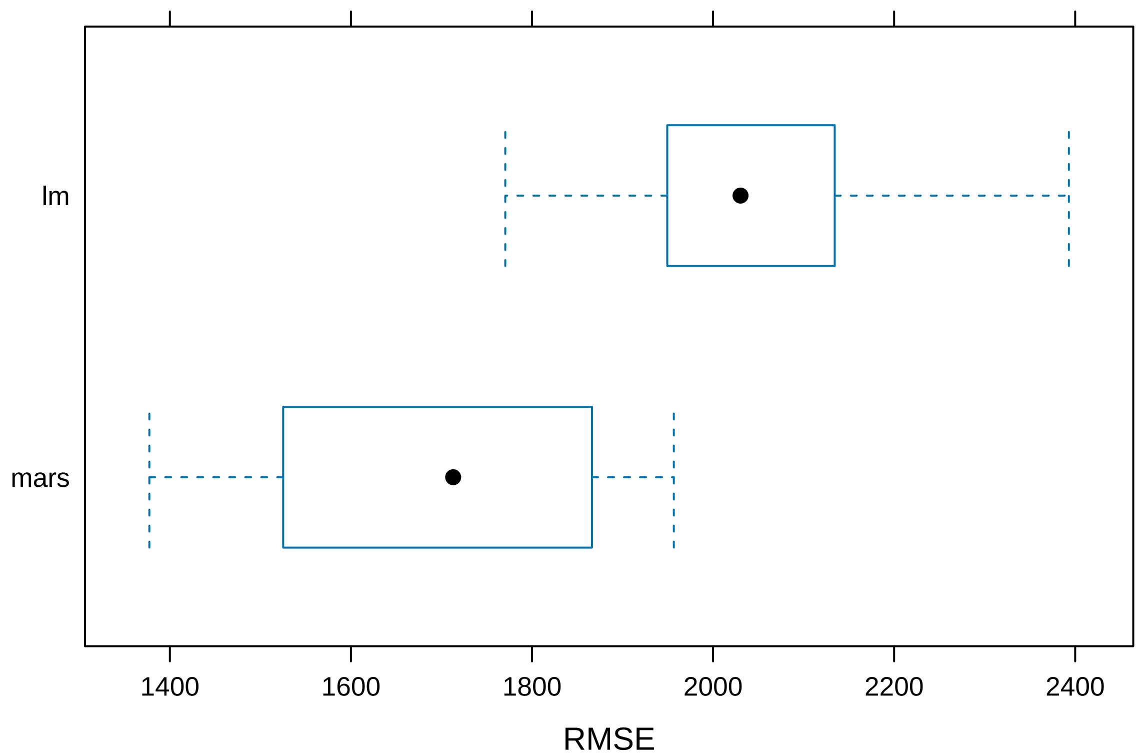
```r
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: lm, mars
## Number of resamples: 10
##
## MAE
##          Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lm   1427.528 1578.173 1615.828 1608.826 1674.692 1752.230    0
## mars 1138.836 1239.576 1320.672 1321.704 1425.698 1497.191    0
##
## RMSE
##          Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lm   1770.494 1950.509 2030.364 2047.597 2114.502 2393.058    0
## mars 1377.340 1548.527 1712.939 1688.556 1831.993 1956.649    0
##
## Rsquared
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lm   0.6577688 0.7142791 0.7249317 0.7204843 0.7408093 0.7704528    0
## mars 0.7384265 0.7672405 0.8068437 0.8053855 0.8427477 0.8636907    0
```

```r
parallelplot(resamp, metric = "RMSE")
```



```r
bwplot(resamp, metric = "RMSE")
```



From the resampling summary, I believe the best model is the MARS since it has the smallest mean and median RMSE value. I would prefer fitting a MARS model over a linear model mostly because of its flexibility. MARS models can capture non-linear relationships, can capture interactions between variables, and are generally more robust.