p8106_hw2

Hao Zheng(hz2770)

2022/3/5

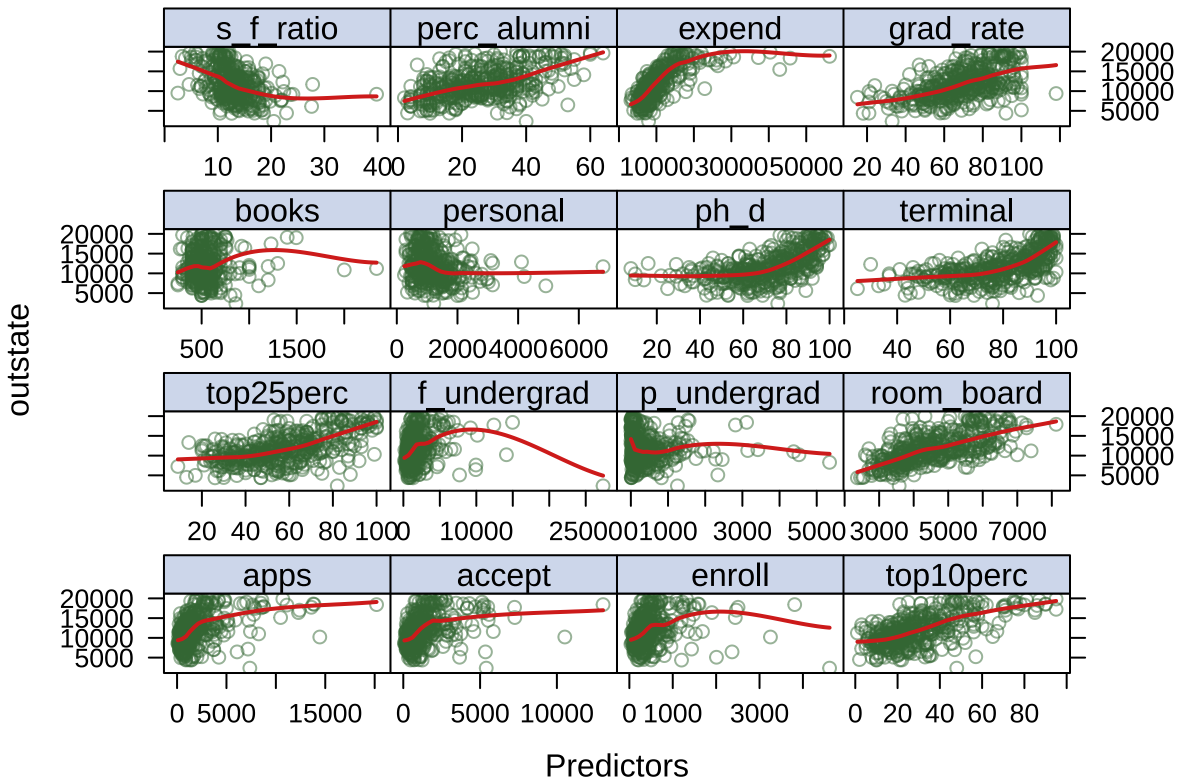
```r
# Data Cleaning
dat =
  read.csv("./data/college.csv")[-1] %>%
  janitor::clean_names() %>%
  na.omit()

# Data Partition
indexTrain <- createDataPartition(y = dat$outstate, p = 0.8, list = FALSE)
trainData <- dat[indexTrain,]
testData <- dat[-indexTrain,]
head(trainData)
```

```
##    apps accept enroll top10perc top25perc f_undergrad p_undergrad outstate
## 2  2186   1924    512        16        29        2683        1227    12280
## 3  1428   1097    336        22        50        1036          99    11250
## 4   417    349    137        60        89         510          63    12960
## 6   587    479    158        38        62         678          41    13500
## 8  1899   1720    489        37        68        1594          32    13868
## 11 1732   1425    472        37        75        1830         110    16548
##    room_board books personal ph_d terminal s_f_ratio perc_alumni expend
## 2        6450   750     1500   29       30      12.2          16  10527
## 3        3750   400     1165   53       66      12.9          30   8735
## 4        5450   450      875   92       97       7.7          37  19016
## 6        3335   500      675   67       73       9.4          11   9727
## 8        4826   450      850   89      100      13.7          37  11487
## 11       5406   500      600   82       88      11.3          31  10932
##    grad_rate
## 2         56
## 3         54
## 4         59
## 6         55
## 8         73
## 11        73
```

Exploratory Data Analysis

```r
theme1 <- trellis.par.get()
theme1$plot.symbol$col <- rgb(.2, .4, .2, .5)
theme1$plot.symbol$psh <- 16
```

```r
theme1$plot.line$col <- rgb(.8, .1, .1, 1)
theme1$plot.line$lwd <- 2
theme1$strip.background$col <- rgb(.0, .2, .6, .2)
trellis.par.set(theme1)

x <- trainData %>% select(-outstate)
y <- trainData$outstate

# scatter plot
featurePlot(x,
            y,
            plot = "scatter",
            span = .5,
            labels = c("Predictors", "outstate"),
            type = c("p", "smooth"),
            layout = c(4,4))
```



From the scatter plot, we can see that most predictors are not linearly associated with the response variable. However, there may exist a linear relationship between the variable `perc_alumni`, `grad_rate`, `room_board` and the response `outstate` respectively.

Smoothing Spline Models

Now let's fit smoothing spline models using `terminal` as the only predictor of `outstate`.

```r
terminal.grid <- seq(from = 40, to = 100, by = 1)
fit.ss <- smooth.spline(trainData$terminal, trainData$outstate, cv = TRUE)
```

```
## Warning in smooth.spline(trainData$terminal, trainData$outstate, cv = TRUE):
## cross-validation with non-unique 'x' values seems doubtful
```

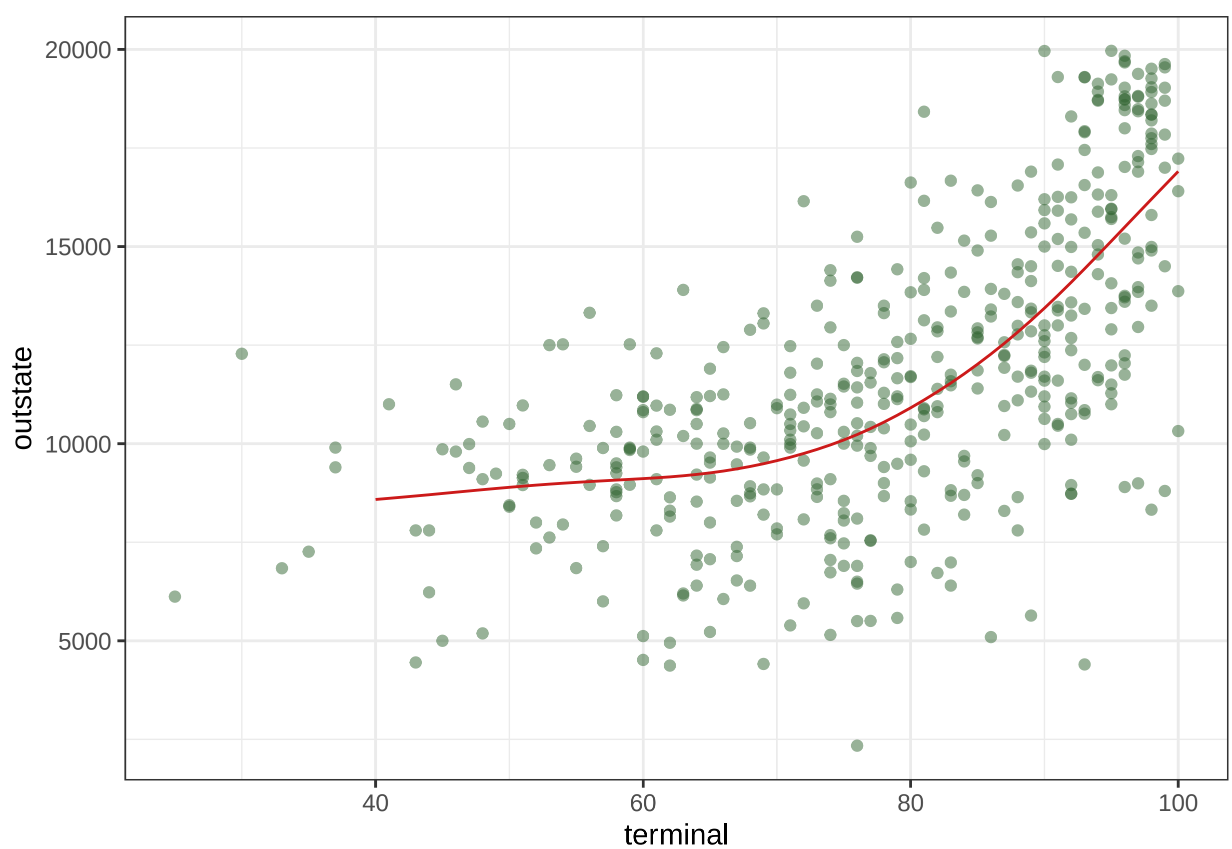
```r
fit.ss$df
```

```
## [1] 4.756147
```

```r
fit.ss$lambda
```

```
## [1] 0.02282439
```

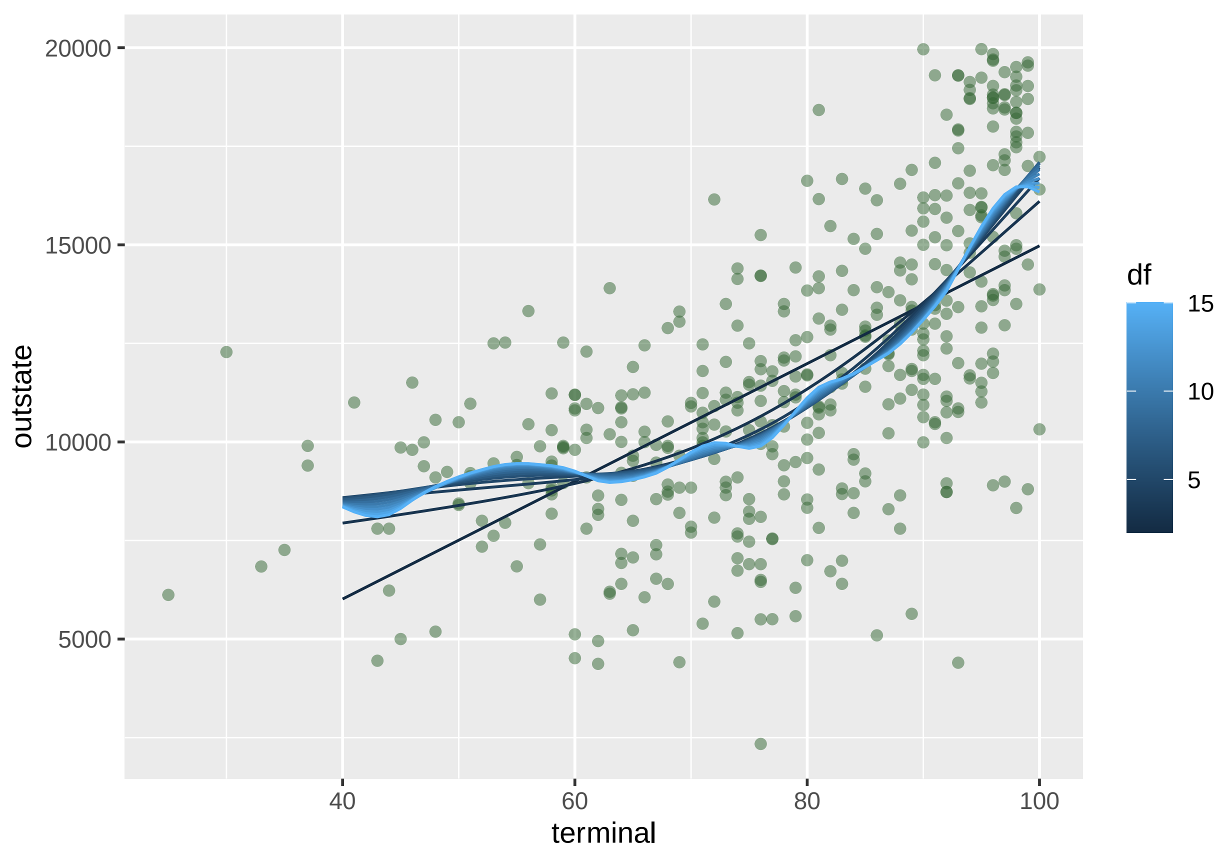
```r
pred.ss <- predict(fit.ss,
                   x = terminal.grid)
pred.ss.df <- data.frame(pred = pred.ss$y,
                         terminal = terminal.grid)

# plot the fit
p <- ggplot(data = trainData, aes(x = terminal, y = outstate)) +
  geom_point(color = rgb(.2, .4, .2, .5))

p +
  geom_line(aes(x = terminal.grid, y = pred), data = pred.ss.df, color = rgb(.8, .1, .1, 1)) + theme_bw
```



The optimal smoothing spline model fitted using the degrees of freedom obtained by generalized cross validation is as above with degrees of freedom 4.7561474. As we can see from the plot, the smoothing spline obtained is quite smooth and fits the data quite well.

Then we also try to fit the model for a range of degrees of freedom to observe the underlying pattern.

```
# Write a function about it
ss.func <- function(degree){
  fit.ss <- smooth.spline(trainData$terminal, trainData$outstate, df = degree)
  pred.ss <- predict(fit.ss,
                     x = terminal.grid)
  pred.ss.df <- data.frame(pred = pred.ss$y,
                           terminal = terminal.grid,
                           df = degree)
}

ss.list <- list()
for (i in 2:15) {
  ss.list <- rbind(ss.list, ss.func(i))
}
ss.data <- as.data.frame(ss.list)

p +
  geom_line(aes(x = terminal, y = pred, group = df, color = df), data = ss.data)
```



From the plot of smoothing spline fit with different degrees of freedom, we can see that when the degrees of freedom is small, the fitted line is quite linear, and it gets more and more wiggly as degrees of freedom increase.

Generalized Additive Models (GAM)

Fit GAM model with all the predictors.

```
set.seed(2022)
ctrl = trainControl(method = "cv", number = 10)

model.gam <- train(x, y,
                 method = "gam",
                 tuneGrid = data.frame(method = "GCV.Cp",
                                       select = TRUE),
                 trControl = ctrl)
model.gam$finalModel
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ s(perc_alumni) + s(terminal) + s(top10perc) + s(books) +
##     s(ph_d) + s(grad_rate) + s(top25perc) + s(s_f_ratio) + s(personal) +
##     s(p_undergrad) + s(enroll) + s(room_board) + s(accept) +
##     s(f_undergrad) + s(apps) + s(expend)
##
## Estimated degrees of freedom:
## 0.910 0.000 0.667 3.482 3.205 3.426 0.000
## 3.908 0.680 0.000 1.000 1.765 3.190 6.681
## 5.832 4.701  total = 40.45
##
## GCV score: 2637015
```

```
summary(model.gam)
```

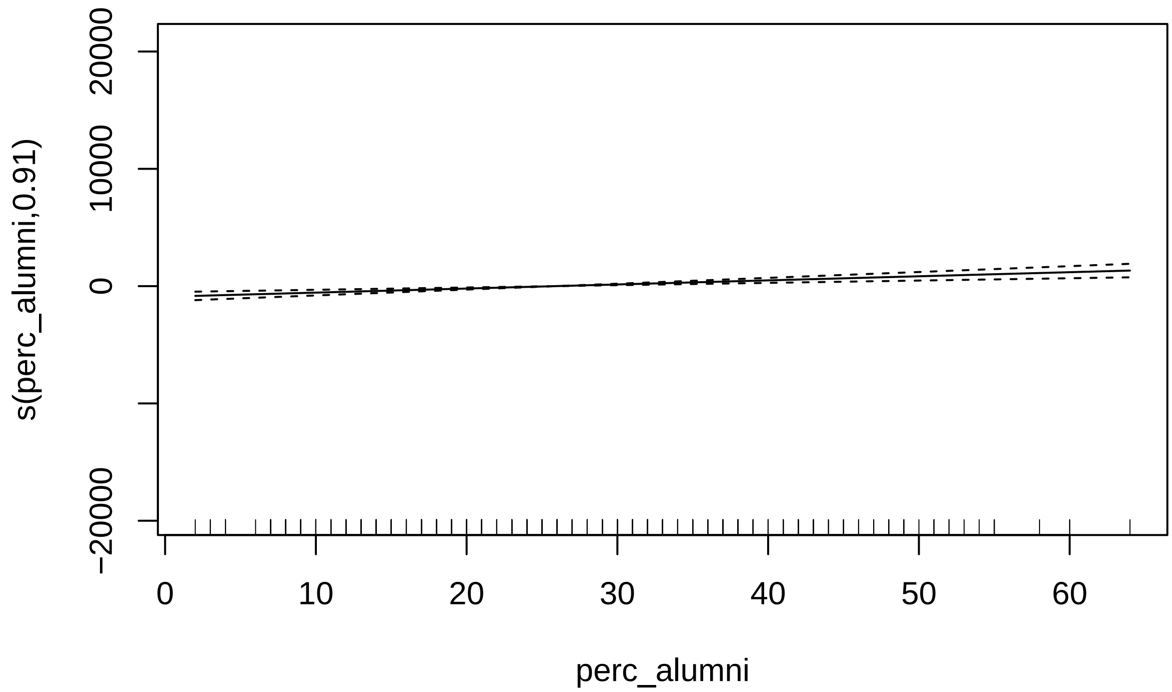
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ s(perc_alumni) + s(terminal) + s(top10perc) + s(books) +
##     s(ph_d) + s(grad_rate) + s(top25perc) + s(s_f_ratio) + s(personal) +
##     s(p_undergrad) + s(enroll) + s(room_board) + s(accept) +
##     s(f_undergrad) + s(apps) + s(expend)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11751.30      72.81   161.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
```

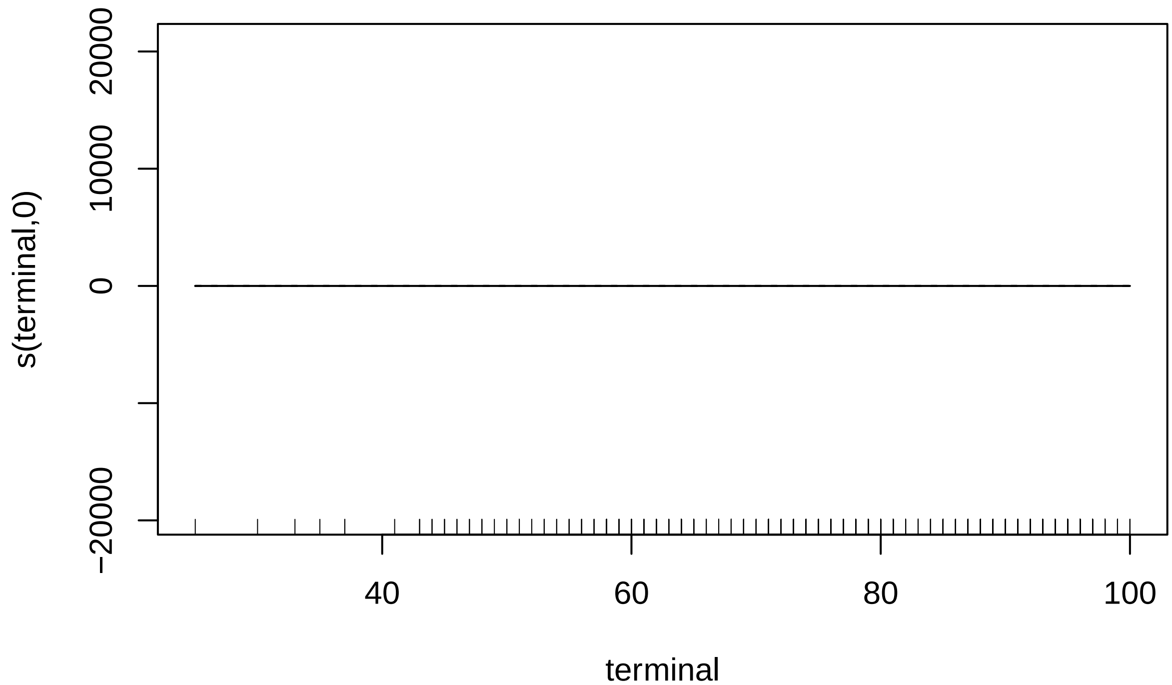
```
##                        edf Ref.df      F  p-value
## s(perc_alumni) 9.103e-01      9  2.360 1.52e-06 ***
## s(terminal)    1.946e-07      9  0.000 0.917535
## s(top10perc)   6.674e-01      9  0.221 0.078160 .
## s(books)       3.482e+00      9  1.452 0.003088 **
## s(ph_d)        3.205e+00      9  1.391 0.002919 **
## s(grad_rate)   3.426e+00      9  1.920 0.000349 ***
## s(top25perc)   2.186e-07      9  0.000 0.696823
## s(s_f_ratio)   3.908e+00      9  1.124 0.020836 *
## s(personal)    6.799e-01      9  0.301 0.043199 *
## s(p_undergrad) 1.755e-07      9  0.000 0.991138
## s(enroll)      1.000e+00      9  1.904 1.84e-05 ***
## s(room_board)  1.765e+00      9  7.611  < 2e-16 ***
## s(accept)      3.190e+00      9  2.112 2.48e-05 ***
## s(f_undergrad) 6.681e+00      9  4.684  < 2e-16 ***
## s(apps)        5.833e+00      9  1.931 0.001214 **
## s(expend)      4.701e+00      9 15.335  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.824   Deviance explained = 83.9%
## GCV = 2.637e+06  Scale est. = 2.4016e+06  n = 453
```

According to the p value, some predictors may not be significant in the GAM model, such as `terminal`, `top25perc` and `p_undergrad`. The deviance explained by the model is 85.3%, adjusted R-squared value is 0.833, which is quite close to 1. The GAM model fits the data quite well.

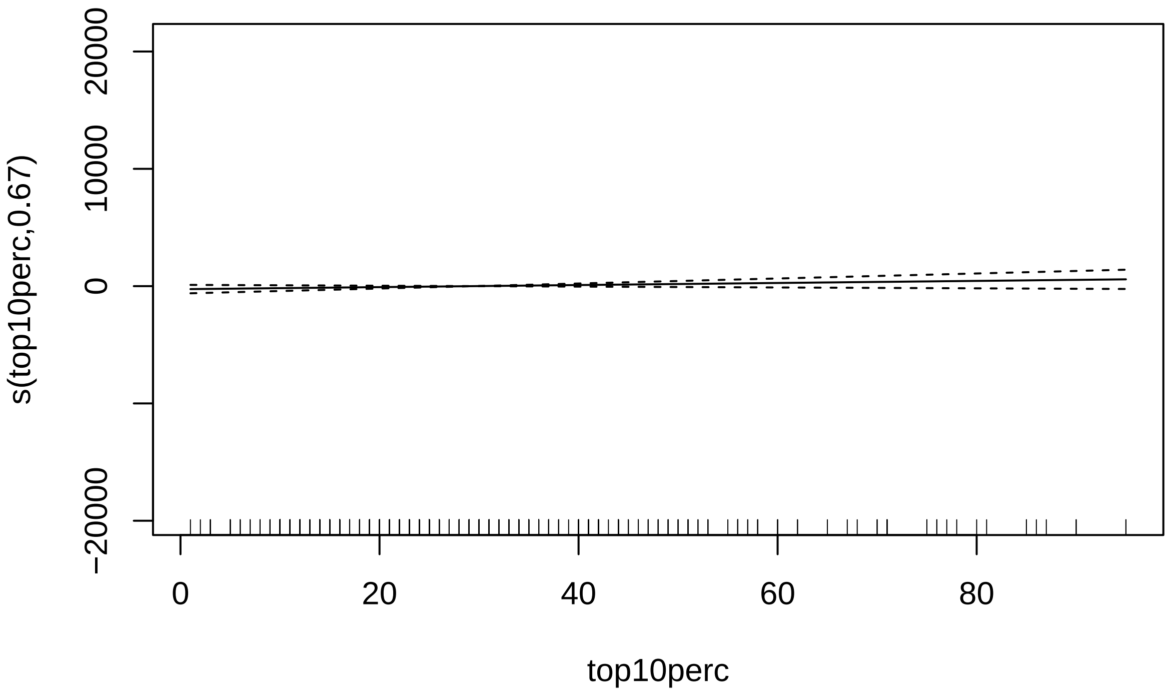
The plots of each predictor against the response variable are as below.

```
plot(model.gam$finalModel)
```

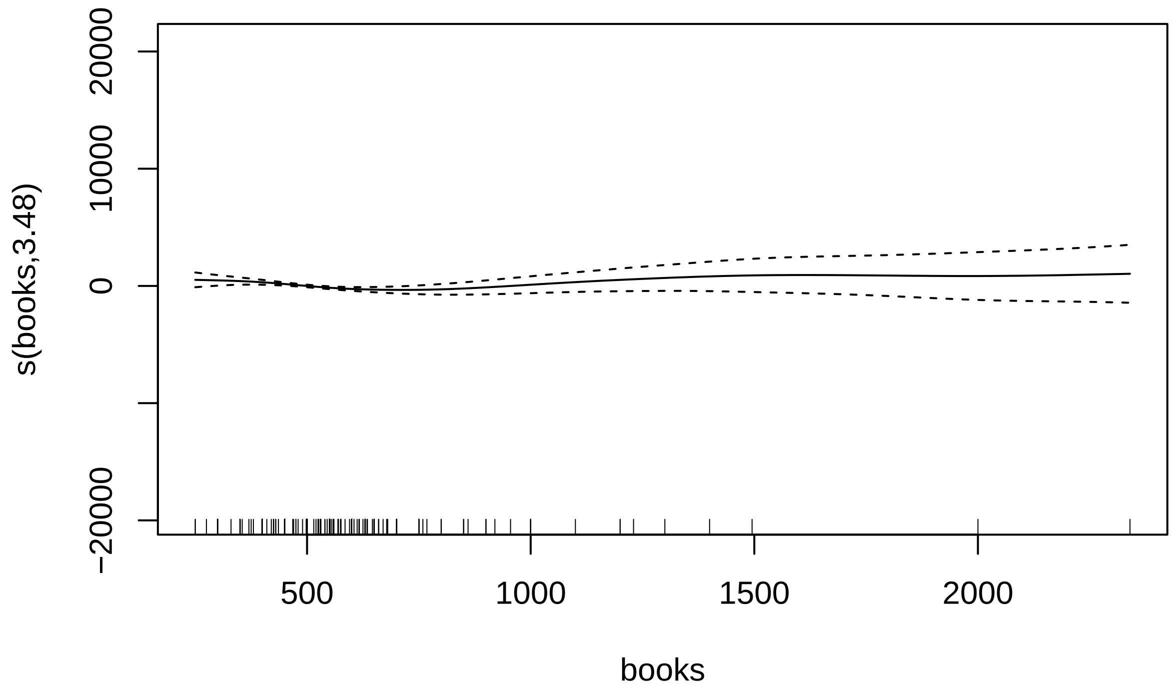

s(perc_alumni,0.91)
perc_alumni
20000
10000
0
−20000
0
10
20
30
40
50
60

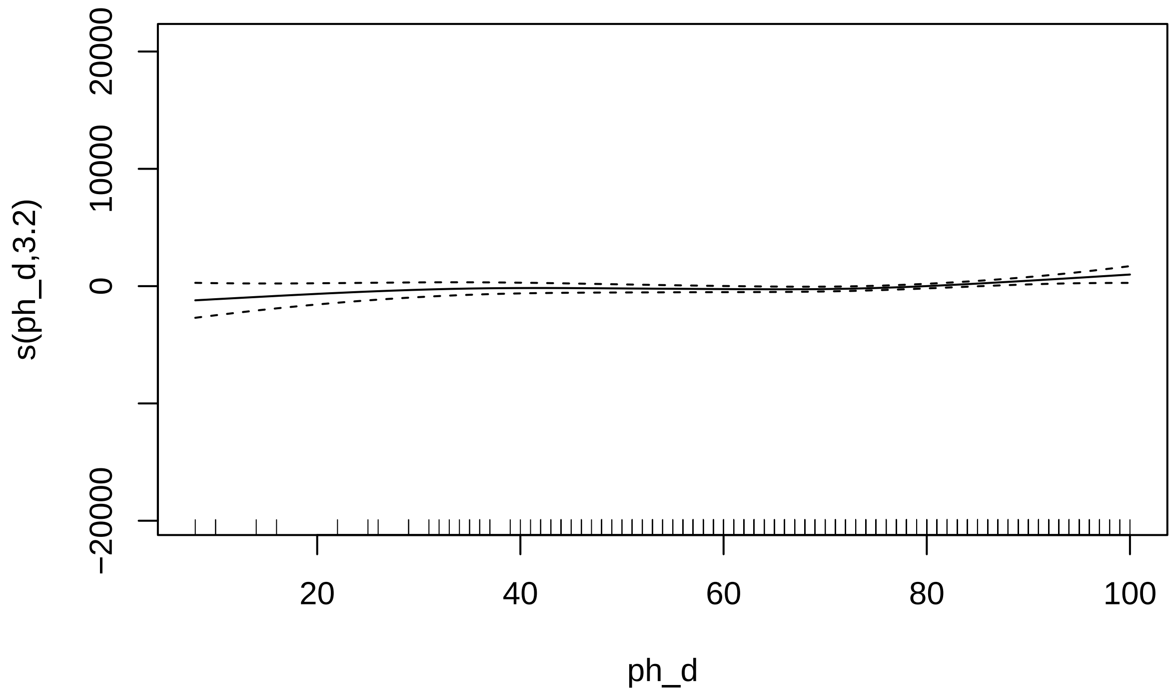


20000
10000
0
−20000
s(terminal,0)
40
60
80
100
terminal

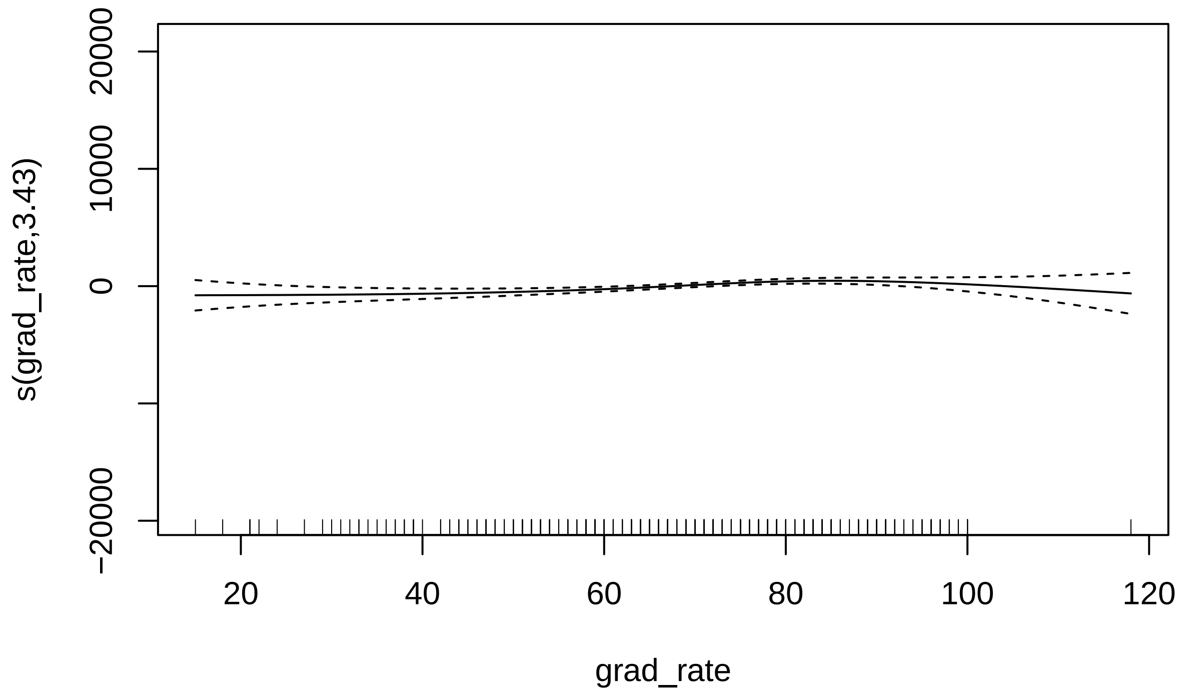


20000
10000
0
−20000
s(top10perc,0.67)
0
20
40
60
80
top10perc

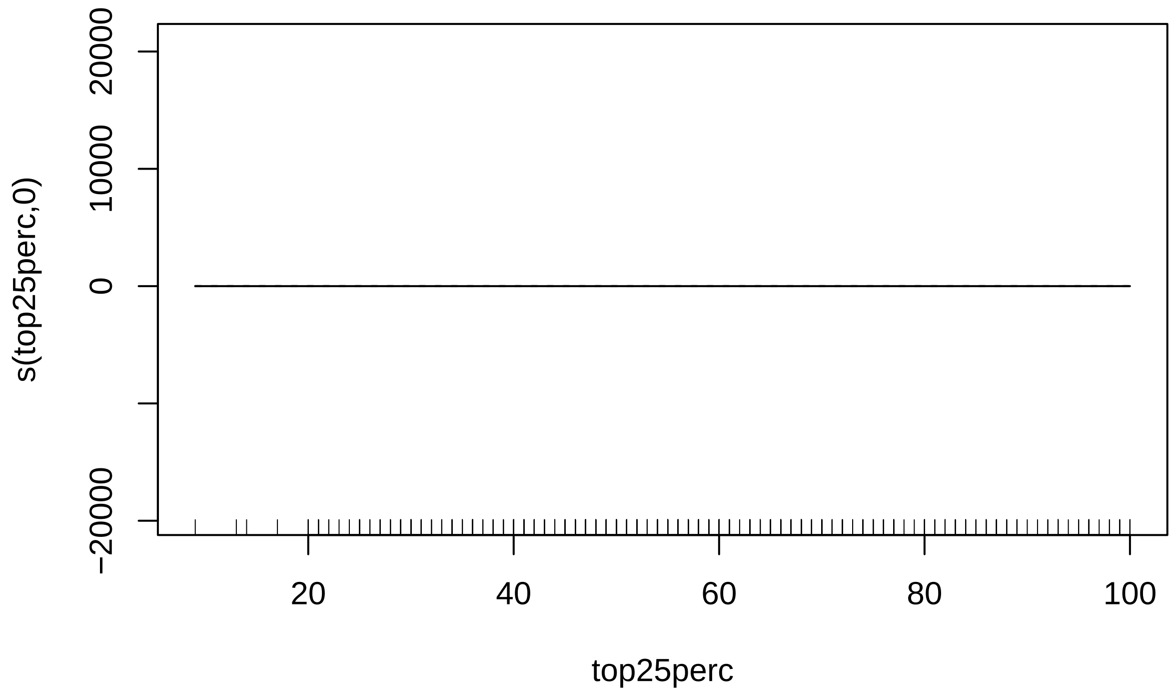

20000
10000
0
−20000
s(books,3.48)
500
1000
1500
2000
books

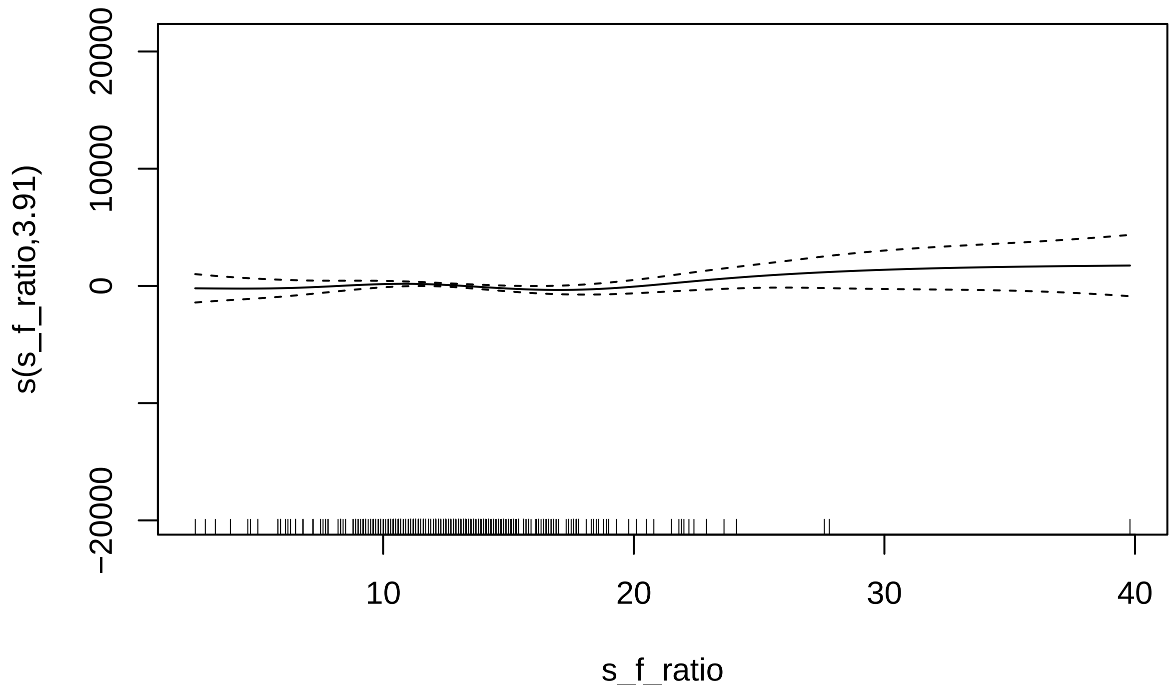


s(ph_d,3.2)
20000
10000
0
−20000
20
40
60
80
100
ph_d

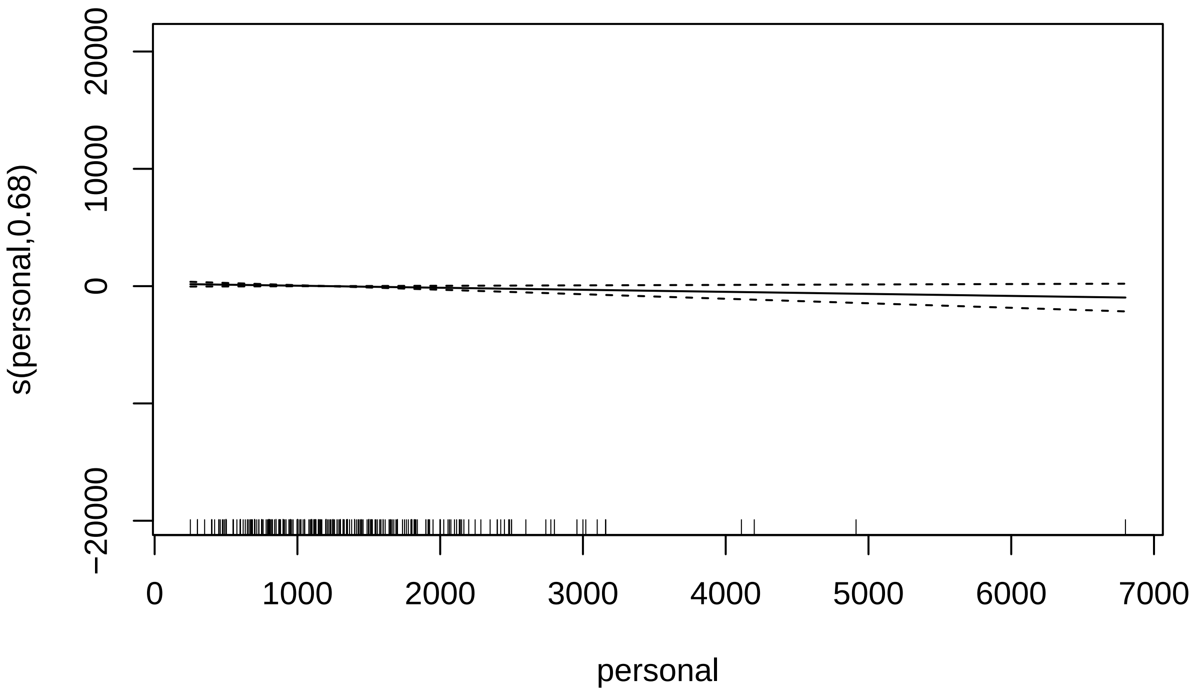


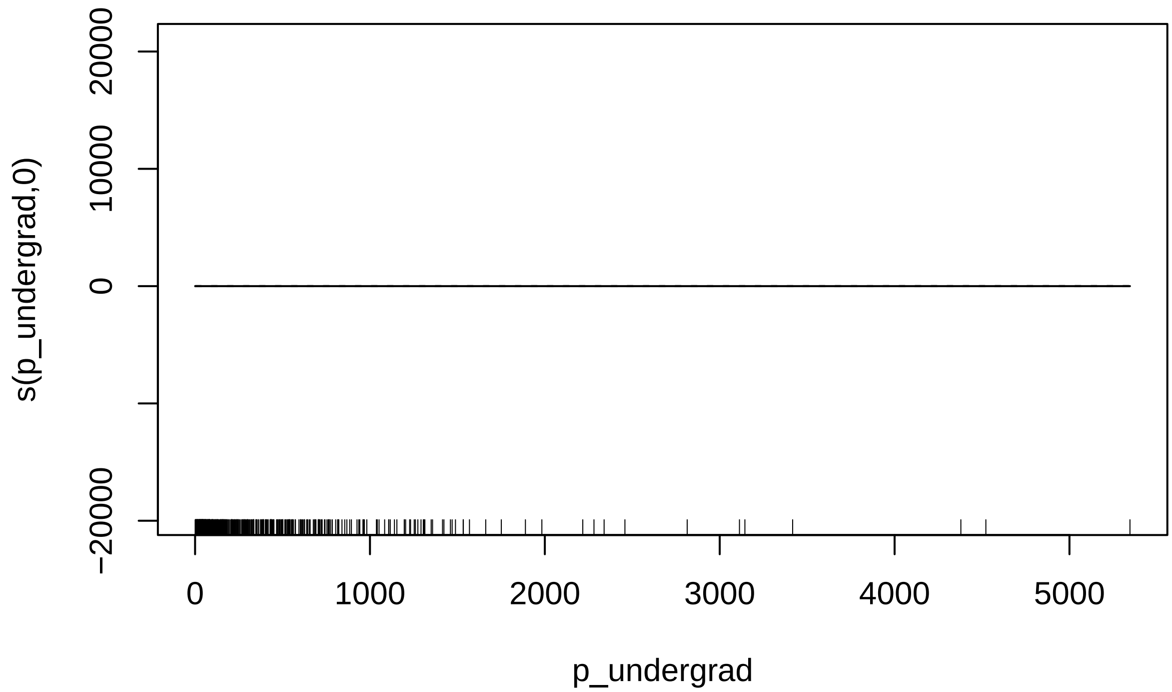
s(grad_rate,3.43)
20000
10000
0
−20000
20
40
60
80
100
120
grad_rate


s(top25perc,0)
20000
10000
0
−20000
20
40
60
80
100
top25perc

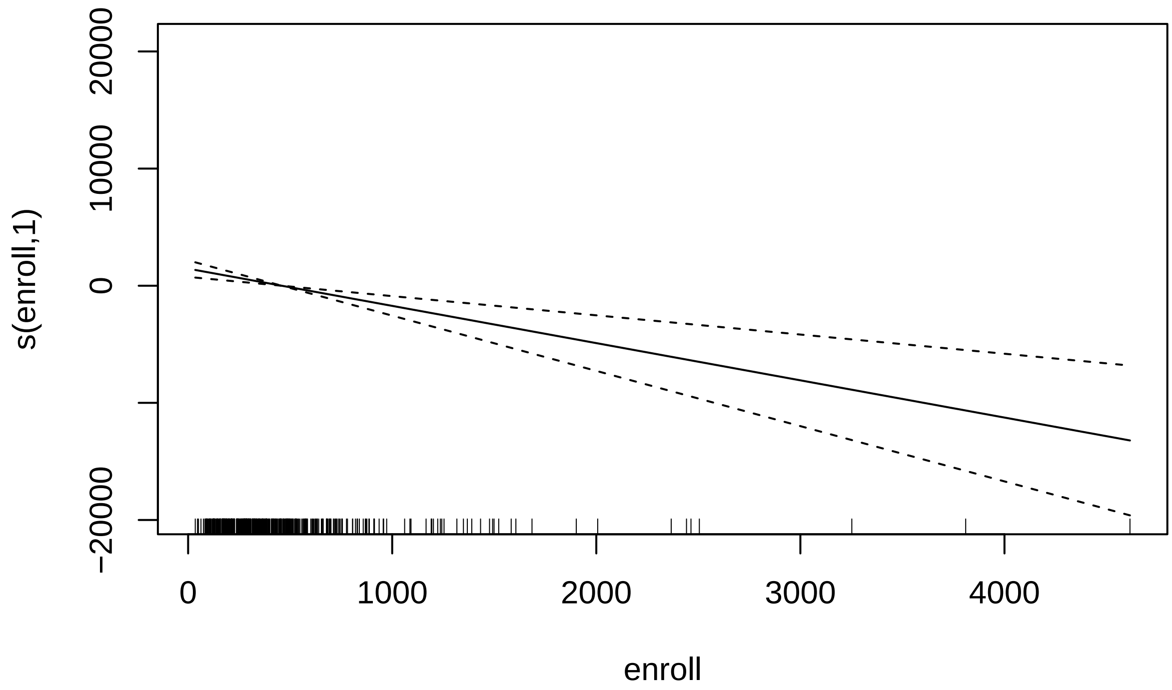


20000
10000
0
−20000
s(s_f_ratio,3.91)
10
20
30
40
s_f_ratio

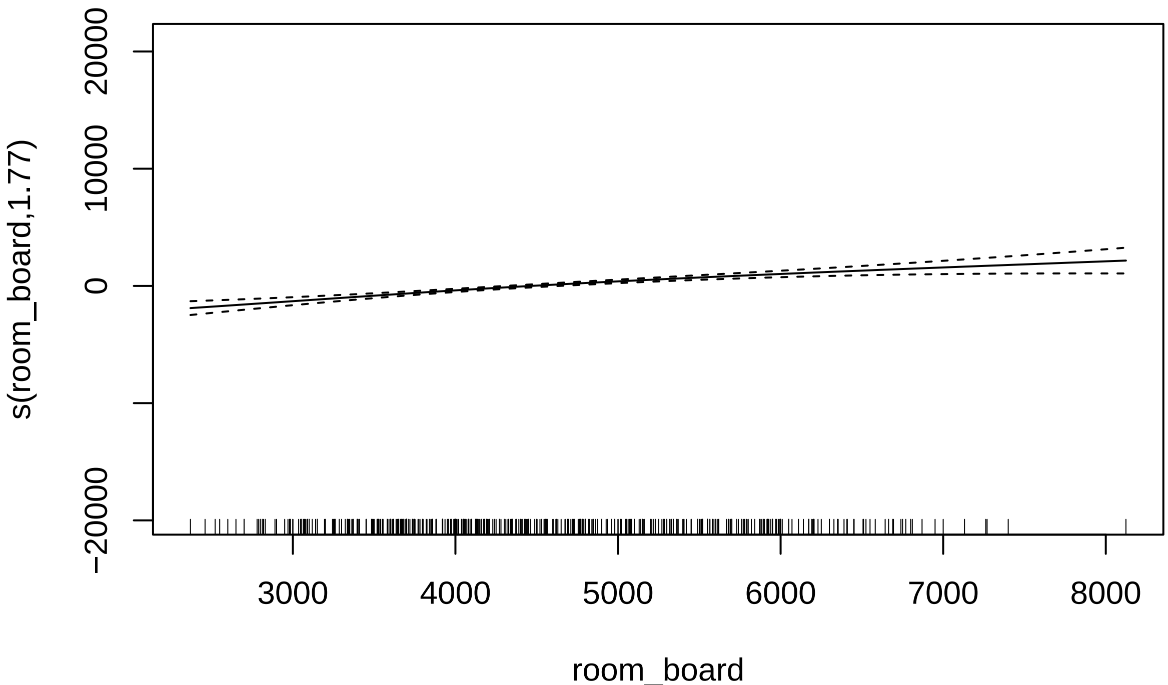

s(personal,0.68)
20000
10000
0
−20000
0
1000
2000
3000
4000
5000
6000
7000
personal

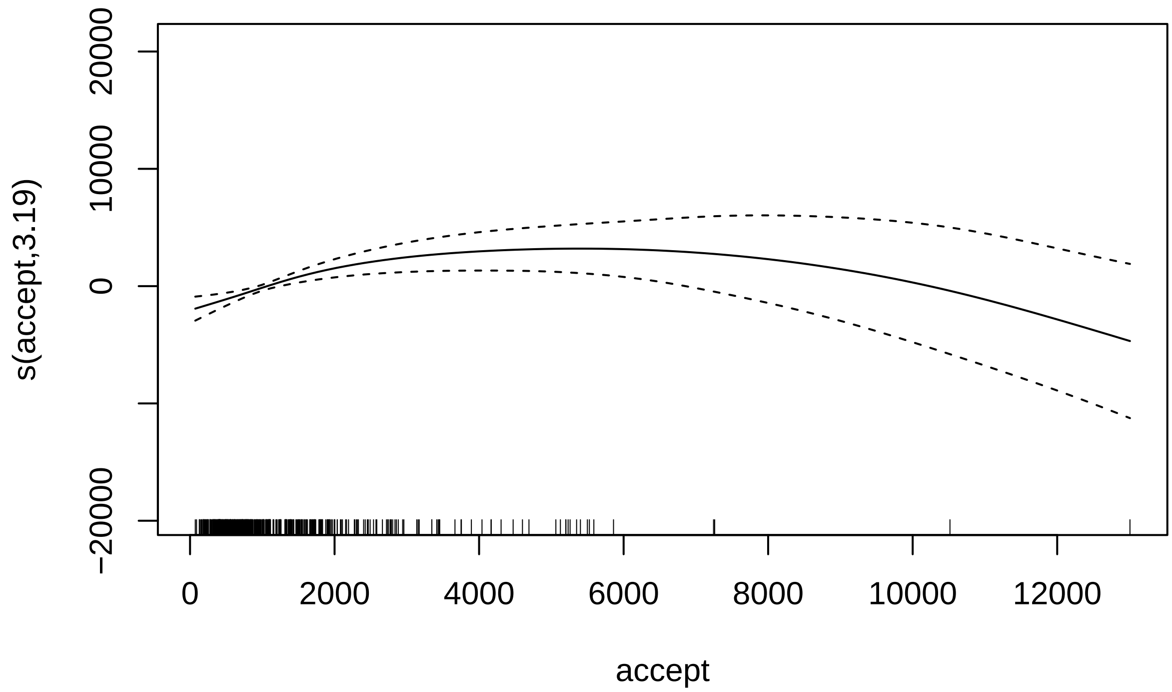


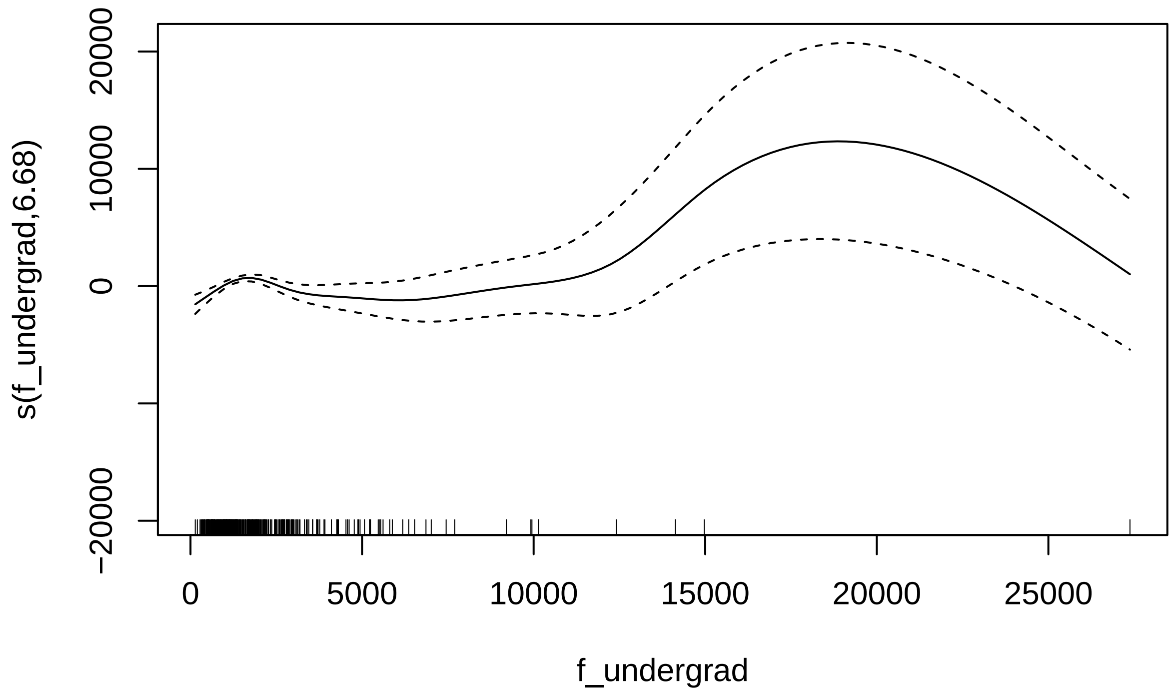
s(p_undergrad,0)
20000
10000
0
−20000
0
1000
2000
3000
4000
5000
p_undergrad

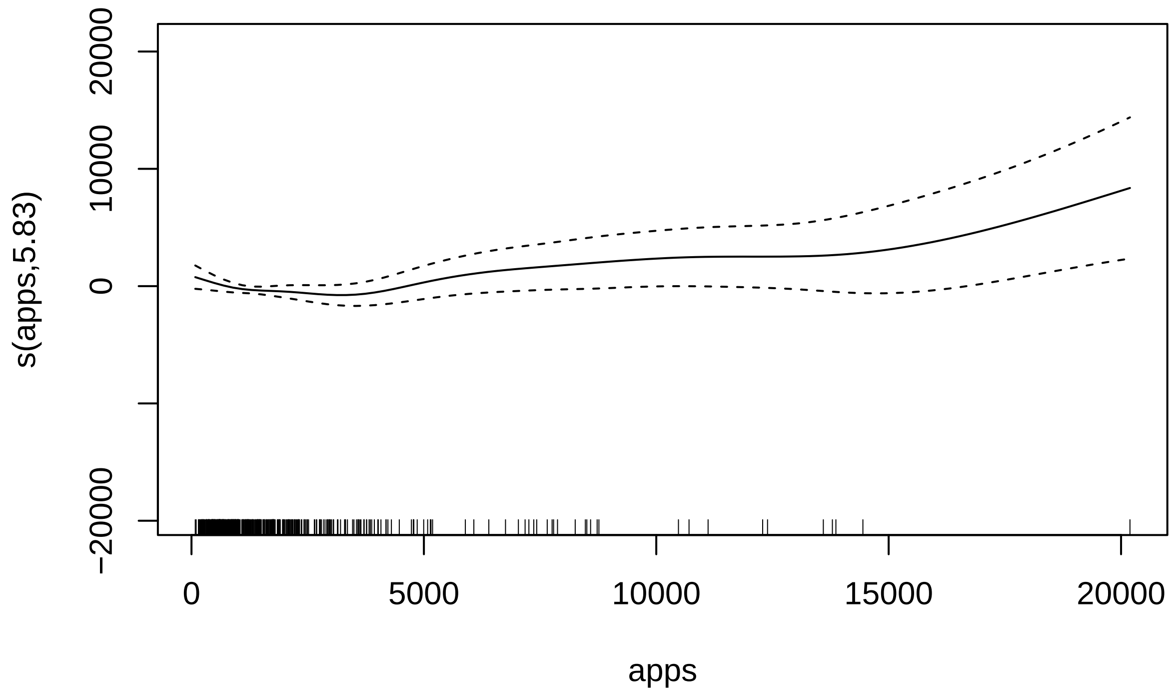


20000
10000
0
−20000
s(enroll,1)
0
1000
2000
3000
4000
enroll

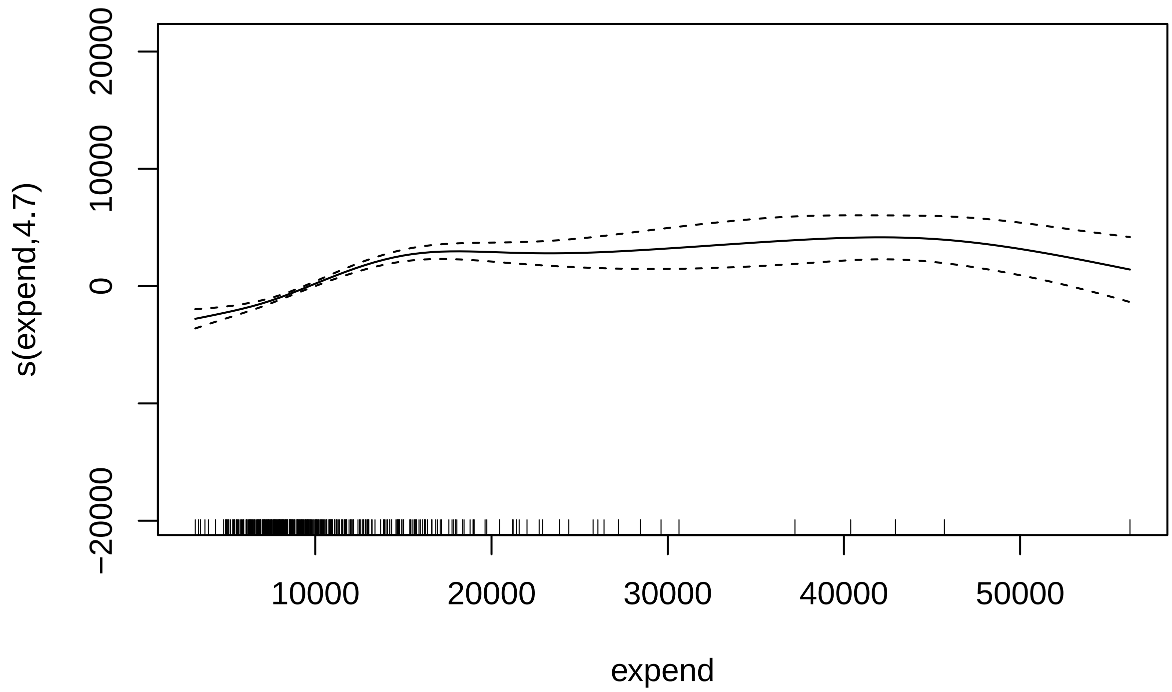

20000
10000
0
−20000
s(room_board,1.77)
3000
4000
5000
6000
7000
8000
room_board


s(accept,3.19)
20000
10000
0
−20000
0
2000
4000
6000
8000
10000
12000
accept


s(f_undergrad,6.68)
−20000
0
10000
20000
0
5000
10000
15000
20000
25000
f_undergrad



20000
10000
0
−20000
s(apps,5.83)
0
5000
10000
15000
20000
apps



Now let's calculate the test error of the GAM model.

```
test_x = testData %>% select(-outstate)

gam.pred <- predict(model.gam, newdata = test_x)

test_error_gam = mean((gam.pred - testData$outstate)^2)
test_error_gam
```

```
## [1] 3423151
```

The test error for the GAM model is 3.4231506×10^6.

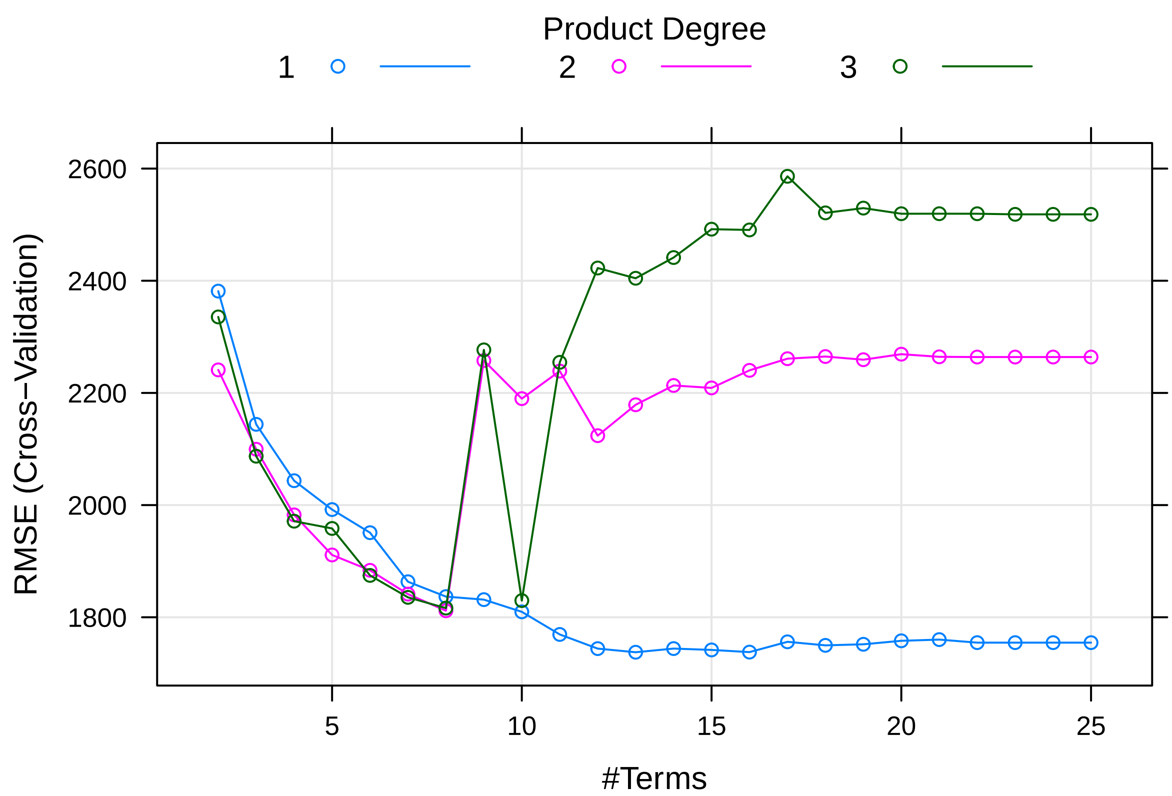
Multivariate Adaptive Regression Spline (MARS)

```
set.seed(2022)

model.mars <- train(x,y,
                    method = "earth",
                    tuneGrid = expand.grid(degree = 1:3,
                                           nprune = 2:25),
                    trControl = ctrl)
```

```
## Loading required package: earth
```

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

## Loading required package: Formula

## Loading required package: plotmo

## Warning: package 'plotmo' was built under R version 4.1.2

## Loading required package: plotrix

## Loading required package: TeachingDemos

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

```
plot(model.mars)
```




```
model.mars$bestTune
```

```
##    nprune degree
## 12     13      1
```

```r
summary(model.mars$finalModel)
```

```
## Call: earth(x=data.frame[453,16], y=c(12280,11250,1...), keepxy=TRUE, degree=1,
##             nprune=13)
##
##                       coefficients
## (Intercept)             13036.6711
## h(apps-1415)                0.4383
## h(1402-accept)             -1.4343
## h(911-enroll)               4.6708
## h(enroll-911)              -2.0638
## h(1274-f_undergrad)        -2.1400
## h(4450-room_board)         -1.1212
## h(room_board-4450)          0.4493
## h(660-books)                3.1164
## h(ph_d-74)                 49.4781
## h(perc_alumni-14)          38.6536
## h(15365-expend)            -0.5678
## h(98-grad_rate)           -18.7622
##
## Selected 13 of 21 terms, and 10 of 16 predictors (nprune=13)
## Termination condition: RSq changed by less than 0.001 at 21 terms
## Importance: expend, room_board, f_undergrad, perc_alumni, apps, enroll, ...
## Number of terms at each degree of interaction: 1 12 (additive model)
## GCV 2848666    RSS 1151942536    GRSq 0.7915596    RSq 0.8131072
```

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

```
##         (Intercept)     h(15365-expend)  h(room_board-4450)  h(4450-room_board)
##       13036.6711496          -0.5678269           0.4493307          -1.1211988
## h(1274-f_undergrad)   h(perc_alumni-14)        h(apps-1415)        h(660-books)
##          -2.1399682          38.6536464           0.4383361           3.1163826
##          h(ph_d-74)       h(enroll-911)       h(911-enroll)      h(1402-accept)
##          49.4780967          -2.0638368           4.6708035          -1.4343254
##     h(98-grad_rate)
##         -18.7621827
```

The final model uses 2 product degree and `nprune` $= 16$. 13 terms are selected of 33 terms among 9 of the 16 predictors. The predictors `expend`, `room_board`, `perc_alumni` and `accept` are of the most importance.

Then we calculate the test error on the test data.

```r
mars.pred <- predict(model.mars, newdata = test_x)

test_error_mars = mean((mars.pred - testData$outstate)^2)
test_error_mars
```

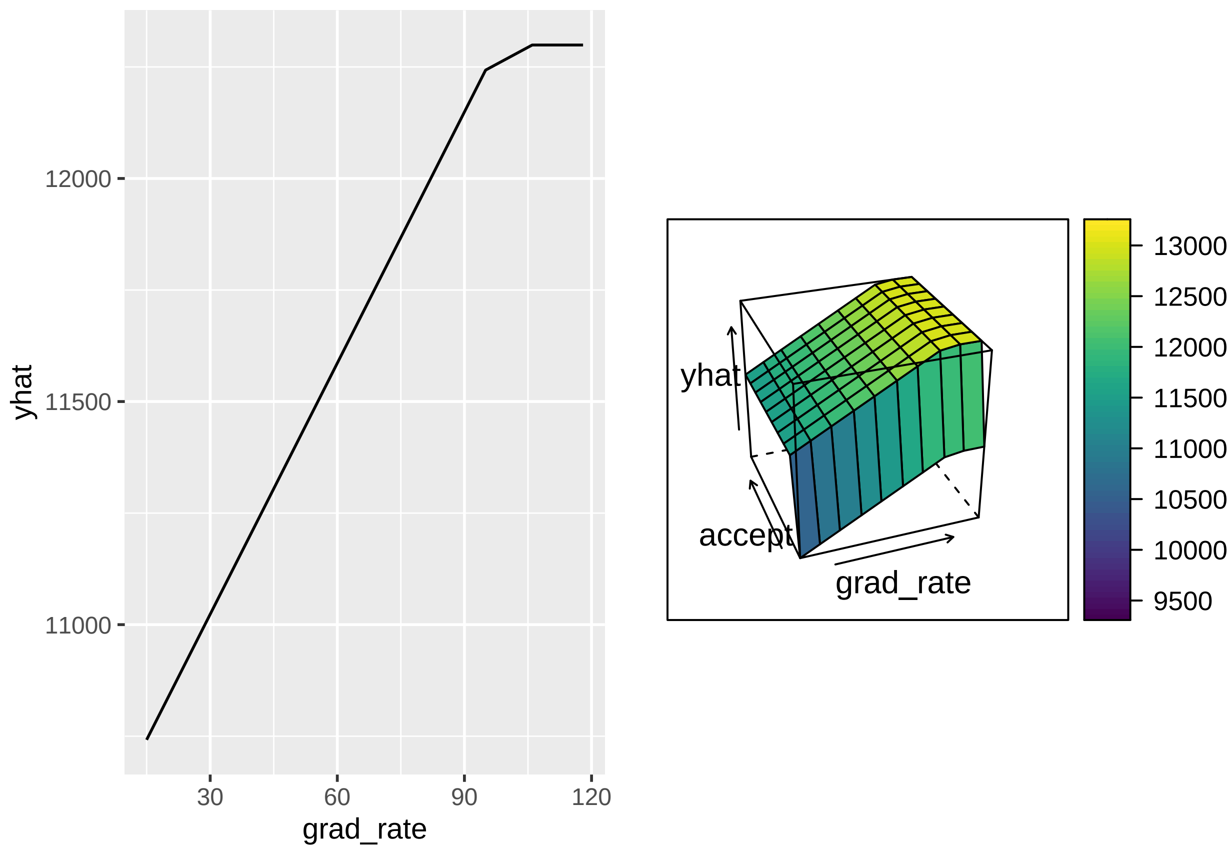
```
## [1] 3551601
```

The test error is 3.5516006×10^6.

```r
# Perform partial dependency plot.
p1 <- pdp::partial(model.mars, pred.var = c("grad_rate"), grid.resolution = 10) %>% autoplot()
p2 <- pdp::partial(model.mars, pred.var = c("grad_rate","accept"), grid.resolution = 10) %>%
  pdp::plotPartial(levelplot = FALSE, zlab = "yhat", drape = TRUE, screen = list(z = 20, x = -60))

grid.arrange(p1, p2, ncol = 2)
```

```
## Warning: Use of `object[[1L]]` is discouraged. Use `.data[[1L]]` instead.
```

```
## Warning: Use of `object[["yhat"]]` is discouraged. Use `.data[["yhat"]]`
## instead.
```



We present two partial dependency plot here, the left one is for `grad_rate` while the right one is for both variables `grad_rate` and `accept`.

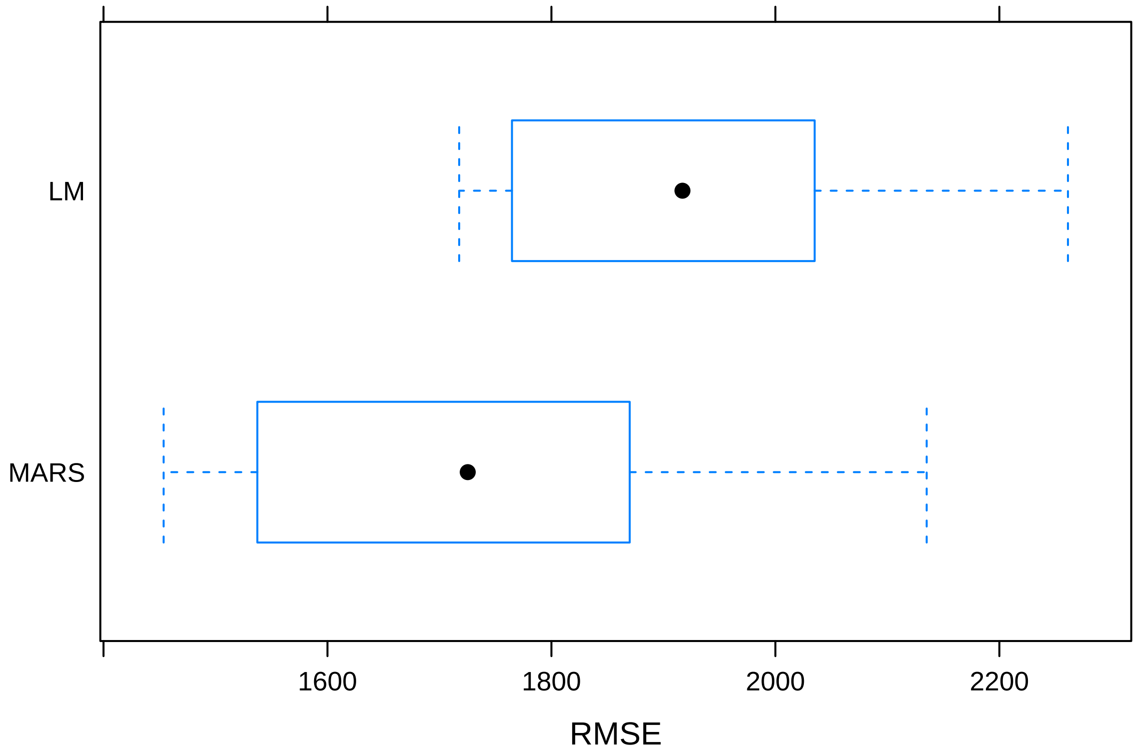
Model Selection

```r
set.seed(2022)
model.lm <- train(x, y,
                  method = "lm",
                  trControl = ctrl)

resamp <- resamples(list(MARS = model.mars,
```

```r
                         LM = model.lm))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: MARS, LM
## Number of resamples: 10
##
## MAE
##          Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## MARS 1137.175 1224.736 1317.743 1367.352 1488.634 1697.165    0
## LM   1199.578 1429.780 1499.041 1559.860 1739.608 1898.703    0
##
## RMSE
##          Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## MARS 1453.683 1551.473 1725.253 1737.627 1844.479 2135.103    0
## LM   1717.608 1770.324 1917.019 1939.742 2032.183 2261.255    0
##
## Rsquared
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## MARS 0.6208247 0.7552459 0.8031437 0.7782638 0.8280139 0.8499193    0
## LM   0.5388922 0.7272526 0.7446076 0.7285649 0.7666724 0.8198768    0
```

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



As we learned in the class, final model selection should be based on our cross-validation results. Since the MARS model has far less RMSE than the linear model, so we prefer the use of model MARS when predicting the out-of-state tuition.