p8106_hw2

Hao Zheng(hz2770)

2022/3/5

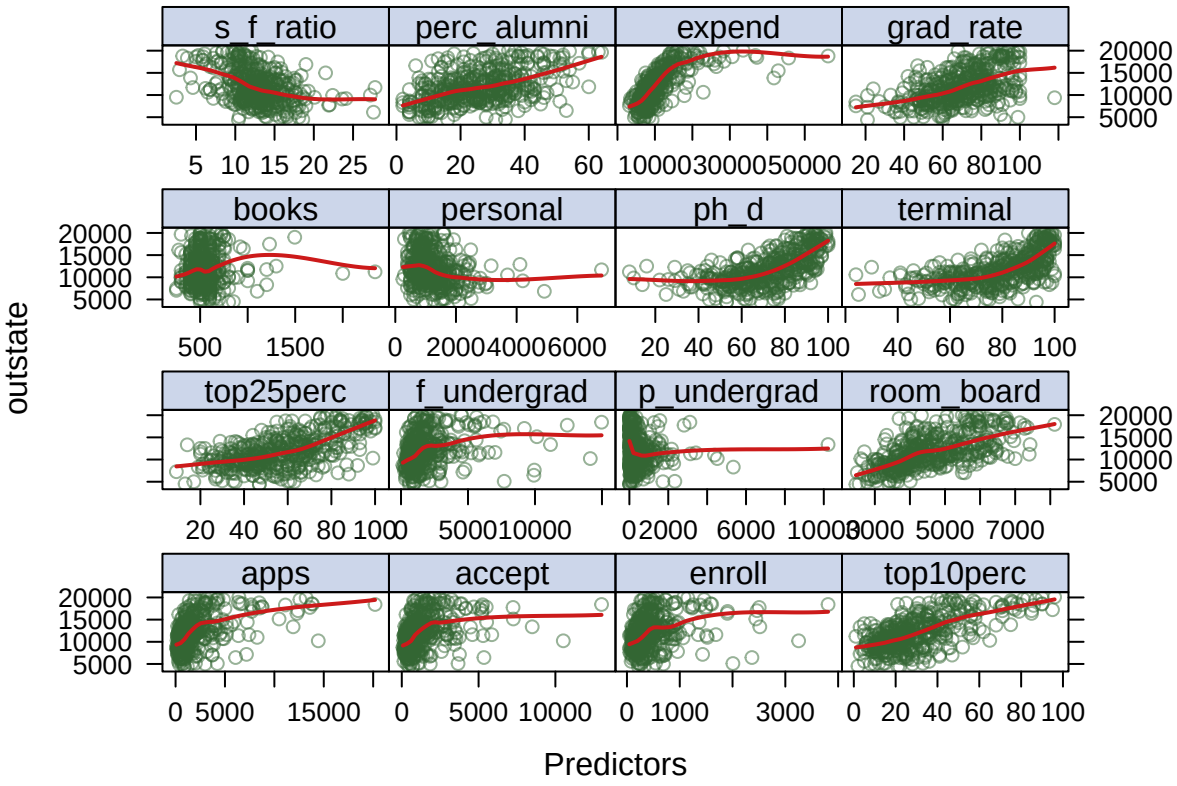
```
# Data Cleaning
dat =
  read.csv("./data/college.csv")[-1] %>%
  janitor::clean_names() %>%
  na.omit()

# 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
## 1 1660   1232    721        23        52        2885         537     7440
## 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
## 5  193    146     55        16        44         249         869     7560
## 6  587    479    158        38        62         678          41    13500
##   room_board books personal ph_d terminal s_f_ratio perc_alumni expend
## 1       3300   450     2200   70       78      18.1          12   7041
## 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
## 5       4120   800     1500   76       72      11.9           2  10922
## 6       3335   500      675   67       73       9.4          11   9727
##   grad_rate
## 1        60
## 2        56
## 3        54
## 4        59
## 5        15
## 6        55
```

Exploratory Data Analysis

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

```
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`.

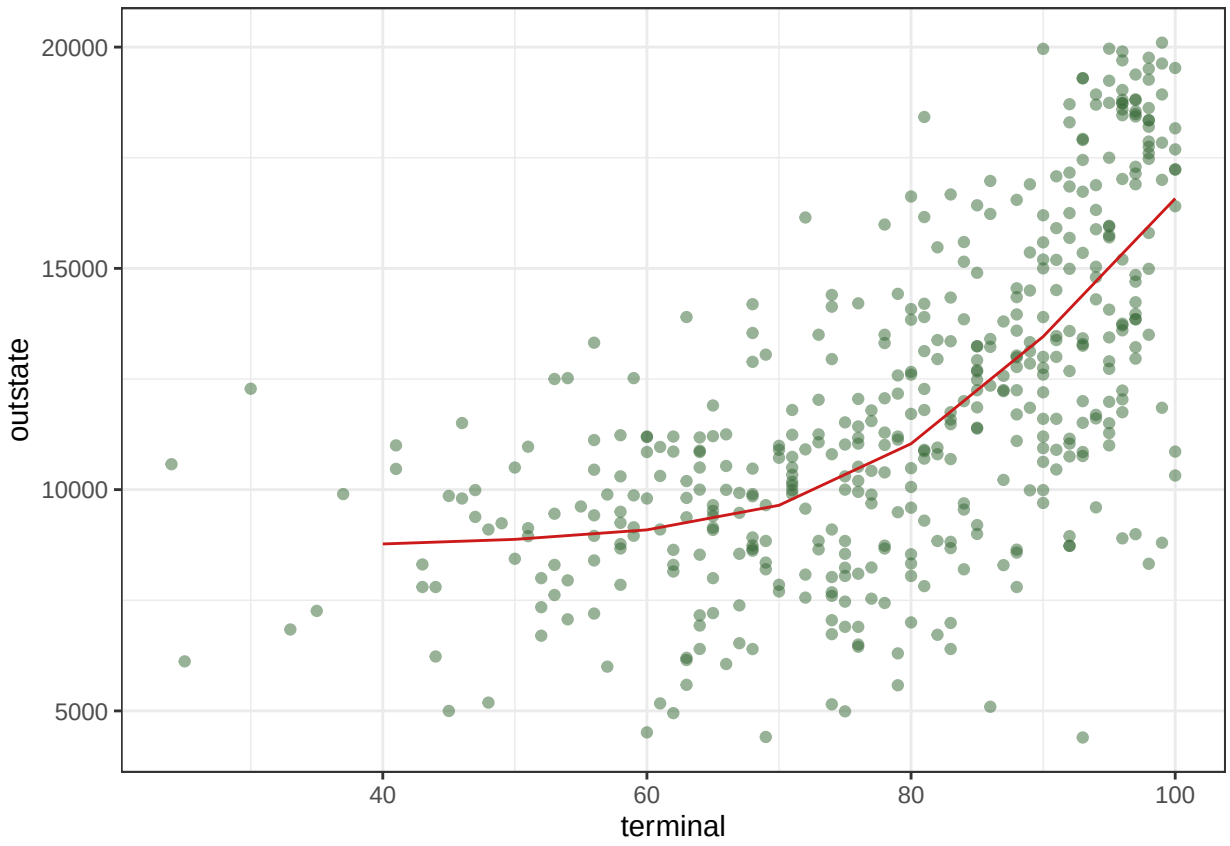
```
terminal.grid <- seq(from = 40, to = 100, by = 10)
fit.ss <- smooth.spline(trainData$terminal, trainData$outstate)
fit.ss$df
```

```
## [1] 4.31623
```

```
fit.ss$lambda
```

```
## [1] 0.03852559
```

```
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 smoothing spline model fitted for a range of degrees of freedom is 4.3162302. Then obtain the degrees of freedom using generalized cross-validation and plot the new fits.

```
fit.ss.cv <- 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
```

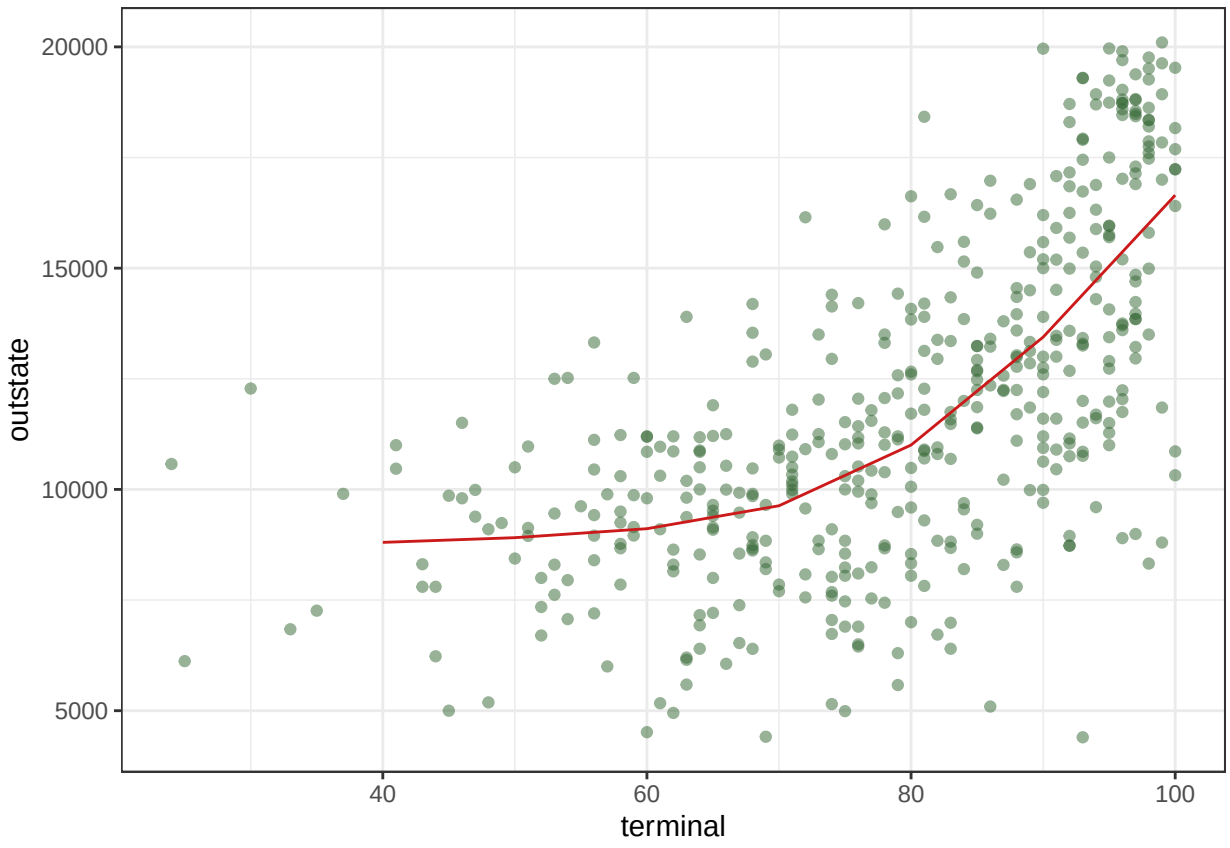
```
fit.ss.cv$df
```

```
## [1] 4.629441
```

```
fit.ss.cv$lambda
```

```
## [1] 0.02725455
```

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

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



Using cross-validation, we obtain the degrees of freedom 4.6294415 with lambda = 0.0272546.

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)
```

```
## Loading required package: mgcv
```

```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'
```

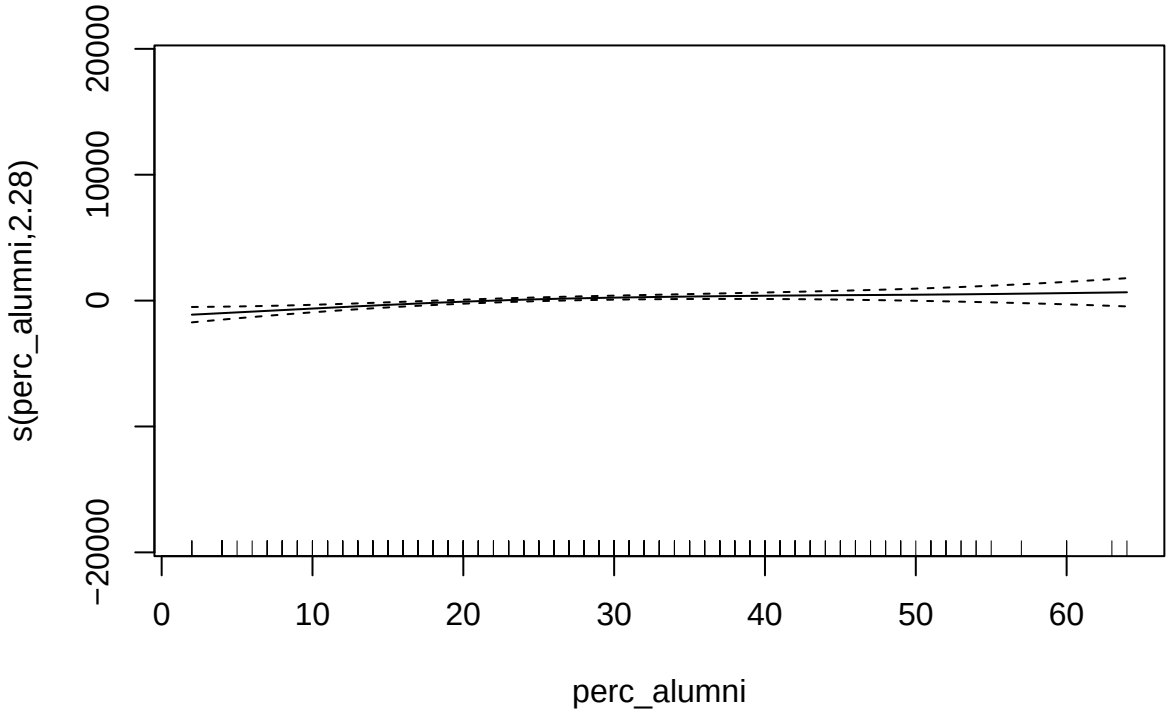
```
## The following object is masked from 'package:dplyr':
##
##     collapse
```

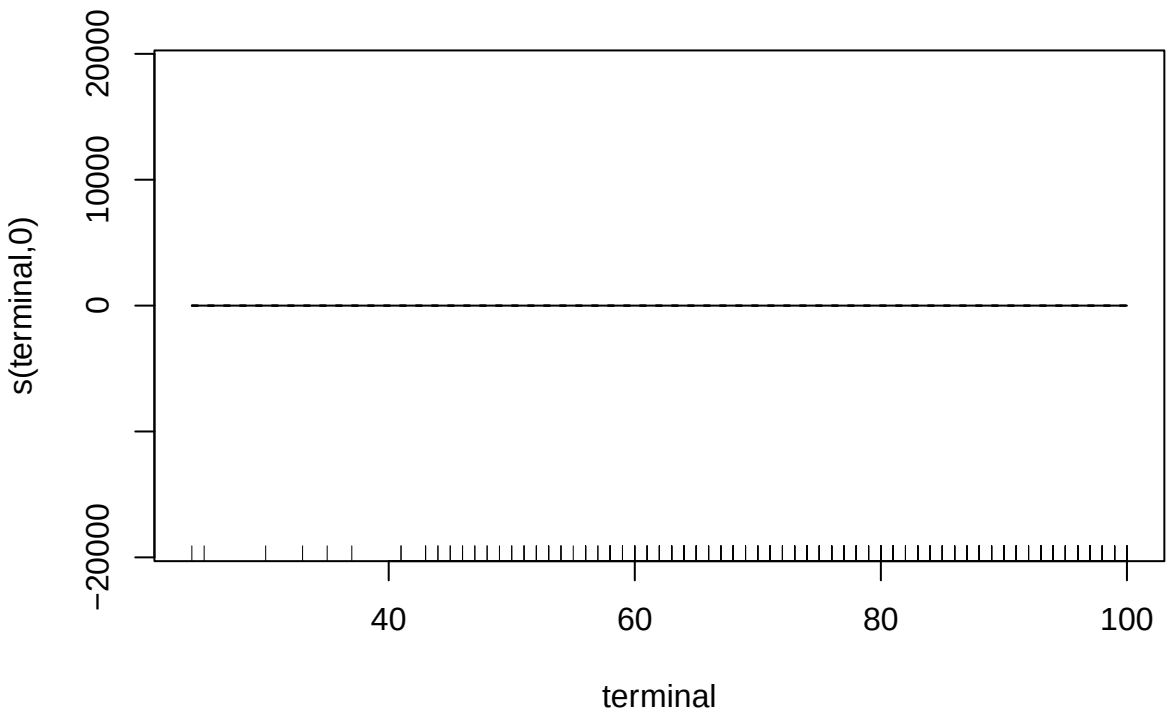
```
## This is mgcv 1.8-36. For overview type 'help("mgcv-package")'.
```

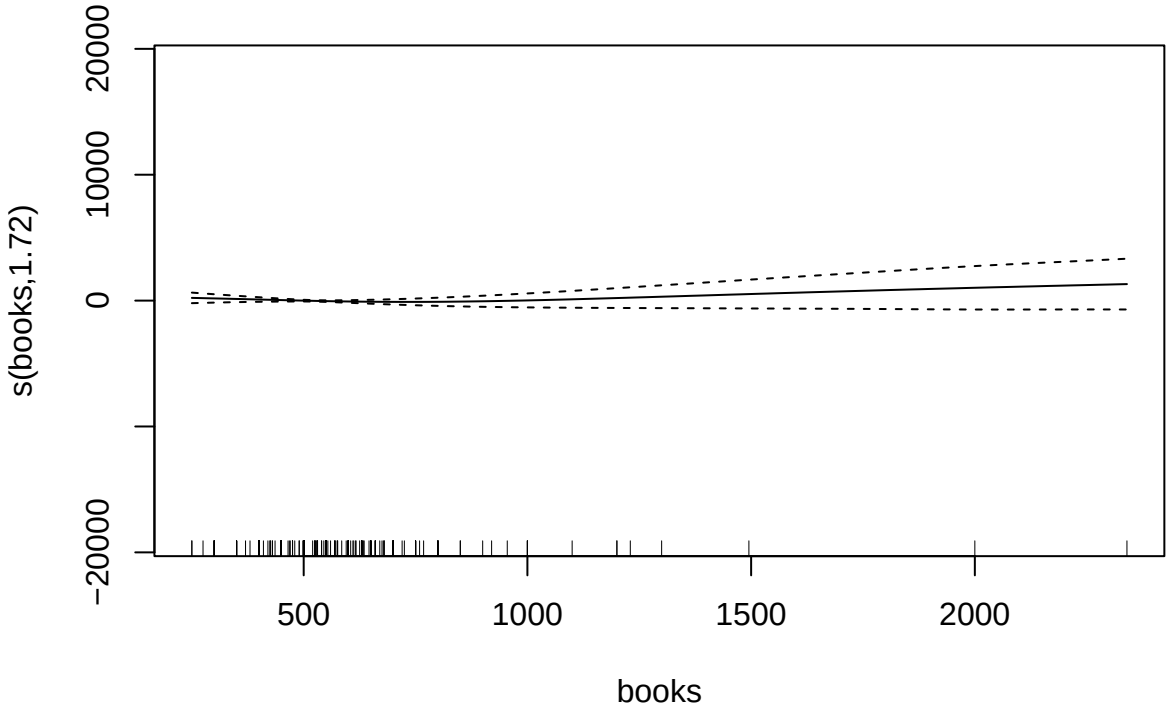
```
model.gam$finalModel
```

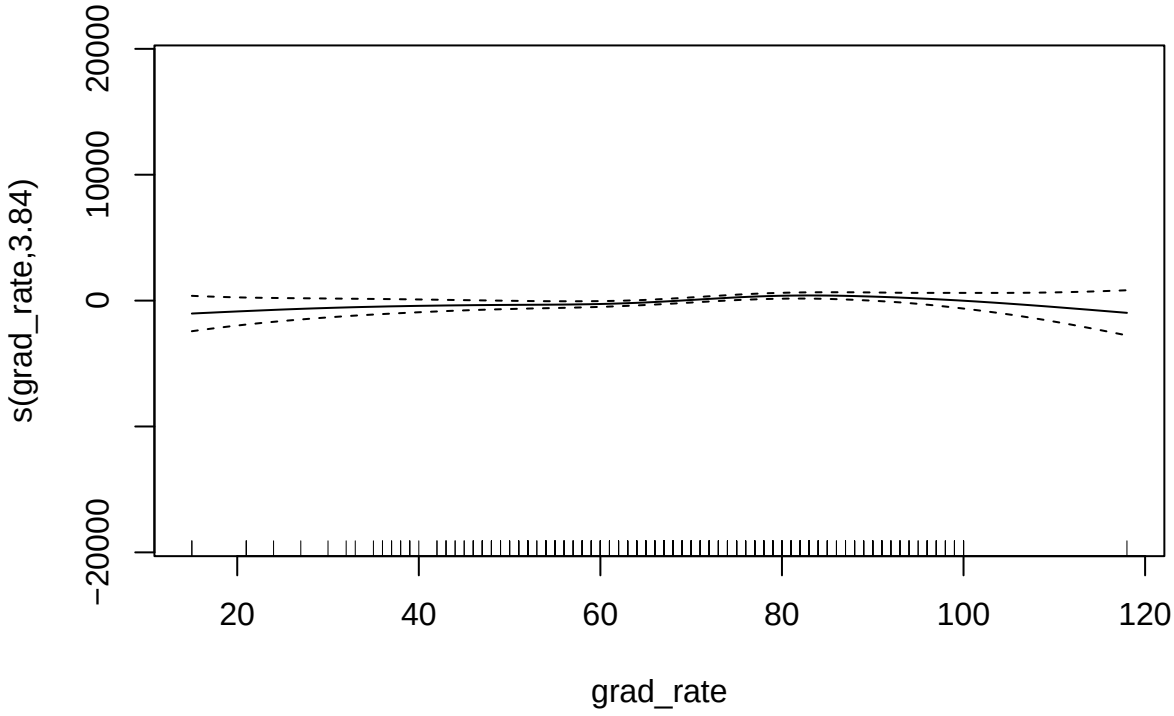
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ s(perc_alumni) + s(terminal) + s(books) + s(grad_rate) +
##     s(ph_d) + s(top10perc) + s(top25perc) + s(s_f_ratio) + s(personal) +
##     s(p_undergrad) + s(room_board) + s(enroll) + s(accept) +
##     s(f_undergrad) + s(apps) + s(expend)
##
## Estimated degrees of freedom:
## 2.277 0.000 1.723 3.840 5.169 5.417 0.612
## 3.626 0.763 0.000 1.879 0.974 3.536 5.824
## 4.304 5.005  total = 45.95
##
## GCV score: 2745565
```

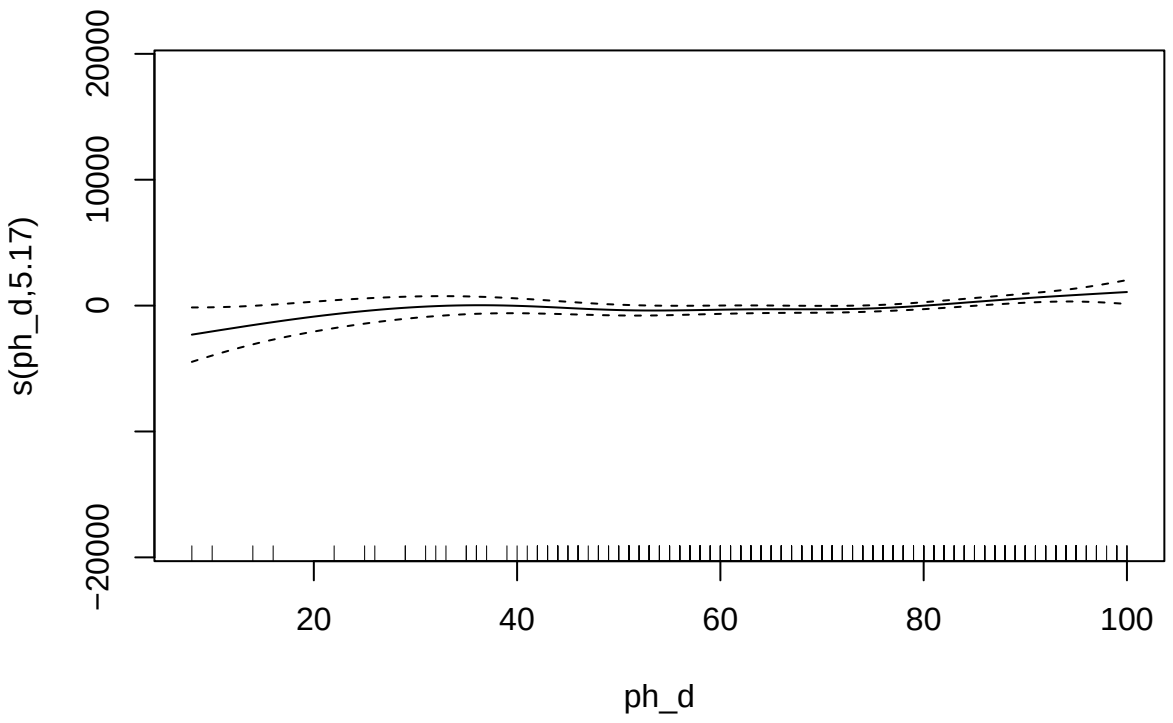
```
plot(model.gam$finalModel)
```

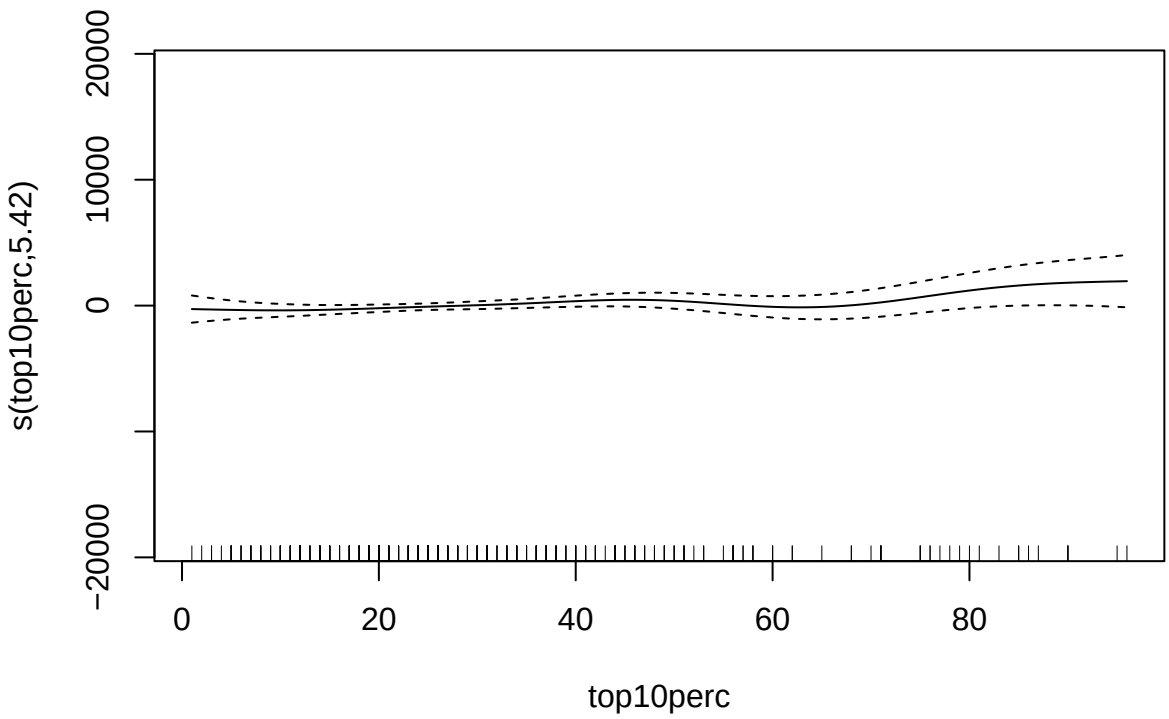

s(perc_alumni,2.28)
20000
10000
0
−20000
0
10
20
30
40
50
60
perc_alumni

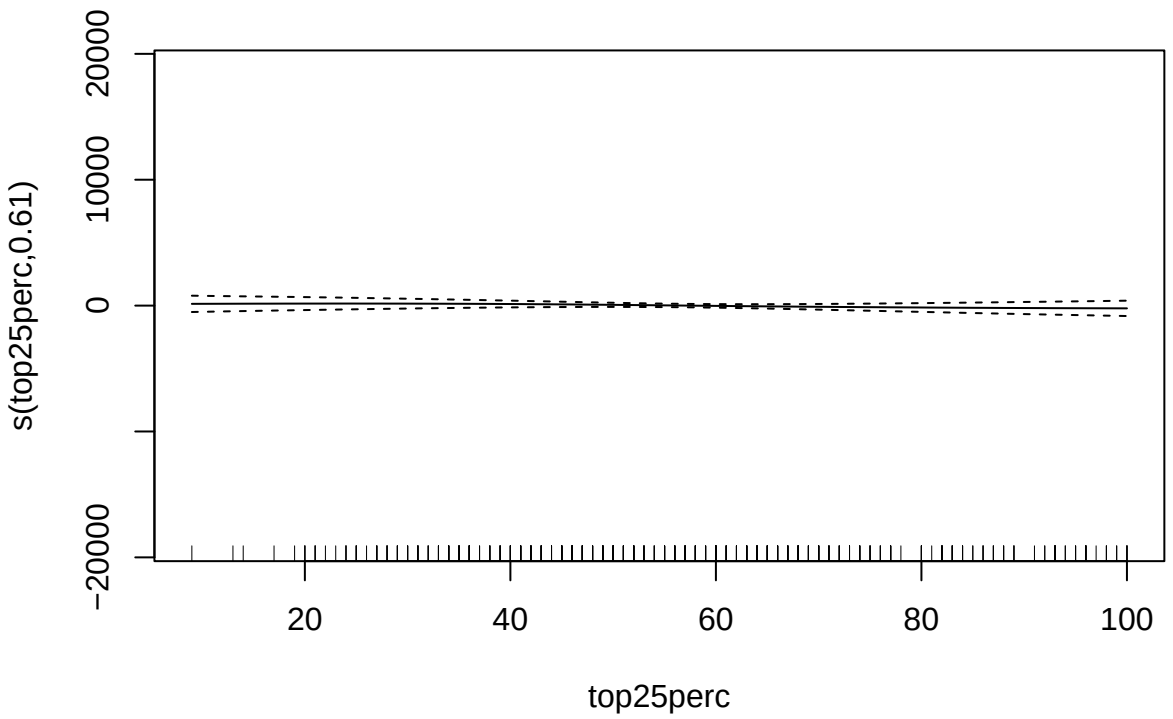

s(terminal,0)
20000
10000
0
−20000
40
60
80
100
terminal

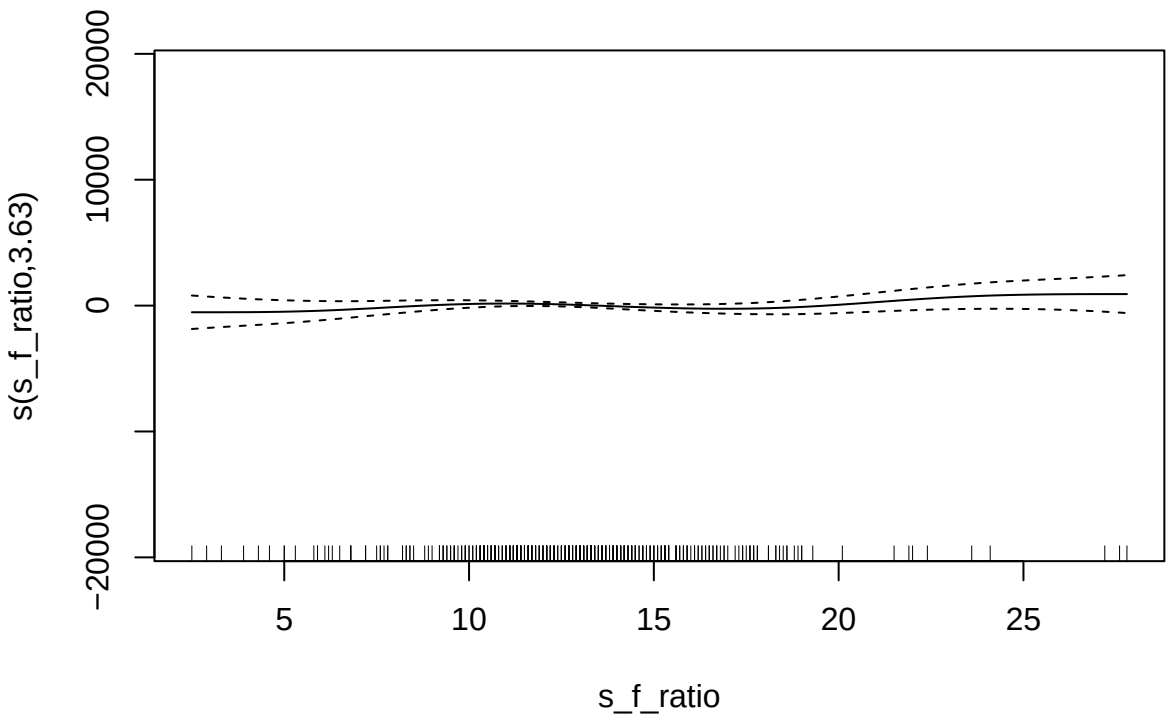

s(books,1.72)
20000
10000
0
−20000
500
1000
1500
2000
books

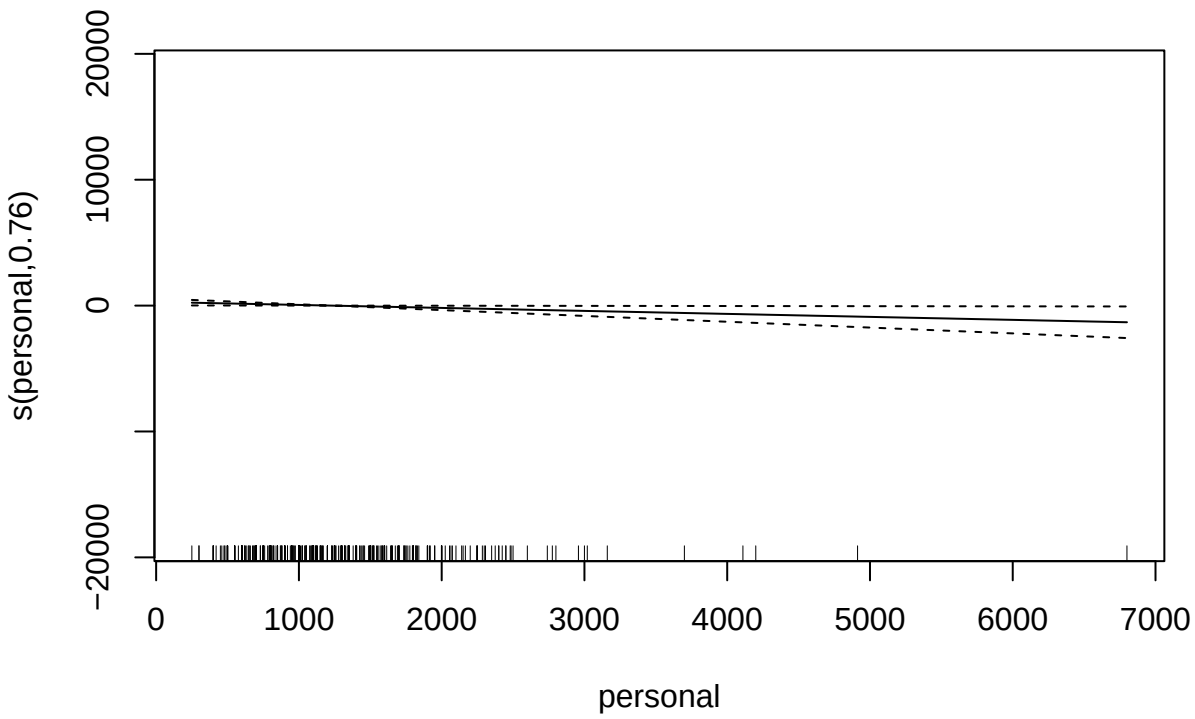

s(grad_rate,3.84)
20000
10000
0
−20000
20
40
60
80
100
120
grad_rate

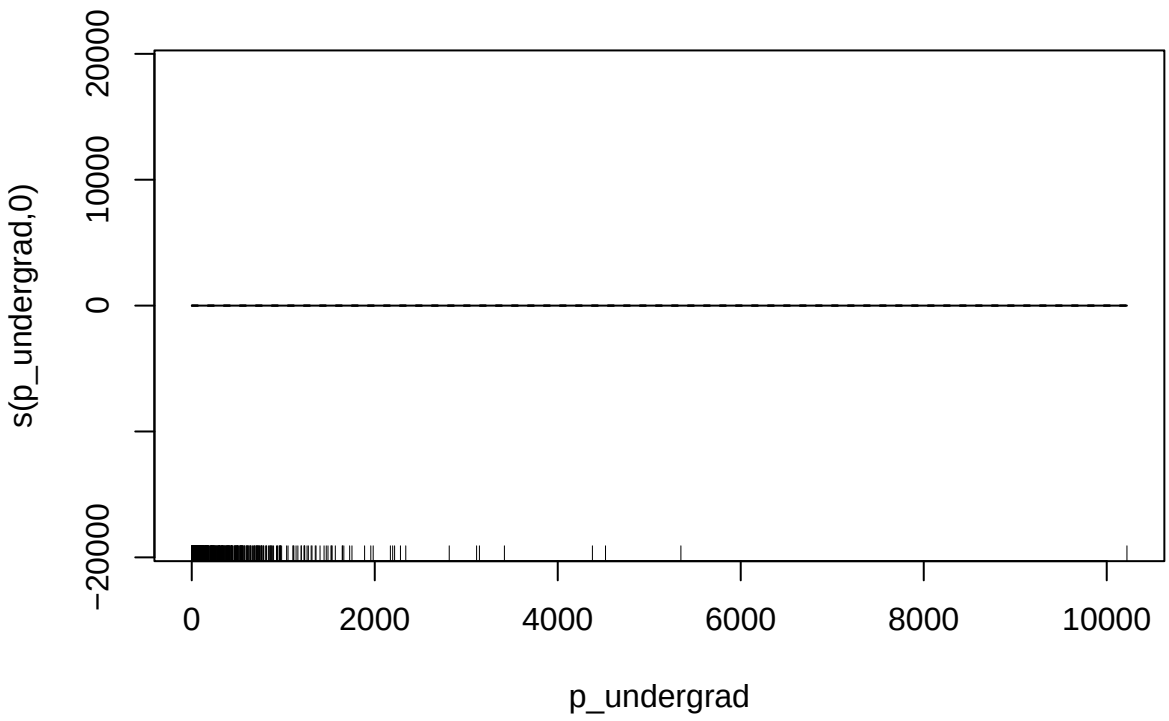

s(ph_d,5.17)
20000
10000
0
−20000
20
40
60
80
100
ph_d

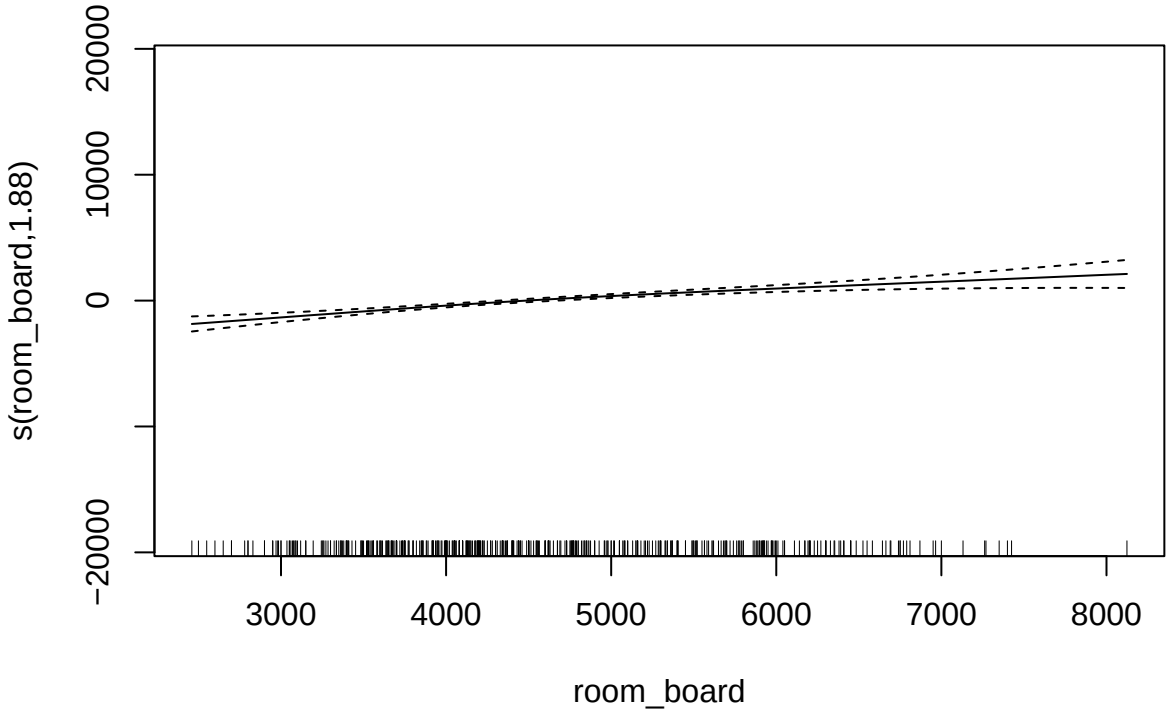

s(top10perc,5.42)
20000
10000
0
−20000
0
20
40
60
80
top10perc

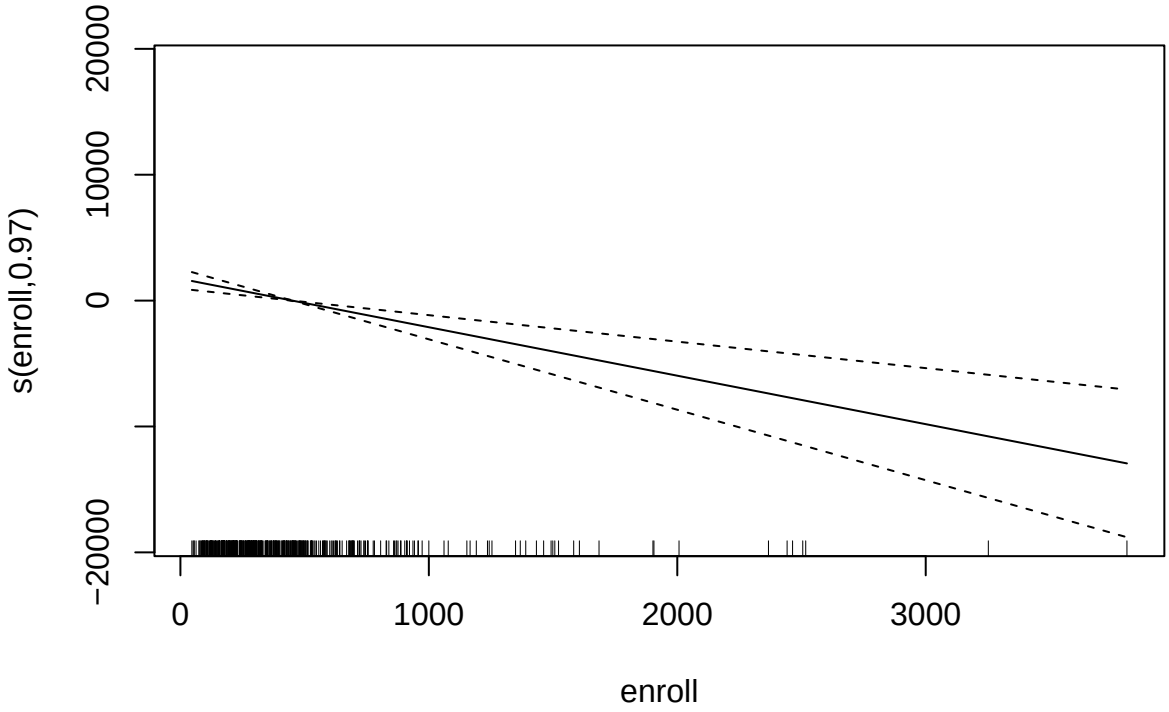

s(top25perc,0.61)
20000
10000
0
−20000
20
40
60
80
100
top25perc

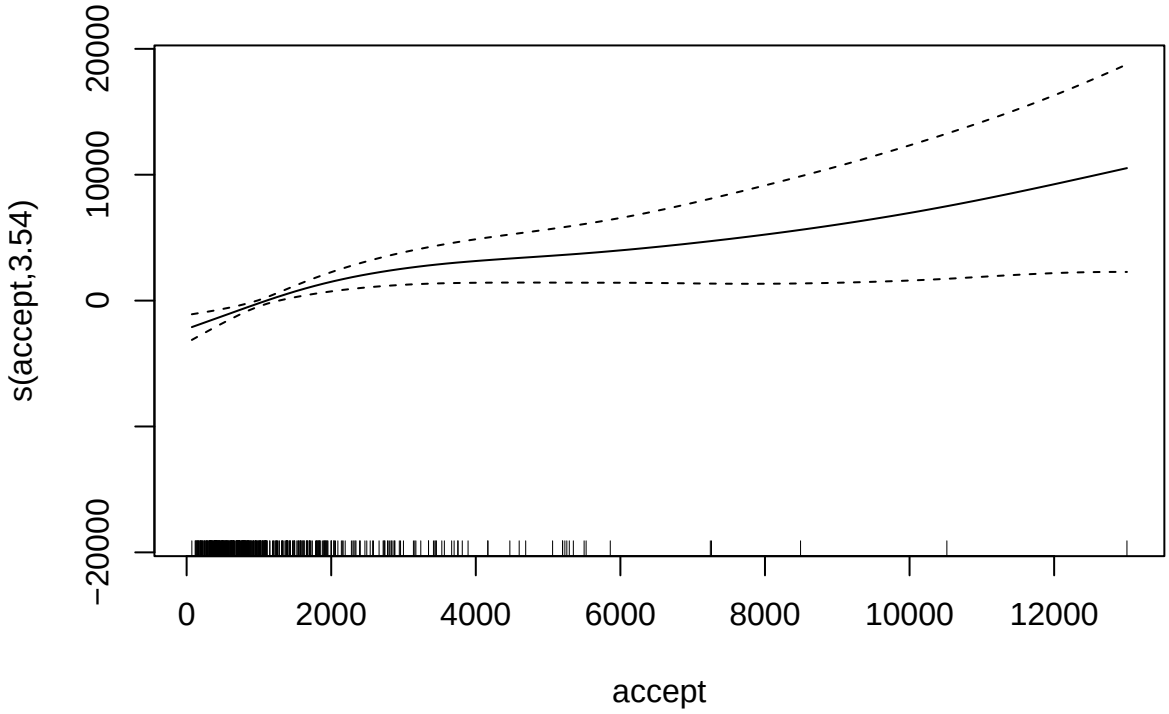

s(s_f_ratio,3.63)
20000
10000
0
−20000
5
10
15
20
25
s_f_ratio

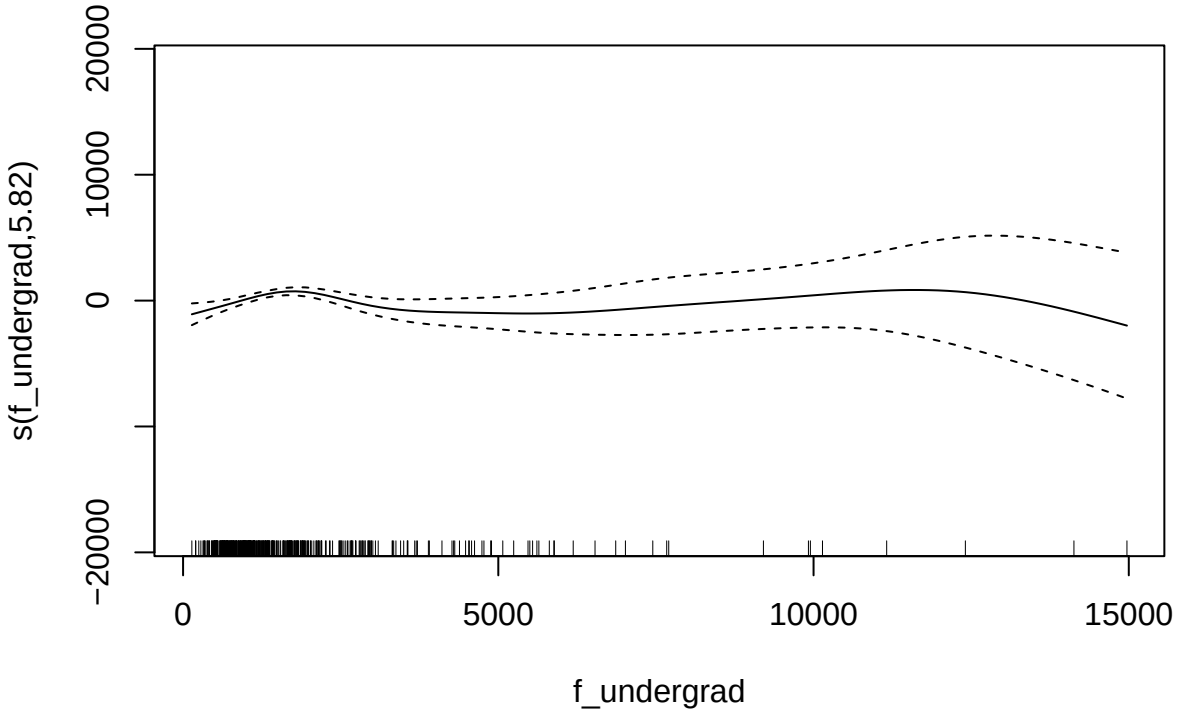

s(personal,0.76)
20000
10000
0
−20000
0
1000
2000
3000
4000
5000
6000
7000
personal

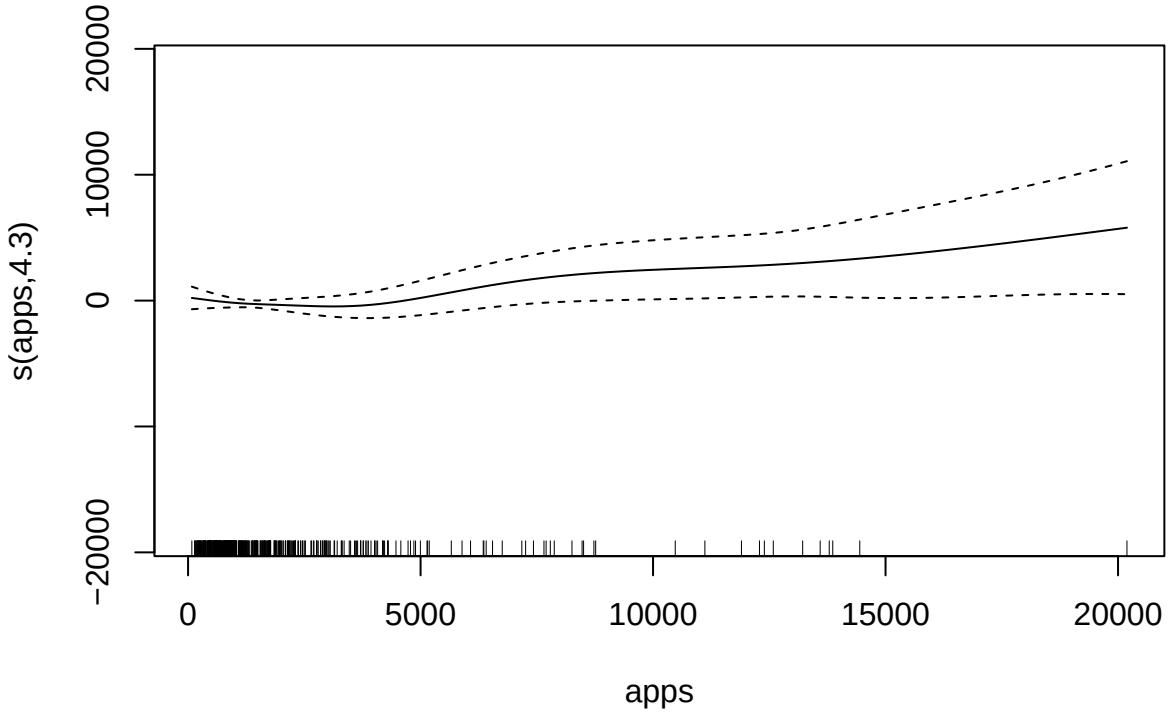

s(p_undergrad,0)
20000
10000
0
−20000
0
2000
4000
6000
8000
10000
p_undergrad

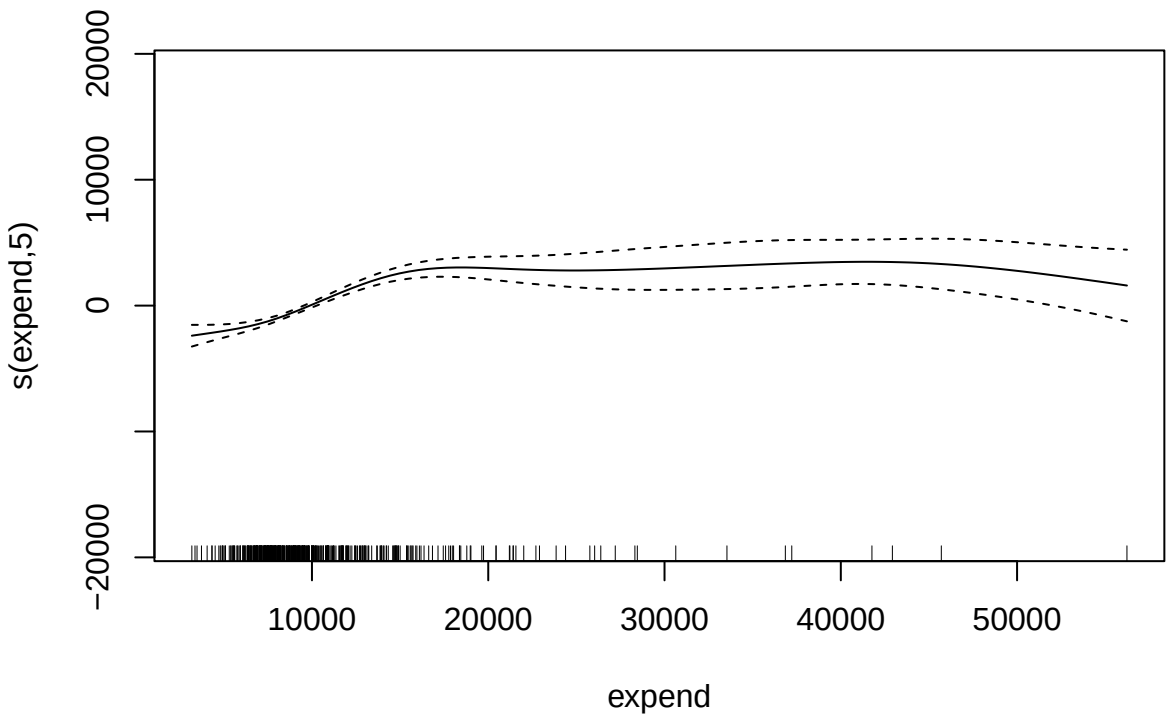

s(room_board,1.88)
20000
10000
0
−20000
3000
4000
5000
6000
7000
8000
room_board


s(enroll,0.97)
20000
10000
0
−20000
0
1000
2000
3000
enroll


s(accept,3.54)
20000
10000
0
−20000
0
2000
4000
6000
8000
10000
12000
accept


s(f_undergrad,5.82)
20000
10000
0
−20000
0
5000
10000
15000
f_undergrad


s(apps,4.3)
20000
10000
0
−20000
0
5000
10000
15000
20000
apps




```
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] 7048354
```

The test error for the GAM model is 7.0483544×10^6.

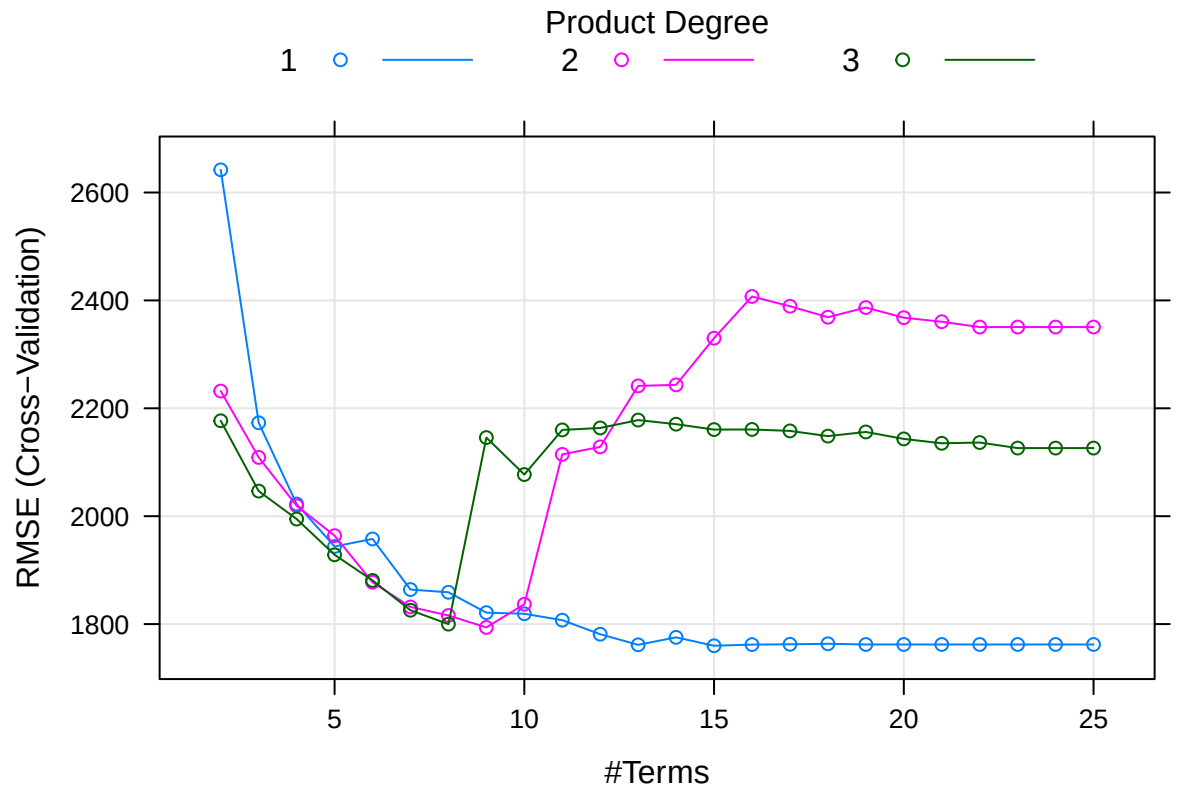
Multivariate Adaptive Regression Spline (MARS)

```
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
## 14     15      1
```

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

```
##         (Intercept)     h(expend-15365)  h(room_board-4500)  h(4500-room_board)
##        9999.1281135          -0.6137213           0.3844399          -1.0488316
##     h(grad_rate-97)     h(97-grad_rate) h(f_undergrad-1270) h(1270-f_undergrad)
##        -190.8558850         -18.4460542          -0.3525970          -1.7773452
##   h(22-perc_alumni)        h(apps-2212)       h(973-enroll)      h(2037-accept)
##         -89.7299783           0.4046535           5.1251675          -1.9839326
##      h(expend-6889)          h(ph_d-79)    h(1300-personal)
##           0.6086850          64.0689926           0.8997170
```

Then we calculate the test error on the test data.

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

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

```
## [1] 3019584
```

The test error is 3.0195837×10^6.

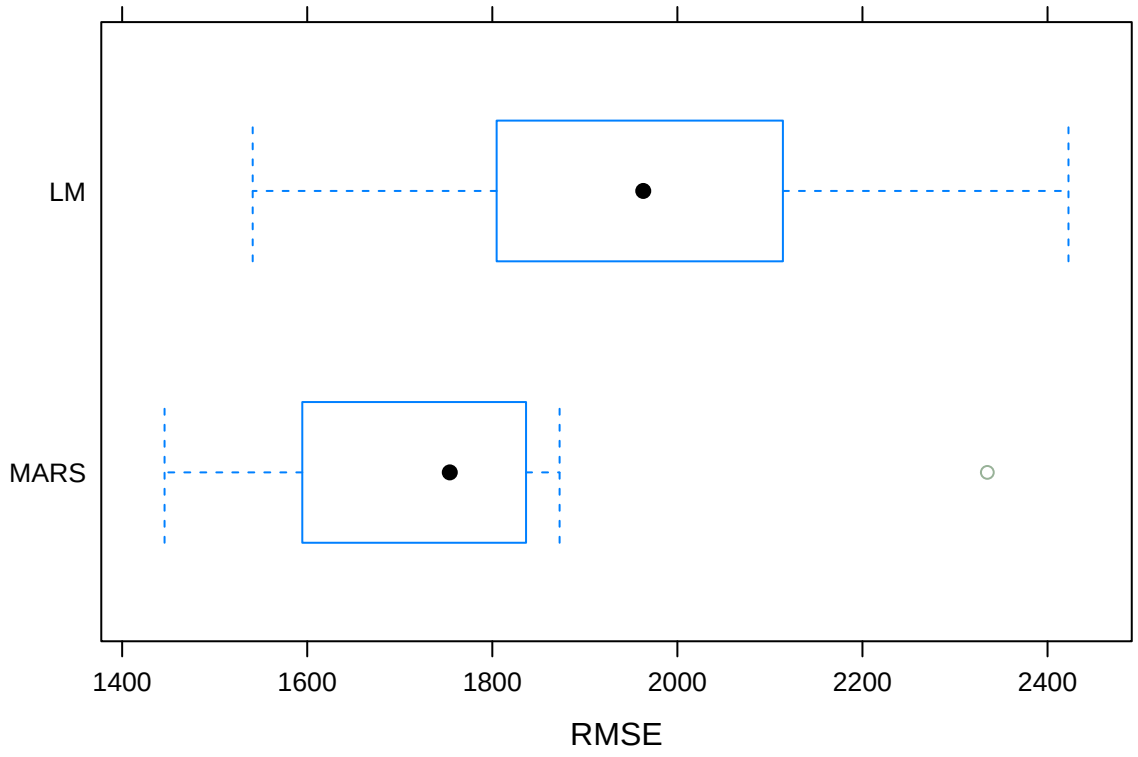
Model Selection

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

resamp <- resamples(list(MARS = model.mars,
                         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 1176.573 1291.42 1373.768 1391.373 1438.079 1779.644    0
## LM   1328.532 1409.13 1588.646 1577.186 1744.718 1788.333    0
##
## RMSE
##          Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## MARS 1445.809 1619.690 1754.058 1759.833 1833.005 2334.945    0
## LM   1541.059 1805.925 1963.117 1961.154 2112.585 2422.643    0
```

```
##
## Rsquared
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## MARS 0.5827976 0.7291782 0.7946951 0.7635709 0.8121574 0.8373916    0
## LM   0.6553657 0.6867260 0.7215434 0.7192811 0.7409446 0.7945244    0
```

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



The MARS model has far less RMSE than the linear model, so we prefer the use of model MARS.