hw2

Johnstone Tcheou

2025-03-04

Contents

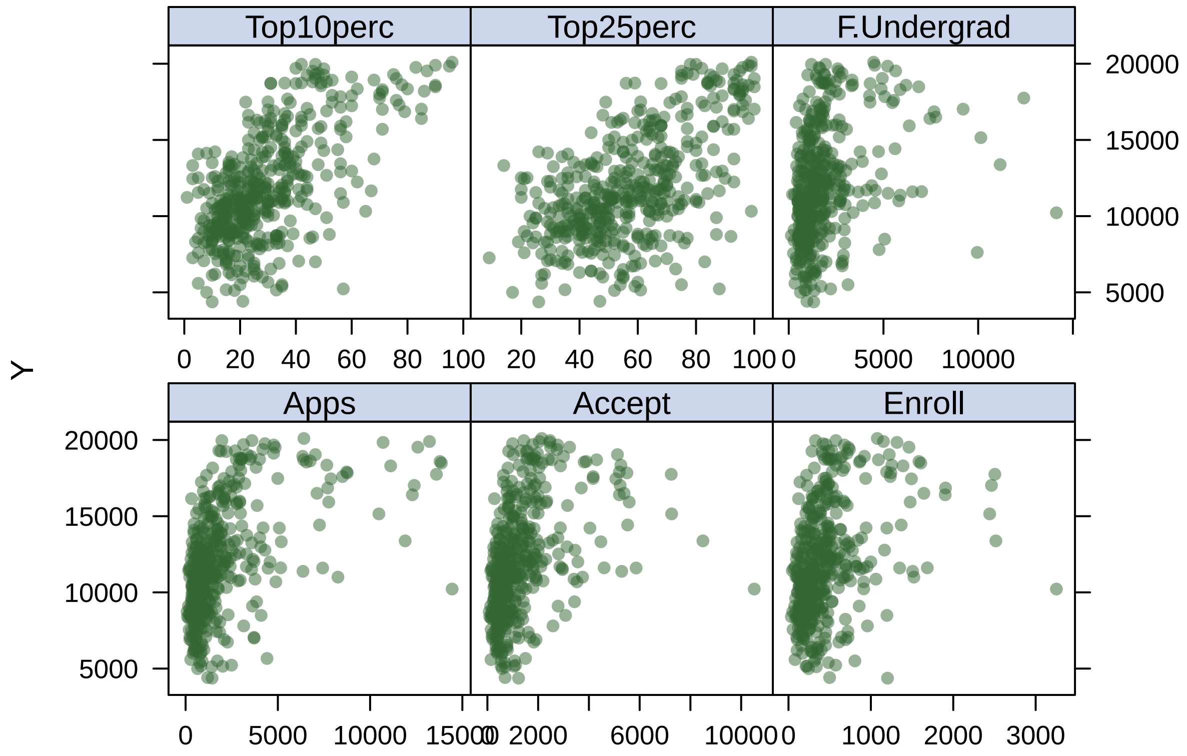
```r
library(caret)
library(tidymodels)
library(splines)
library(mgcv)
library(pdp)
library(earth)
library(ggplot2)
```

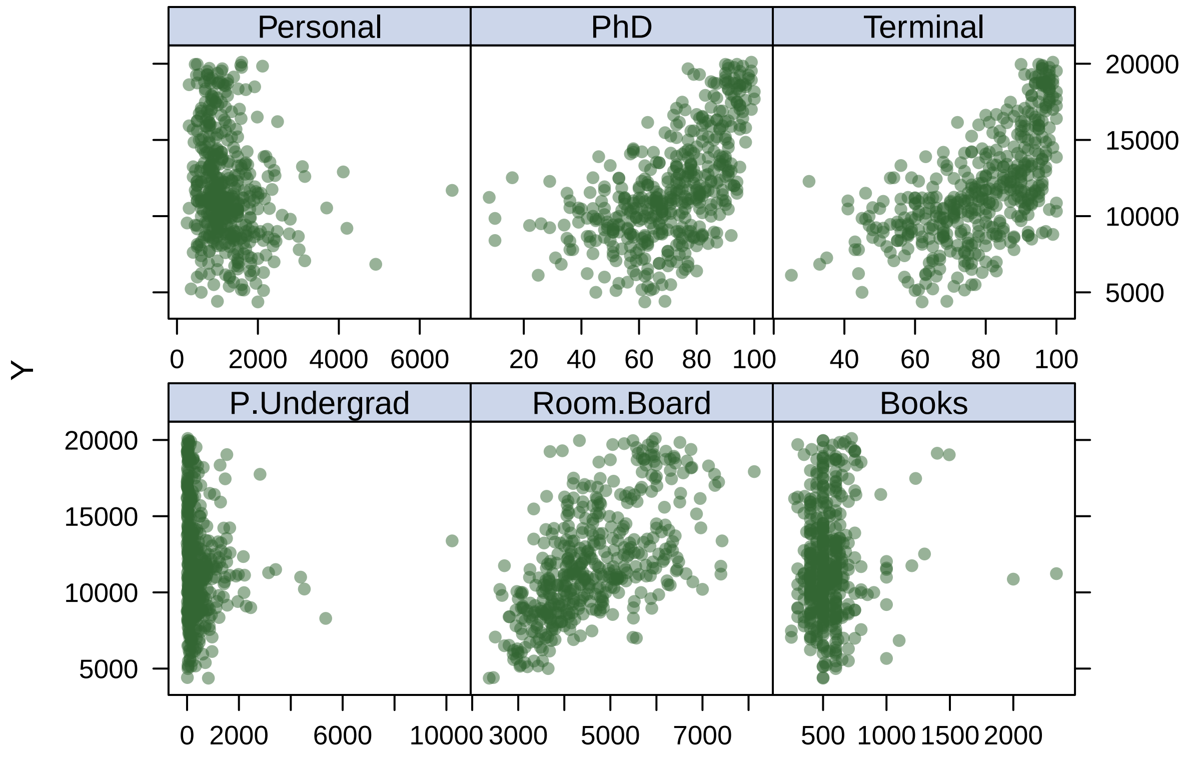
First, we need to set a seed for reproducibility. Then we will import the data and conduct a training/testing split, using 80% for training and 20% for testing.

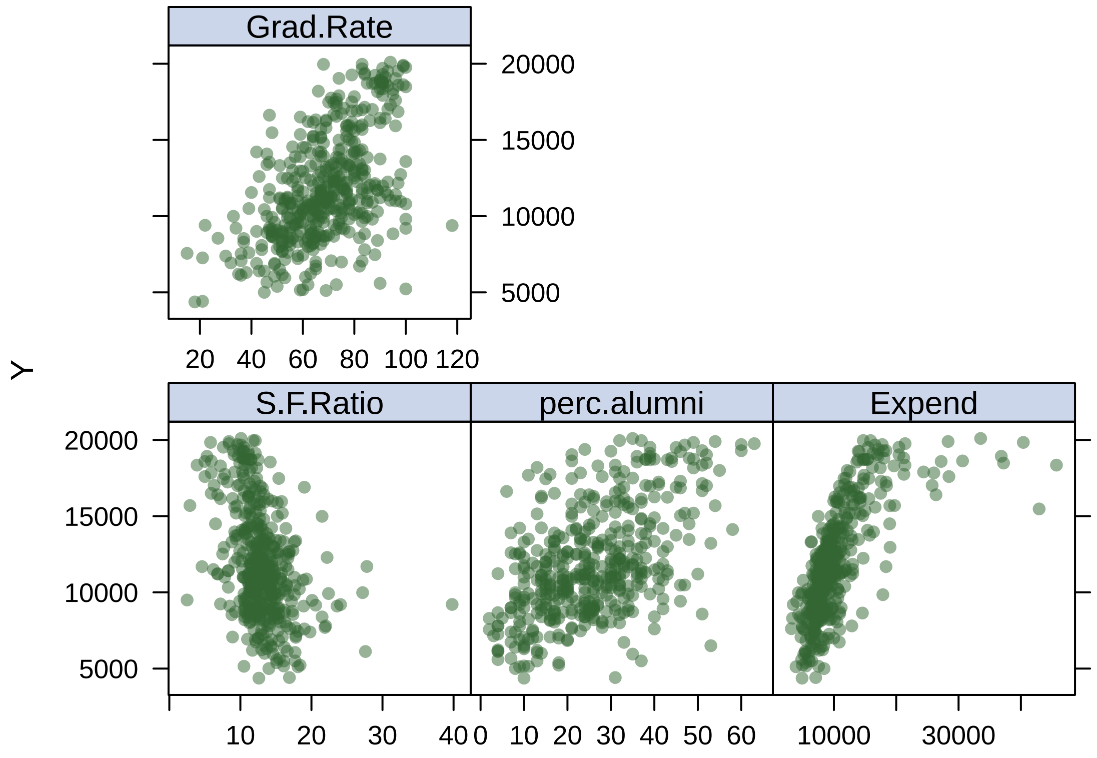
```r
set.seed(81062)
college <- read.csv("College.csv")

# college <- college |> na.omit()

data_split <- initial_split(college, prop = 0.8)
training <- training(data_split)
testing <- testing(data_split)

training_y <- training$Outstate
training_x <- model.matrix(Outstate ~ ., training)[,-1]
training_predictors <- training_x[, (ncol(training_x)-15):(ncol(training_x))]

testing_y <- testing$Outstate
testing_x <- model.matrix(Outstate ~ ., testing)[,-1]
testing_predictors <- testing_x[, (ncol(testing_x)-15):(ncol(testing_x))]
```

Except for the dummy variables created for each college, the other predictors are continuous. We can examine their relationships with our outcome variable, `Outstate` - out of state tuition. With the training data, only the predictors with indexes 452-467 are used and the College predictors are discarded.

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

featurePlot(training_predictors, training_y, plot = "scatter", labels = c("", "Y"),
            type = c("p"), layout = c(3, 2))
```


Top10perc
Top25perc
F.Undergrad
Apps
Accept
Enroll
Y
20000
15000
10000
5000
0
20
40
60
80
100
0
5000
10000
15000
2000
6000
10000
1000
2000
3000


Personal
PhD
Terminal
P.Undergrad
Room.Board
Books
Y
20000
15000
10000
5000
0
2000
4000
6000
20
40
60
80
100
6000
10000
3000
5000
7000
500
1000
1500
2000

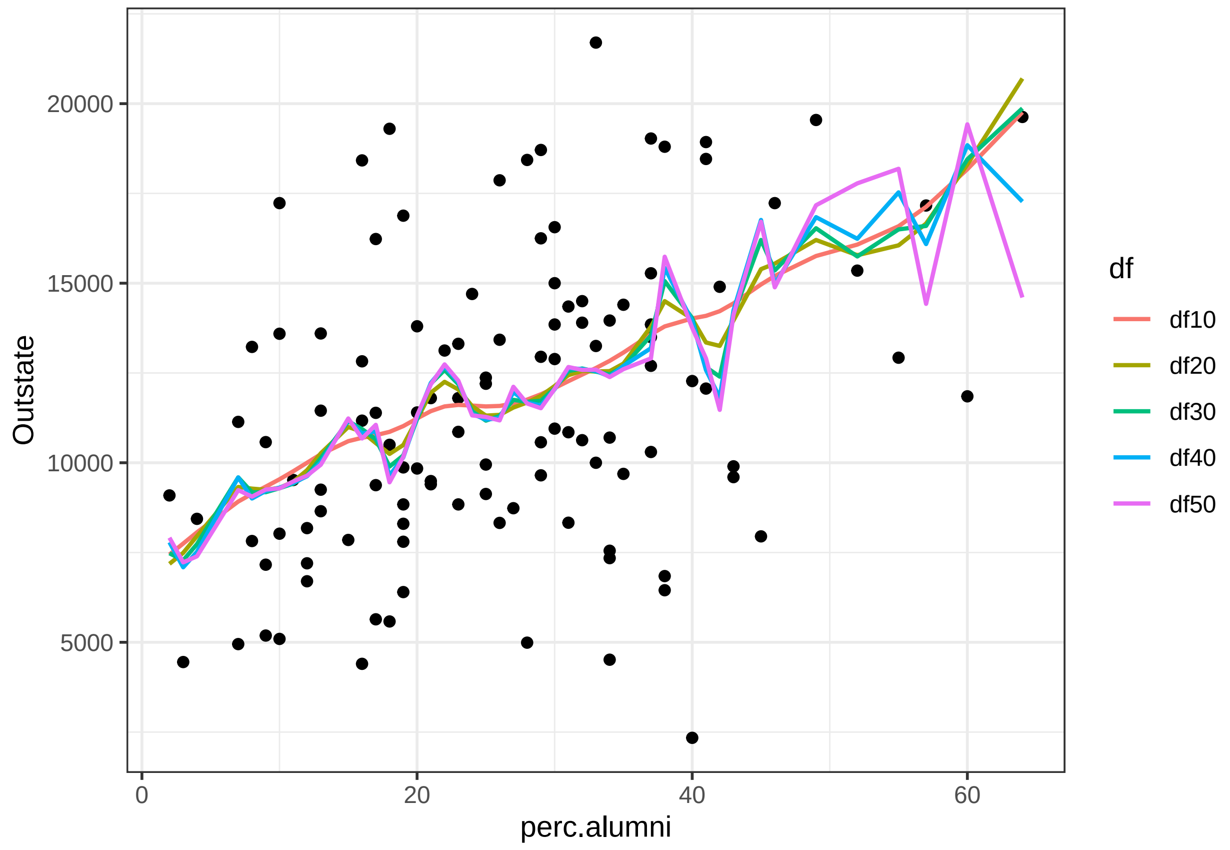


Potential variables with nonlinear relationships with `Outstate` include `Apps`, `Accept`, `Enroll`, `F.Undergrad`, `Expend`, `S.F.Ratio`, `PhD`, and `Terminal` (both of which appear to have some exponential aspect).

Fitting smoothing spline

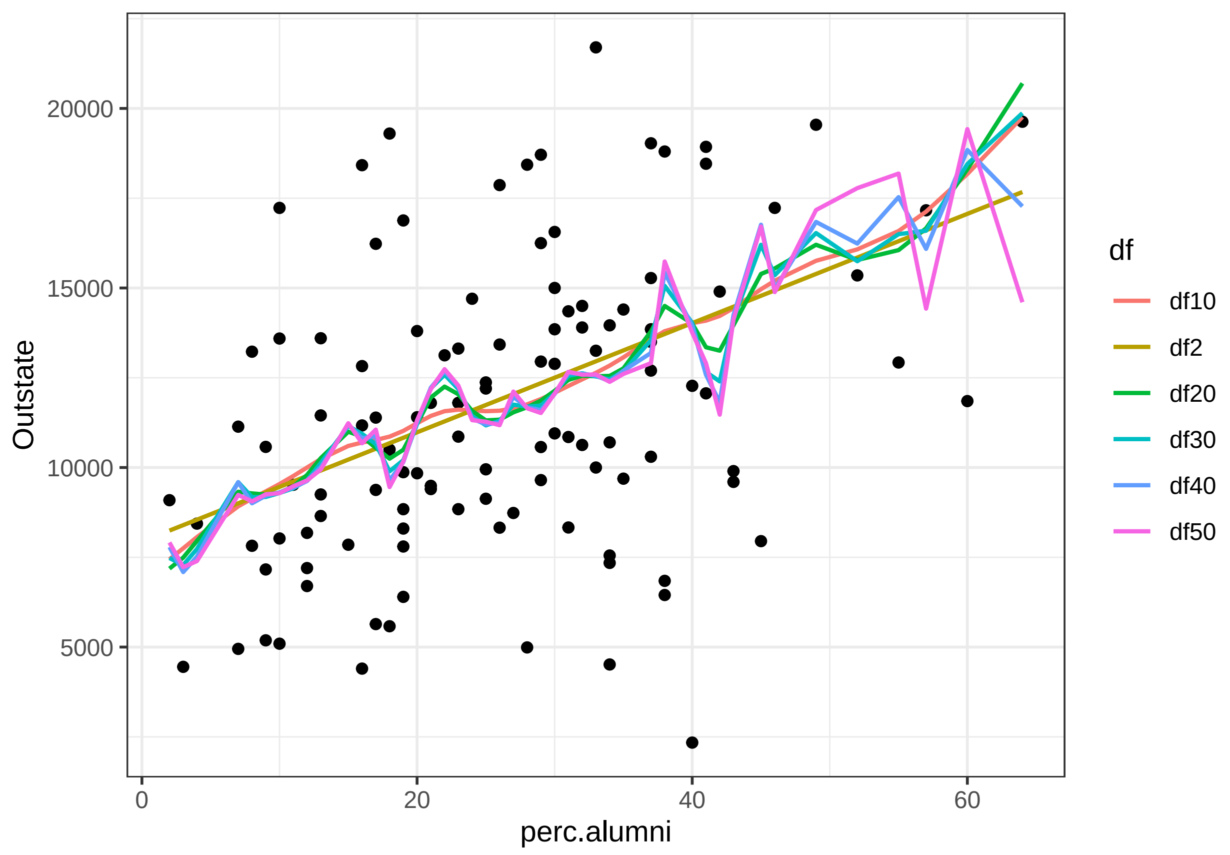
The range of degrees of freedom is from 1 to the number of unique predictor values, which is 57. Therefore, a range of degrees of freedom is tested, ranging from 10 to 50 by an increment of 10.

```
ss.10 <-
  smooth.spline(
    training_predictors[, "perc.alumni"],
    training_y,
    df = 10
  )

ss.20 <-
  smooth.spline(
    training_predictors[, "perc.alumni"],
    training_y,
    df = 20
  )

ss.30 <-
  smooth.spline(
    training_predictors[, "perc.alumni"],
    training_y,
    df = 30
  )

ss.40 <-
  smooth.spline(
    training_predictors[, "perc.alumni"],
    training_y,
    df = 40
  )

ss.50 <-
  smooth.spline(
    training_predictors[, "perc.alumni"],
    training_y,
    df = 50
  )

pred.ss.10 <-
  predict(
    ss.10,
    x = testing_x[, "perc.alumni"],
  )

pred.ss.20 <-
  predict(
    ss.20,
    x = testing_x[, "perc.alumni"]
  )

pred.ss.30 <-
  predict(
    ss.30,
    x = testing_x[, "perc.alumni"]
  )

pred.ss.40 <-
  predict(
    ss.40,
    x = testing_x[, "perc.alumni"]
  )

pred.ss.50 <-
  predict(
    ss.50,
    x = testing_x[, "perc.alumni"]
  )

combined <-
  cbind(
    x = testing_x[, "perc.alumni"],
    df10 = as.data.frame(pred.ss.10)[, "y"],
    df20 = as.data.frame(pred.ss.20)[, "y"],
    df30 = as.data.frame(pred.ss.30)[, "y"],
    df40 = as.data.frame(pred.ss.40)[, "y"],
    df50 = as.data.frame(pred.ss.50)[, "y"]
  ) |>
  as.data.frame() |>
  pivot_longer(
    !x,
    names_to = "df",
    values_to = "pred"
  )

ggplot(aes(x = perc.alumni, y = Outstate), data = testing) +
  geom_point() +
  geom_line(aes(x = x, y = pred, col = df), linewidth = 0.75, data = combined) +
  theme_bw()
```

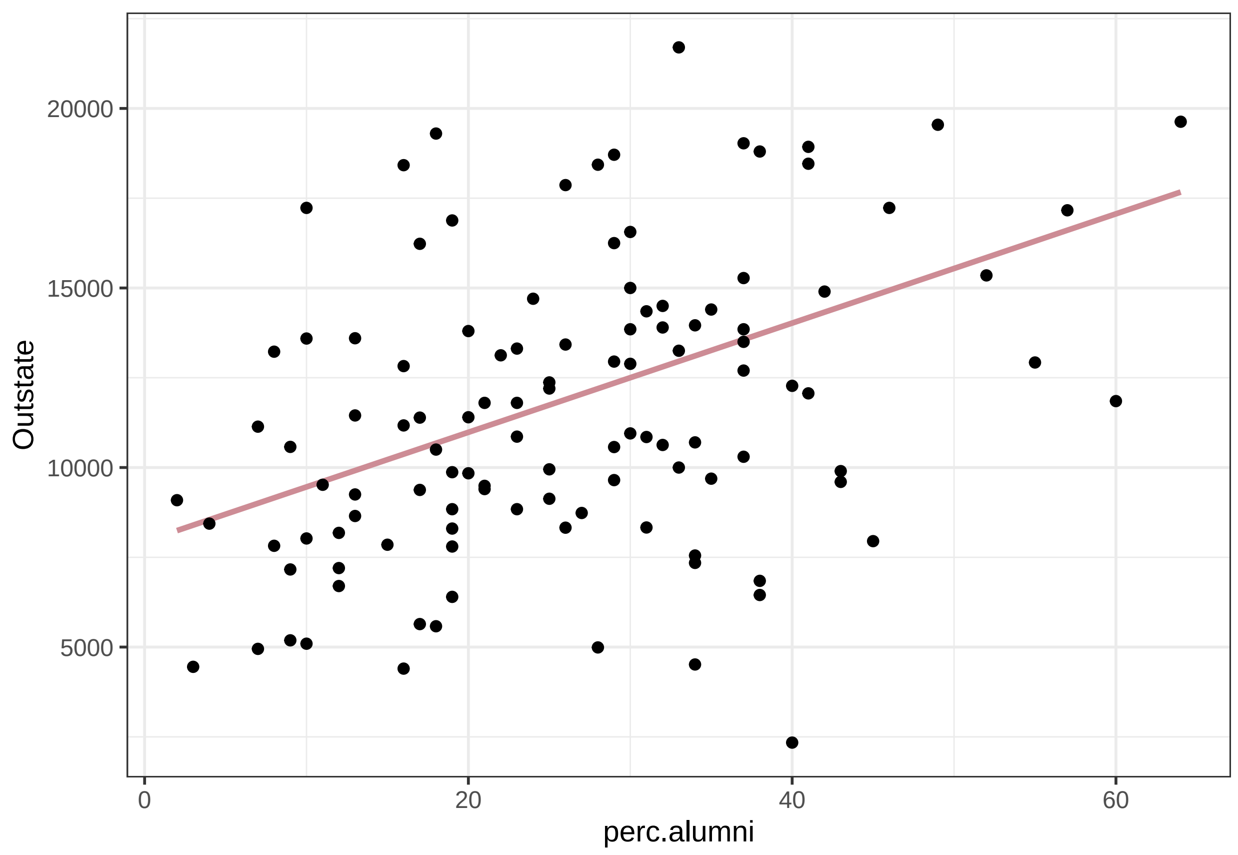


As visualized above, when fitting the predicted values for a given degree of freedom to the testing dataset, as the degrees of freedom increases, the roughness of the fitted curve increases. With degrees of freedom = 10, the curve is at its smoothest, but with degrees of freedom = 50, the curve is most jagged. Plots for degrees of freedom > 10 look to be overfitting the data and are too jagged to capture the approximately linear relation, so the optimal degree of freedom is likely between 1 and 10.

```
ss <-
  smooth.spline(
    training_predictors[, "perc.alumni"],
    training_y
  )

pred.ss <-
  predict(
    ss,
    x = testing_x[, "perc.alumni"]
  )

combined_optimal <-
  cbind(
    x = testing_x[, "perc.alumni"],
    df2 = as.data.frame(pred.ss)[, "y"],
    df10 = as.data.frame(pred.ss.10)[, "y"],
    df20 = as.data.frame(pred.ss.20)[, "y"],
    df30 = as.data.frame(pred.ss.30)[, "y"],
    df40 = as.data.frame(pred.ss.40)[, "y"],
    df50 = as.data.frame(pred.ss.50)[, "y"]
  ) |>
  as.data.frame() |>
  pivot_longer(
    !x,
    names_to = "df",
    values_to = "pred"
  )

ggplot(aes(x = perc.alumni, y = Outstate), data = testing) +
  geom_point() +
  geom_line(aes(x = x, y = pred, col = df), linewidth = 0.75, data = combined_optimal) +
  theme_bw()
```



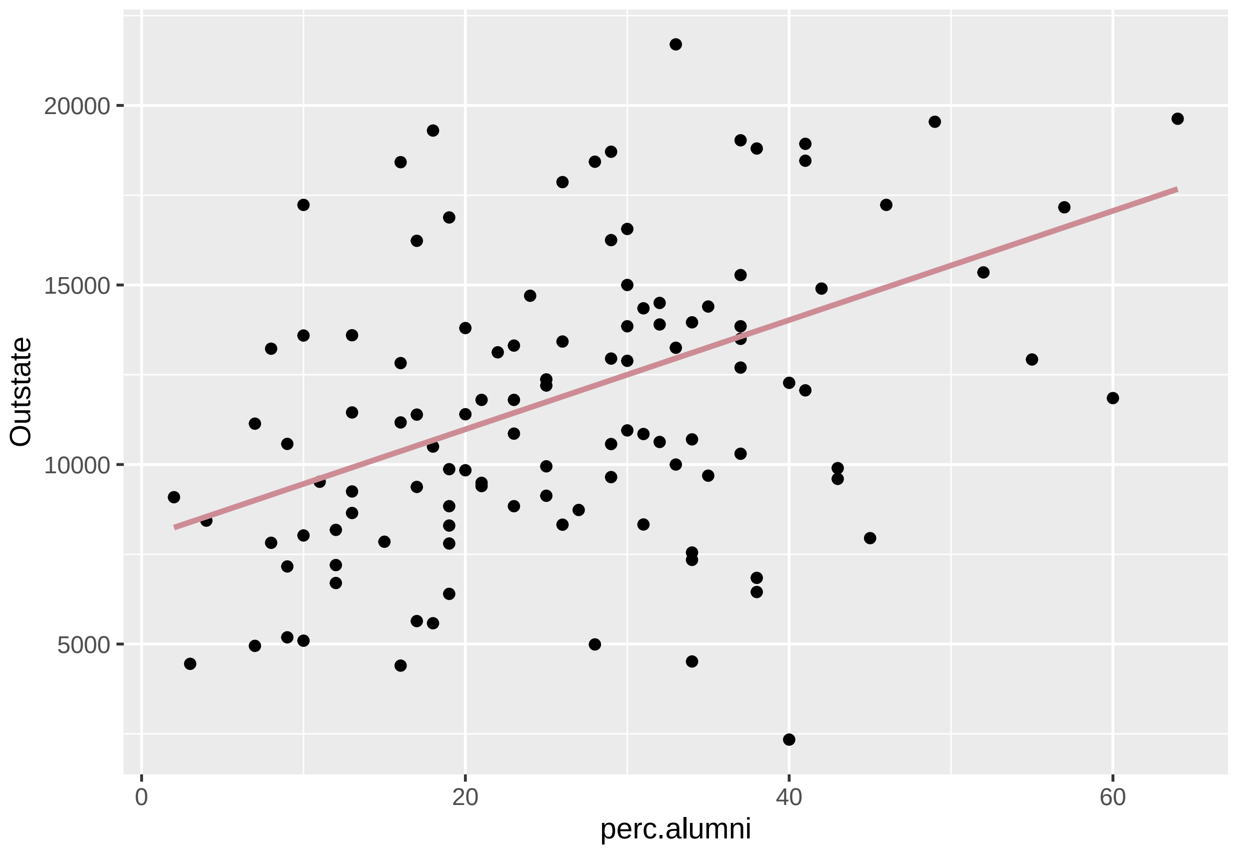
```
pred.ss |>
  as.data.frame() |>
  ggplot(aes(x = x, y = y)) +
  geom_line(linewidth = 1, color = "lightpink3")  +
  geom_point(aes(x = testing_x[, "perc.alumni"], y = testing_y)) +
  labs(y = "Outstate", x = "perc.alumni") +
  theme_bw()
```



```r
ss$df
```

```
## [1] 2.000232
```

```r
ggplot(aes(x = perc.alumni, y = Outstate), data = testing) +
  geom_point() +
  geom_line(aes(x = testing_x[, "perc.alumni"], y = as.data.frame(pred.ss)[, "y"]), linewidth = 1, colo
```



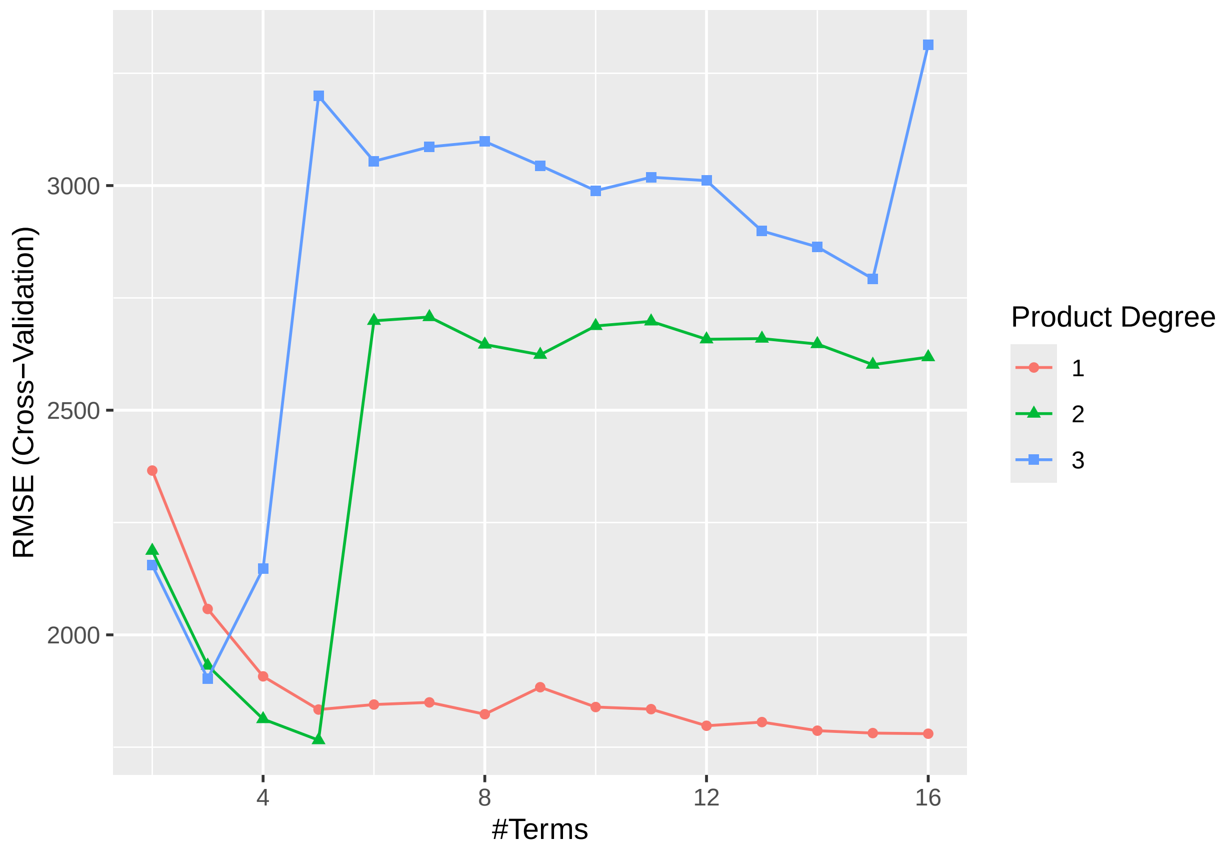
When not passing a prespecified degree of freedom to `smooth.spline`, it automatically uses generalized cross validation to get the optimal degree of freedom, , which is used to get the optimal fit for smoothing spline. This is plotted above with the testing dataset to compare with the prior range of degrees of freedom.

Fitting multivariate adaptive regression spline

To get the optimal number of terms and degree of interactions, ensure that the plotted RMSE has approximately a U shaped curve and the lowest RMSE is plotted. A range of number of terms is from 2:16, the total number of predictors.

```
set.seed(81062)

ctrl1 <- trainControl(method = "cv", number = 10)

mars_grid <- expand.grid(degree = 1:3,
                         nprune = 2:16)

mars <-
  train(
    training_predictors, training_y,
    method = "earth",
    tuneGrid = mars_grid,
    trControl = ctrl1
  )
```

```r
ggplot(mars)
```



```r
mars$bestTune
```

```
##    nprune degree
## 19      5      2
```

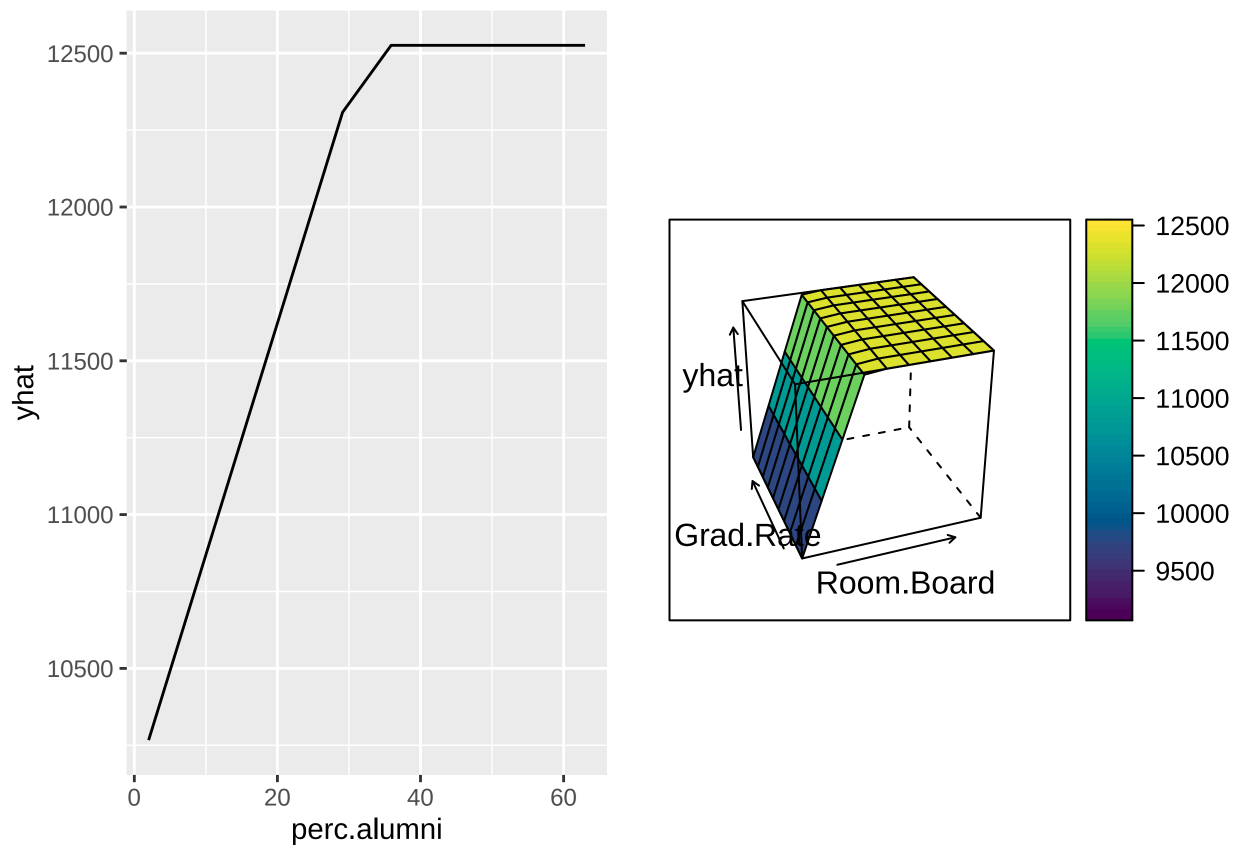
```r
coef(mars$finalModel)
```

```
##        (Intercept)    h(15411-Expend) h(4310-Room.Board)     h(1577-Accept)
##      17786.1851738         -0.6858256         -1.5746231         -1.3476546
##  h(32-perc.alumni)
##        -75.2960753
```

The ideal MARS model has a degree of features of 2 and a number of terms of 5.

Arbitrary predictors in the model of `Room.Board` and `Grad.Rate` were selected to include in the partial dependence plot.

```r
partial1 <-
  pdp::partial(
    mars, pred.var = c("perc.alumni"),
    grid.resolution = 10
  ) |>
  autoplot()

partial2 <-
  pdp::partial(
    mars, pred.var = c("Room.Board", "Grad.Rate"),
    grid.resolution = 10
  ) |>
  pdp::plotPartial(
    levelplot = FALSE,
    zlab = "yhat",
    drape = TRUE,
    screen = list(z = 20, x = -60)
  )

gridExtra::grid.arrange(partial1, partial2, ncol = 2)
```



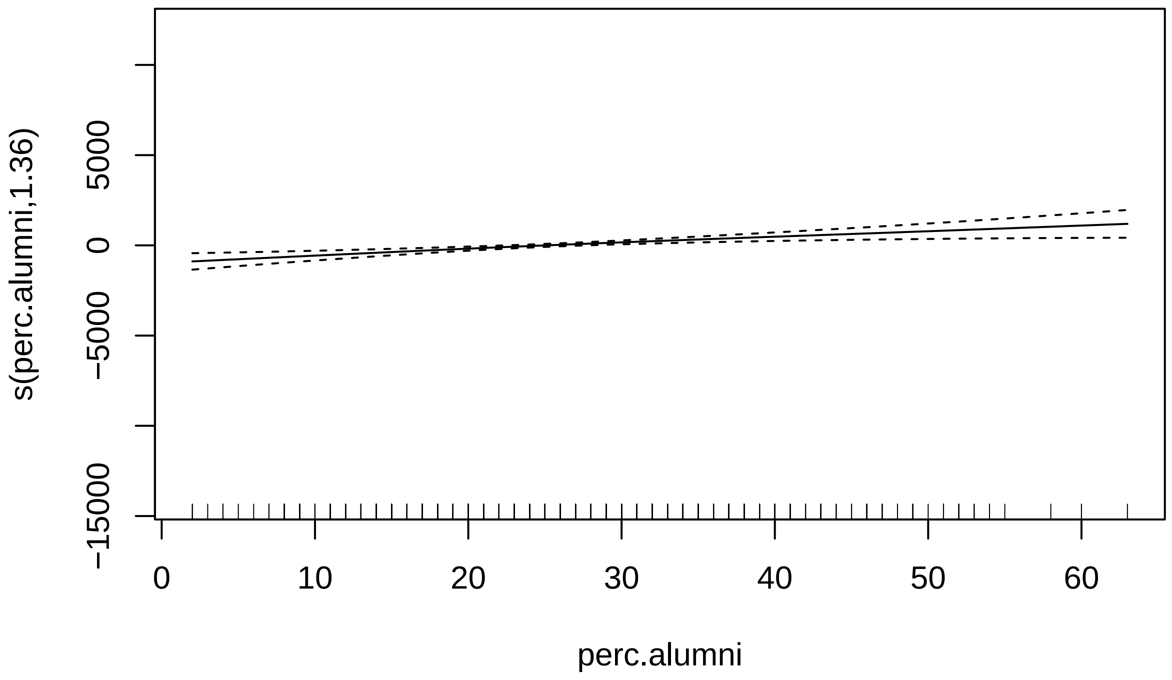
Over the 10-fold cross validation, the mean CV RMSE across each fold is 1765.6194815.

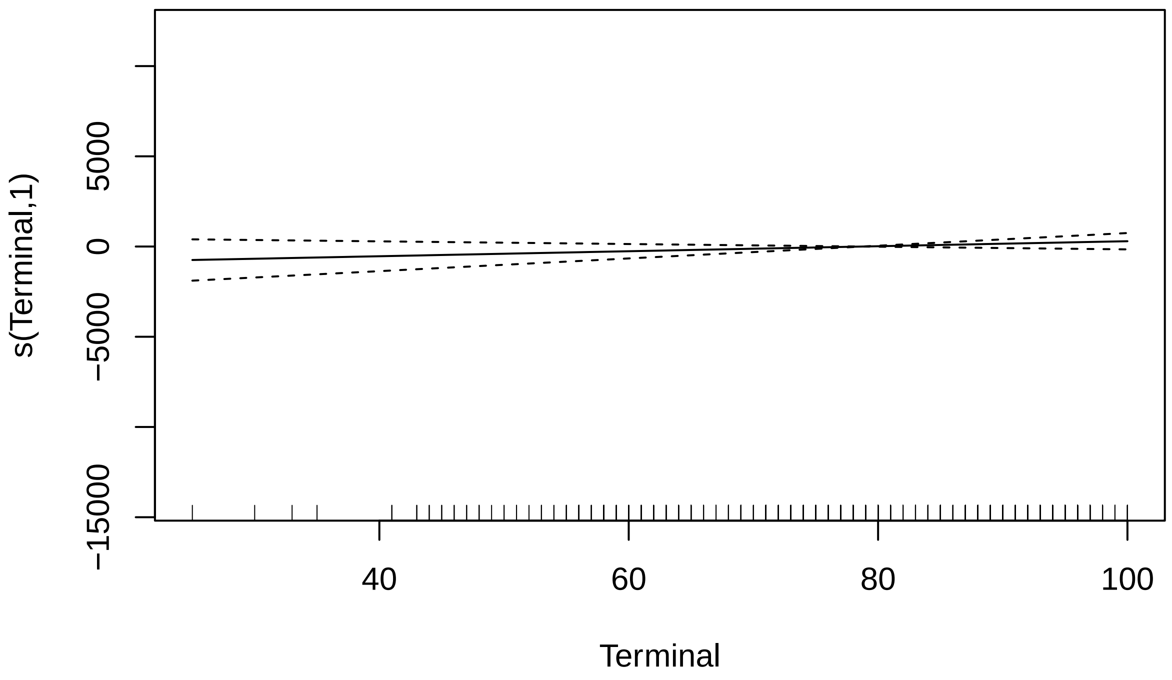
```
mars_pred <- predict(mars, newdata = testing)

mars_pred_error <- mean((mars_pred - testing_y)^2)
```

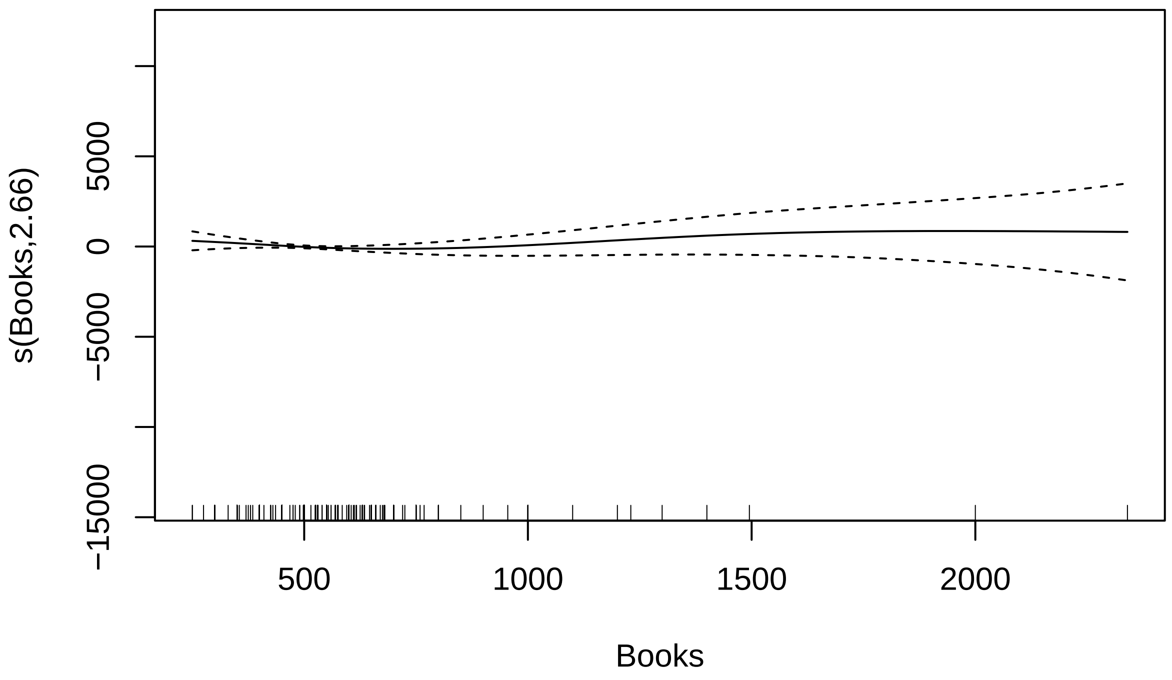
When fit to the testing dataset, the MARS model has a test MSE of 4.8924208×10^6.

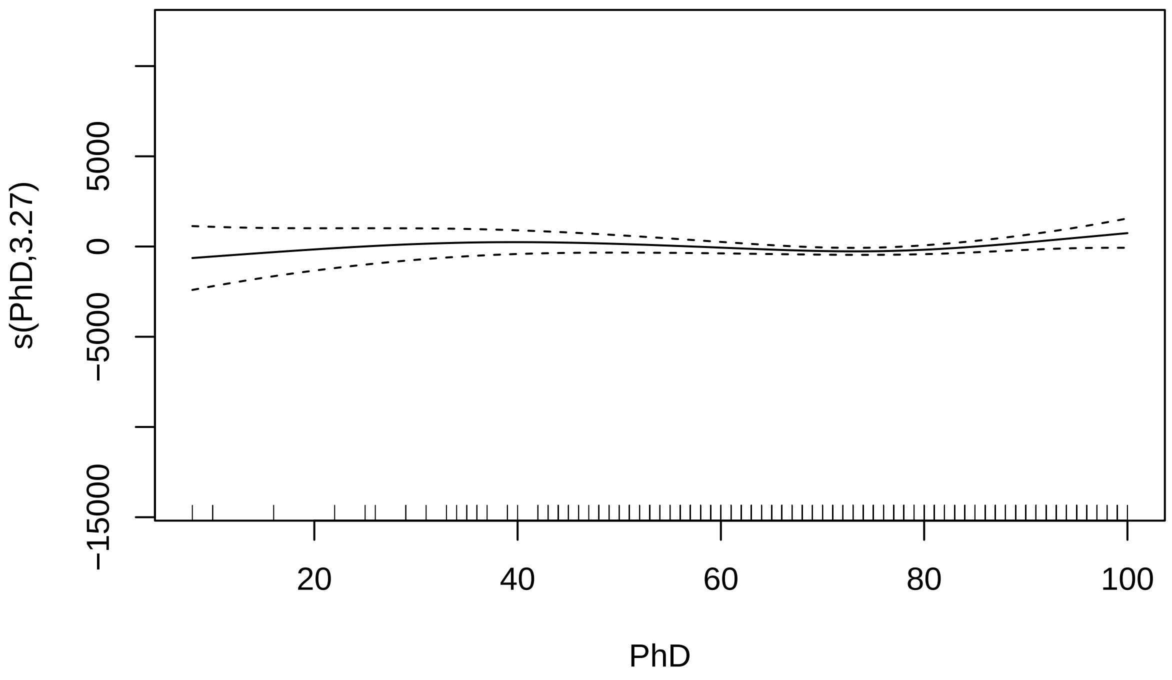
Fitting generalized additive model (GAM)

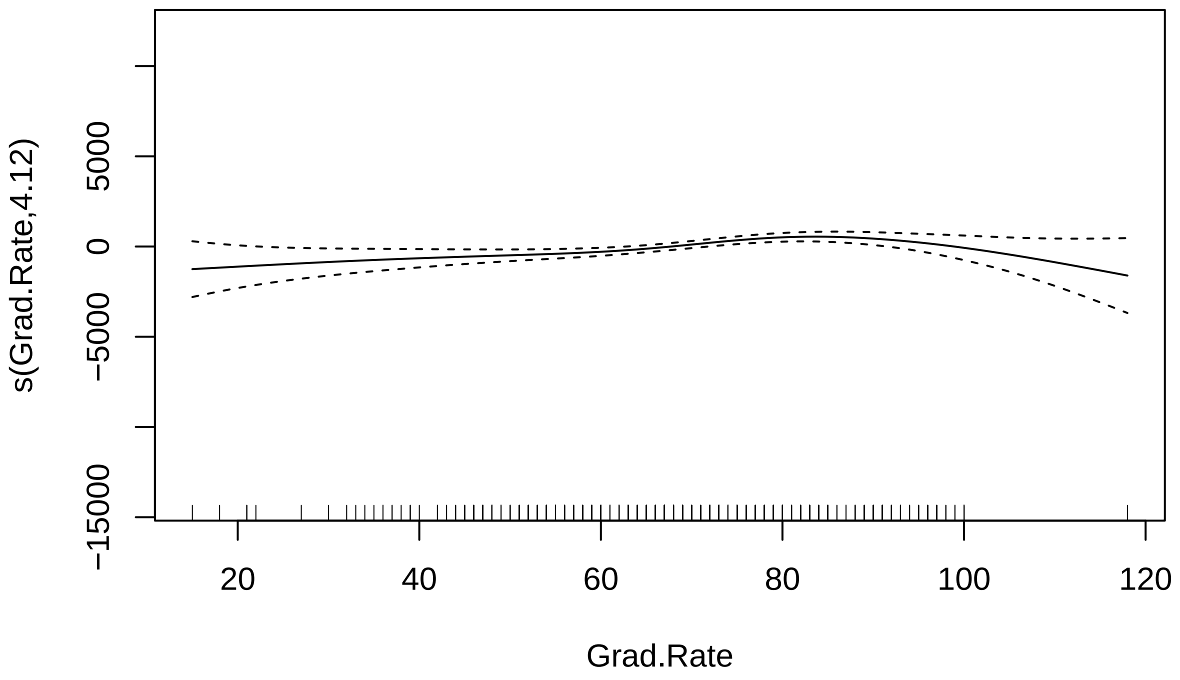
```r
set.seed(81062)

gam <- gam(Outstate ~ 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),
    data = training)

plot(gam)
```

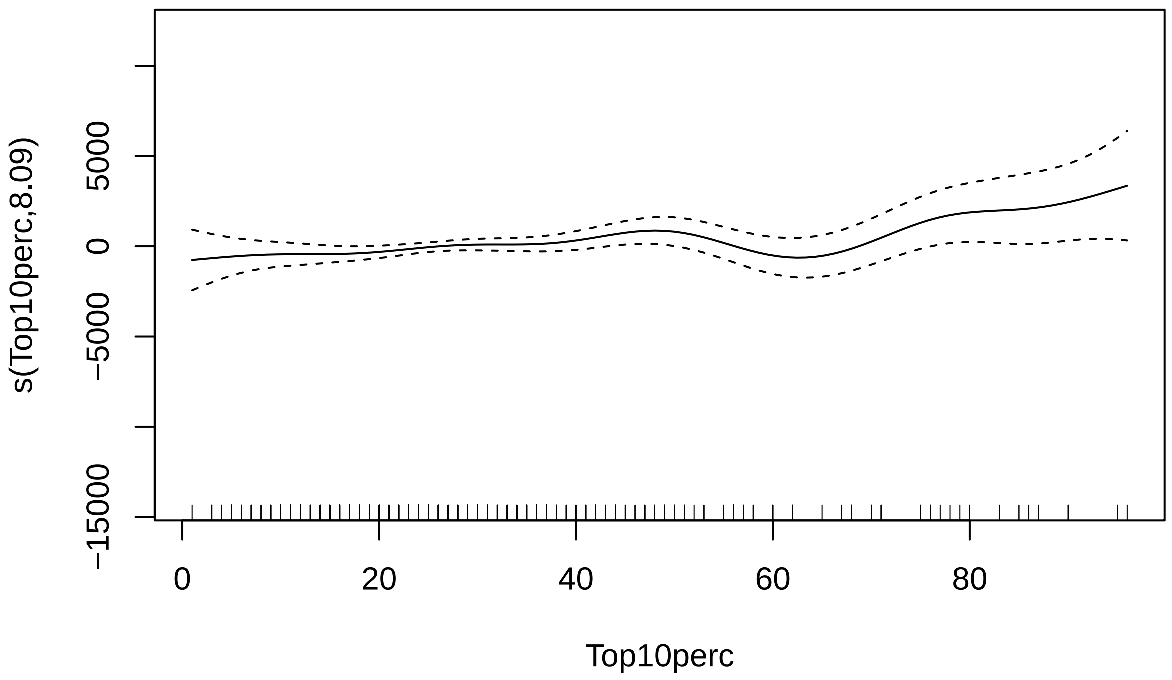


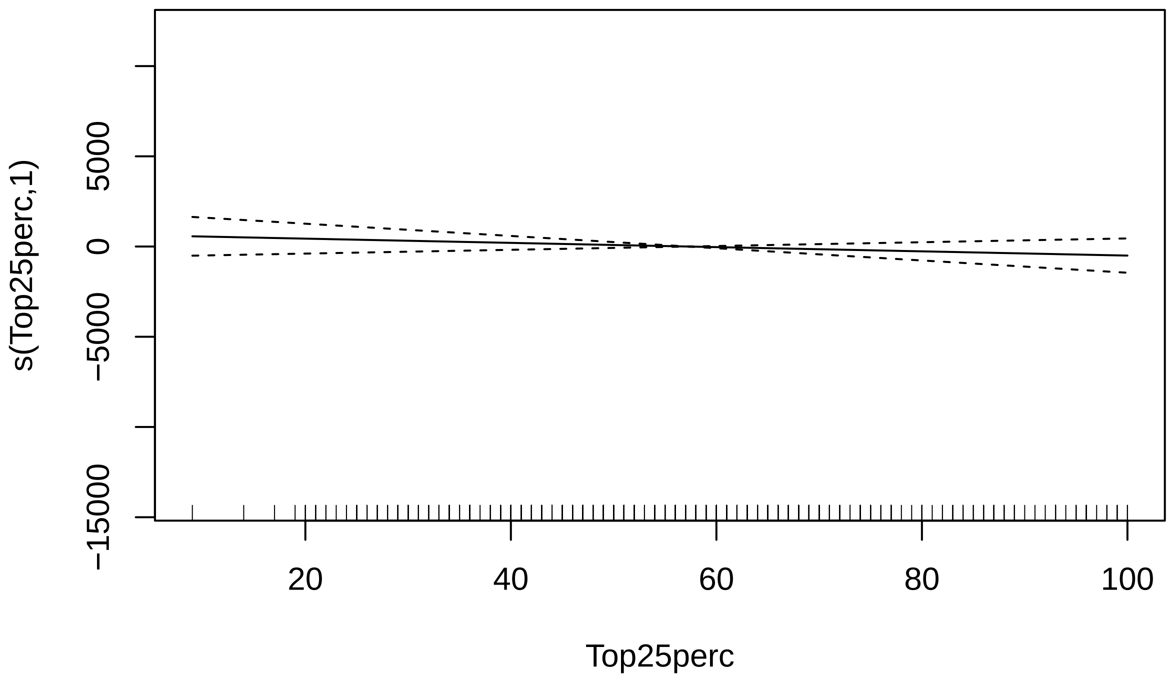

s(Terminal,1)
5000
0
−5000
−15000
40
60
80
100
Terminal


s(Books,2.66)
5000
0
−5000
−15000
500
1000
1500
2000
Books

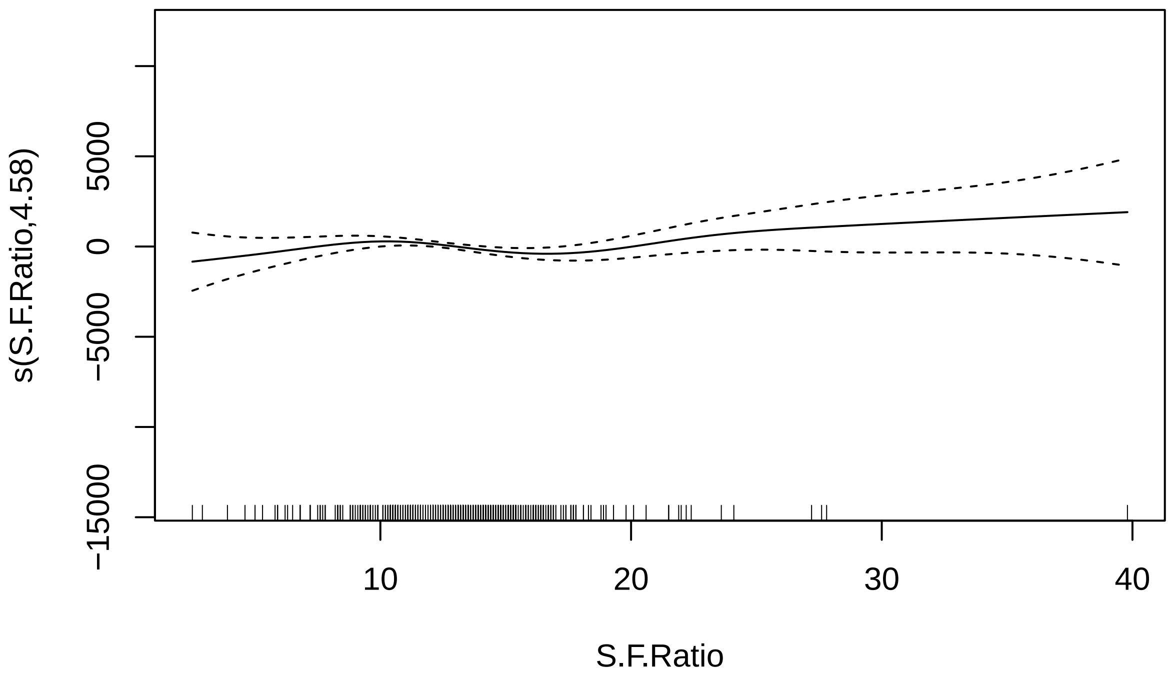

s(PhD,3.27)
5000
0
−5000
−15000
20
40
60
80
100
PhD


s(Grad.Rate,4.12)
5000
0
−5000
−15000
20
40
60
80
100
120
Grad.Rate

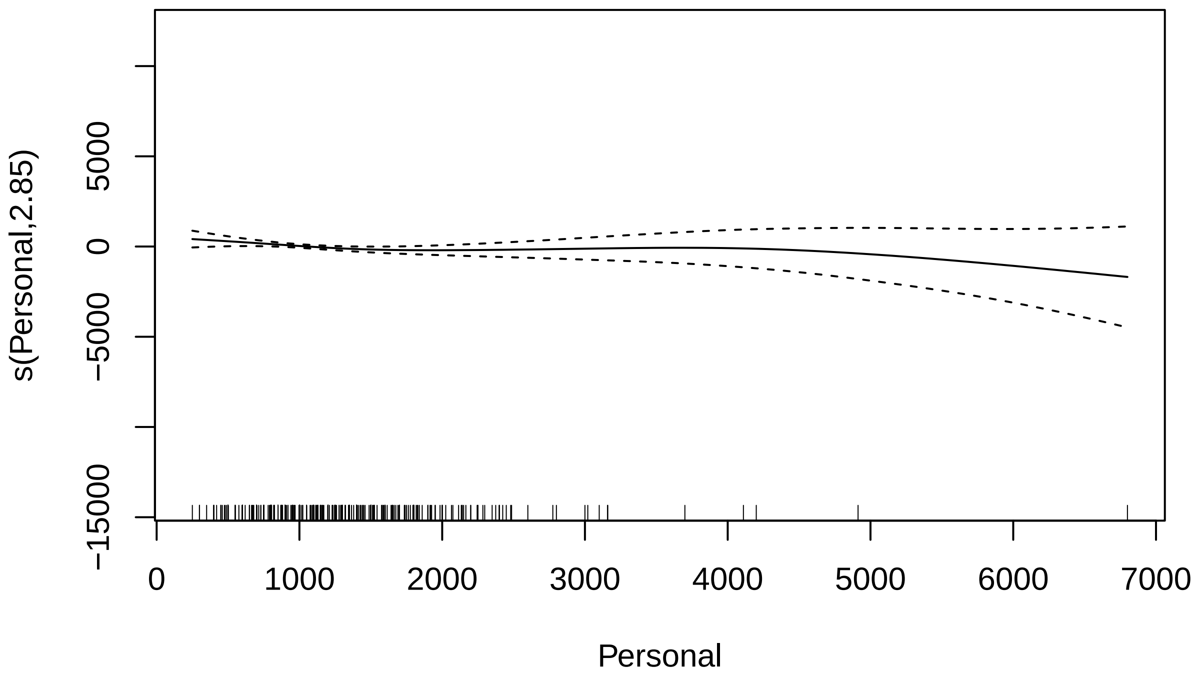

s(Top10perc,8.09)
5000
0
−5000
−15000
0
20
40
60
80
Top10perc



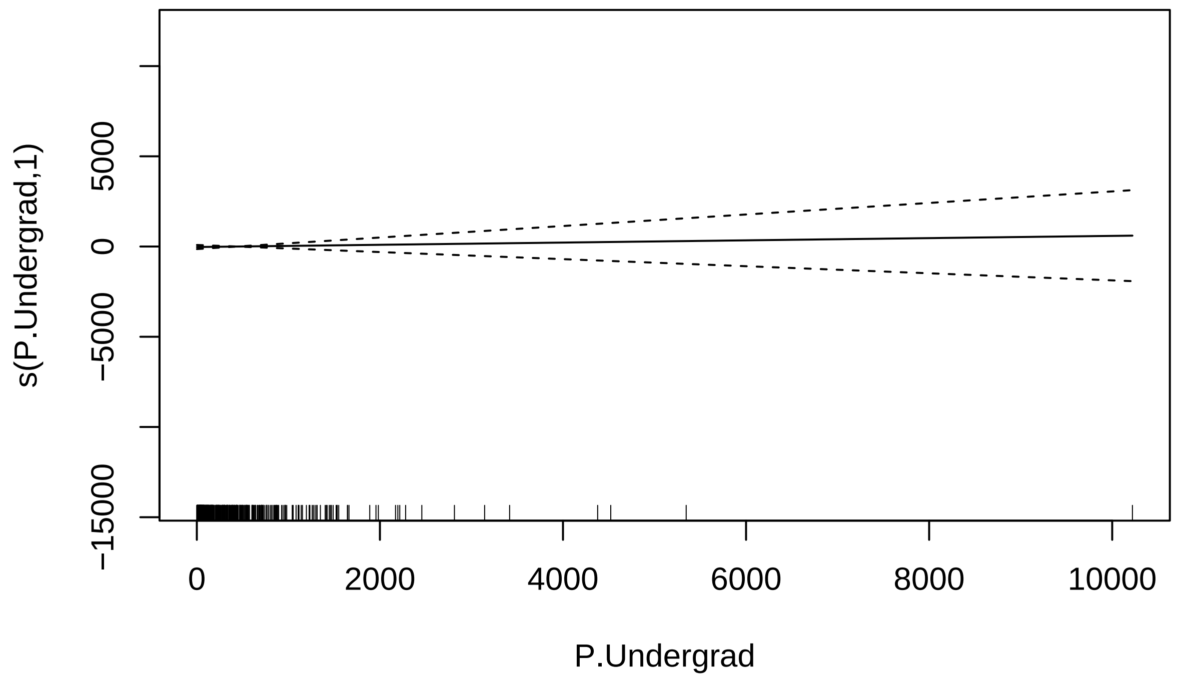
s(Top25perc,1)
5000
0
−5000
−15000
20
40
60
80
100
Top25perc



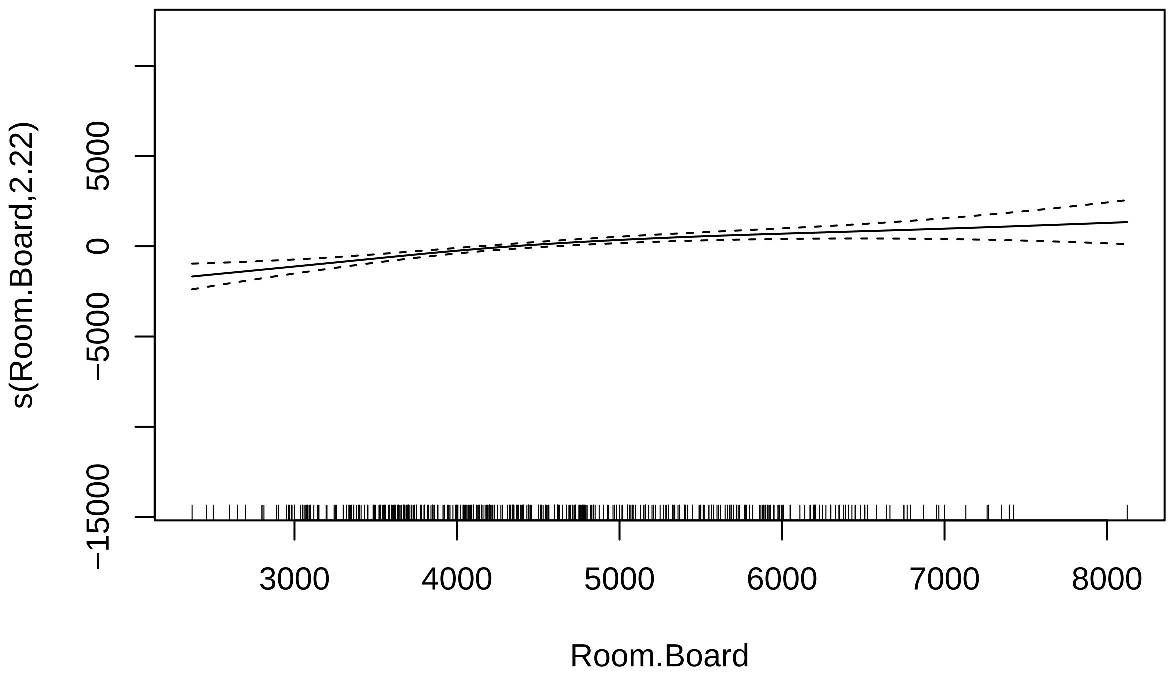
s(S.F.Ratio,4.58)
5000
0
−5000
−15000
10
20
30
40
S.F.Ratio

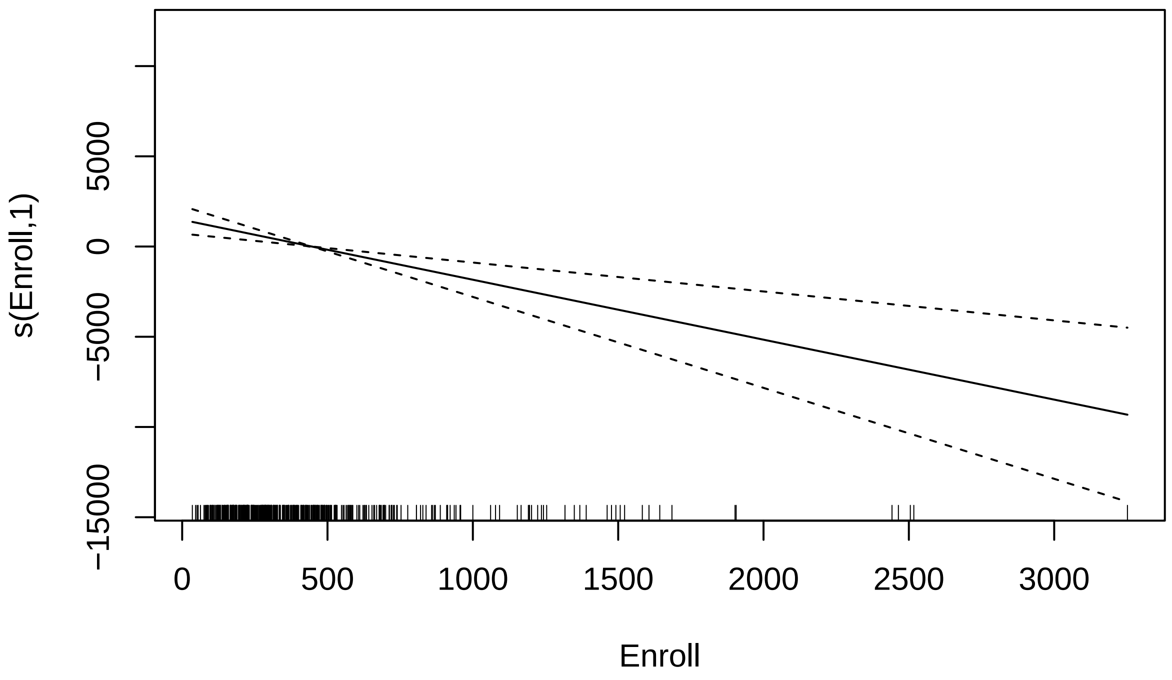


s(Personal,2.85)
5000
0
−5000
−15000
0
1000
2000
3000
4000
5000
6000
7000
Personal

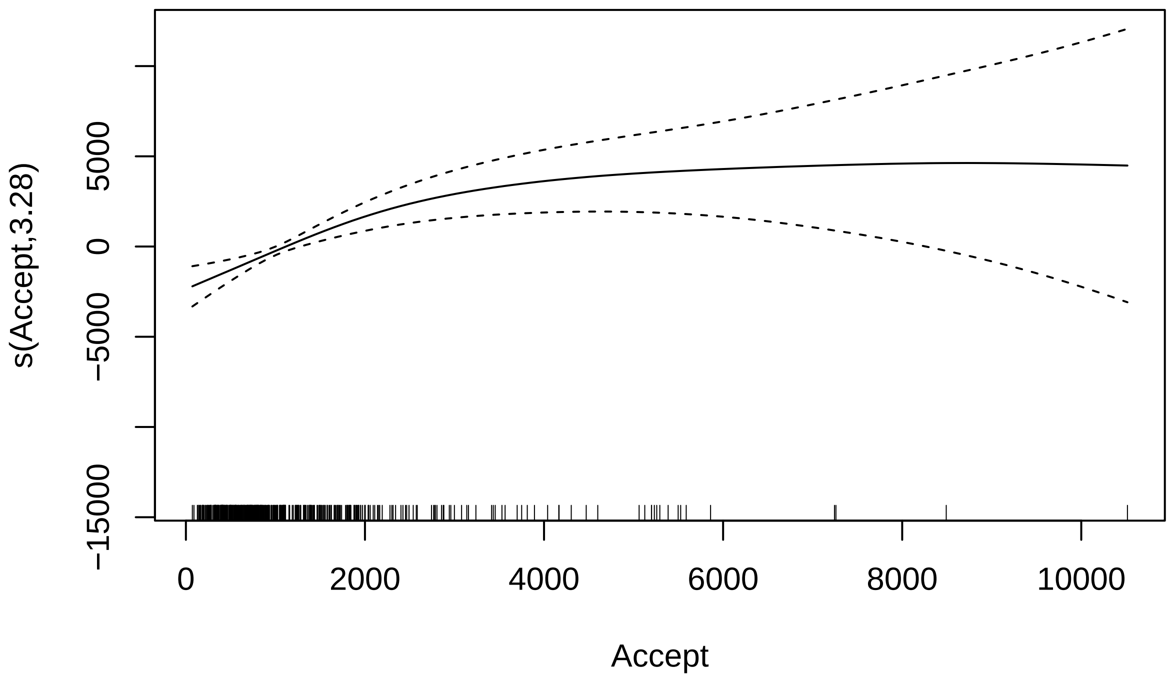


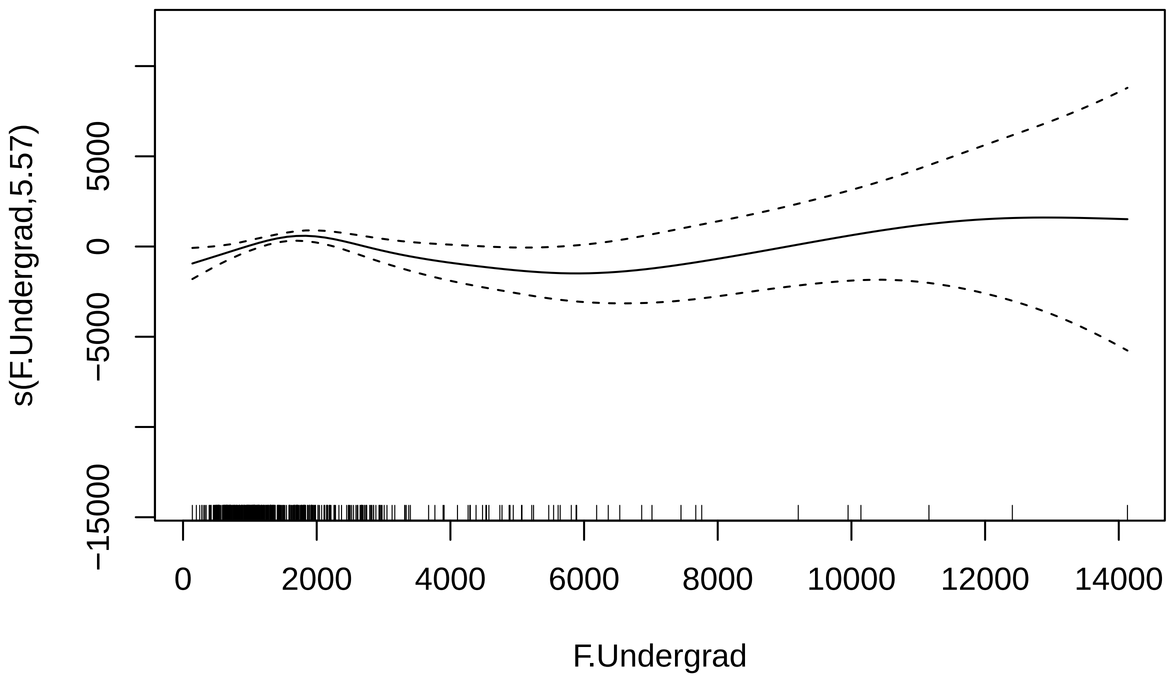
s(P.Undergrad,1)
5000
0
−5000
−15000
0
2000
4000
6000
8000
10000
P.Undergrad

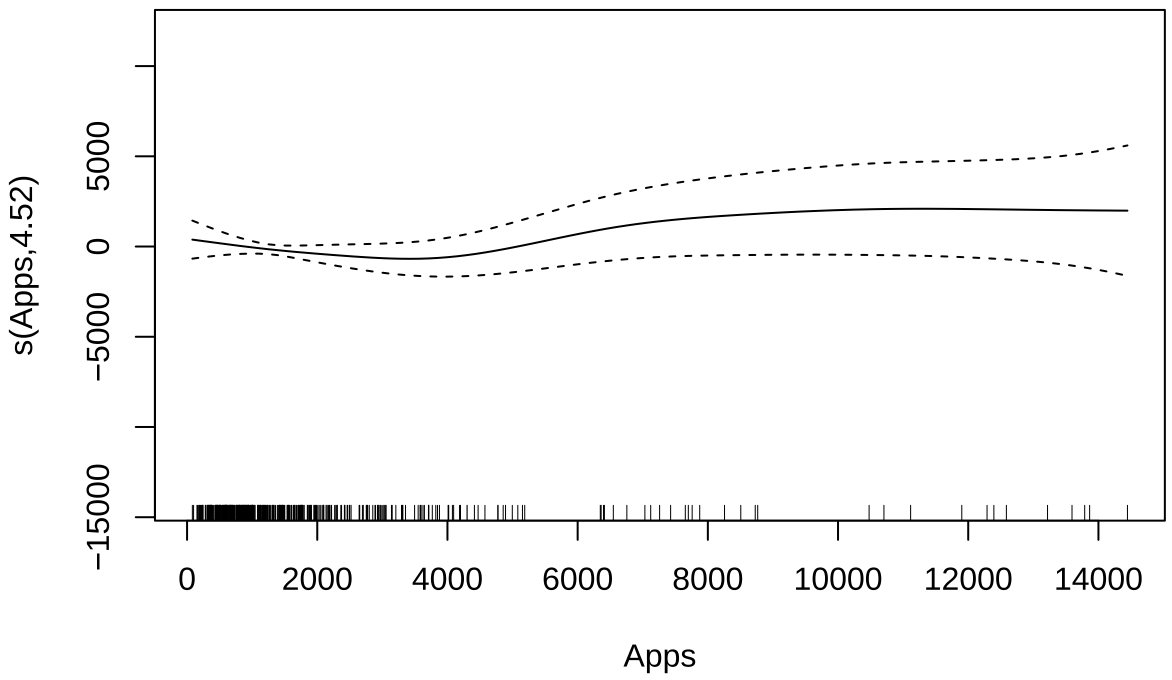

s(Room.Board,2.22)
5000
0
−5000
−15000
3000
4000
5000
6000
7000
8000
Room.Board

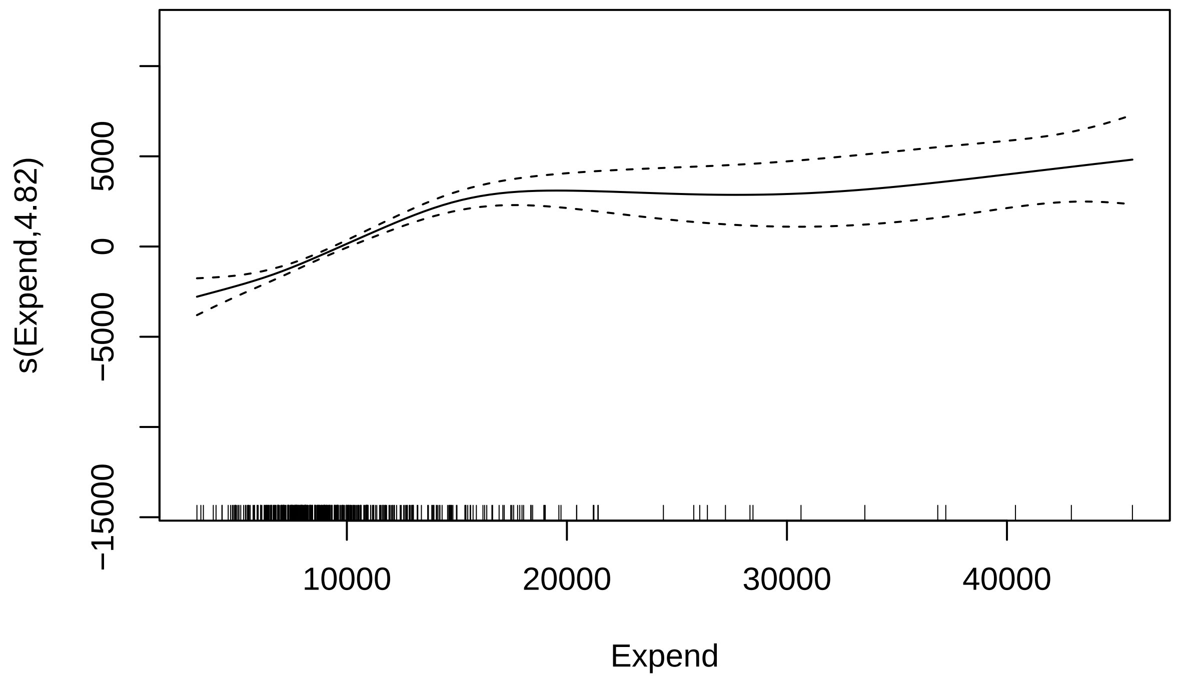


s(Enroll,1)
−15000
−5000
0
5000
0
500
1000
1500
2000
2500
3000
Enroll


s(Accept,3.28)
5000
0
−5000
−15000
0
2000
4000
6000
8000
10000
Accept


s(F.Undergrad,5.57)
5000
0
−5000
−15000
0
2000
4000
6000
8000
10000
12000
14000
F.Undergrad


s(Apps,4.52)
5000
0
−5000
−15000
0
2000
4000
6000
8000
10000
12000
14000
Apps



```r
gam_2 <- train(
    training_predictors, training_y,
    method = "gam",
    trControl = ctrl1
  )

gam_2$bestTune
```

```
##   select method
## 1  FALSE GCV.Cp
```

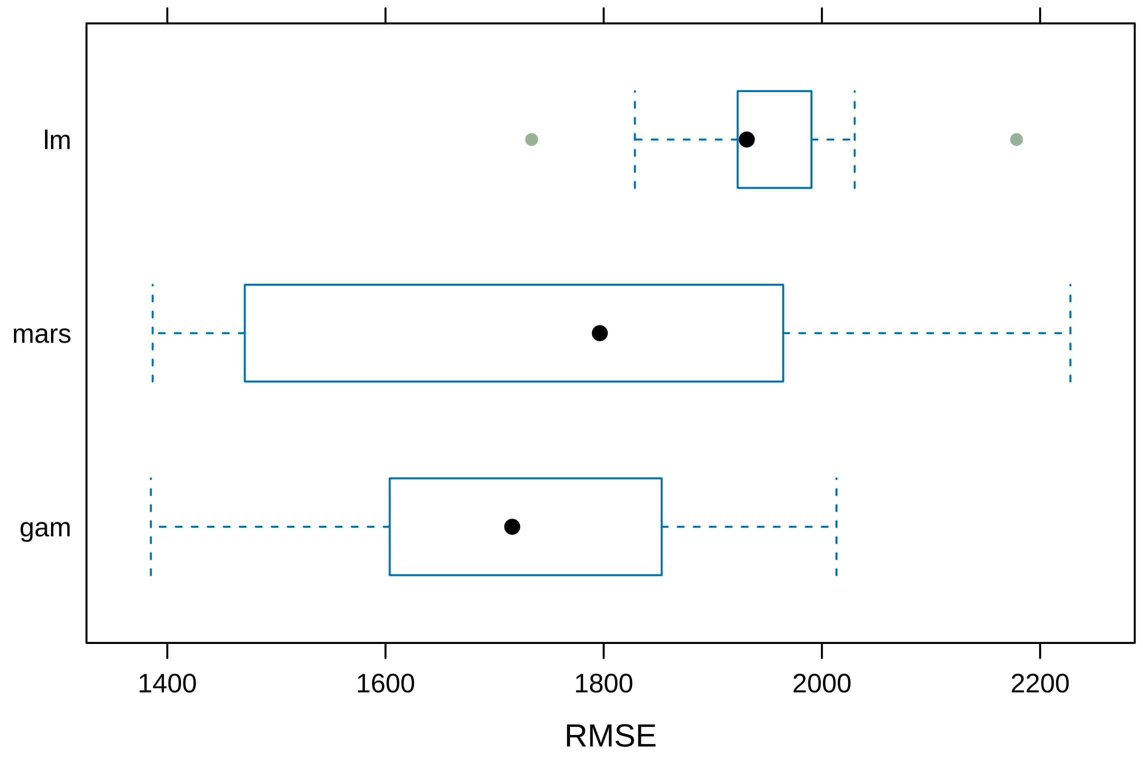
```r
gam_2$finalModel
```

```
##
## 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:
## 1.36 1.00 2.66 3.27 4.12 8.09 1.00
## 4.58 2.85 1.00 2.22 1.00 3.28 5.57
## 4.52 4.82  total = 52.34
##
## GCV score: 2545855
```

```r
gam_pred <- predict(gam, newdata = testing)

gam_pred_error <- mean((gam_pred - testing_y)^2)
```

When fit to the testing dataset, the GAM has a test error MSE of 4.2132774×10^6.

```r
lm <-
  train(
    Outstate ~ perc.alumni + Terminal + Books + PhD +
    Grad.Rate + Top10perc + Top25perc + S.F.Ratio +
    Personal + P.Undergrad + Room.Board + Enroll +
    Accept + F.Undergrad + Apps + Expend,
    data = training,
    method = "lm",
    trControl = ctrl1
  )

resamp <-
  resamples(
    list(
      gam = gam_2,
      mars = mars,
      lm = lm
    )
  )

mars_rmse <-mean(resamp$values$`mars~RMSE`)
gam_rmse <- mean(resamp$values$`gam~RMSE`)
lm_rmse <- mean(resamp$values$`lm~RMSE`)

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



```r
lm_pred <- predict(lm, newdata = testing)

lm_pred_error <- mean((lm_pred - testing_y)^2)
```

For this particular dataset, I would favor MARS over a linear model to predict out of state tuition. When comparing the CV mean RMSEs, the MARS model has a mean RMSE of 1765.6194815, while the linear model has a mean RMSE of 1942.6879676. While only slightly higher, this is likely from the non-linear relations out of state tuition has with some predictors like `perc.alumni`, `P.Undergrad`, and `F.Undergrad`. Additionally, MARS is better able to account for interaction.

More broadly, it cannot be generalized whether MARS or linear models are preferred. It depends on a given dataset. If the dataset can be simply predicted with a linear relationship, without much interaction, then a linear model may be preferable, especially since it is easily interpretable and provides 95% CI. However, if there is considerable interaction and non-linear relations, then MARS may be preferred.