Homework 2

Yuki Joyama

2024-03-11

```
# load libraries
library(tidyverse)
library(rsample) # split data
library(caret)
library(splines)
library(mgcv)
library(earth)
library(ggplot2)
library(vip)
```

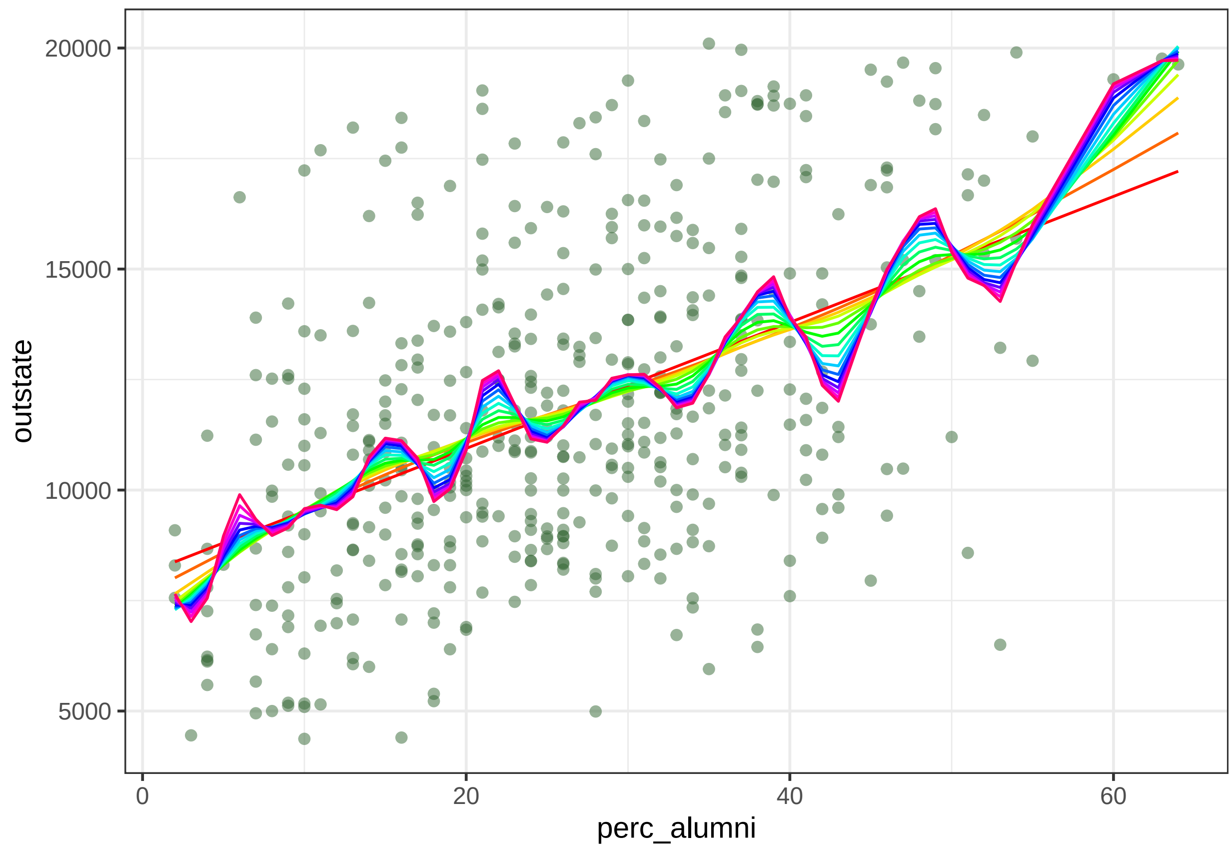
```
# read csv files
df = read_csv("./College.csv") |>
  janitor::clean_names() |>
  dplyr::select(-college) |>
  dplyr::select(outstate, everything())

# partition (training:test=80:20)
set.seed(100)
data_split = initial_split(df, prop = .80)
train = training(data_split)
test = testing(data_split)
```

The college data is split into train (80%) and test (20%).

(a) Smoothing Spline

```
# Function to fit smoothing spline model and return predicted values
fit_spline_model <- function(df, df_value) {
  fit.ss <- smooth.spline(df$perc_alumni, y = df$outstate, df = df_value)
  pred.ss <- predict(fit.ss, x = df$perc_alumni)
  return(data.frame(pred = pred.ss$y, perc = df$perc_alumni))
}

# Function to plot smoothed lines with different colors
plot_smooth_lines <- function(train, df_values, colors) {
  p <- ggplot(data = train, aes(x = perc_alumni, y = outstate)) +
    geom_point(color = rgb(.2, .4, .2, .5))

  for (i in seq_along(df_values)) {
```

```
    df_value <- df_values[i]
    color <- colors[i]

    pred.ss.df <- fit_spline_model(train, df_value)

    p <- p + geom_line(aes(x = perc, y = pred), data = pred.ss.df, color = color)
  }

  p <- p + theme_bw()
  return(p)
}

# Set range of dfs
df_values <- c(seq(2, 30, by = 2))
colors <- rainbow(length(df_values))

# Plot smoothed lines
plot_smooth_lines(train, df_values, colors)
```

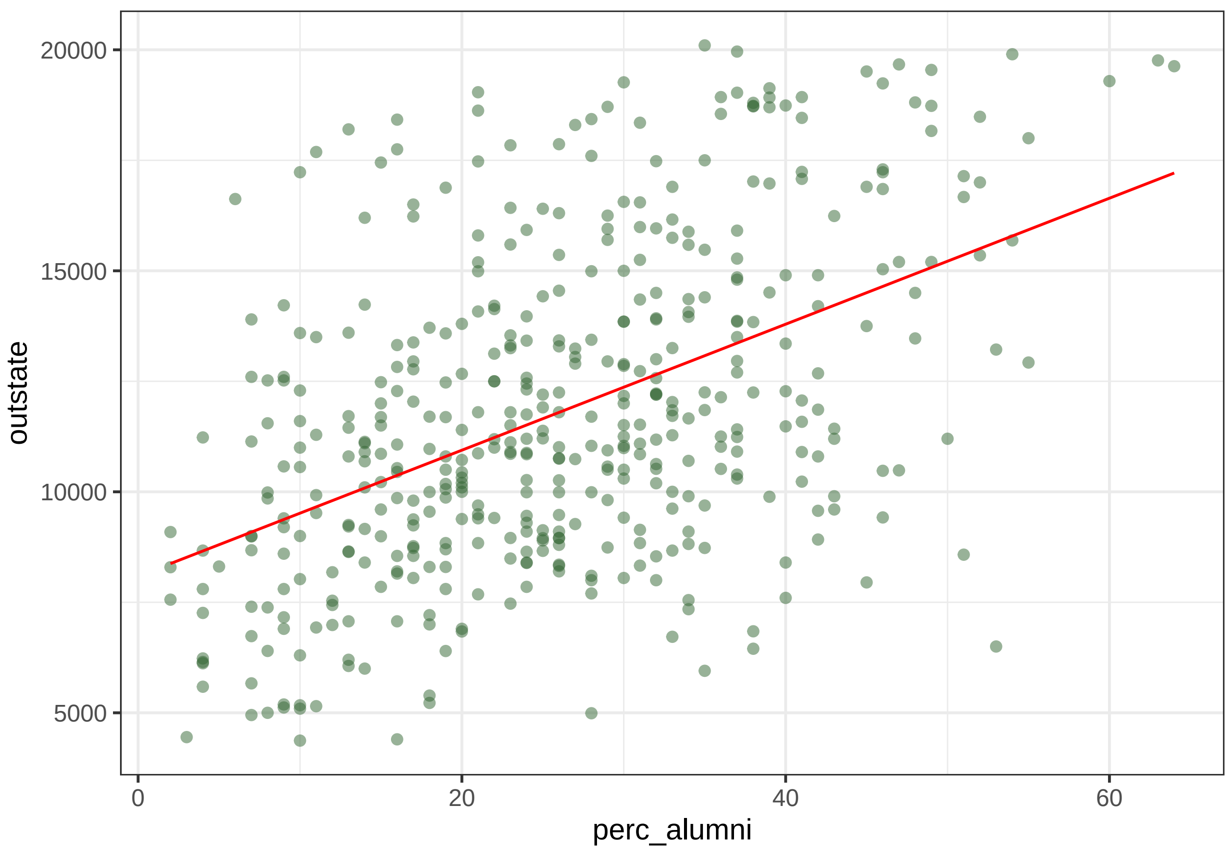


I set the range of degree of freedom (df) from 2 to 30 by 2 (2, 4, 6, ..., 28, 30). The plot shows that as df increases, the fitted lines become more wiggly.

To find the optimal df for the model, I will use Generalized cross-validation.

```
# refit the model using GCV
fit.ss <- smooth.spline(train$perc_alumni, y = train$outstate, cv = FALSE) # determine tuning parameter
```

```
pred.ss <- predict(
  fit.ss,
  x = train$perc_alumni
)

pred.ss.df <- data.frame(
  pred = pred.ss$y,
  perc = train$perc_alumni
)

# plot
p <- ggplot(
  data = train,
  aes(x = perc_alumni, y = outstate)
) +
  geom_point(color = rgb(.2, .4, .2, .5))

p + geom_line(
  aes(x = perc, y = pred),
  data = pred.ss.df,
  color = "red"
) + theme_bw()
```



The selected df was 2 and the plot of this optimal fit is shown above.

(b) Multivariate Adaptive Regression Splines (MARS)

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

```
set.seed(100)

# fit mars model
model.mars <- train(
  x = train[2:17],
  y = train$outstate,
  method = "earth",
  tuneGrid = expand.grid(degree = 1:5, nprune = 2:30),
  metric = "RMSE",
  trControl = ctrl
)

summary(model.mars$finalModel)
```

```
## Call: earth(x=tbl_df[452,16], y=c(11710,11000,1...), keepxy=TRUE, degree=1,
##             nprune=14)
##
##                       coefficients
## (Intercept)             16332.5703
## h(apps-2095)                0.4519
## h(1673-accept)             -1.8329
## h(accept-1673)              0.5027
## h(903-enroll)               3.0624
## h(1251-f_undergrad)        -1.5629
## h(f_undergrad-1251)        -0.7315
## h(4980-room_board)         -0.9381
## h(ph_d-81)                111.2806
## h(8.3-s_f_ratio)         -396.6214
## h(27-perc_alumni)         -56.8440
## h(14820-expend)            -0.5965
## h(98-grad_rate)           -19.3291
## h(grad_rate-98)          -230.4300
##
## Selected 14 of 21 terms, and 10 of 16 predictors (nprune=14)
## Termination condition: RSq changed by less than 0.001 at 21 terms
## Importance: expend, room_board, perc_alumni, accept, ph_d, f_undergrad, ...
## Number of terms at each degree of interaction: 1 13 (additive model)
## GCV 2669172    RSS 1066635330    GRSq 0.7986773    RSq 0.8212206
```

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

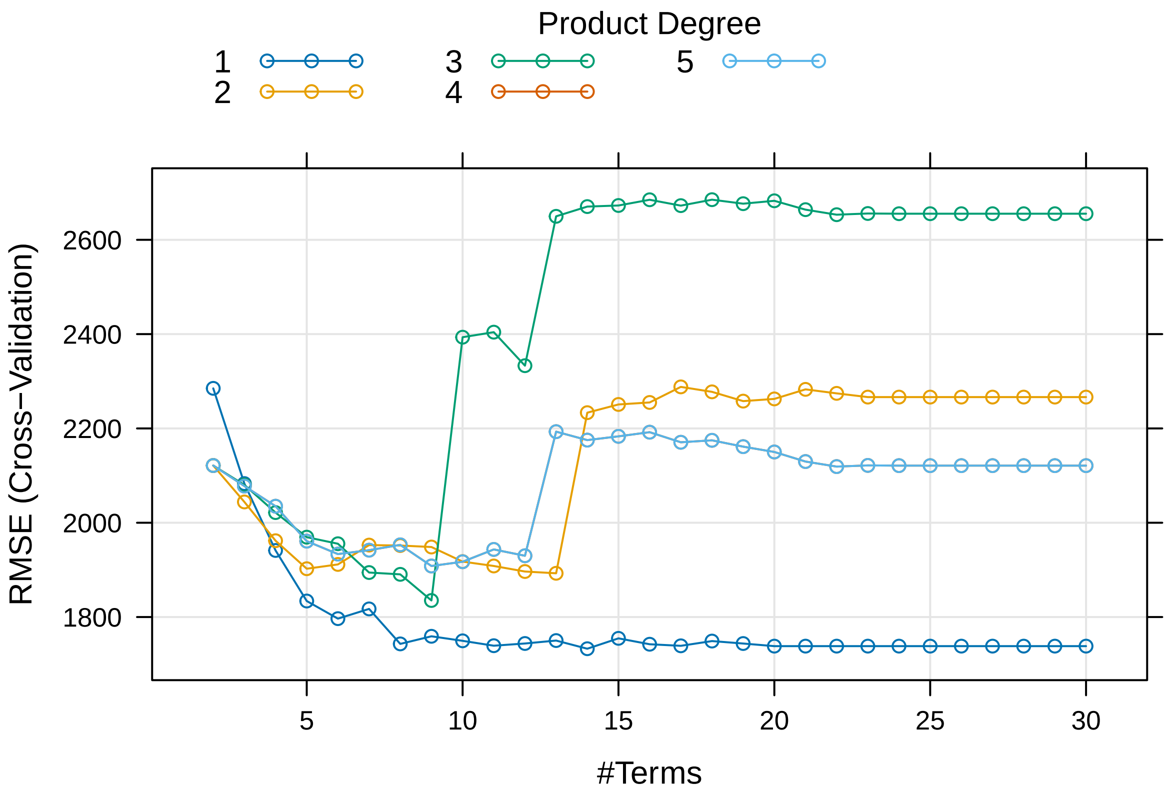
```
##         (Intercept)     h(14820-expend)  h(4980-room_board) h(f_undergrad-1251)
##       16332.5702752          -0.5965124          -0.9381077          -0.7315103
```

```
## h(1251-f_undergrad)   h(27-perc_alumni)        h(apps-2095)          h(ph_d-81)
##          -1.5628566         -56.8439656           0.4518761         111.2806223
##      h(accept-1673)      h(1673-accept)      h(903-enroll)     h(grad_rate-98)
##           0.5027107          -1.8329192           3.0623761        -230.4299780
##     h(98-grad_rate)    h(8.3-s_f_ratio)
##         -19.3291086        -396.6213719
```

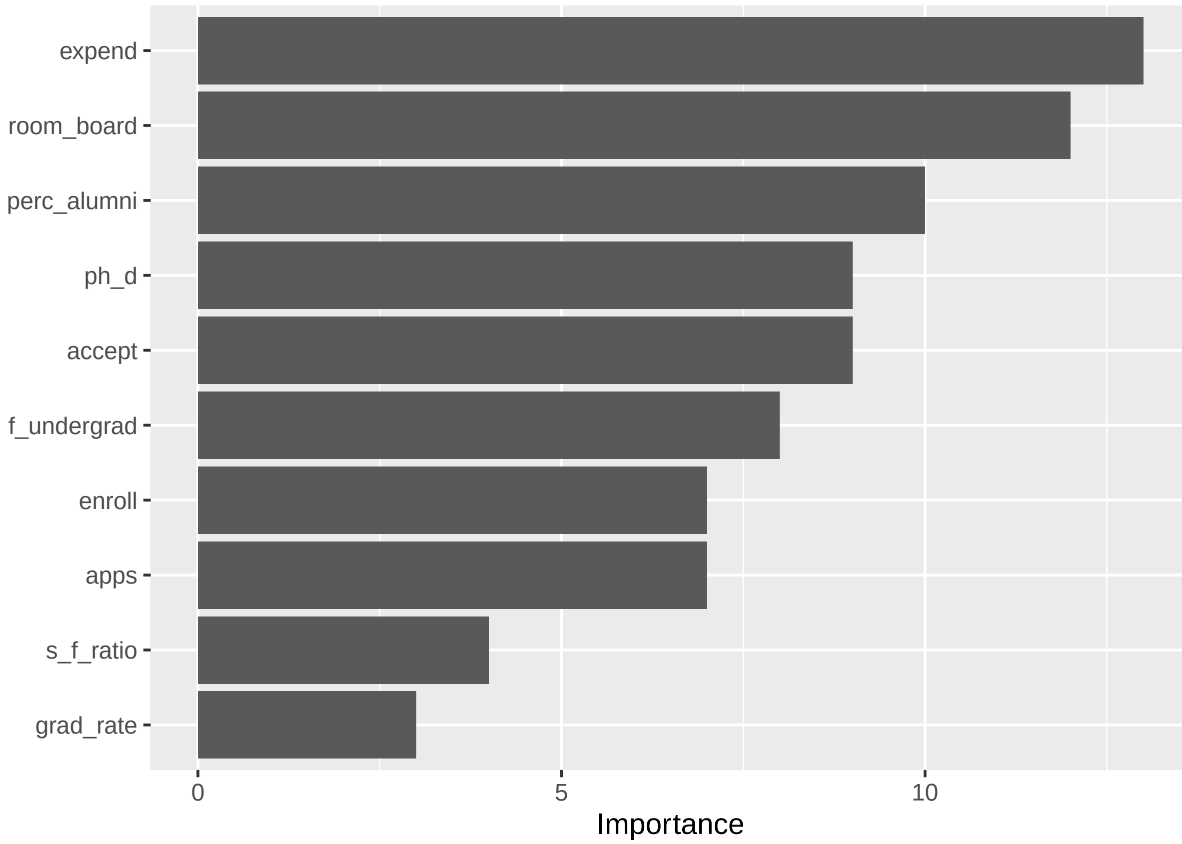
```r
# best tuning parameters
model.mars$bestTune
```

```
##    nprune degree
## 13     14      1
```

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

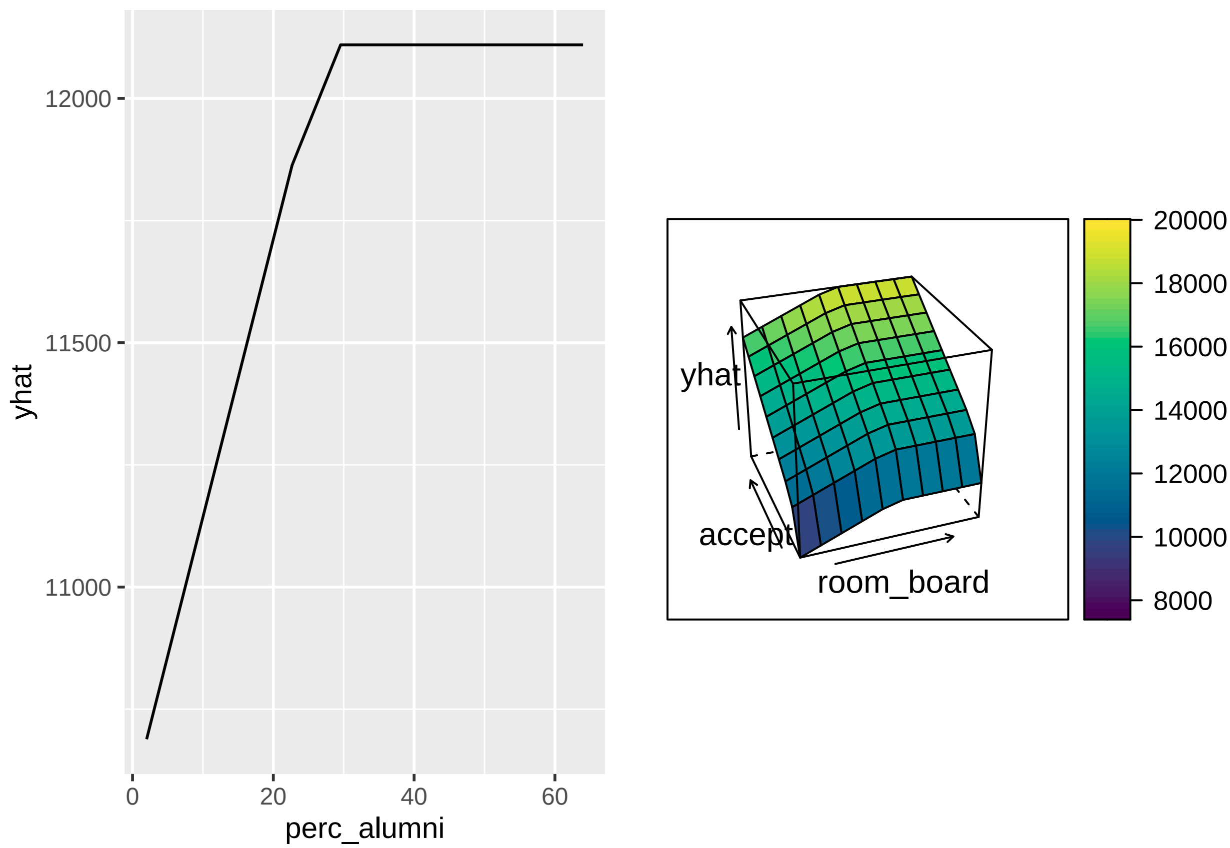



```r
# relative variable importance
vip(model.mars$finalModel, type = "nsubsets")
```



The final model can be expressed as the following:
$\hat{y}$ = 16357.0585 + 1.2722 × h(2095 - apps) + 0.4506 × h(apps - 2095) - 3.3926 × h(1673 - accept) + 0.5027 × h(accept - 1673) + 3.2937 × h(903 - enroll) - 1.6789 × h(1251 - f_undergrad) - 0.7292 × h(f_undergrad - 1251) - 0.9345 × h(4980 - room_board) + 118.0087 × h(ph_d - 81) - 424.6454 × h(8.3 - s_f_ratio) - 53.5431 × h(27 - perc_alumni) - 0.6097 × h(14820 - expend) - 21.6454 × h(98 - grad_rate) - 219.9621 × h(grad_rate - 98)
where $h(.)$ is hinge function.

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

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



The above plot shows the partial dependence plot of `perc_alumni`, `room_board`, and `accept`.

```
# Obtain the test error
mars.pred <- predict(model.mars, newdata = test)
mean((mars.pred - pull(test, "outstate"))^2) # test error
```

```
## [1] 4831325
```

The test error is 4.02×10^6

(c) Generalized Additive Model (GAM)

```
set.seed(100)
model.gam <- train(x = train[2:17],
                   y = train$outstate,
                   method = "gam",
                   metric = "RMSE",
                   trControl = ctrl)

# check the best tune
model.gam$bestTune
```

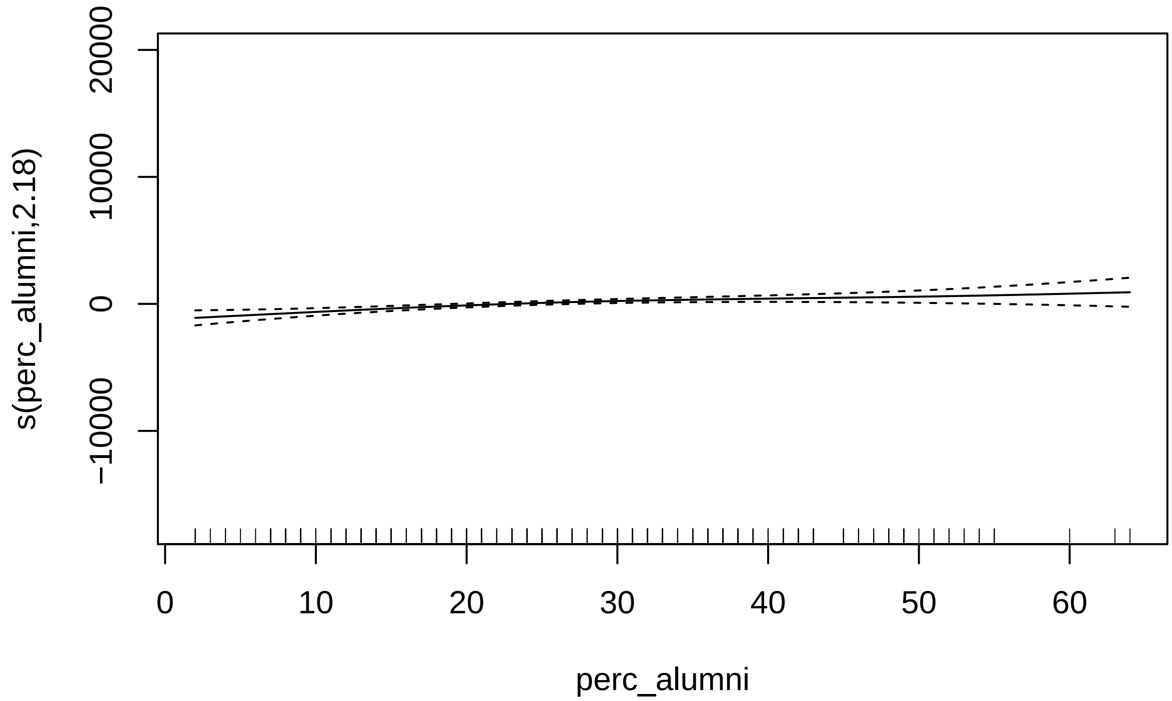
```
##   select method
## 1  FALSE GCV.Cp
```

```r
# check final model
summary(model.gam$finalModel)
```

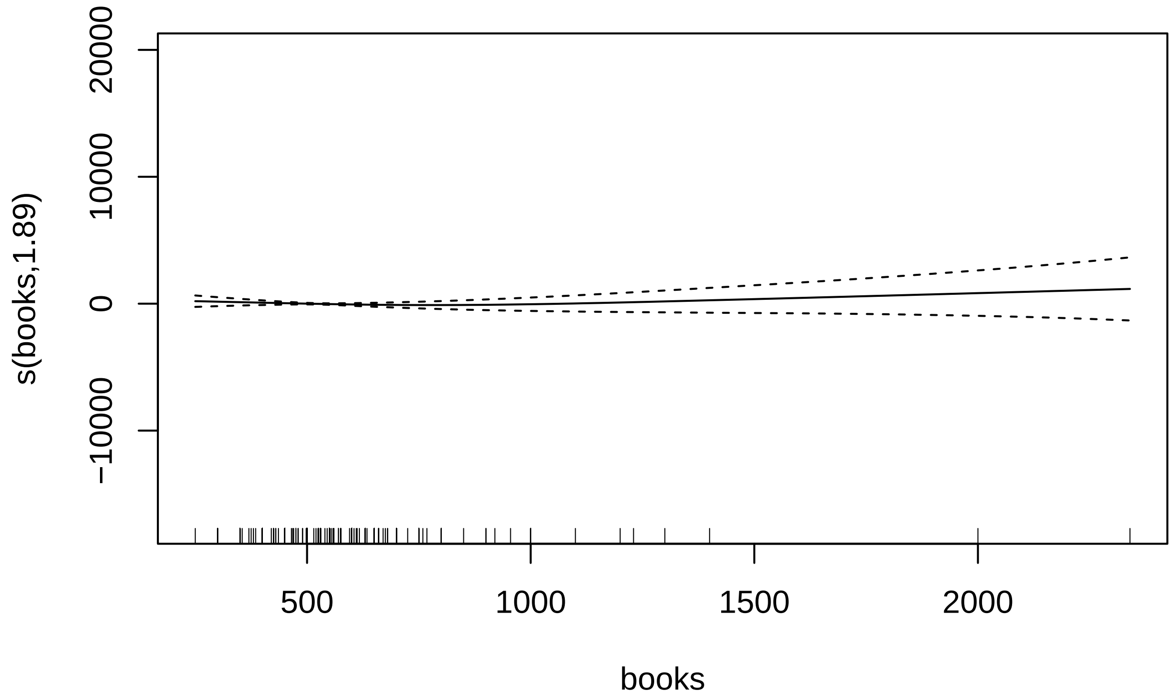
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ s(perc_alumni) + s(terminal) + s(books) + s(top10perc) +
##     s(ph_d) + s(grad_rate) + s(top25perc) + s(s_f_ratio) + s(personal) +
##     s(p_undergrad) + s(room_board) + s(enroll) + s(f_undergrad) +
##     s(accept) + s(apps) + s(expend)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11793.1       72.4   162.9   <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) 2.175  2.767  7.033 0.000353 ***
## s(terminal)    1.000  1.000  0.823 0.364699
## s(books)       1.891  2.352  1.112 0.419994
## s(top10perc)   1.331  1.595  1.441 0.336862
## s(ph_d)        4.512  5.533  2.853 0.010508 *
## s(grad_rate)   3.786  4.746  2.704 0.023566 *
## s(top25perc)   1.000  1.000  0.477 0.490400
## s(s_f_ratio)   4.188  5.207  3.381 0.004749 **
## s(personal)    1.294  1.526  1.138 0.224949
## s(p_undergrad) 1.000  1.000  0.021 0.883627
## s(room_board)  2.076  2.626 15.849  < 2e-16 ***
## s(enroll)      1.000  1.000  8.956 0.002935 **
## s(f_undergrad) 5.941  7.023  3.733 0.000607 ***
## s(accept)      4.257  5.240  3.750 0.002170 **
## s(apps)        5.040  6.085  2.746 0.012259 *
## s(expend)      5.172  6.282 18.821  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.821   Deviance explained = 83.9%
## GCV = 2.6419e+06  Scale est. = 2.3691e+06  n = 452
```

The model includes all the predictors.
Given the estimated degrees of freedom, I will select nonlinear terms and plot each of them.

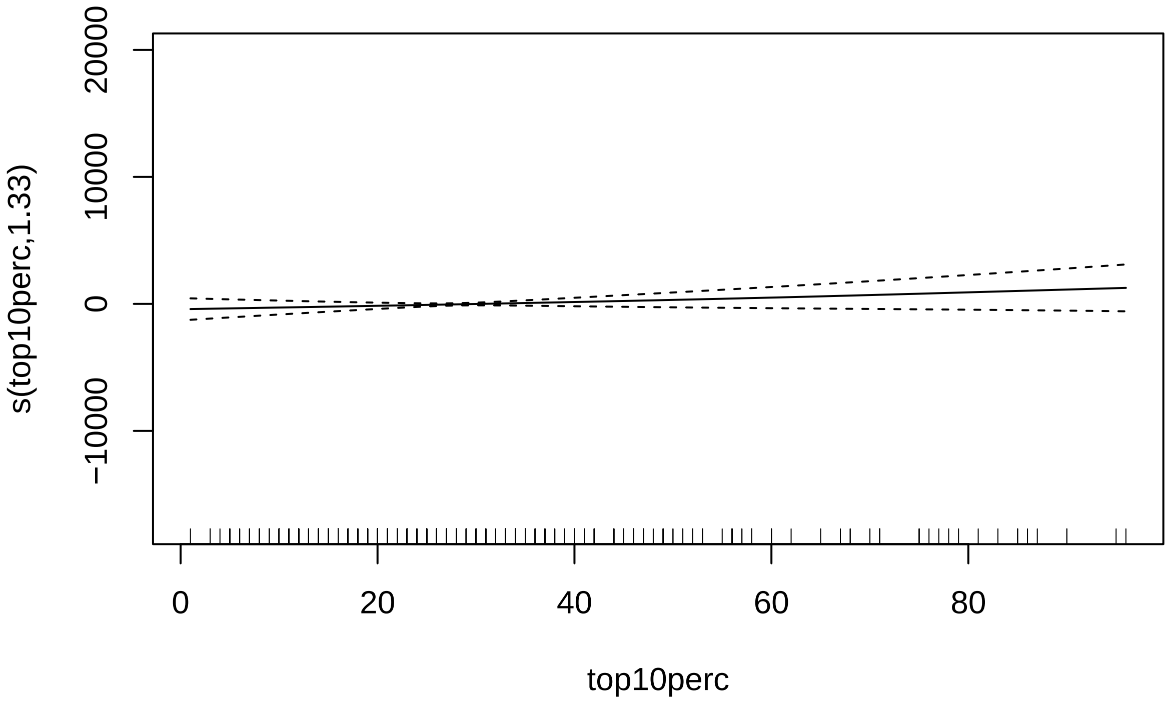
```r
# plot
plot(model.gam$finalModel, select = 1)
```



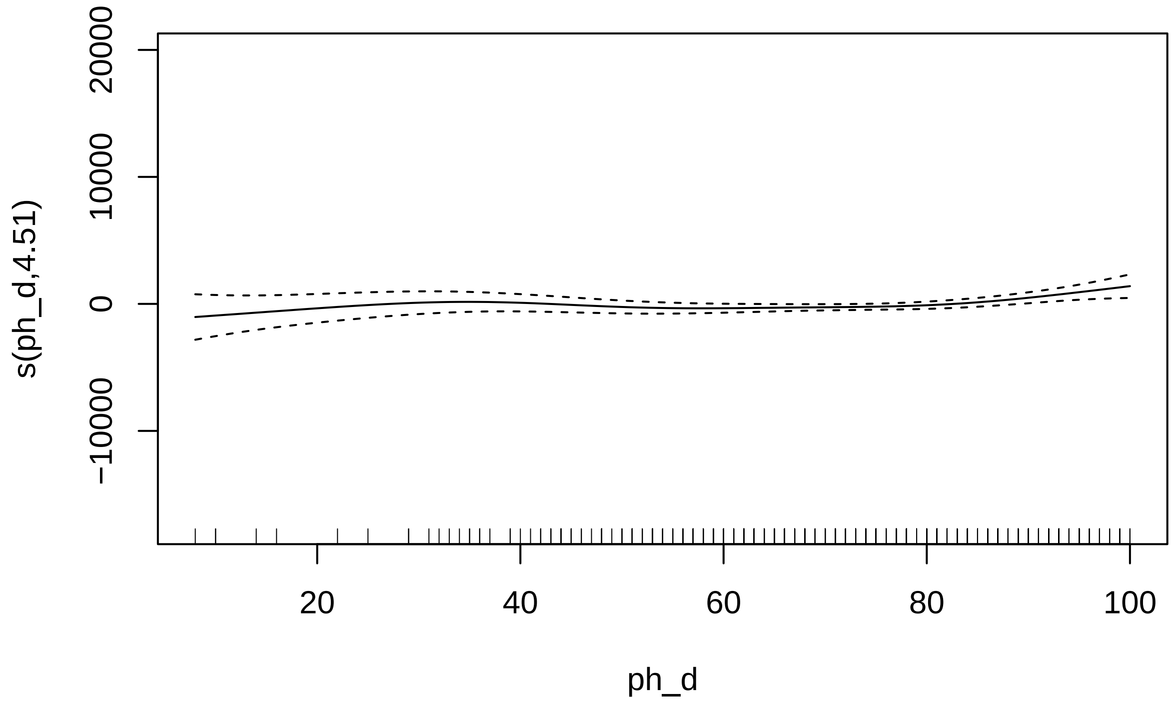

```r
plot(model.gam$finalModel, select = 3)
```



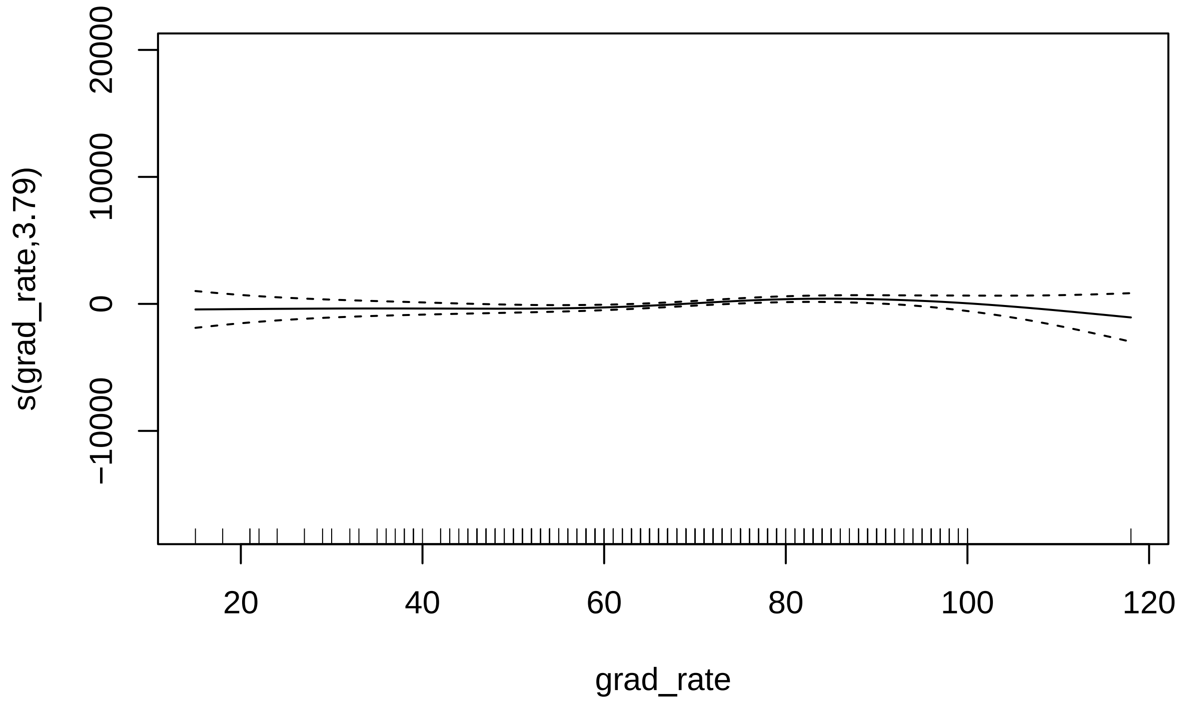

```
plot(model.gam$finalModel, select = 4)
```



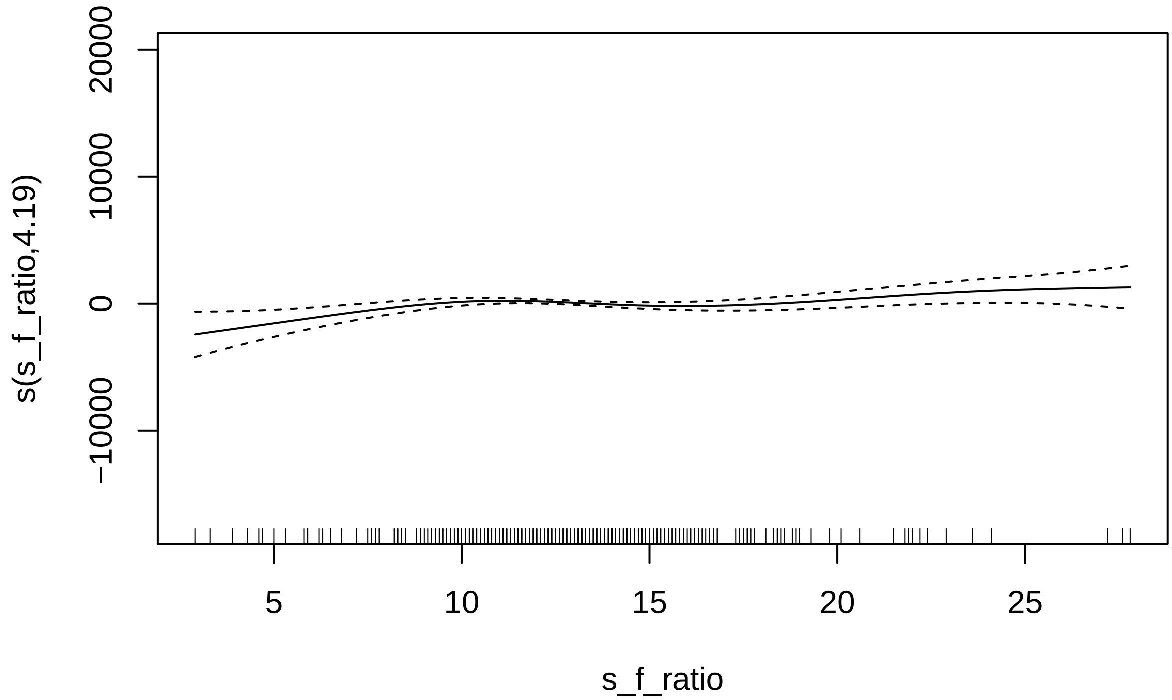

```
plot(model.gam$finalModel, select = 5)
```



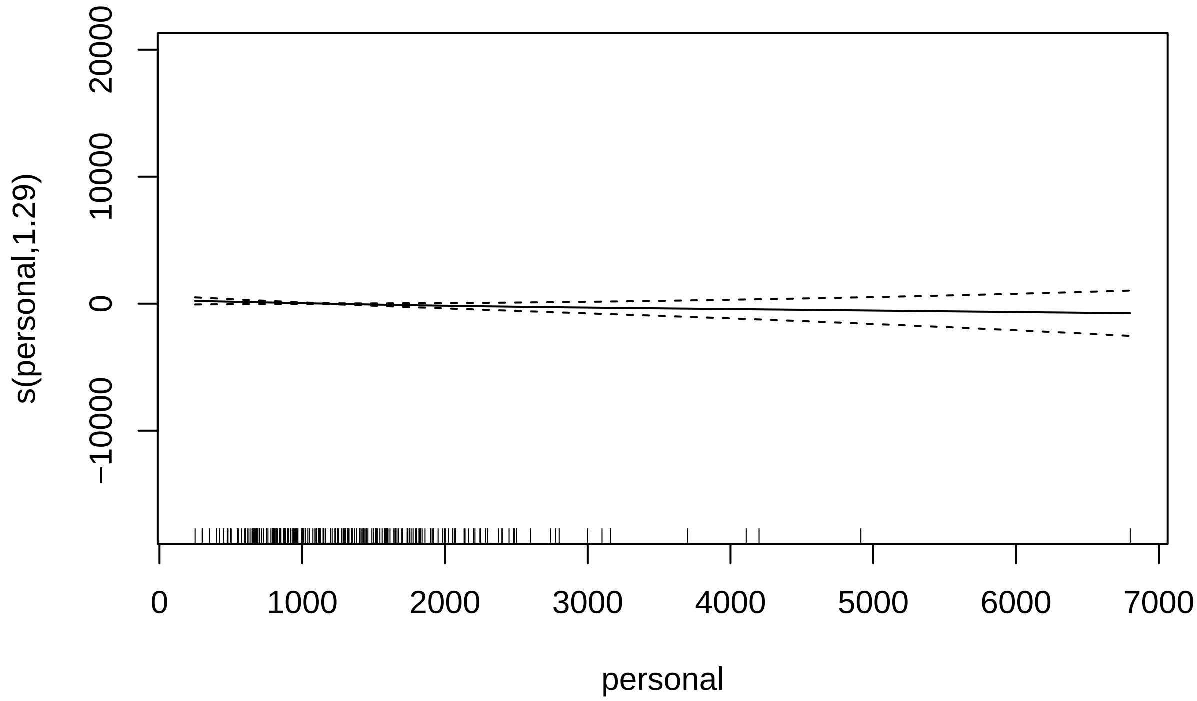

```
plot(model.gam$finalModel, select = 6)
```



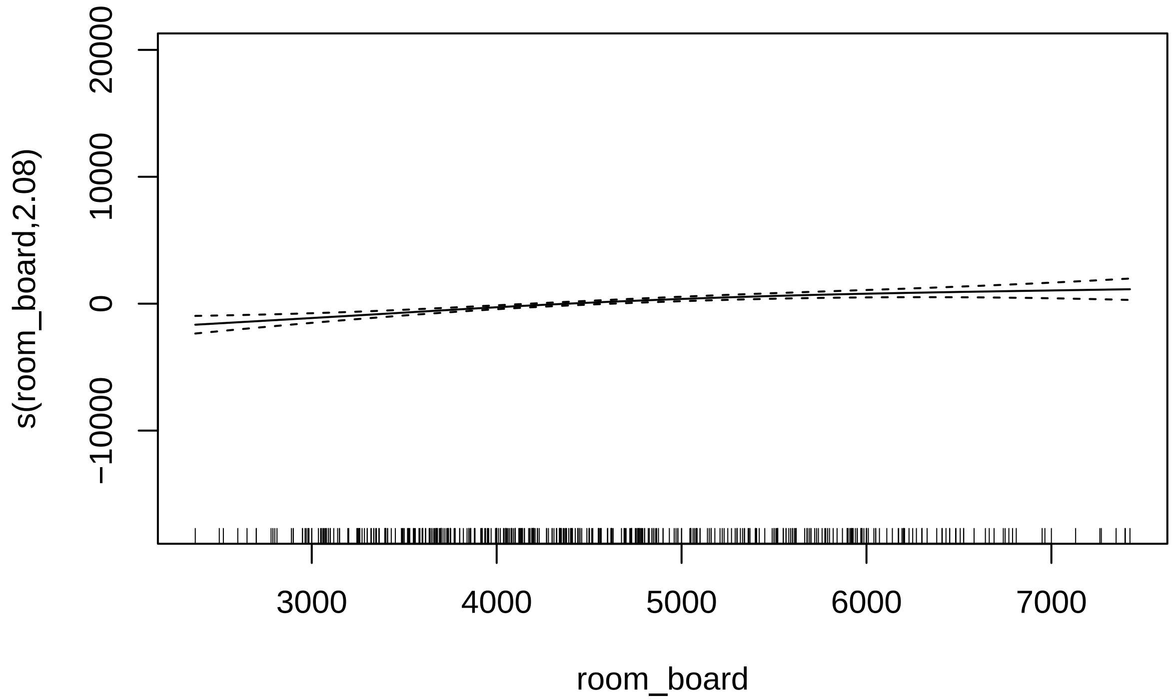

```
plot(model.gam$finalModel, select = 8)
```



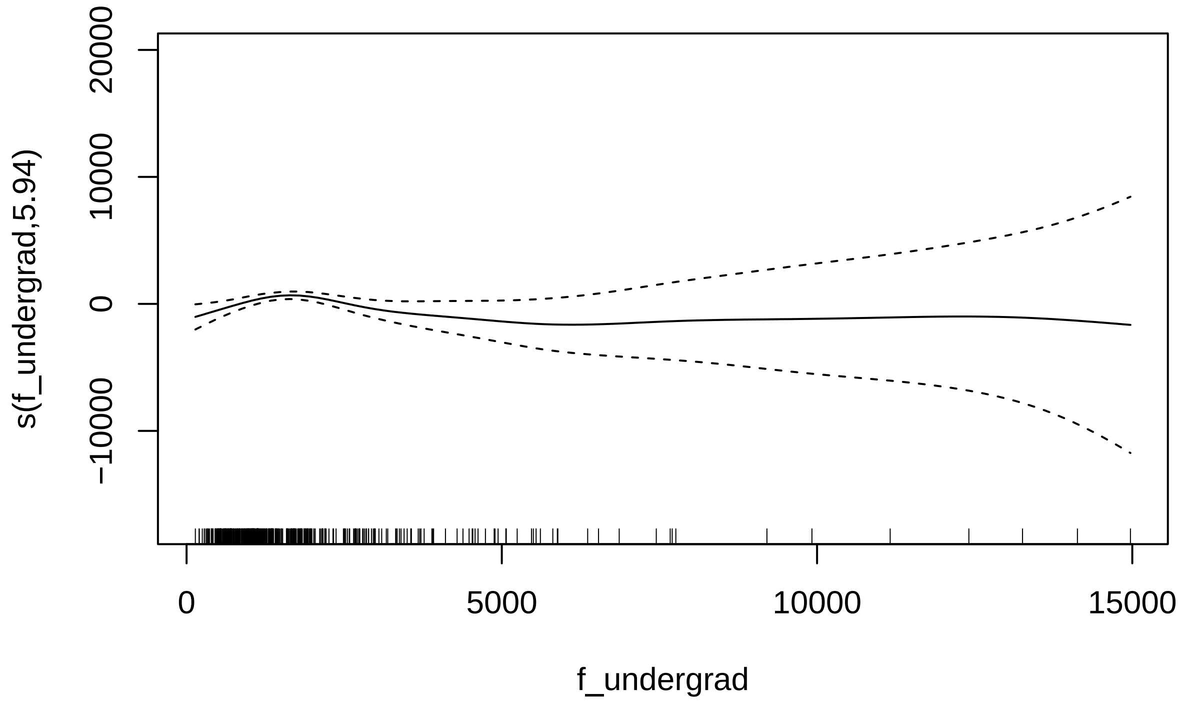

```
plot(model.gam$finalModel, select = 9)
```



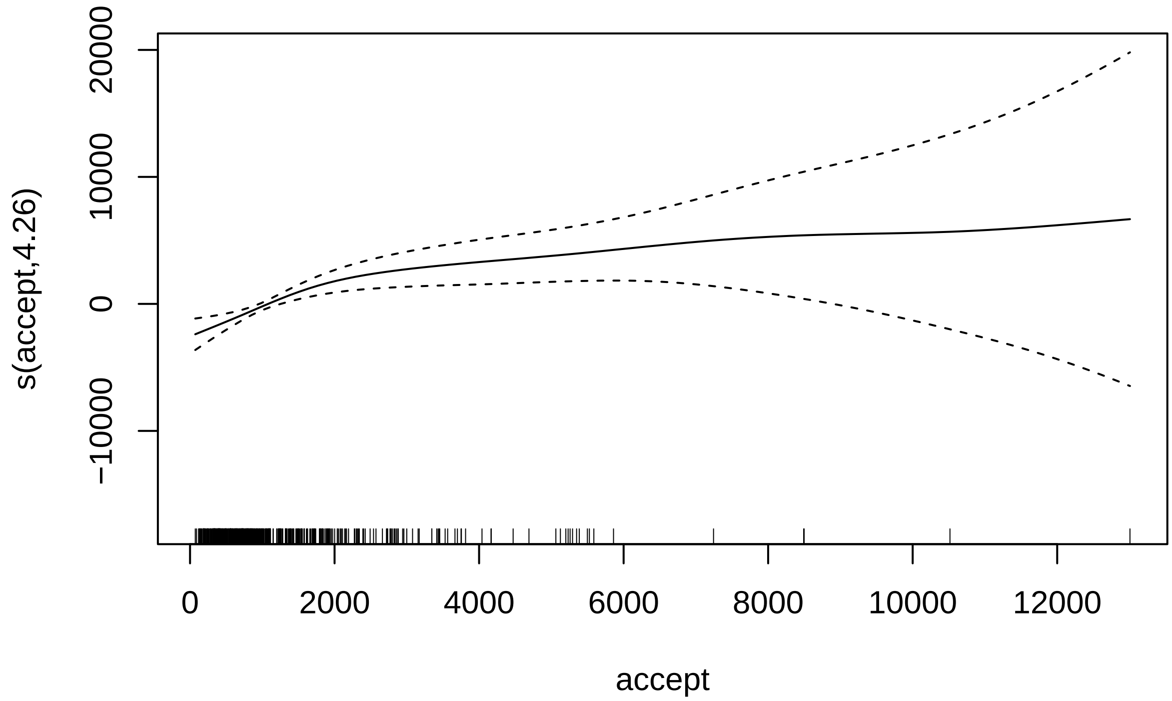

```
plot(model.gam$finalModel, select = 11)
```



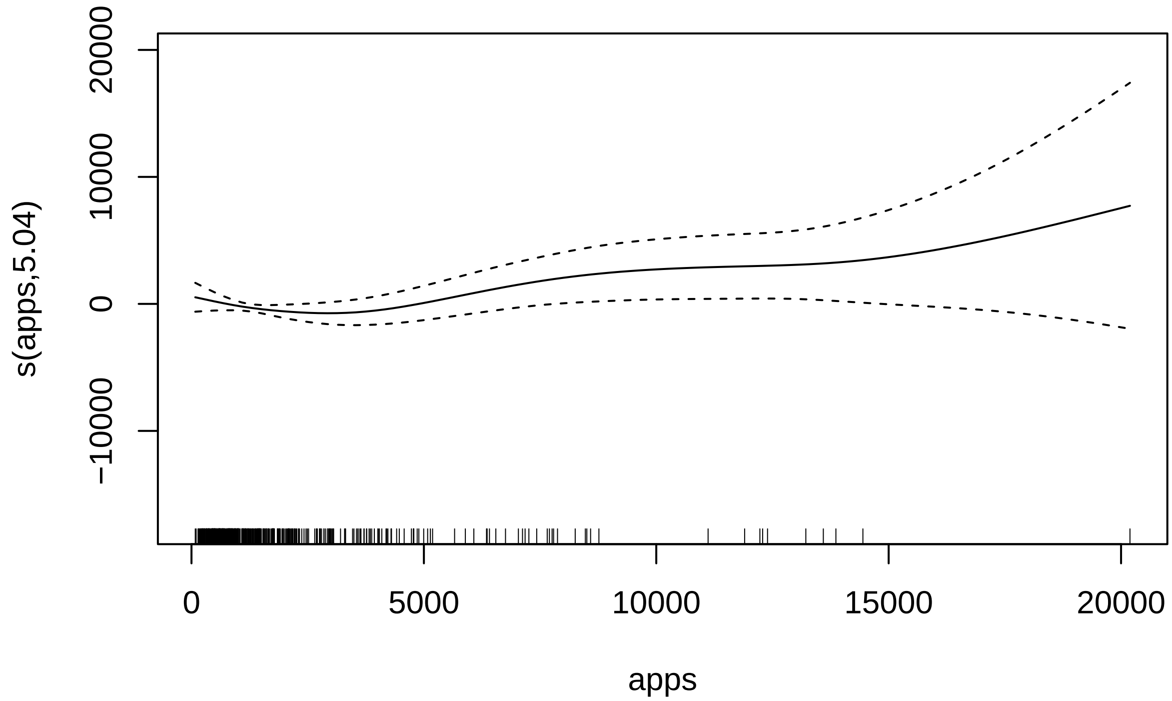

```
plot(model.gam$finalModel, select = 13)
```



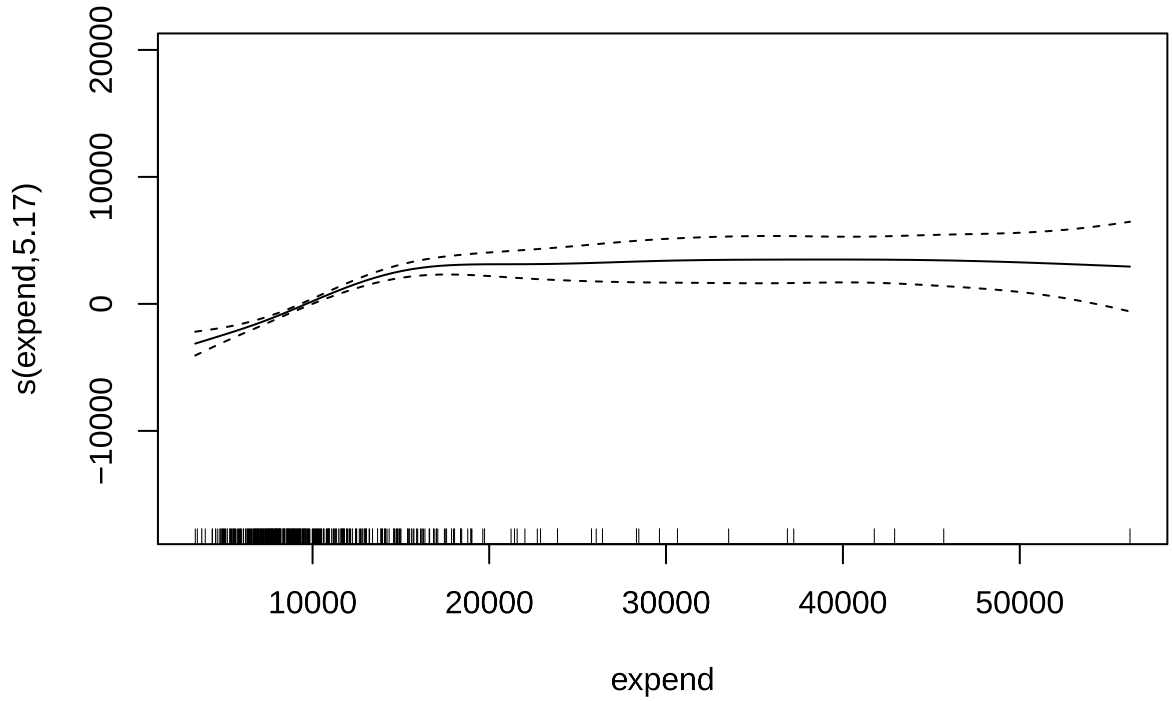

```r
plot(model.gam$finalModel, select = 14)
```




```r
plot(model.gam$finalModel, select = 15)
```




```
plot(model.gam$finalModel, select = 16)
```



For s(f_undergrad, 5.94), s(accept, 4.26), and s(apps, 5.04), we can see that the variances of their effects tend to increase as the corresponding predictor values increase, relative to the other covariates in the model.

```
# Obtain the test error
gam.pred <- predict(model.gam, newdata = test)
mean((gam.pred - pull(test, "outstate"))^2) # test error
```

```
## [1] 3834108
```

The test error is 3.83×10^6

(d) MARS vs linear model

```
# fit a linear model
set.seed(100)
model.lm <- train(x = train[2:17],
                  y = train$outstate,
                  method = "lm",
                  metric = "RMSE",
                  trControl = ctrl)

# check the model
summary(model.lm$finalModel)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6070.3 -1296.8   132.5  1337.0  4936.7
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 397.13325  956.36006   0.415  0.67816
## apps          0.05871    0.12520   0.469  0.63936
## accept        1.32210    0.21588   6.124 2.04e-09 ***
## enroll       -2.95772    1.09446  -2.702  0.00715 **
## top10perc    31.54737   15.49917   2.035  0.04241 *
## top25perc    -1.04688   12.34384  -0.085  0.93245
## f_undergrad  -0.25525    0.23437  -1.089  0.27673
## p_undergrad  -0.24355    0.14895  -1.635  0.10274
## room_board    0.89276    0.10895   8.194 2.85e-15 ***
## books         0.38723    0.54416   0.712  0.47709
## personal     -0.41074    0.14816  -2.772  0.00581 **
## ph_d         20.67055   10.47976   1.972  0.04919 *
## terminal     24.92126   11.63291   2.142  0.03272 *
## s_f_ratio   -11.23630   33.82499  -0.332  0.73991
## perc_alumni  39.51108    9.58650   4.122 4.51e-05 ***
## expend        0.13580    0.02684   5.060 6.21e-07 ***
## grad_rate    15.11448    6.86537   2.202  0.02822 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1933 on 435 degrees of freedom
## Multiple R-squared:  0.7277, Adjusted R-squared:  0.7176
## F-statistic: 72.64 on 16 and 435 DF,  p-value: < 2.2e-16
```

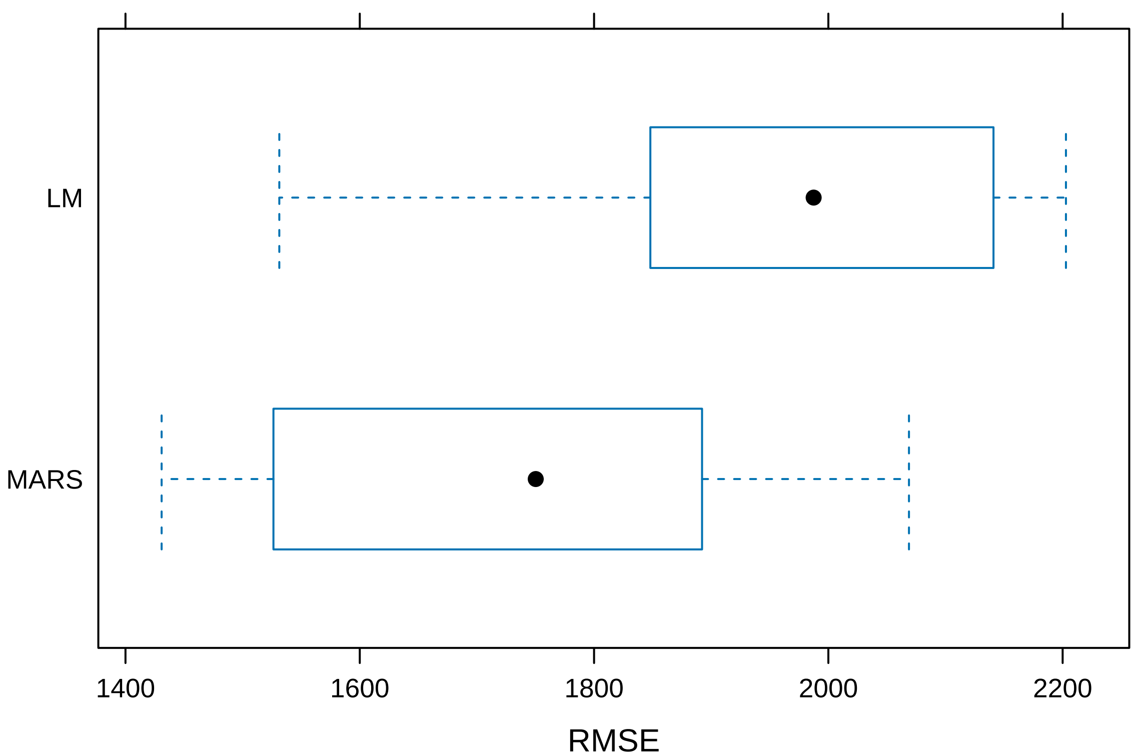
I used 10-fold cross validation to train the linear model including all the predictors. Now, let's compare the RMSEs between the two models using the resampling method.

```r
# resampling
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 1112.071 1226.045 1342.280 1349.000 1449.221 1624.525    0
## LM   1266.458 1452.715 1656.125 1610.315 1763.862 1846.010    0
##
## RMSE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## MARS 1430.852 1562.792 1750.215 1732.842 1878.424 2068.824    0
## LM   1531.323 1873.067 1987.467 1964.473 2137.328 2202.838    0
##
## Rsquared
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## MARS 0.7083646 0.7369688 0.7710106 0.7746155 0.8200407 0.8382725    0
## LM   0.6072036 0.6854601 0.7040280 0.7094711 0.7335389 0.8023868    0
```

```r
# visualize RMSEs
bwplot(resamp, metric = "RMSE")
```



Both the plot and summary output suggest that the MARS model outperforms the linear model in predicting out-of-state tuition. Therefore, in this case, I would prefer the MARS model.
In general applications, I think the superiority of a MARS model over a linear model depends on various factors. A MARS model may outperform a linear model when the data exhibits nonlinear relationships. However, if the data suggests linearity and parsimony is preferred for interpretability and computational efficiency, then a linear model may be favored.