P8106_HW2

Naomi Simon-Kumar ns3782

02/03/2025

Contents

Loading libraries

```
# Load libraries
library(earth)
library(tidyverse)
library(ISLR)
library(pls)
library(caret)
library(tidymodels)
library(pdp)
library(ggplot2)
library(mgcv)
```

Partition into training and testing set

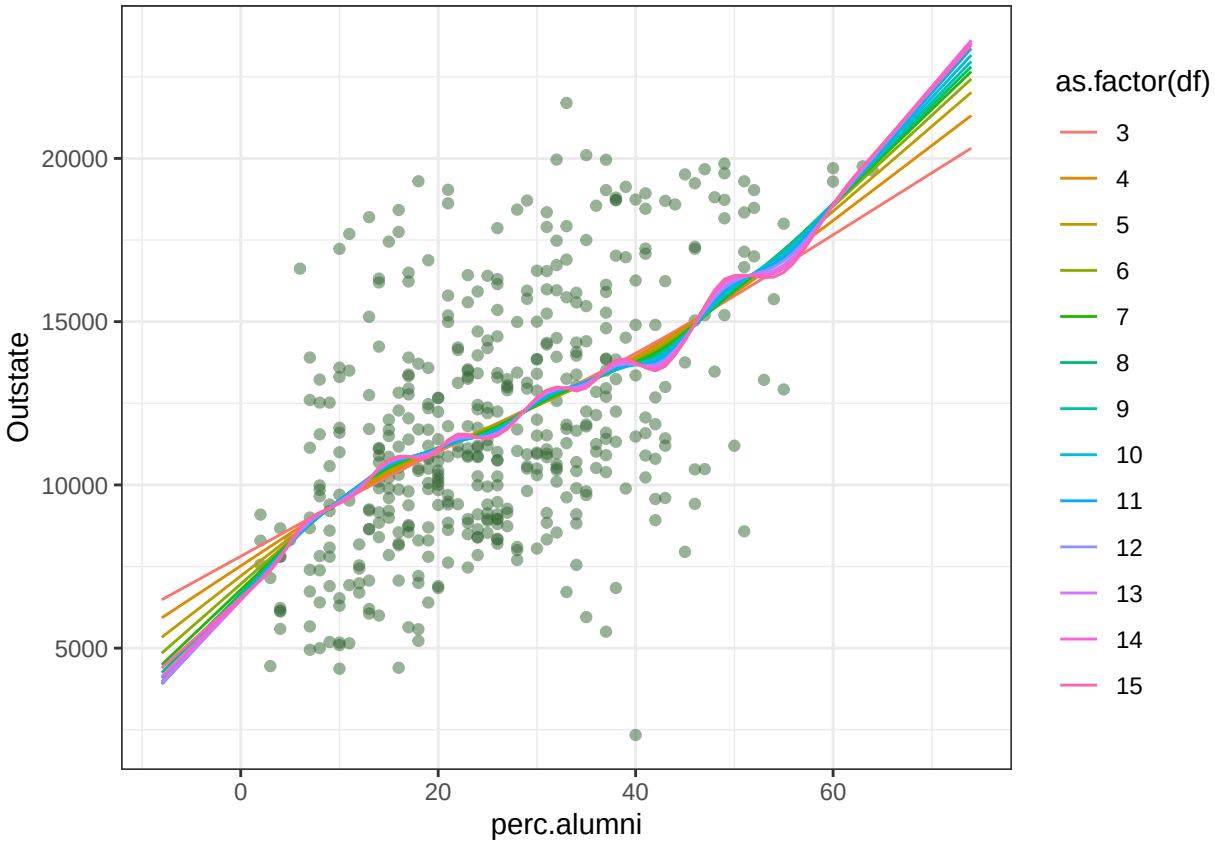
```
# Read in dataset
college <- read.csv("College.csv")

# Remove NAs
college <- na.omit(college)

# Set seed for reproducibility
set.seed(299)

# Split data into training and testing data
```

```r
data_split <- initial_split(college, prop = 0.8)


# Extract the training and test data
training_data <- training(data_split)
testing_data <- testing(data_split)
```

Question (a): Smoothing spline models

```r
# Set seed for reproducibility
set.seed(299)

# Fit smoothing spline
fit.ss <- smooth.spline(training_data$perc.alumni, training_data$Outstate)

# Define a grid for smooth predictions using dataset range
# Will illustrate curves beyond dataset boundary
perc.alumni.grid <- seq(from = min(training_data$perc.alumni) - 10,
                        to = max(training_data$perc.alumni) + 10,
                        by = 1)

# Create dataframe of degrees of freedom range
df_values <- seq(3, 15, by = 1)

# Create dataframe to store smoothing spline predictions
ss.predictions <- data.frame()

# Use for loop to populate ss.predictions dataframe
# Code Source: https://www.rdocumentation.org/packages/openintro/versions/2.4.0/topics/loop

for (df in df_values) {

  # Plot smoothing spline curves for different degrees of freedom
  # Code Source: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/smooth.spline

  fit <- smooth.spline(training_data$perc.alumni, training_data$Outstate, df = df)
  pred <- predict(fit, x = perc.alumni.grid)

  # Store pred for current df
  temp_df <- data.frame(
    perc.alumni = perc.alumni.grid,
    pred = pred$y,
    df = df
  )

  ss.predictions <- rbind(ss.predictions, temp_df) # Add to ss.predictions
}

# Scatter plot of perc.alumni vs Outstate
ss.p <- ggplot(training_data, aes(x = perc.alumni, y = Outstate)) +
  geom_point(color = rgb(.2, .4, .2, .5))
```

```
# Add smoothing spline curves for df range 3-15
ss.p +
  geom_line(aes(x = perc.alumni, y = pred, color = as.factor(df)),
            data = ss.predictions) + theme_bw()
```



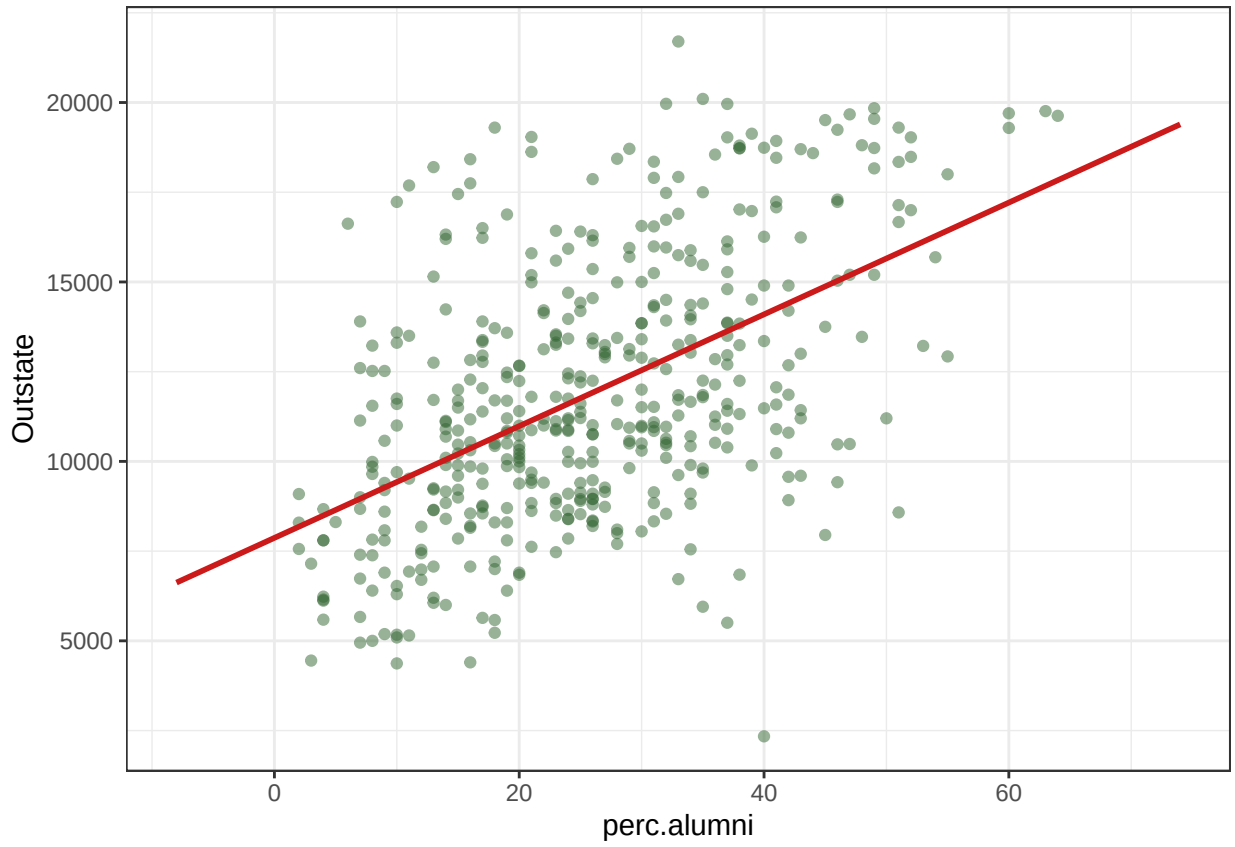
The plot I produced represents the fitted smoothing spline curves for each degree of freedom between the range of 3 and 15. It can be observed that as the degrees of freedom increases over this range, the smoothing spline goes from underfitting, to overfitting of the data. Specifically, as the degrees of freedom increases beyond ~10, the spline curve becomes highly wiggly, particularly at extreme values of perc.alumni, indicating overfitting of the data. For lower df (i.e., 3–5), the spline appears quite smooth.

```
# Set seed for reproducibility
set.seed(299)

# Fits a smoothing spline with automatically selected df using generalized cross-validation
fit.ss.gcv <- smooth.spline(training_data$perc.alumni, training_data$Outstate)

gcv_df <- fit.ss.gcv$df  # Extract optimal df selected by GCV
print(gcv_df) # 2.000237 appears to be the optimal df selected by GCV
```

```
## [1] 2.000237
```

```
# Predict values using GCV-selected df
pred.ss.gcv <- predict(fit.ss.gcv, x = perc.alumni.grid)
```

```r
# Convert predictions to dataframe
ss.optimal <- data.frame(
  perc.alumni = perc.alumni.grid,
  pred = pred.ss.gcv$y
)

# Add optimal spline curve to the plot
ss.p +
  geom_line(aes(x = perc.alumni, y = pred), data = ss.optimal,
  color = rgb(.8, .1, .1, 1), linewidth = 1) + theme_bw()
```




```r
# Checking lambda value

lambda_value <- fit.ss.gcv$lambda
print(lambda_value) # High lambda value 2384.249
```

```
## [1] 2384.249
```

For my smoothing spline model, I selected a degree of freedom of approximately 2, as determined by Generalized Cross-Validation (GCV). The GCV method minimises a criterion balancing model fit against model complexity, and in this case, selected a nearly linear model (df approx. 2.000237). This is evident in the plot above where the optimal smoothing spline appears as essentially a straight line, indicating that the relationship between percentage of alumni who donate and out-of-state tuition is best represented linearly.

While I explored models with higher degrees of freedom (ranging from 3 to 15), these more complex models with greater flexibility showed increasing wiggliness, especially at extreme values of perc.alumni as discussed, indicating potential overfitting. The GCV criterion effectively penalised this unnecessary complexity and favored the simpler model. From what we have covered in class, we know that smoothing splines can range from very flexible (high df) to very smooth, with a minimum of 2 degrees of freedom. As discussed in class, when lambda approaches infinity, the function becomes linear, and this model has a relatively large lambda value (2384.249), explaining why the optimal model with df approximately 2 appears as a straight line. This relationship between lambda and degrees of freedom demonstrates how the roughness penalty controls the smoothness of the function.

The GCV has selected a model at approximately this minimum value, resulting in a nearly linear fit that would indicate out-of-state tuition (outstate) increases steadily with the percentage of alumni who donate (perc.alumni), without needing to be represented by a more complex nonlinear relationship.

Question (b): Multivariate Adaptive Regression Spline (MARS) model

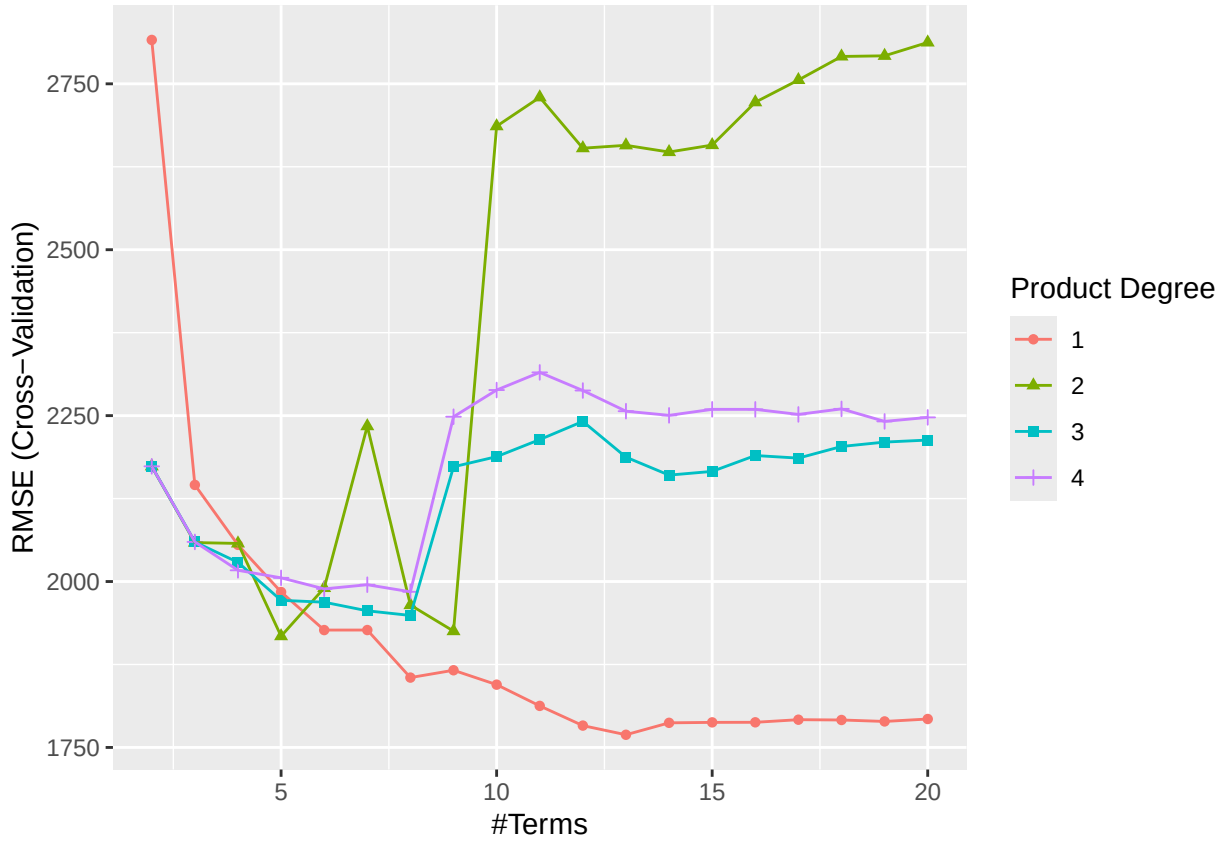
In order to fit a piecewise linear mode using MARs, it is necessary to undertake a grid search to identify the optimal combination of the two hyperparameters (the degree of interactions, and the number of retained terms) that minimise prediction error.

In the college dataset, there are greater than 15 predictors - we can therefore expect some interactions to potentially be important to the model. Therefore, I opted to set `degree` to 1:4, allowing me to represent an appropriate level of interactions without too much complexity. I initially tried a range of 1:3, but extended this to see if it would impact the patterns.

For the number of retained terms (`nprune`), I set the range to 2:25 because I assumed this provides a good balance between model simplicity and flexibility given our dataset size. With 565 observations and 17 variables, setting the maximum number of terms to 25 allows us to avoid overfitting while still allowing the model to capture complexity in the data.

```
# Set seed for reproducibility
set.seed(299)

# Set 10-fold CV
ctrl1 <- trainControl(method = "cv", number = 10)

# Remove College column
training_data <- training_data %>% select(-College)
testing_data <- testing_data %>% select(-College)

# Specify MARS grid - first attempt using Professor's example
# mars_grid <- expand.grid(
#  degree = 1:3, # degree of interactions
#  nprune = 2:15  # no. of retained terms
# )

# Specify MARS grid - expanding grid to observe pattern
mars_grid <- expand.grid(
  degree = 1:4, # degree of interactions
  nprune = 2:20  # no. of retained terms
)

# Fit data on training set
mars.fit <- train(Outstate ~ .,
                  data = training_data,
```

```
                  method = "earth",
                  tuneGrid = mars_grid,
                  trControl = ctrl1)

# Fit the ggplot

ggplot(mars.fit)
```



To ensure I identified the optimal model complexity, I expanded the grid search to include degree = 1:4 and nprune = 2:20. This allowed me to observe how prediction error varied across different interaction levels and numbers of retained terms.

From the results, I observed a clear dip in RMSE when degree = 1, followed by stabilisation, indicating that interactions beyond first-order did not meaningfully improve predictive performance. When I increased the interaction degree, RMSE values became more variable rather than decreasing steadily, which indicated there was no reliable reduction in prediction error. Given this, I selected degree = 1 as the optimal choice. The range nprune = 2:20 was also appropriate, as RMSE initially decreased as nprune increased but then stabilised, indicating that adding more terms would not significantly improve prediction accuracy.

```
# Set seed for reproducibility
set.seed(299)

# Optimal parameters
mars.fit$bestTune
```

```
##    nprune degree
```

```
## 12     13      1
```

```
# Final model coefficients
mars_coef <- coef(mars.fit$finalModel)

mars_coef
```

```
##          (Intercept)     h(Expend-15003)     h(83-Grad.Rate)  h(Room.Board-4138)
##       11503.4960291          -0.6259264         -28.7113025           0.3277437
##  h(4138-Room.Board) h(F.Undergrad-1365)   h(16-perc.alumni)    h(1300-Personal)
##           -1.3252744          -1.1507561        -107.9892002           1.1616572
##      h(Expend-5664) h(F.Undergrad-3566)           h(PhD-81)      h(2059-Accept)
##            0.6356263           0.8655538          64.9589711          -1.5784786
##        h(Apps-3294)
##            0.3322853
```

The best-tuned model selected `degree = 1` and `nprune = 13`. Therefore, the final model includes 13 retained terms with no interaction effects.

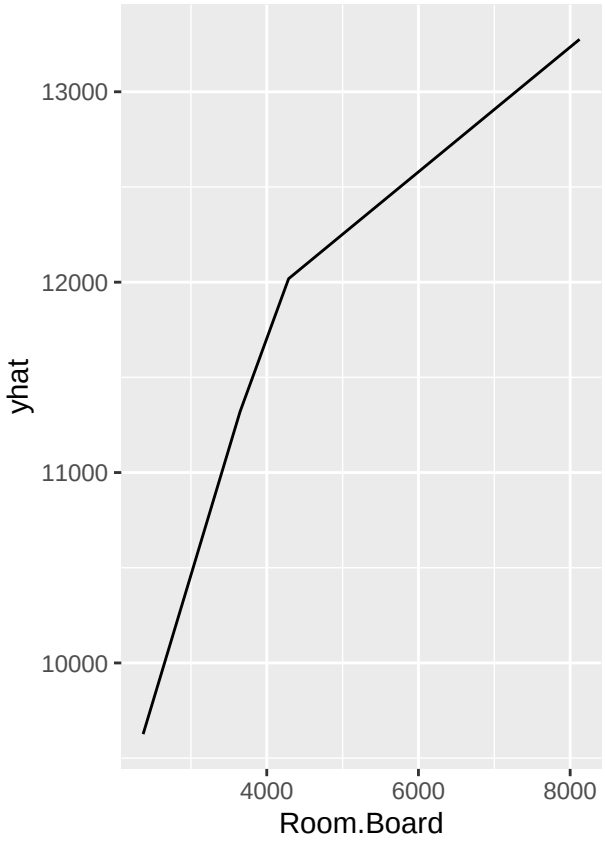
The final `regression equation` (degree = 1, nprune = 13) is as follows:

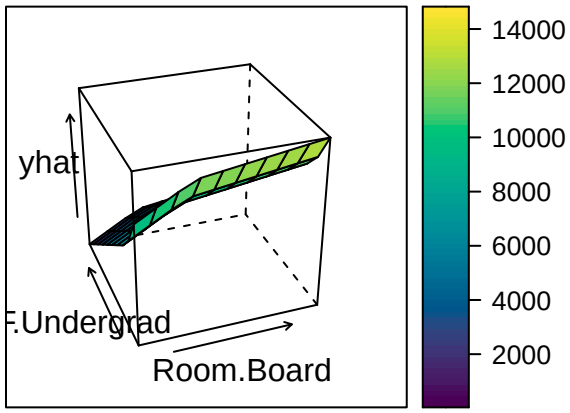
$$
\begin{aligned}
\hat{y} = {} & 11503.4960291 \\
& - 1.3252744 \cdot h(4138 - \text{Room.Board}) - 0.6259264 \cdot h(\text{Expend} - 15003) \\
& - 28.7113025 \cdot h(83 - \text{Grad.Rate}) + 0.3277437 \cdot h(\text{Room.Board} - 4138) \\
& - 1.1507561 \cdot h(\text{F.Undergrad} - 1365) - 107.9897002 \cdot h(16 - \text{perc.alumni}) \\
& + 1.1616572 \cdot h(1300 - \text{Personal}) + 0.6356263 \cdot h(\text{Expend} - 5664) \\
& + 0.8655538 \cdot h(\text{F.Undergrad} - 3566) + 64.9589711 \cdot h(\text{PhD} - 81) \\
& - 1.5784786 \cdot h(2059 - \text{Accept}) + 0.3322853 \cdot h(\text{Apps} - 3294)
\end{aligned}
$$

where h(x-c) represents hinge functions, defined as h(x-c) = max(0, x-c).

Next, creating a `Partial Dependency Plot of a selected predictor`, Room.Board.

```
# Set seed for reproducibility
set.seed(299)

# PDP for Room.Board (single predictor)
p1 <- pdp::partial(mars.fit, pred.var = c("Room.Board"), grid.resolution = 10) |>
  autoplot()

# PDP for Room.Board and F.Undergrad (interaction effect)
p2 <- pdp::partial(mars.fit, pred.var = c("Room.Board", "F.Undergrad"),
                   grid.resolution = 10) |>
  pdp::plotPartial(levelplot = FALSE, zlab = "yhat", drape = TRUE,
                   screen = list(z = 20, x = -60))

# Show side by side
gridExtra::grid.arrange(p1, p2, ncol = 2)
```





The left partial dependence plot of the predictor Room.Board (room and board costs) shows out-of-state tuition rises sharply as Room.Board costs increase up to around $4,200, then increases more gradually. This indicates that Room.Board increases beyond this threshold have a smaller, but still clear effect on out of state tuition.

On the right, the interaction plot includes F.Undergrad (number of full-time undergraduates). Based on this, it appears that Room.Board clearly drives great variation in tuition predictions, but at higher Room.Board values, larger undergraduate enrollment is also associated with higher tuition. This suggests an effect where institutions with both high Room.Board and high enrollment tend to charge the highest out-of-state tuition.

Next, obtaining the test error:

```
# Set seed for reproducibility
set.seed(299)

# Obtain response variable from testing data
y_testing <- testing_data$Outstate

# Predict on test data using trained MARS model
y_pred_MARS <- predict(mars.fit, newdata = testing_data)

# Compute RMSE (Test Error)
test_rmse <- sqrt(mean((y_testing - y_pred_MARS)^2))

# Print test RMSE
test_rmse
```

```
## [1] 1553.758
```

Therefore, the test error is `1553.758`.

Question (c): Generalized Additive Model (GAM)

First, I will proceed with constructing a GAM model, allowing us to mix non-linear and linear terms and build a model estimating the relationship between the outcome (`Outstate`) and predictors in the provided dataset.

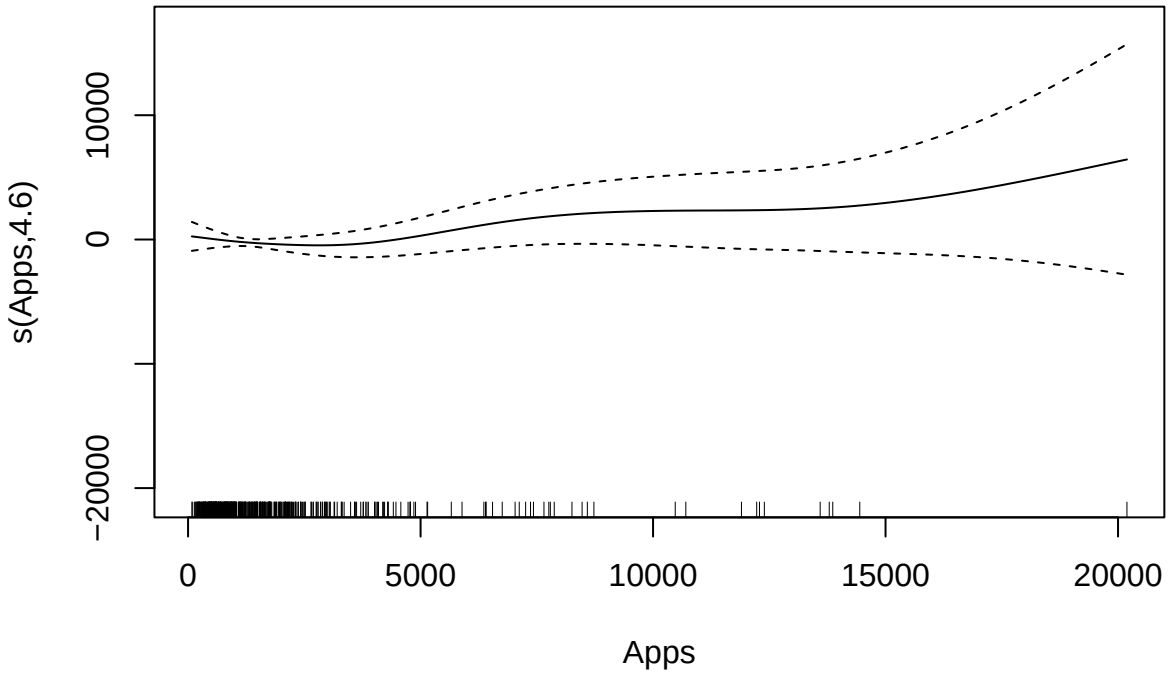
```
# Set seed for reproducibility
set.seed(299)

# Fit a GAM model, using training data
gam_outstate <- gam(Outstate ~ s(Apps) + s(Accept) + s(Enroll) +
                            s(Top10perc) + s(Top25perc) + s(F.Undergrad) +
                            s(P.Undergrad) + s(Room.Board) + s(Books) +
                            s(Personal) + s(PhD) + s(Terminal) + s(S.F.Ratio) +
                            s(perc.alumni) + s(Expend) + s(Grad.Rate),
                  data = training_data)

# Summary
summary(gam_outstate)
```

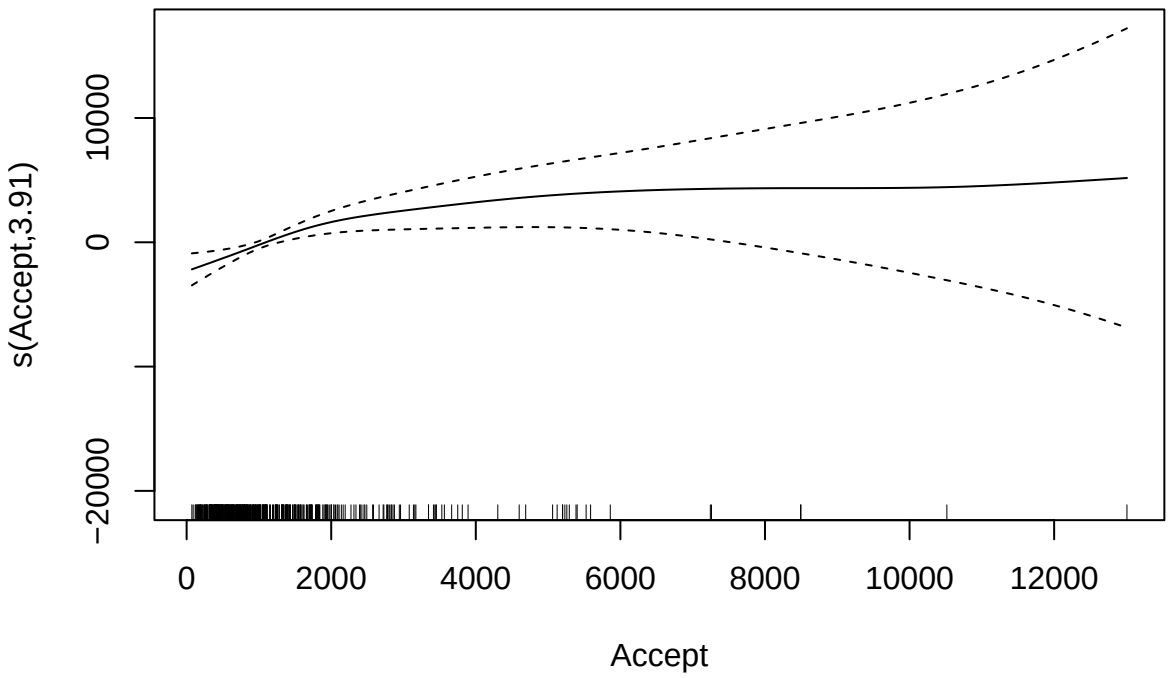
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Outstate ~ s(Apps) + s(Accept) + s(Enroll) + s(Top10perc) + s(Top25perc) +
##     s(F.Undergrad) + s(P.Undergrad) + s(Room.Board) + s(Books) +
##     s(Personal) + s(PhD) + s(Terminal) + s(S.F.Ratio) + s(perc.alumni) +
##     s(Expend) + s(Grad.Rate)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11894.77      76.29   155.9   <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(Apps)         4.596  5.588  1.523 0.171933
## s(Accept)       3.907  4.806  2.910 0.015091 *
## s(Enroll)       1.000  1.000 12.048 0.000575 ***
## s(Top10perc)    3.382  4.277  1.163 0.311423
## s(Top25perc)    1.000  1.000  0.482 0.487801
## s(F.Undergrad) 6.316  7.300  3.157 0.002751 **
## s(P.Undergrad) 1.000  1.000  0.142 0.706481
## s(Room.Board)  2.054  2.611 13.862 4.78e-07 ***
## s(Books)        2.051  2.564  1.917 0.171686
## s(Personal)     2.646  3.315  2.100 0.097372 .
## s(PhD)          4.223  5.197  1.420 0.250407
## s(Terminal)     1.835  2.326  0.797 0.433270
## s(S.F.Ratio)    3.495  4.403  1.482 0.196306
```

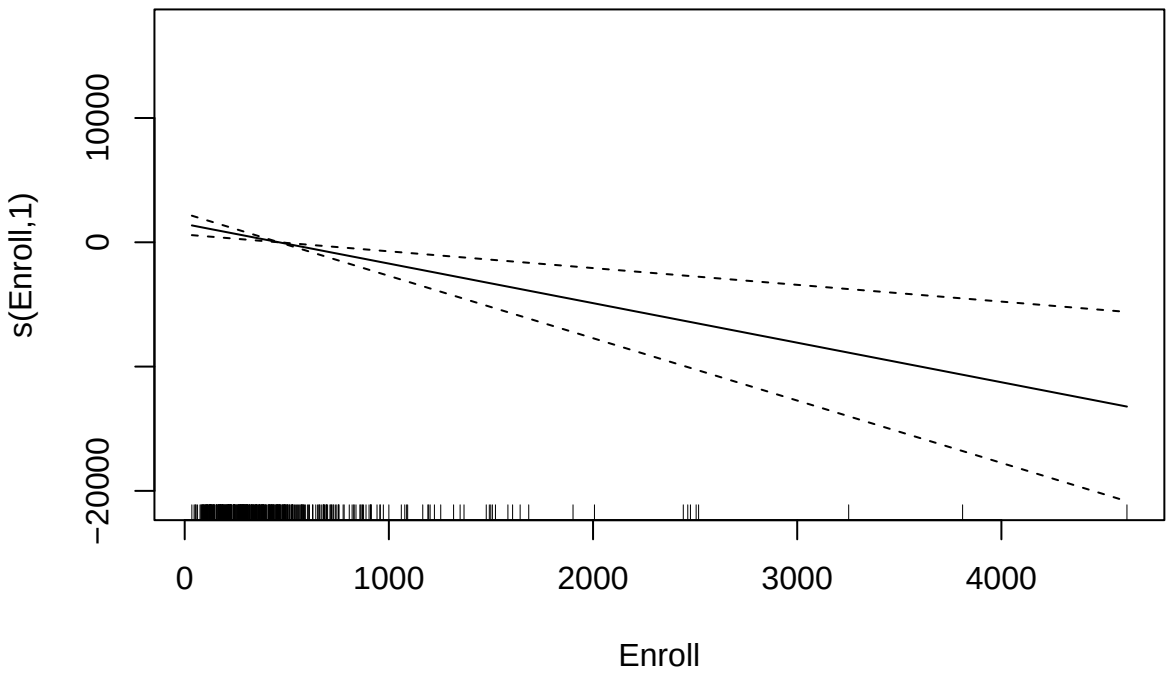
```
## s(perc.alumni) 1.772  2.243  7.175 0.000725 ***
## s(Expend)      6.509  7.580 16.485  < 2e-16 ***
## s(Grad.Rate)   3.712  4.672  2.871 0.019277 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.809   Deviance explained =   83%
## GCV = 2.9615e+06  Scale est. = 2.6307e+06  n = 452
```

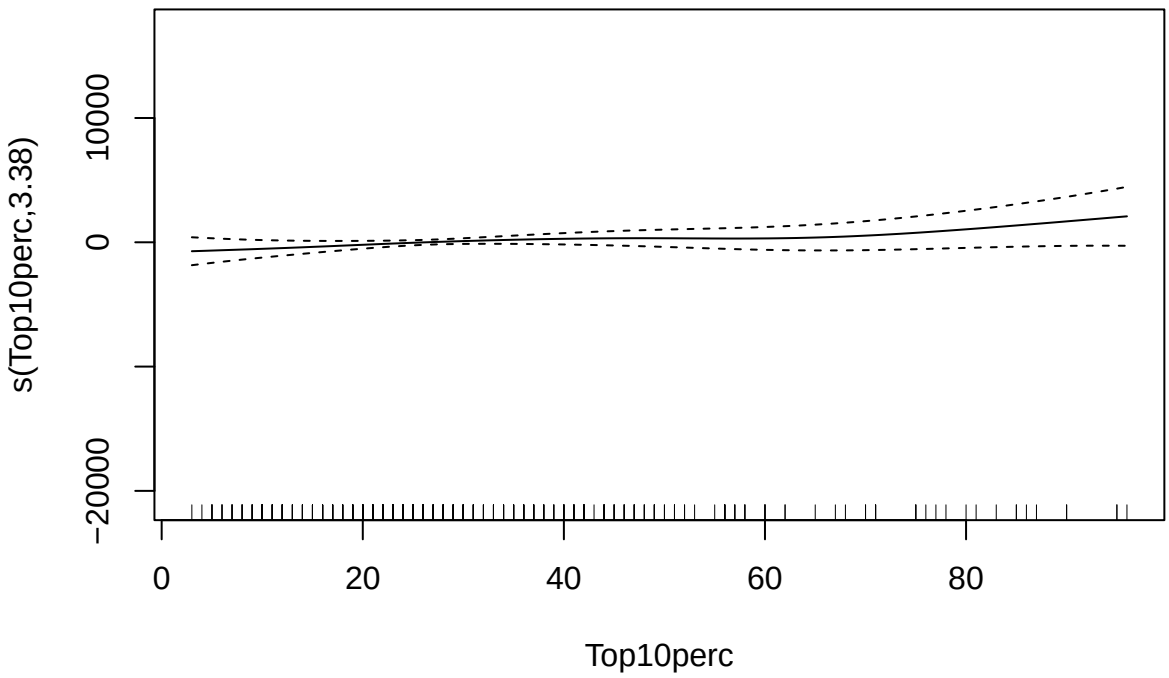
Based on the GAM model summary, many of the predictors have estimated degrees of freedom (edf) greater than 1, indicating that they are non-linear terms. There are also predictors that are linear or likely to be linear (edf around 1).

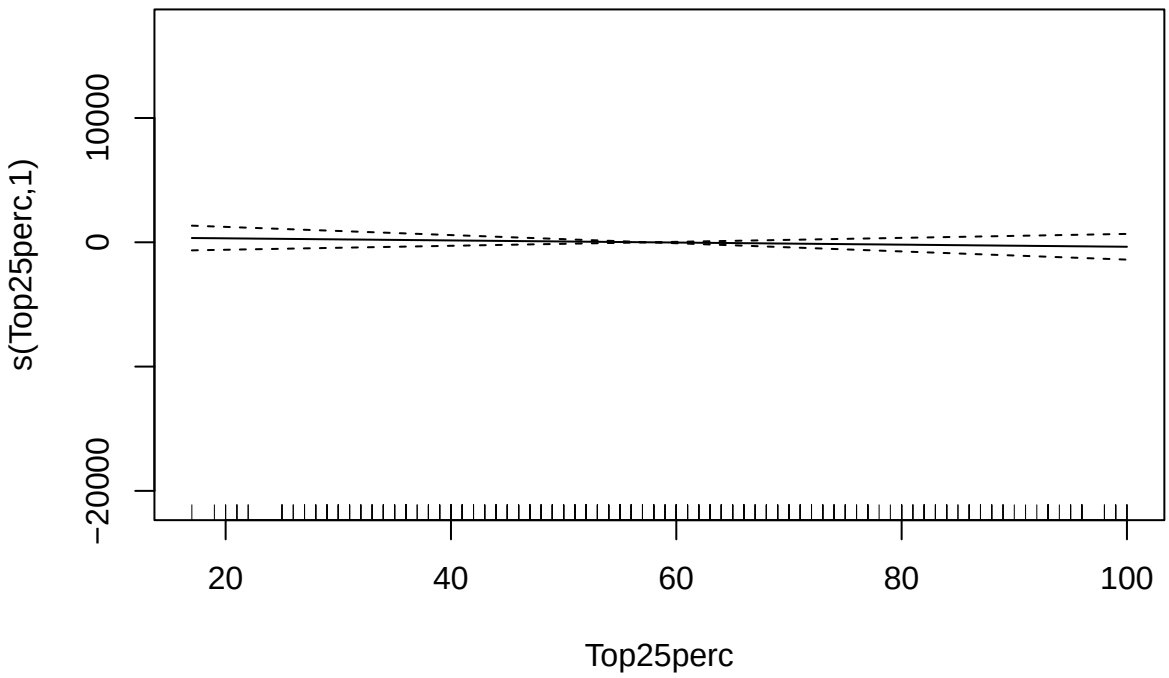
```
# Visualise predictors
plot(gam_outstate)
```

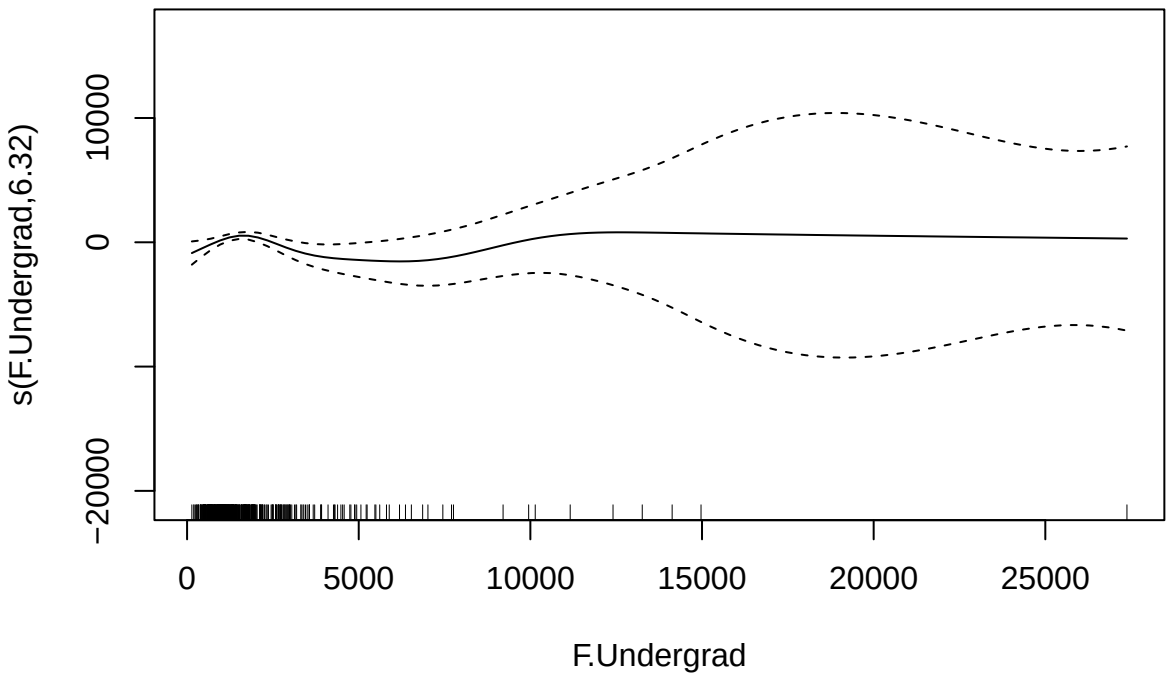


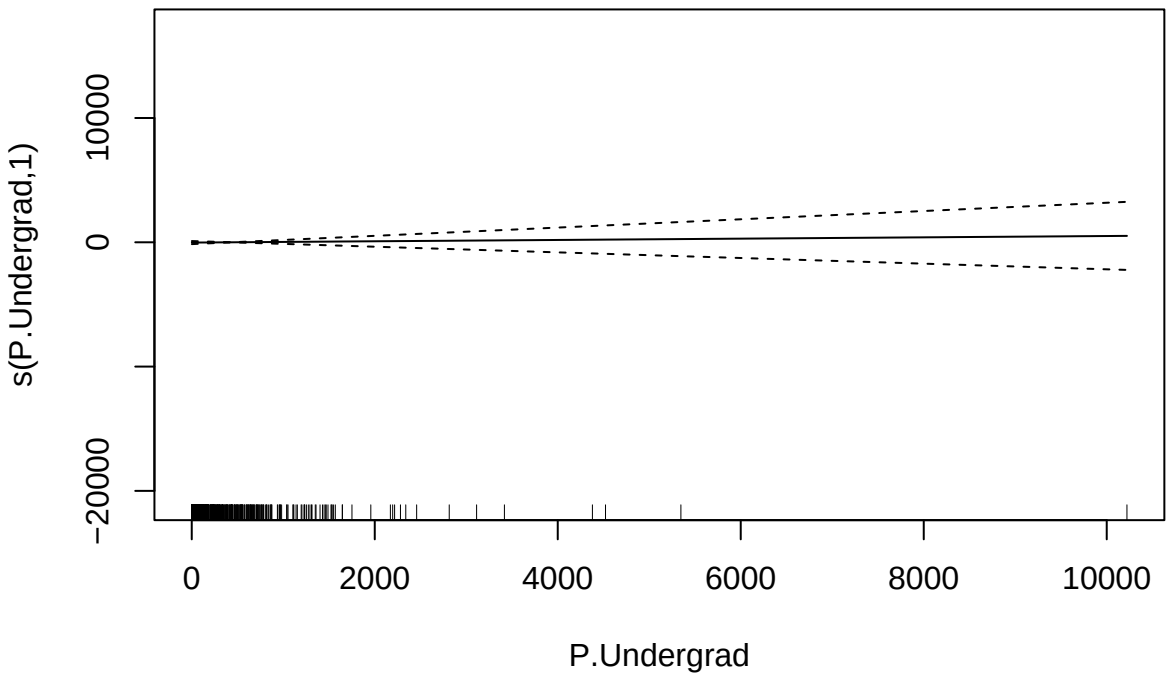

s(Accept,3.91)
10000
0
−20000
0
2000
4000
6000
8000
10000
12000
Accept

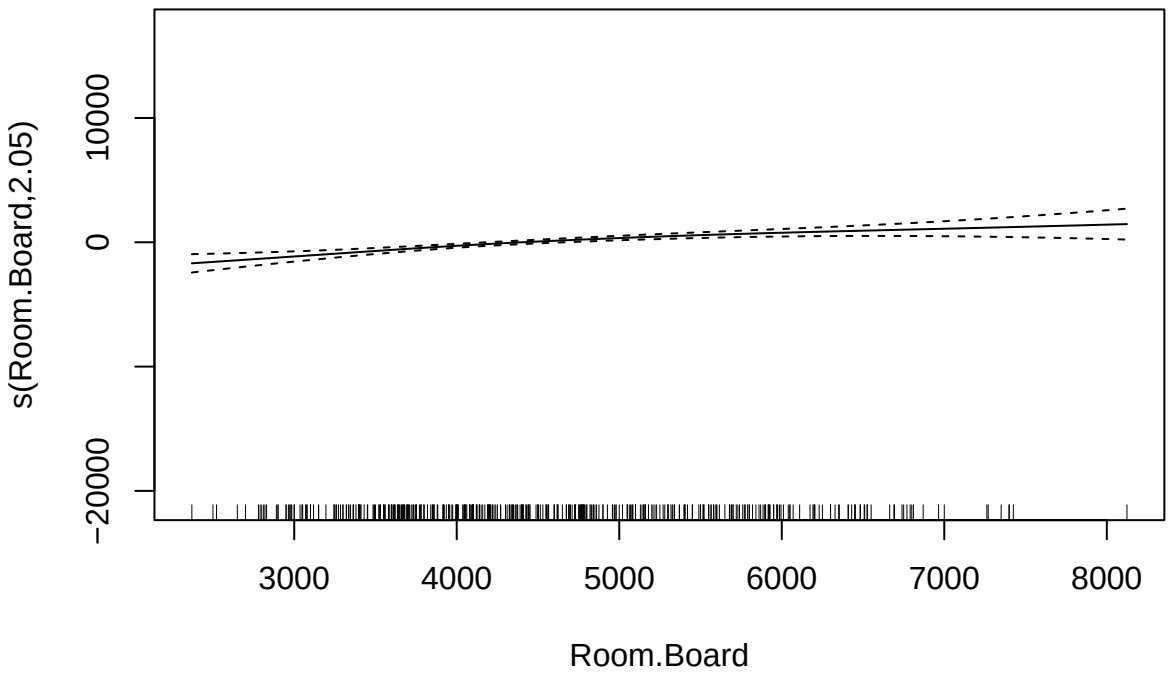

s(Enroll,1)
10000
0
−20000
0
1000
2000
3000
4000
Enroll

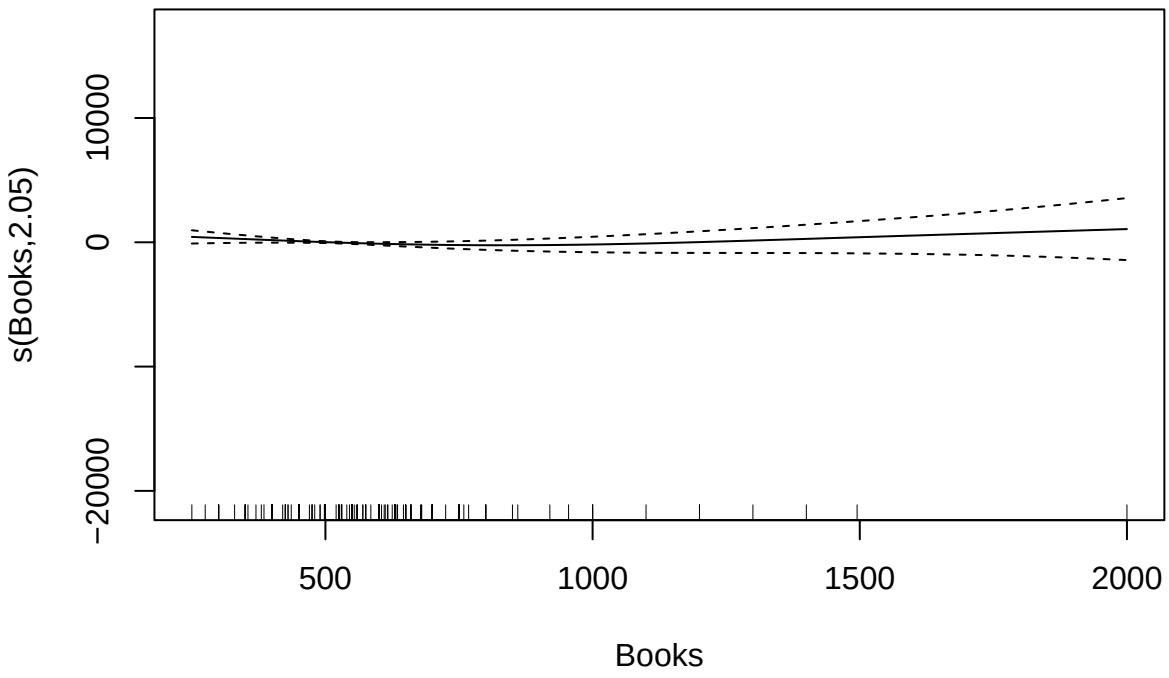

s(Top10perc,3.38)
10000
0
−20000
0
20
40
60
80
Top10perc

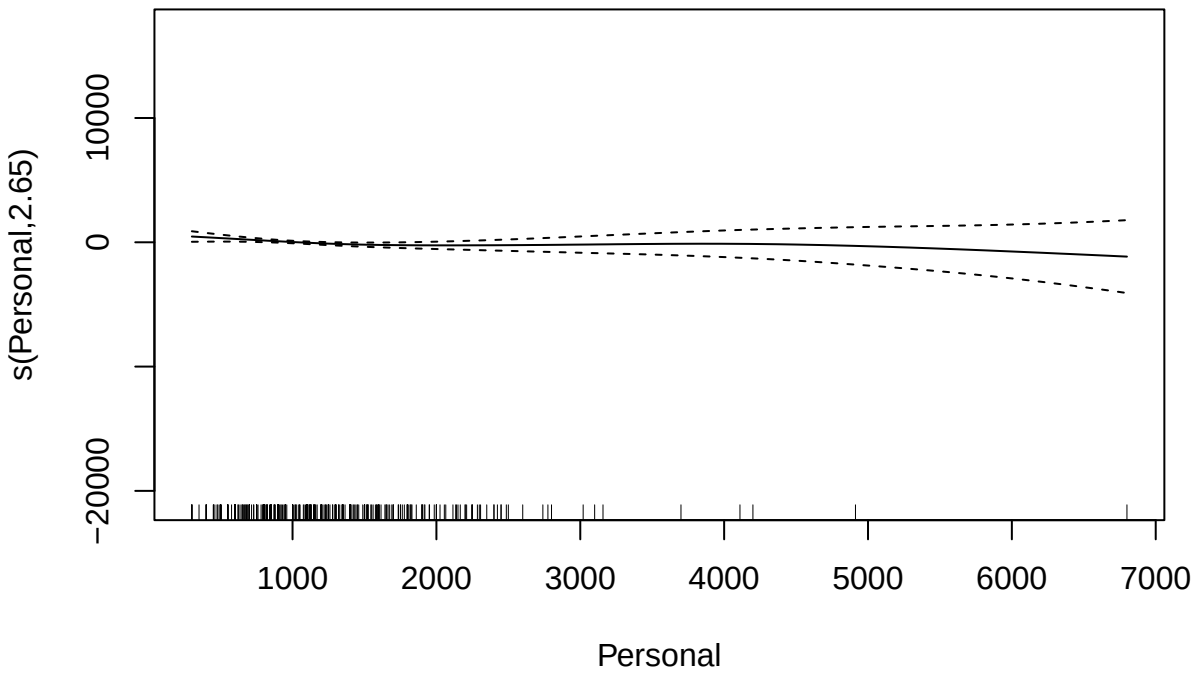

s(Top25perc,1)
10000
0
−20000
20
40
60
80
100
Top25perc

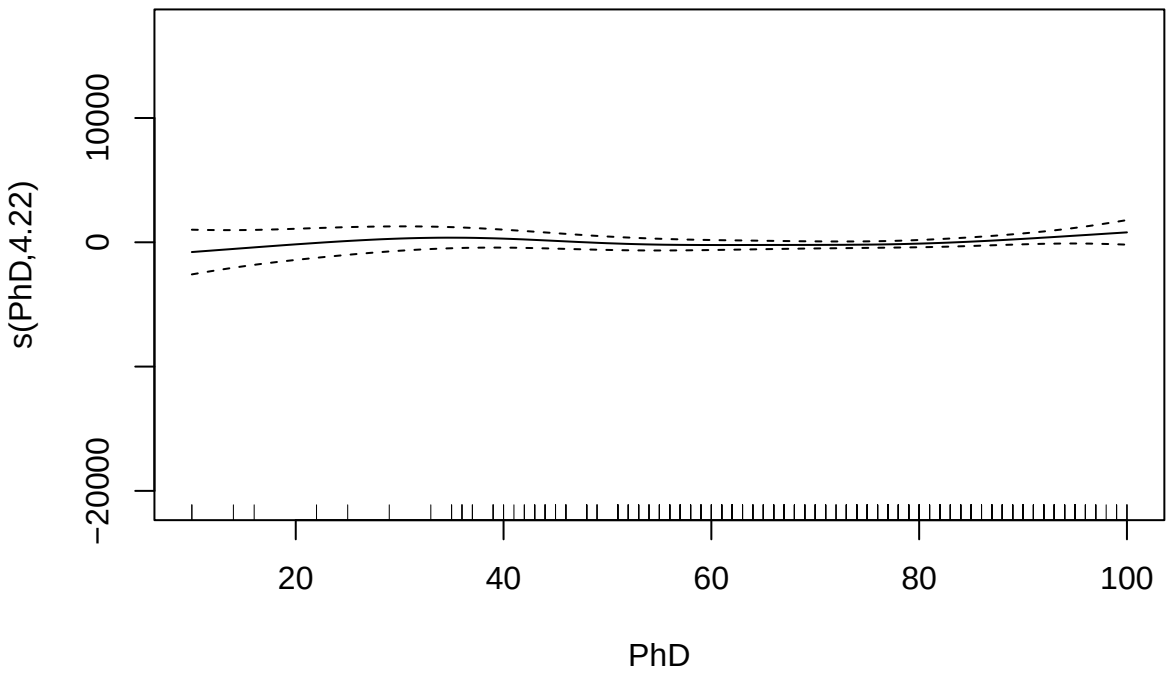

s(F.Undergrad,6.32)
10000
0
−20000
0
5000
10000
15000
20000
25000
F.Undergrad

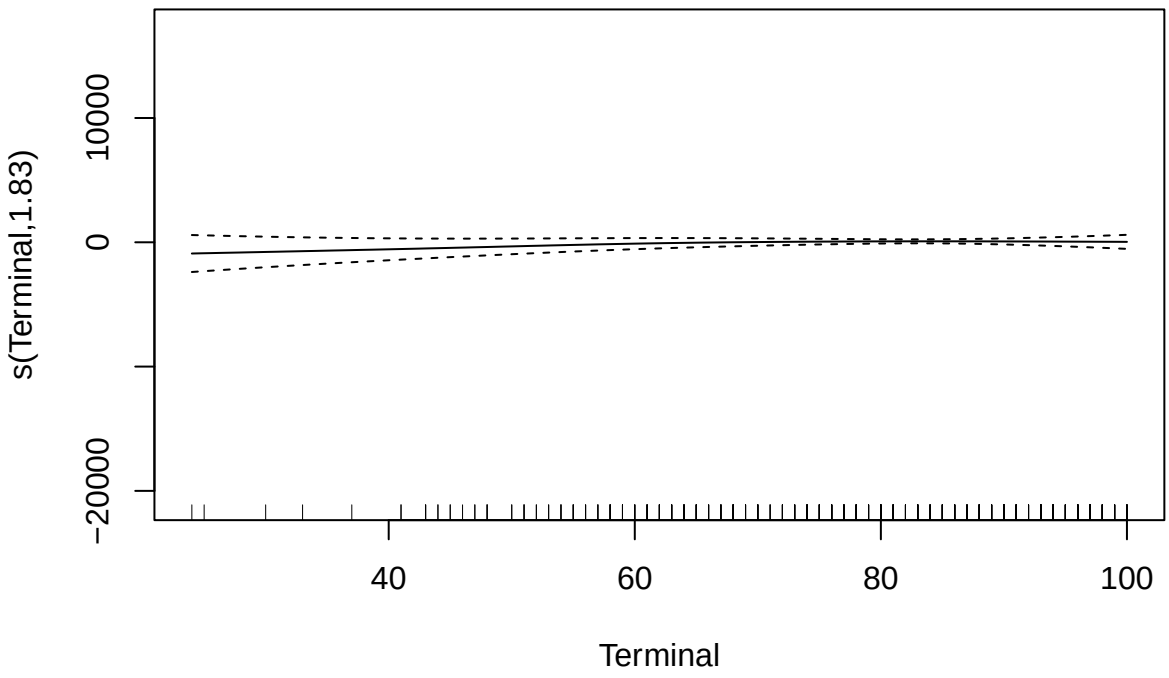

s(P.Undergrad,1)
10000
0
−20000
0
2000
4000
6000
8000
10000
P.Undergrad

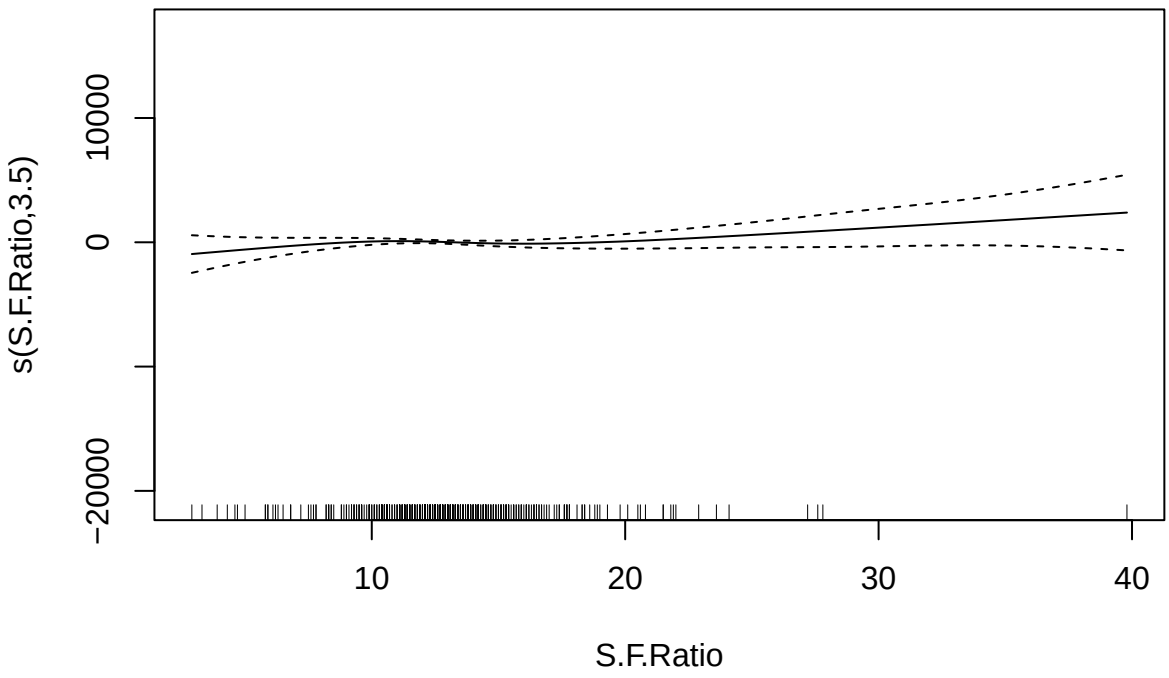

s(Room.Board,2.05)
10000
0
−20000
3000
4000
5000
6000
7000
8000
Room.Board

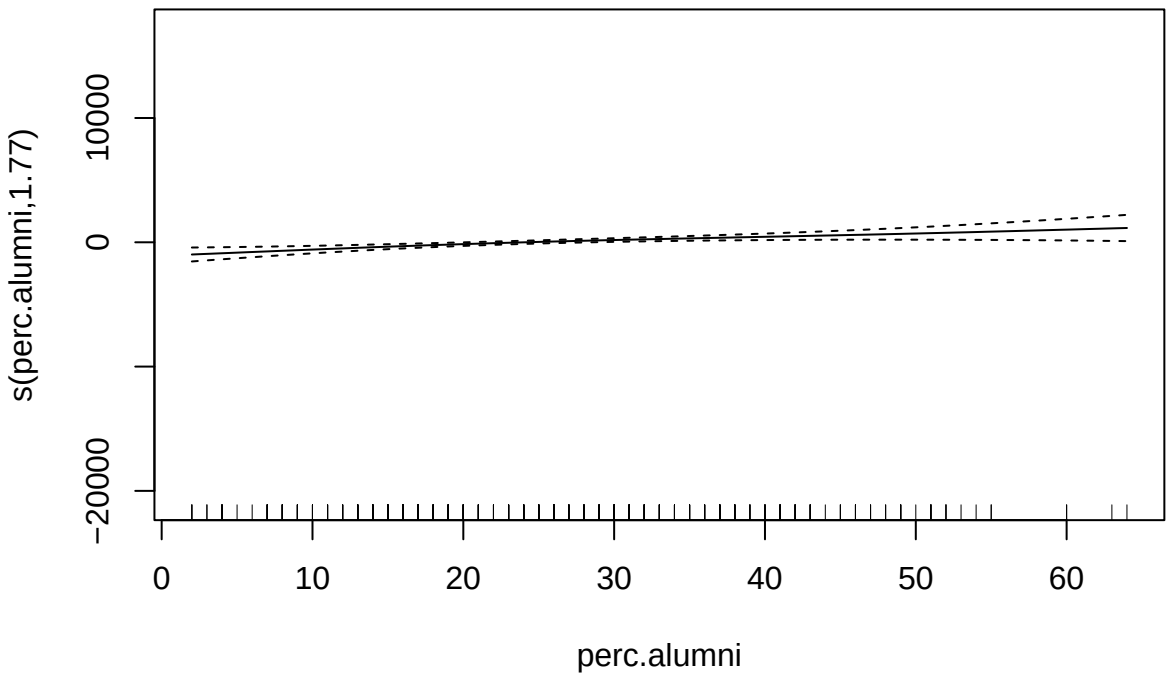

s(Books,2.05)
10000
0
−20000
500
1000
1500
2000
Books

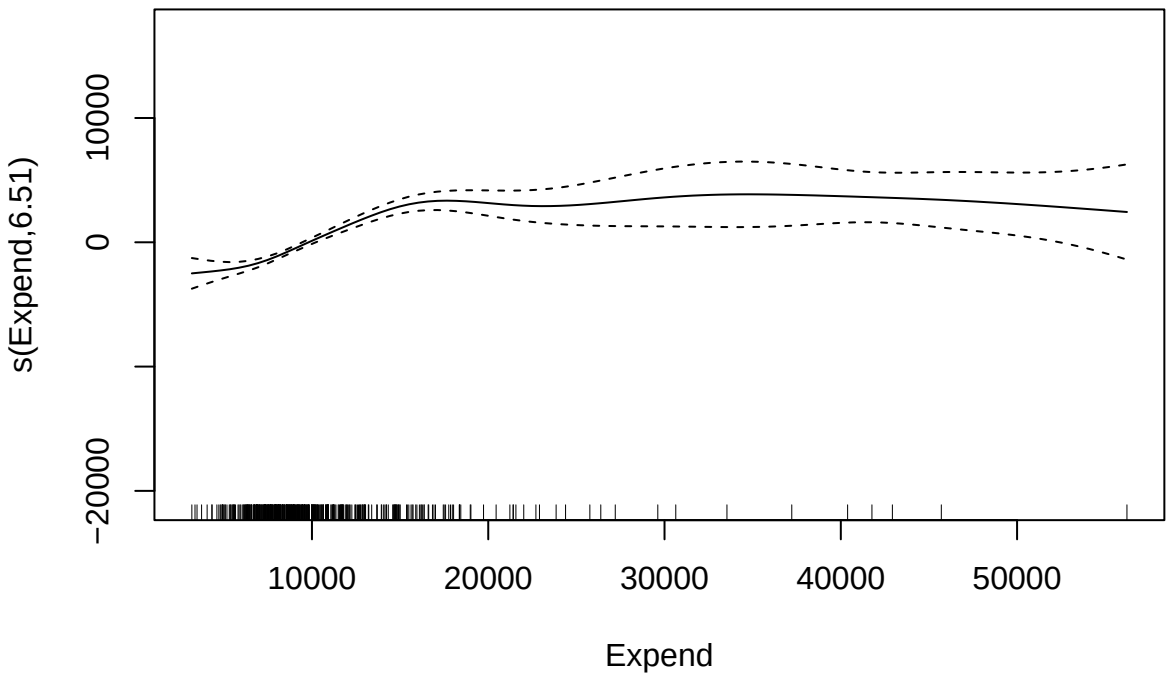

s(Personal,2.65)
10000
0
−20000
1000
2000
3000
4000
5000
6000
7000
Personal

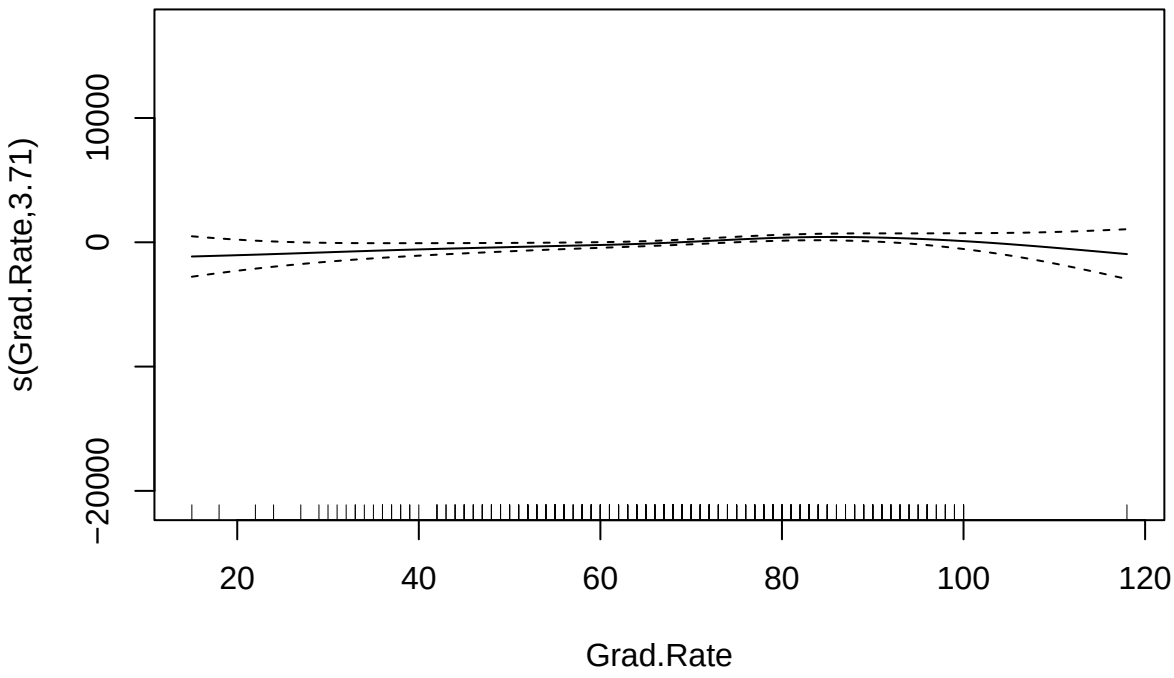

s(PhD,4.22)
10000
0
−20000
20
40
60
80
100
PhD


s(Terminal,1.83)
10000
0
−20000
40
60
80
100
Terminal


s(S.F.Ratio,3.5)
10000
0
−20000
10
20
30
40
S.F.Ratio


s(perc.alumni,1.77)
10000
0
−20000
0
10
20
30
40
50
60
perc.alumni


s(Expend,6.51)
10000
0
−20000
10000
20000
30000
40000
50000
Expend



Based on the plots above, Enroll, Top25perc, and P.Undergrad show linear relationships with out-of-state tuition, consistent with their estimated degrees of freedom (edf around 1) in the model summary. Enroll appears to be the only significant linear term, based on the model summary ($p < 0.05$).

Other predictors, like Apps, Accept, and F.Undergrad among others show some nonlinearity, which is also reflected in their higher edf values. However, not all nonlinear terms are statistically significant. For example, Apps and Top10perc have higher edf values but non-significant p-values, suggesting their effects may not be meaningful. However, nonlinear terms like Accept, F.Undergrad, Room.Board, perc.alumni, and Expend are significant.

Next, finding the test error of the model:

```
# Set seed for reproducibility
set.seed(299)

# Prediction using testing data
y_pred_gam <- predict(gam_outstate, newdata = testing_data)

# Test error/RMSE
test_rmse_gam <- sqrt(mean((testing_data$Outstate - y_pred_gam)^2))

test_rmse_gam
```

```
## [1] 1514.57
```

The test error of this GAM model is `1514.57`.

Question (d): Preferred model for predicting out-of-state tuition

First, I will fit a linear model using the training data, and compute the test error (RMSE) for comparison.

```
# Set seed for reproducibility
set.seed(299)

# Fit Linear model using training data
lm.fit <- train(Outstate ~ .,
                data = training_data,
                method = "lm",
                trControl = ctrl1)

# Predict on training data
y_pred_lm_train <- predict(lm.fit, newdata = training_data)

# training RMSE
train_rmse_lm <- sqrt(mean((training_data$Outstate - y_pred_lm_train)^2))
train_rmse_lm
```

```
## [1] 1946.786
```

```
# Predict on training data using MARS model
y_pred_mars_train <- predict(mars.fit, newdata = training_data)

# Compute Training RMSE for MARS Model
train_rmse_mars <- sqrt(mean((training_data$Outstate - y_pred_mars_train)^2))

# Print Training RMSE
print(train_rmse_mars)
```

```
## [1] 1645.373
```

Based on the results above computing training data RMSE, I would favor a MARS model over a linear model for predicting out of state tuition in this dataset. The MARS model performs better at predicting out of state tuition with a **training RMSE of 1645.37** compared to **1946.79 for the linear model**.

Unlike the linear model, the MARS model can represent significant non-linear relationships, for example with Room.Board, where there is a steep increase in tuition up to approximately $4,200 followed by a more gradual effect. As discussed, the partial dependence plots clearly show several non-linear terms are significant, which is an important consideration when choosing an appropriate model.

The MARS model's automatic feature selection identifies the most important predictors and their optimal cut points, creating a model that shows specific thresholds where relationships between predictors and tuition change. This approach works well for data like this case, where relationships often change at certain thresholds rather than following strictly linear patterns. Given the presence of nonlinearity in variables such as Expend, F.Undergrad, and Room.Board, the MARS model is better suited for this dataset than a linear model.

Personally, I wouldn't consider a MARS model to be universally superior to a linear model in all cases - it would really depend on the data in question (including the relationship between predictors and the outcome), as well as the research objectives. For instance, linear models would be most appropriate when the true relationships are linear/close to linear, and when interpretability is a key consideration. Linear models may also be preferred if there is limited data, i.e., to avoid overfitting the data, which may be the

case with MARS in smaller datasets. One of the concepts we discussed in class was parsimony; that is, we should aim to choose the simplest appropriate model. If a linear model represents the relationships in the data well, and is sufficiently accurate in prediction, it would not be preferable to select a more complex MARS model. That being said, I would note that MARS models are better for cases where there are clear non-linear relationships (such as in this college dataset), and also where the dataset is sizeable enough to reliably identify complex patterns. MARS can also manage interactions between variables. Therefore, I would argue that in general applications the choice of MARS or a linear model should be guided by the nature of the data and aims of the intended research analysis.