Data Science II, Homework 2

Will Simmons

16 March 2020

```r
shh <- suppressMessages

shh(library(tidyverse))
shh(library(readr))
shh(library(splines))
shh(library(caret))
shh(library(mgcv))
```

Prompt

In this exercise, we build nonlinear models using the `College` data. The dataset contains statistics for 565 US Colleges from the 1995 issue of US News and World Report.

The response variable is the out-of-state tuition (Outstate). The predictors are

- Apps: Number of applications received
- Accept: Number of applications accepted
- Enroll: Number of new students enrolled
- Top10perc: Pct. new students from top 10
- Top25perc: Pct. new students from top 25
- F.Undergrad: Number of fulltime undergraduates
- P.Undergrad: Number of parttime undergraduates
- Room.Board: Room and board costs
- Books: Estimated book costs
- Personal: Estimated personal spending
- PhD: Pct. of faculty with Ph.D.'s
- Terminal: Pct. of faculty with terminal degree
- S.F.Ratio: Student/faculty ratio
- perc.alumni: Pct. alumni who donate
- Expend: Instructional expenditure per student
- Grad.Rate: Graduation rate

In what follows, use the data excluding statistics for Columbia University (i.e., the 125th observation) to train the models.

First, I'll import the data.

```r
data =
  read_csv('./data/College.csv')
```

```
## Parsed with column specification:
## cols(
##   College = col_character(),
##   Apps = col_double(),
##   Accept = col_double(),
##   Enroll = col_double(),
##   Top10perc = col_double(),
##   Top25perc = col_double(),
##   F.Undergrad = col_double(),
##   P.Undergrad = col_double(),
##   Outstate = col_double(),
##   Room.Board = col_double(),
##   Books = col_double(),
##   Personal = col_double(),
##   PhD = col_double(),
##   Terminal = col_double(),
##   S.F.Ratio = col_double(),
##   perc.alumni = col_double(),
##   Expend = col_double(),
##   Grad.Rate = col_double()
## )
```

```
train =
  data[-125,]

test =
  data[125,]
```

A. Create scatter plots of response vs. predictors.

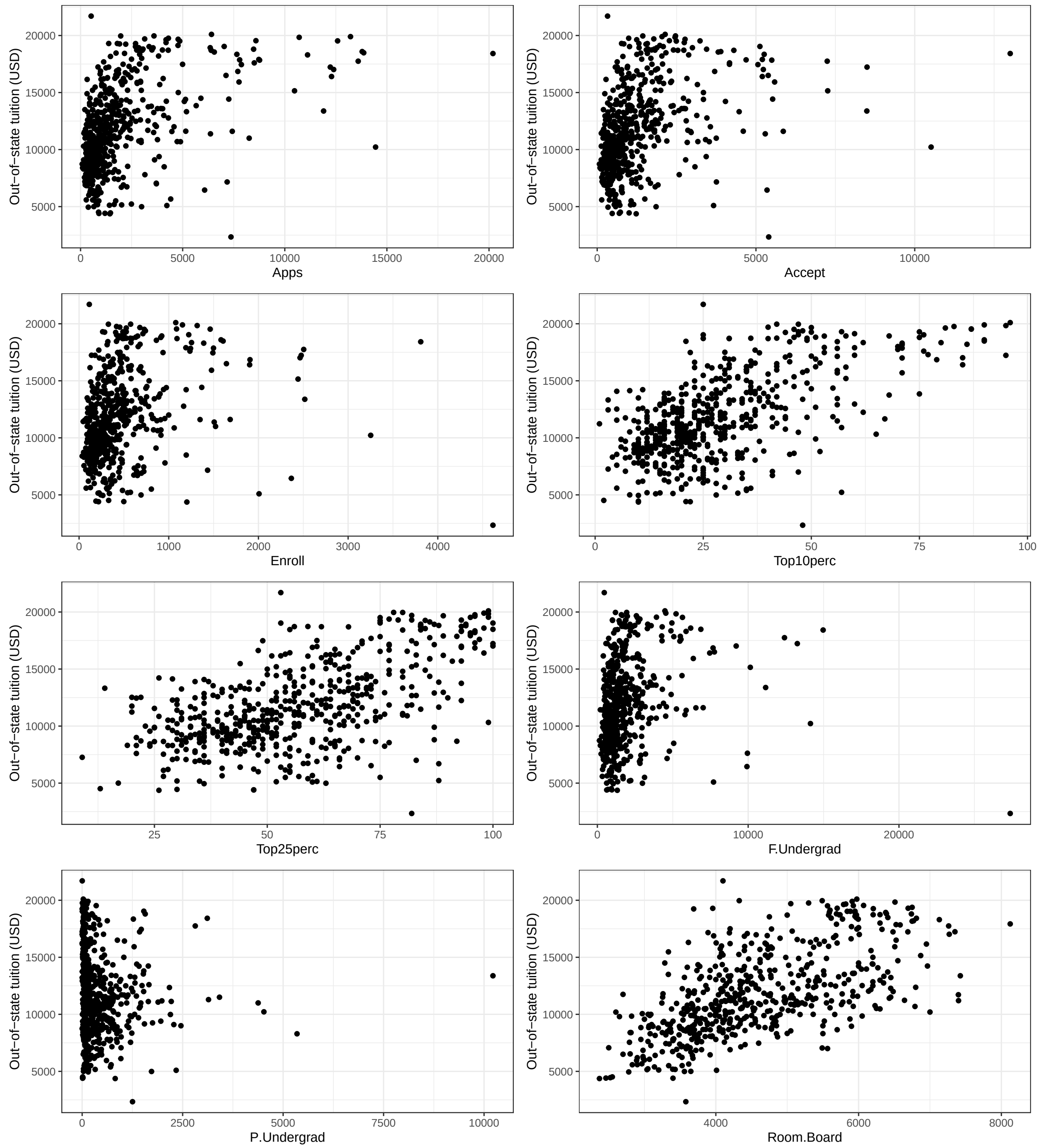
I'll create a function to plot the list of predictors, then use `purrr:map()` to iterate.

```
set.seed(1)

# Create function to map
plot_predictors = function(predictor) {

  train %>%
  # ggplot(aes(x = .data[[predictor]], y = Outstate)) +
  ggplot(aes_string(x = predictor, y = 'Outstate')) +
  geom_point() +
  # geom_smooth(method = 'loess',
              # color = 'IndianRed') +
  theme_bw() +
  labs(
    y = "Out-of-state tuition (USD)"
  )

}

# Create list of predictors, using nsmes() and set_names()
predictors = names(train)[-c(1,9)] %>%
  set_names()  # This purrr function helps with mapping lists
```

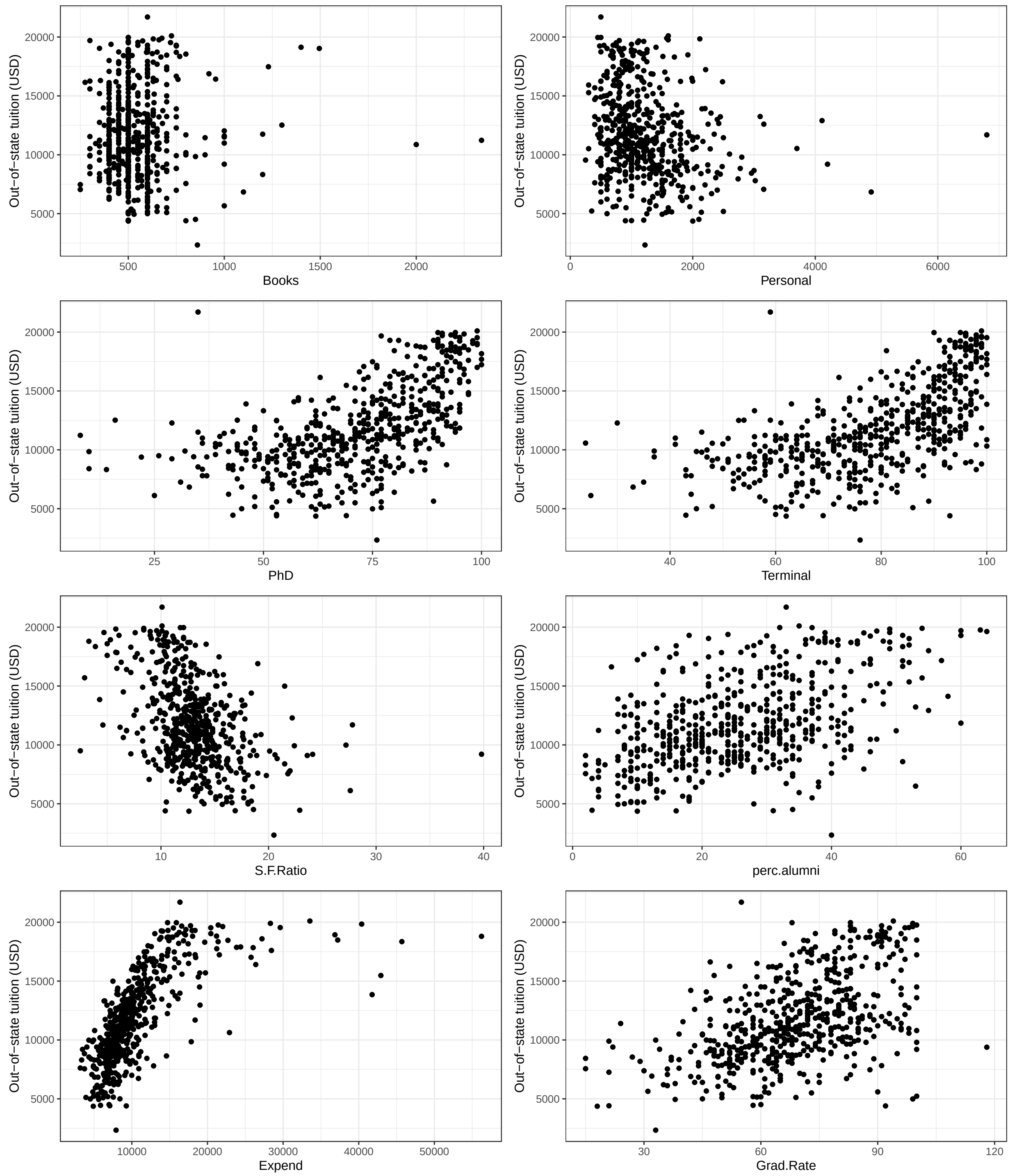
```r
# Iterate over function and list of predictors
all_plots =
  map(.x = predictors, ~plot_predictors(.x))
```

```r
# First half of the figure
cowplot::plot_grid(plotlist = all_plots[1:8],
                   ncol = 2) %>%
  patchwork::wrap_elements() +
  ggtitle("Figure 1. Outcome (out-of-state tuition) vs. exposure scatterplots\n") +
  theme(plot.title = element_text(hjust = .5,
                                  size = 20))
```

Figure 1. Outcome (out−of−state tuition) vs. exposure scatterplots




```r
# Second half (without title - will go onto another page)
cowplot::plot_grid(plotlist = all_plots[9:16],
                   ncol = 2) %>%
  patchwork::wrap_elements()
```



B1. Fit a smoothing spline model using Terminal as the only predictor of Outstate for a range of degrees of freedom, and plot the resulting fits. Describe the results obtained.

To illustrate fit across a range of different degrees of freedom, I will:

- Select 10 random integer DF values between the minimum DF value for this predictor-outcome pair (2) and the maximum (65)

- Plot each fit over the predictor-outcome scatter separately

```r
set.seed(1)

smooth_models = function(deg) {

  smooth.spline(x = train$Terminal,
                y = train$Outstate,
                df = deg)

}

# Creating list of random DF values between min (2) and max (65)
set.seed(1)
dfs = sample(2:65, 10) %>% sort()

# Fitting 10 models with DFs
random_df_models =
  map(.x = dfs, ~smooth_models(.x))

# Creating x-value prediction grid we can use to plot our predicted nonlinear model
terminal_lims = range(train$Terminal)
terminal_grid = seq(from = terminal_lims[1], to = terminal_lims[2])

# Predicting values and plotting for each DF
smooth_plots = function(model) {

  smooth_predict = predict(random_df_models[[model]],
                           x = terminal_grid)

  smooth_predict_df = tibble(pred = smooth_predict$y,
                             Terminal = terminal_grid)

  all_plots[[12]] +
  geom_line(aes(x = Terminal,
                y = pred),
            data = smooth_predict_df,
            color = 'IndianRed',
            size = 0.8) +
  theme_bw() +
  labs(title = paste0("DF = ", dfs[[model]], " (Equiv. DF = ", random_df_models[[model]]$df %>% round(4
       x = "",
       y = "") +
  theme(plot.title = element_text(hjust = 0.5))

}

# Map over different DF values
DF_smooth_plots =
  map(.x = 1:10, ~smooth_plots(.x))       # 1:10 because I know I fitted 10 DFs
```

Here, we see that as the chosen DF value increases, model flexibility also increases.

```r
DF_smooth_plots[[5]] = DF_smooth_plots[[5]] + labs(y = "Out-of-state tuition (USD)")
DF_smooth_plots[[9]] = DF_smooth_plots[[9]] + labs(x = "% faculty with terminal degree")
DF_smooth_plots[[10]] = DF_smooth_plots[[10]] + labs(x = "% faculty with terminal degree")


# Figure 2
cowplot::plot_grid(plotlist = DF_smooth_plots,
                   ncol = 2) %>%
  patchwork::wrap_elements() +
  ggtitle("Figure 2. Outcome (out-of-state tuition) vs. percent faculty with terminal degree") +
  labs(subtitle = "Fit with smoothing splines using 10 random DF values") +
  theme(plot.title = element_text(hjust = .5,
                                  size = 16),
        plot.subtitle = element_text(hjust = 0.5,
                                     size = 12))
```

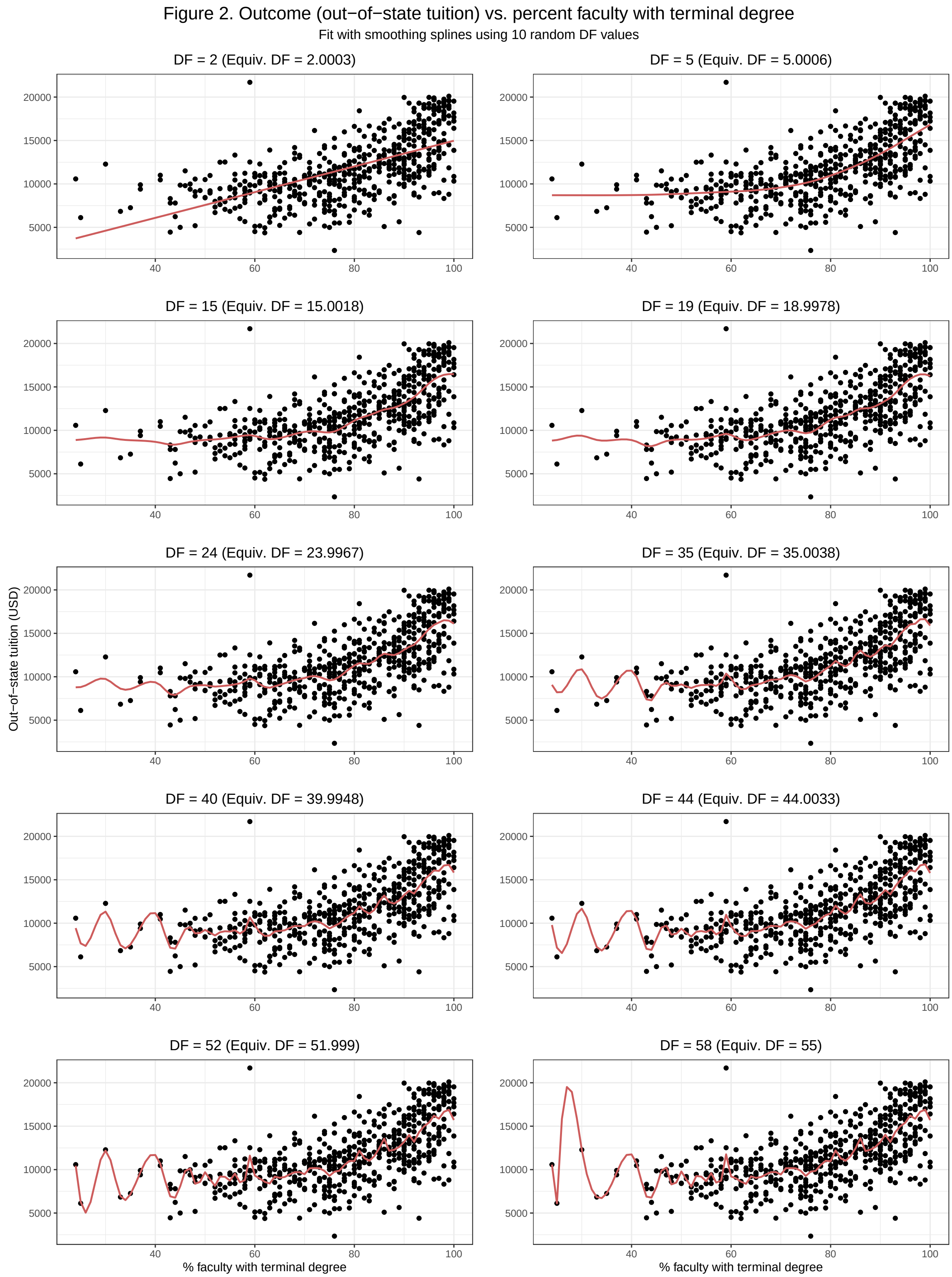


Figure 2. Outcome (out−of−state tuition) vs. percent faculty with terminal degree
Fit with smoothing splines using 10 random DF values
DF = 2 (Equiv. DF = 2.0003)
DF = 5 (Equiv. DF = 5.0006)
DF = 15 (Equiv. DF = 15.0018)
DF = 19 (Equiv. DF = 18.9978)
DF = 24 (Equiv. DF = 23.9967)
DF = 35 (Equiv. DF = 35.0038)
DF = 40 (Equiv. DF = 39.9948)
DF = 44 (Equiv. DF = 44.0033)
DF = 52 (Equiv. DF = 51.999)
DF = 58 (Equiv. DF = 55)
Out−of−state tuition (USD)
% faculty with terminal degree
20000
15000
10000
5000
40
60
80
100

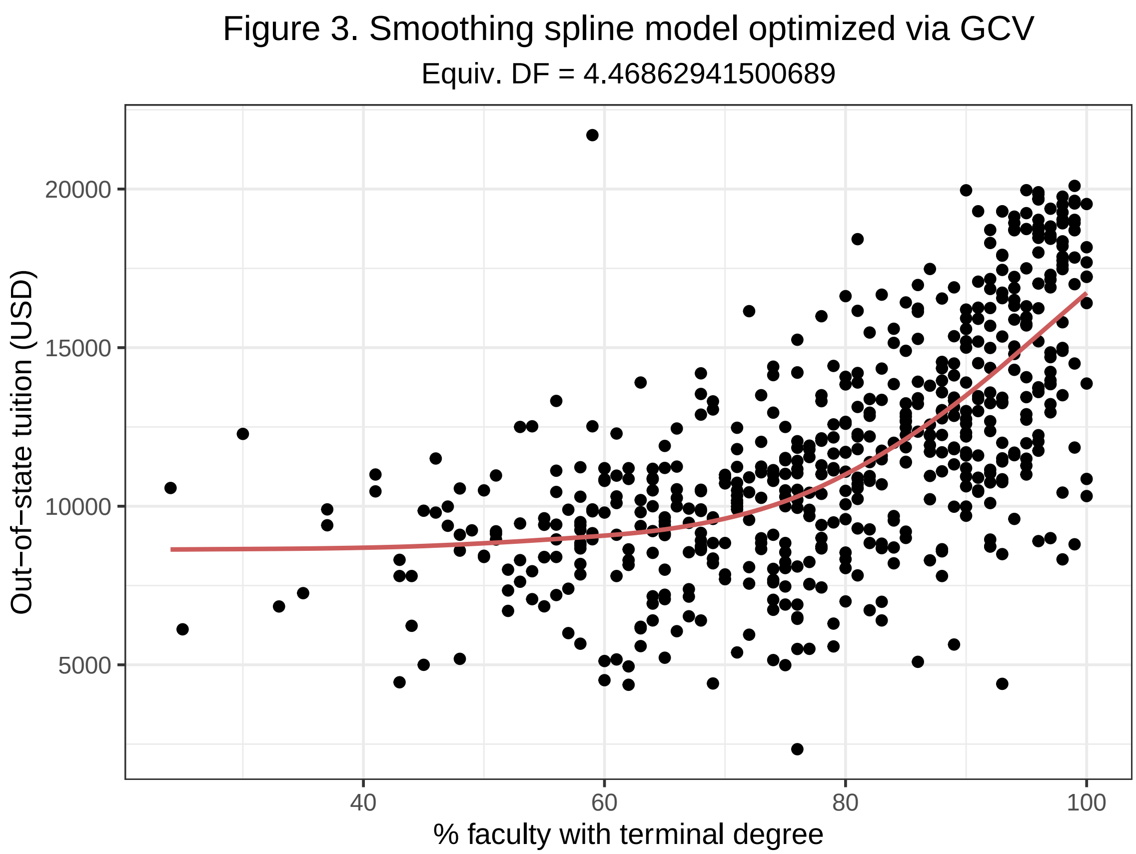
B2. Fit a smoothing spline model using Terminal as the only predictor of Outstate for the degree of freedom obtained by generalized cross-validation, and plot the resulting fits. Describe the results obtained.

Here, the `smooth.spline()` function automatically chooses DF using GCV.

```
set.seed(1)

# Fitting smooth spline model automatically (via GCV in-built in function)
smooth_model_gcv =
  smooth.spline(x = train$Terminal,
                y = train$Outstate)

# Predicted values given grid of X values bounded by minimum and maximum x values (terminal_grid)
smooth_gcv_predict = predict(smooth_model_gcv,
                             x = terminal_grid)

# Creating tibble to use in plot - will help plot curve
smooth_gcv_predict_df = tibble(pred = smooth_gcv_predict$y,
                               Terminal = terminal_grid)
```

In the model with DF optimized via generalized cross-validation, we can see that there is a good balance between model flexibility (variance) and bias.

```
# Plotting against all_plots[[12]] - raw scatter for Terminal vs. Outstate
all_plots[[12]] +
geom_line(aes(x = Terminal,
              y = pred),
          data = smooth_gcv_predict_df,
          color = 'IndianRed',
          size = 0.8) +
theme_bw() +
labs(title = "Figure 3. Smoothing spline model optimized via GCV",
     subtitle = paste0("Equiv. DF = ", smooth_model_gcv$df),
     x = "% faculty with terminal degree",
     y = "Out-of-state tuition (USD)") +
theme(plot.title = element_text(hjust = 0.5),
      plot.subtitle = element_text(hjust = 0.5))
```



C. Fit a generalized additive model (GAM) using all the predictors. Plot the results and explain your findings.

Here, I'll fit a GAM model with `caret::train(method = "gam")`, using repeated CV to tune the model hyperparameters.

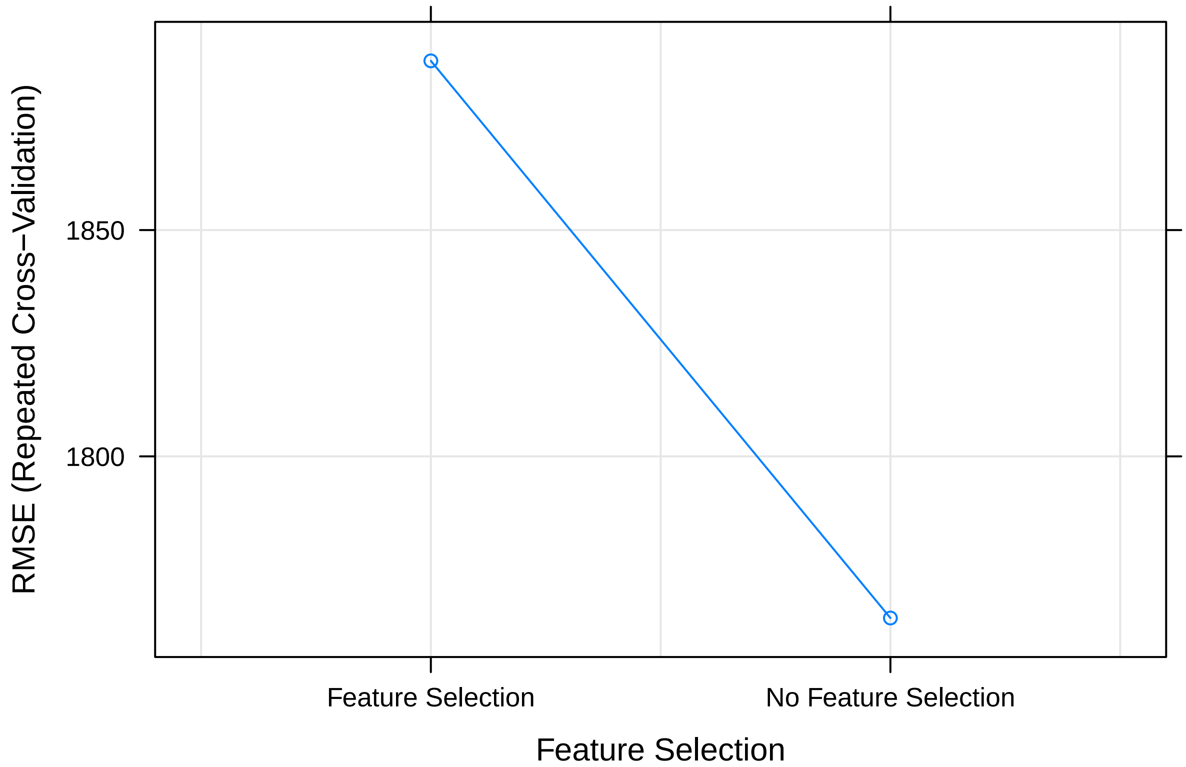
```
set.seed(1)

gam_tr_control = trainControl(method = "repeatedcv", number = 10, repeats = 2,
                              verboseIter = TRUE)

full_formula =
  reformulate(predictors, "Outstate")

gam_fit =
  train(full_formula,
        data = train,
        method = "gam",
        tuneGrid = data.frame(method = "GCV.Cp", select = c(TRUE, FALSE)),
        trControl = gam_tr_control)
```

```
## + Fold01.Rep1: method=GCV.Cp, select= TRUE
## - Fold01.Rep1: method=GCV.Cp, select= TRUE
## + Fold01.Rep1: method=GCV.Cp, select=FALSE
## - Fold01.Rep1: method=GCV.Cp, select=FALSE
```

```
## + Fold02.Rep1: method=GCV.Cp, select= TRUE
## - Fold02.Rep1: method=GCV.Cp, select= TRUE
## + Fold02.Rep1: method=GCV.Cp, select=FALSE
## - Fold02.Rep1: method=GCV.Cp, select=FALSE
## + Fold03.Rep1: method=GCV.Cp, select= TRUE
## - Fold03.Rep1: method=GCV.Cp, select= TRUE
## + Fold03.Rep1: method=GCV.Cp, select=FALSE
## - Fold03.Rep1: method=GCV.Cp, select=FALSE
## + Fold04.Rep1: method=GCV.Cp, select= TRUE
## - Fold04.Rep1: method=GCV.Cp, select= TRUE
## + Fold04.Rep1: method=GCV.Cp, select=FALSE
## - Fold04.Rep1: method=GCV.Cp, select=FALSE
## + Fold05.Rep1: method=GCV.Cp, select= TRUE
## - Fold05.Rep1: method=GCV.Cp, select= TRUE
## + Fold05.Rep1: method=GCV.Cp, select=FALSE
## - Fold05.Rep1: method=GCV.Cp, select=FALSE
## + Fold06.Rep1: method=GCV.Cp, select= TRUE
## - Fold06.Rep1: method=GCV.Cp, select= TRUE
## + Fold06.Rep1: method=GCV.Cp, select=FALSE
## - Fold06.Rep1: method=GCV.Cp, select=FALSE
## + Fold07.Rep1: method=GCV.Cp, select= TRUE
## - Fold07.Rep1: method=GCV.Cp, select= TRUE
## + Fold07.Rep1: method=GCV.Cp, select=FALSE
## - Fold07.Rep1: method=GCV.Cp, select=FALSE
## + Fold08.Rep1: method=GCV.Cp, select= TRUE
## - Fold08.Rep1: method=GCV.Cp, select= TRUE
## + Fold08.Rep1: method=GCV.Cp, select=FALSE
## - Fold08.Rep1: method=GCV.Cp, select=FALSE
## + Fold09.Rep1: method=GCV.Cp, select= TRUE
## - Fold09.Rep1: method=GCV.Cp, select= TRUE
## + Fold09.Rep1: method=GCV.Cp, select=FALSE
## - Fold09.Rep1: method=GCV.Cp, select=FALSE
## + Fold10.Rep1: method=GCV.Cp, select= TRUE
## - Fold10.Rep1: method=GCV.Cp, select= TRUE
## + Fold10.Rep1: method=GCV.Cp, select=FALSE
## - Fold10.Rep1: method=GCV.Cp, select=FALSE
## + Fold01.Rep2: method=GCV.Cp, select= TRUE
## - Fold01.Rep2: method=GCV.Cp, select= TRUE
## + Fold01.Rep2: method=GCV.Cp, select=FALSE
## - Fold01.Rep2: method=GCV.Cp, select=FALSE
## + Fold02.Rep2: method=GCV.Cp, select= TRUE
## - Fold02.Rep2: method=GCV.Cp, select= TRUE
## + Fold02.Rep2: method=GCV.Cp, select=FALSE
## - Fold02.Rep2: method=GCV.Cp, select=FALSE
## + Fold03.Rep2: method=GCV.Cp, select= TRUE
## - Fold03.Rep2: method=GCV.Cp, select= TRUE
## + Fold03.Rep2: method=GCV.Cp, select=FALSE
## - Fold03.Rep2: method=GCV.Cp, select=FALSE
## + Fold04.Rep2: method=GCV.Cp, select= TRUE
## - Fold04.Rep2: method=GCV.Cp, select= TRUE
## + Fold04.Rep2: method=GCV.Cp, select=FALSE
## - Fold04.Rep2: method=GCV.Cp, select=FALSE
## + Fold05.Rep2: method=GCV.Cp, select= TRUE
## - Fold05.Rep2: method=GCV.Cp, select= TRUE
```

```
## + Fold05.Rep2: method=GCV.Cp, select=FALSE
## - Fold05.Rep2: method=GCV.Cp, select=FALSE
## + Fold06.Rep2: method=GCV.Cp, select= TRUE
## - Fold06.Rep2: method=GCV.Cp, select= TRUE
## + Fold06.Rep2: method=GCV.Cp, select=FALSE
## - Fold06.Rep2: method=GCV.Cp, select=FALSE
## + Fold07.Rep2: method=GCV.Cp, select= TRUE
## - Fold07.Rep2: method=GCV.Cp, select= TRUE
## + Fold07.Rep2: method=GCV.Cp, select=FALSE
## - Fold07.Rep2: method=GCV.Cp, select=FALSE
## + Fold08.Rep2: method=GCV.Cp, select= TRUE
## - Fold08.Rep2: method=GCV.Cp, select= TRUE
## + Fold08.Rep2: method=GCV.Cp, select=FALSE
## - Fold08.Rep2: method=GCV.Cp, select=FALSE
## + Fold09.Rep2: method=GCV.Cp, select= TRUE
## - Fold09.Rep2: method=GCV.Cp, select= TRUE
## + Fold09.Rep2: method=GCV.Cp, select=FALSE
## - Fold09.Rep2: method=GCV.Cp, select=FALSE
## + Fold10.Rep2: method=GCV.Cp, select= TRUE
## - Fold10.Rep2: method=GCV.Cp, select= TRUE
## + Fold10.Rep2: method=GCV.Cp, select=FALSE
## - Fold10.Rep2: method=GCV.Cp, select=FALSE
## Aggregating results
## Selecting tuning parameters
## Fitting select = FALSE, method = GCV.Cp on full training set
```

```r
plot(gam_fit)
```




```
# gam_fit$finalModel$fitted.values

# Email Angel with gam() question above
  # Which method?
  # How to plot?
  # What predictor to plot?
```

D. Fit a multivariate adaptive regression spline (MARS) model using all the predictors. Report the final model. Present the partial dependence plot of an arbitrary predictor in your final model.

```
shh(library(pdp))
shh(library(earth))

set.seed(1)

mars_grid = expand.grid(degree = 1:3,
                         nprune = 2:34)

mars_control = trainControl(method = "repeatedcv", number = 10, repeats = 2,
                            verboseIter = TRUE)

mars_model = train(full_formula,
                   data = train,
                   method = "earth",
```

```
                    tuneGrid = mars_grid,
                    trControl = mars_control)
```

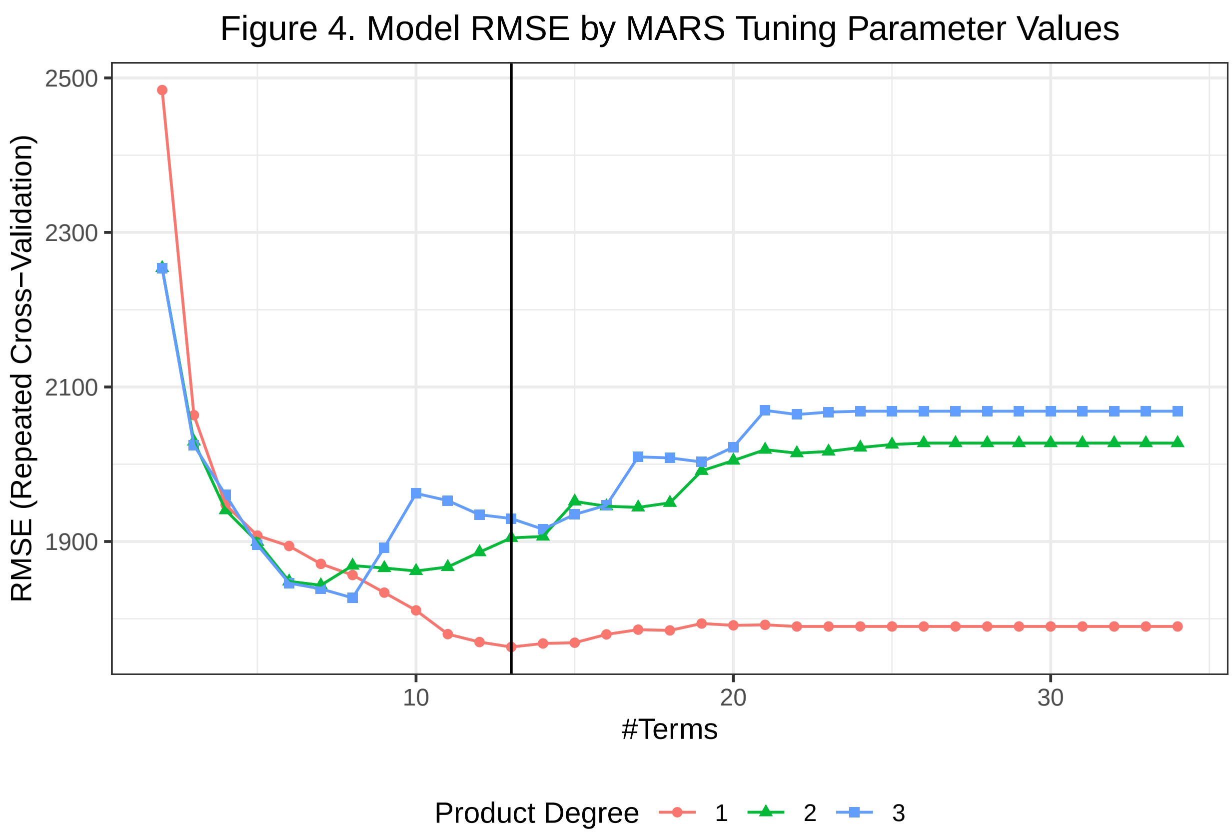
```
## + Fold01.Rep1: degree=1, nprune=34
## - Fold01.Rep1: degree=1, nprune=34
## + Fold01.Rep1: degree=2, nprune=34
## - Fold01.Rep1: degree=2, nprune=34
## + Fold01.Rep1: degree=3, nprune=34
## - Fold01.Rep1: degree=3, nprune=34
## + Fold02.Rep1: degree=1, nprune=34
## - Fold02.Rep1: degree=1, nprune=34
## + Fold02.Rep1: degree=2, nprune=34
## - Fold02.Rep1: degree=2, nprune=34
## + Fold02.Rep1: degree=3, nprune=34
## - Fold02.Rep1: degree=3, nprune=34
## + Fold03.Rep1: degree=1, nprune=34
## - Fold03.Rep1: degree=1, nprune=34
## + Fold03.Rep1: degree=2, nprune=34
## - Fold03.Rep1: degree=2, nprune=34
## + Fold03.Rep1: degree=3, nprune=34
## - Fold03.Rep1: degree=3, nprune=34
## + Fold04.Rep1: degree=1, nprune=34
## - Fold04.Rep1: degree=1, nprune=34
## + Fold04.Rep1: degree=2, nprune=34
## - Fold04.Rep1: degree=2, nprune=34
## + Fold04.Rep1: degree=3, nprune=34
## - Fold04.Rep1: degree=3, nprune=34
## + Fold05.Rep1: degree=1, nprune=34
## - Fold05.Rep1: degree=1, nprune=34
## + Fold05.Rep1: degree=2, nprune=34
## - Fold05.Rep1: degree=2, nprune=34
## + Fold05.Rep1: degree=3, nprune=34
## - Fold05.Rep1: degree=3, nprune=34
## + Fold06.Rep1: degree=1, nprune=34
## - Fold06.Rep1: degree=1, nprune=34
## + Fold06.Rep1: degree=2, nprune=34
## - Fold06.Rep1: degree=2, nprune=34
## + Fold06.Rep1: degree=3, nprune=34
## - Fold06.Rep1: degree=3, nprune=34
## + Fold07.Rep1: degree=1, nprune=34
## - Fold07.Rep1: degree=1, nprune=34
## + Fold07.Rep1: degree=2, nprune=34
## - Fold07.Rep1: degree=2, nprune=34
## + Fold07.Rep1: degree=3, nprune=34
## - Fold07.Rep1: degree=3, nprune=34
## + Fold08.Rep1: degree=1, nprune=34
## - Fold08.Rep1: degree=1, nprune=34
## + Fold08.Rep1: degree=2, nprune=34
## - Fold08.Rep1: degree=2, nprune=34
## + Fold08.Rep1: degree=3, nprune=34
## - Fold08.Rep1: degree=3, nprune=34
## + Fold09.Rep1: degree=1, nprune=34
## - Fold09.Rep1: degree=1, nprune=34
```

```
## + Fold09.Rep1: degree=2, nprune=34
## - Fold09.Rep1: degree=2, nprune=34
## + Fold09.Rep1: degree=3, nprune=34
## - Fold09.Rep1: degree=3, nprune=34
## + Fold10.Rep1: degree=1, nprune=34
## - Fold10.Rep1: degree=1, nprune=34
## + Fold10.Rep1: degree=2, nprune=34
## - Fold10.Rep1: degree=2, nprune=34
## + Fold10.Rep1: degree=3, nprune=34
## - Fold10.Rep1: degree=3, nprune=34
## + Fold01.Rep2: degree=1, nprune=34
## - Fold01.Rep2: degree=1, nprune=34
## + Fold01.Rep2: degree=2, nprune=34
## - Fold01.Rep2: degree=2, nprune=34
## + Fold01.Rep2: degree=3, nprune=34
## - Fold01.Rep2: degree=3, nprune=34
## + Fold02.Rep2: degree=1, nprune=34
## - Fold02.Rep2: degree=1, nprune=34
## + Fold02.Rep2: degree=2, nprune=34
## - Fold02.Rep2: degree=2, nprune=34
## + Fold02.Rep2: degree=3, nprune=34
## - Fold02.Rep2: degree=3, nprune=34
## + Fold03.Rep2: degree=1, nprune=34
## - Fold03.Rep2: degree=1, nprune=34
## + Fold03.Rep2: degree=2, nprune=34
## - Fold03.Rep2: degree=2, nprune=34
## + Fold03.Rep2: degree=3, nprune=34
## - Fold03.Rep2: degree=3, nprune=34
## + Fold04.Rep2: degree=1, nprune=34
## - Fold04.Rep2: degree=1, nprune=34
## + Fold04.Rep2: degree=2, nprune=34
## - Fold04.Rep2: degree=2, nprune=34
## + Fold04.Rep2: degree=3, nprune=34
## - Fold04.Rep2: degree=3, nprune=34
## + Fold05.Rep2: degree=1, nprune=34
## - Fold05.Rep2: degree=1, nprune=34
## + Fold05.Rep2: degree=2, nprune=34
## - Fold05.Rep2: degree=2, nprune=34
## + Fold05.Rep2: degree=3, nprune=34
## - Fold05.Rep2: degree=3, nprune=34
## + Fold06.Rep2: degree=1, nprune=34
## - Fold06.Rep2: degree=1, nprune=34
## + Fold06.Rep2: degree=2, nprune=34
## - Fold06.Rep2: degree=2, nprune=34
## + Fold06.Rep2: degree=3, nprune=34
## - Fold06.Rep2: degree=3, nprune=34
## + Fold07.Rep2: degree=1, nprune=34
## - Fold07.Rep2: degree=1, nprune=34
## + Fold07.Rep2: degree=2, nprune=34
## - Fold07.Rep2: degree=2, nprune=34
## + Fold07.Rep2: degree=3, nprune=34
## - Fold07.Rep2: degree=3, nprune=34
## + Fold08.Rep2: degree=1, nprune=34
## - Fold08.Rep2: degree=1, nprune=34
```

```
## + Fold08.Rep2: degree=2, nprune=34
## - Fold08.Rep2: degree=2, nprune=34
## + Fold08.Rep2: degree=3, nprune=34
## - Fold08.Rep2: degree=3, nprune=34
## + Fold09.Rep2: degree=1, nprune=34
## - Fold09.Rep2: degree=1, nprune=34
## + Fold09.Rep2: degree=2, nprune=34
## - Fold09.Rep2: degree=2, nprune=34
## + Fold09.Rep2: degree=3, nprune=34
## - Fold09.Rep2: degree=3, nprune=34
## + Fold10.Rep2: degree=1, nprune=34
## - Fold10.Rep2: degree=1, nprune=34
## + Fold10.Rep2: degree=2, nprune=34
## - Fold10.Rep2: degree=2, nprune=34
## + Fold10.Rep2: degree=3, nprune=34
## - Fold10.Rep2: degree=3, nprune=34
## Aggregating results
## Selecting tuning parameters
## Fitting nprune = 13, degree = 1 on full training set
```

As we can see, RMSE is minimized at `degree` =1 and `nprune` =13. The final model is presented below in a table of terms and estimated coefficients:

```r
ggplot(mars_model) +
  labs(title = "Figure 4. Model RMSE by MARS Tuning Parameter Values") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom") +
  geom_vline(xintercept = 13)
```




```r
tibble(
  variables =
    labels(mars_model$finalModel$coefficients)[[1]],
  coefficients =
    mars_model$finalModel$coefficients[,"y"]
  ) %>%
  knitr::kable(caption = "Table 1. Final MARS Model: Terms and Est. Coefficients") %>%
  kableExtra::kable_styling(full_width = F)
```

```
## Warning in kableExtra::kable_styling(., full_width = F): Please specify format
## in kable. kableExtra can customize either HTML or LaTeX outputs. See https://
## haozhu233.github.io/kableExtra/ for details.
```

Table 1: Table 1. Final MARS Model: Terms and Est. Coefficients

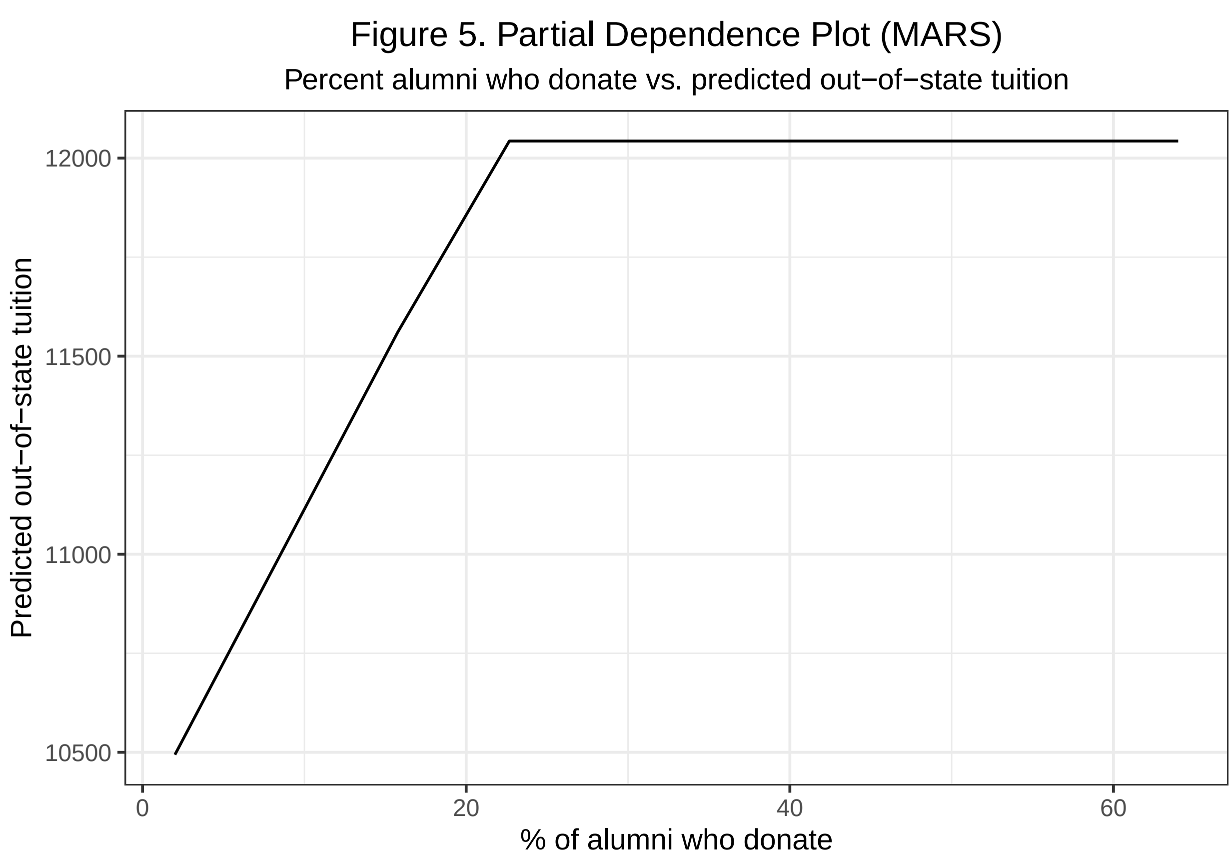
variables	coefficients
(Intercept)	11099.4030981
h(Expend-15365)	-0.7308623
h(4450-Room.Board)	-1.2860756
h(Grad.Rate-97)	-205.1619640
h(97-Grad.Rate)	-24.3424023
h(F.Undergrad-1355)	-0.3435602
h(1355-F.Undergrad)	-1.3670697
h(22-perc.alumni)	-77.4621692

variables	coefficients
h(Apps-3712)	0.4247957
h(1300-Personal)	0.9815494
h(913-Enroll)	4.6968931
h(2193-Accept)	-2.0136737
h(Expend-6881)	0.7344018

Finally, I'll plot a partial dependence plot of an arbitrary predictor, `Accept`. This plot shows that the marginal effect of the predictor `perc.alumni`, the percent of alumni who donate.

As we can see in this partial dependence plot, the average trend of percent of alumni who donate vs. tuition is positive - i.e. as the percent of alumni who donate increases, so does out-of-state tuition, on average.

```
partial(mars_model,
        pred.var = "perc.alumni",
        grid.resolution = 10) %>%
  ggplot(aes(x = perc.alumni, y = yhat)) +
  geom_line() +
  labs(title = "Figure 5. Partial Dependence Plot (MARS)",
       subtitle = "Percent alumni who donate vs. predicted out-of-state tuition",
       y = "Predicted out-of-state tuition",
       x = "% of alumni who donate") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))
```



E. Based on the above GAM and MARS models, predict the out-of-state tuition of Columbia University.

```
# Filter Columbia data to only predictors
test_pred =
  test %>%
  select(
    predictors
  )
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(predictors)` instead of `predictors` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
# Using GAM
gam_prediction =
  predict(gam_fit,
          newdata = test_pred)
# GAM PREDICTION: 17728.51

# Using MARS
mars_prediction =
  predict(mars_model,
          newdata = test_pred)
# MARS PREDICTION: 18144.31
```

The GAM model predicts out-of-state tuition for Columbia University to be \$$1.7728506 \times 10^4$, and the MARS model predicts \$$1.814431 \times 10^4$.