Data Science II, Homework 2

Will Simmons

16 March 2020

```r
shh <- suppressMessages

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

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,]
```

1. Create scatter plots of response vs. predictors.

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

```
# 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

# 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))
```

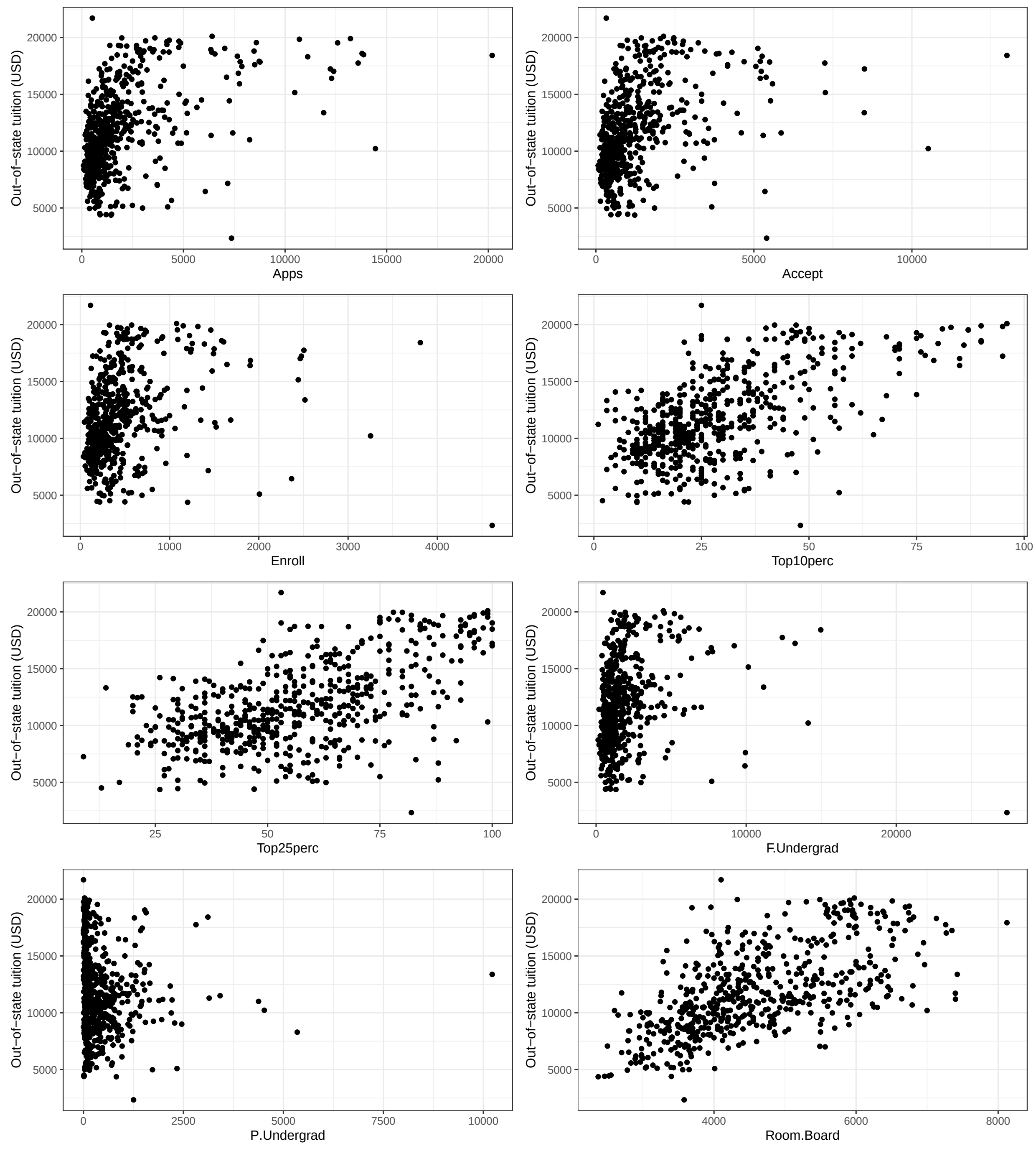
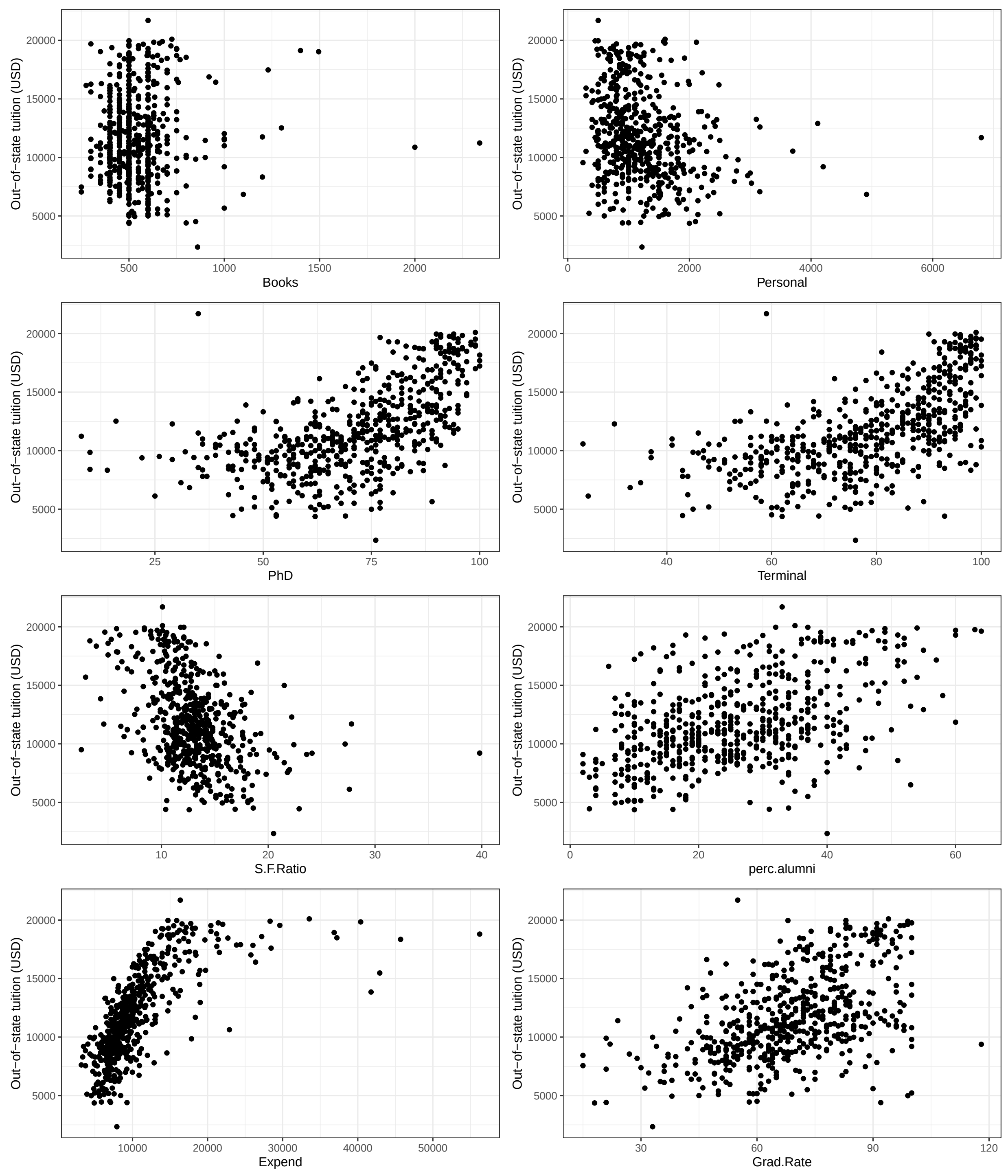


Figure 1: Outcome (out−of−state tuition) vs. exposure scatterplots

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



2a. 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.

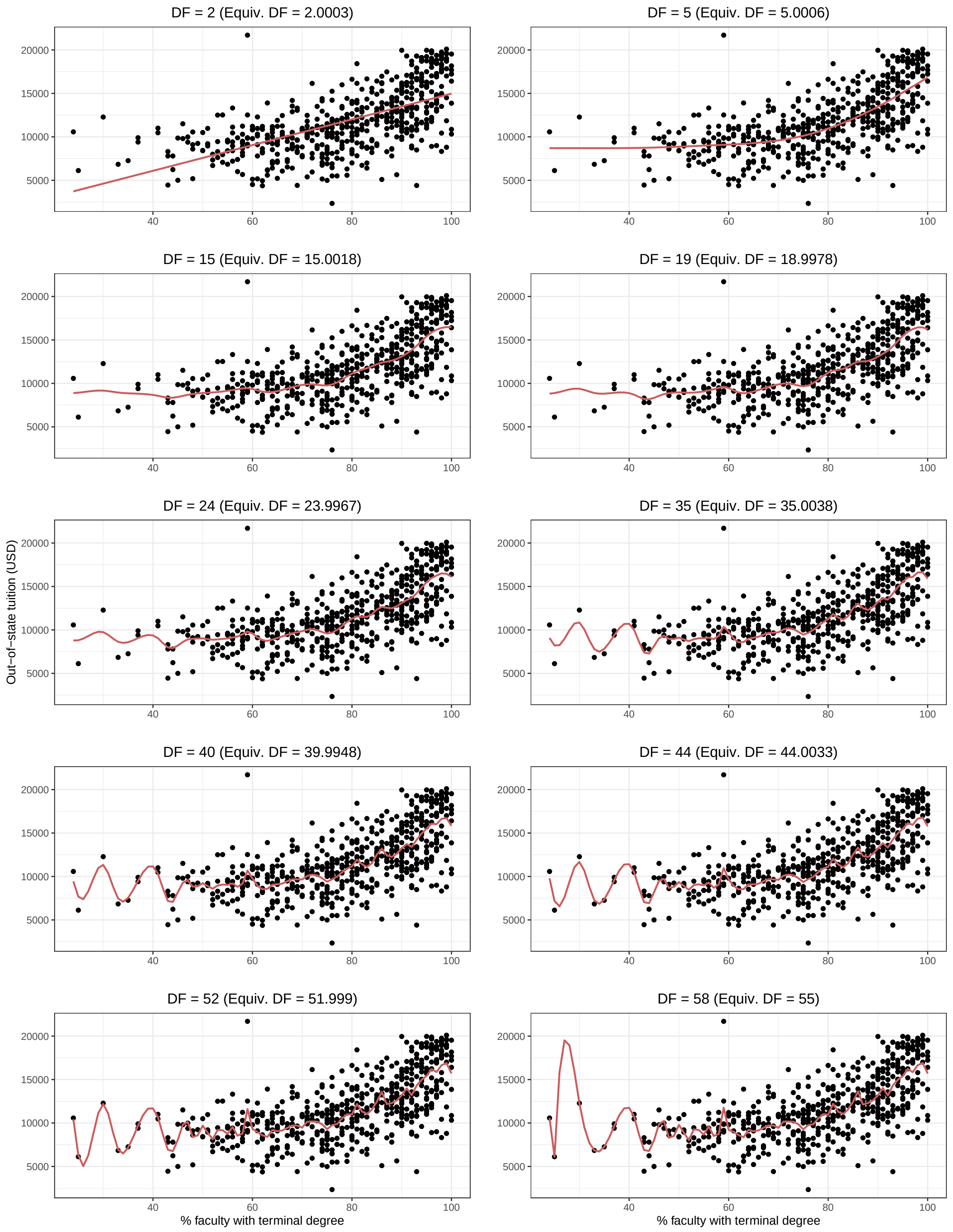
To illustrate fit across a range of different degrees of freedom, I will:

- 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
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
```

```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

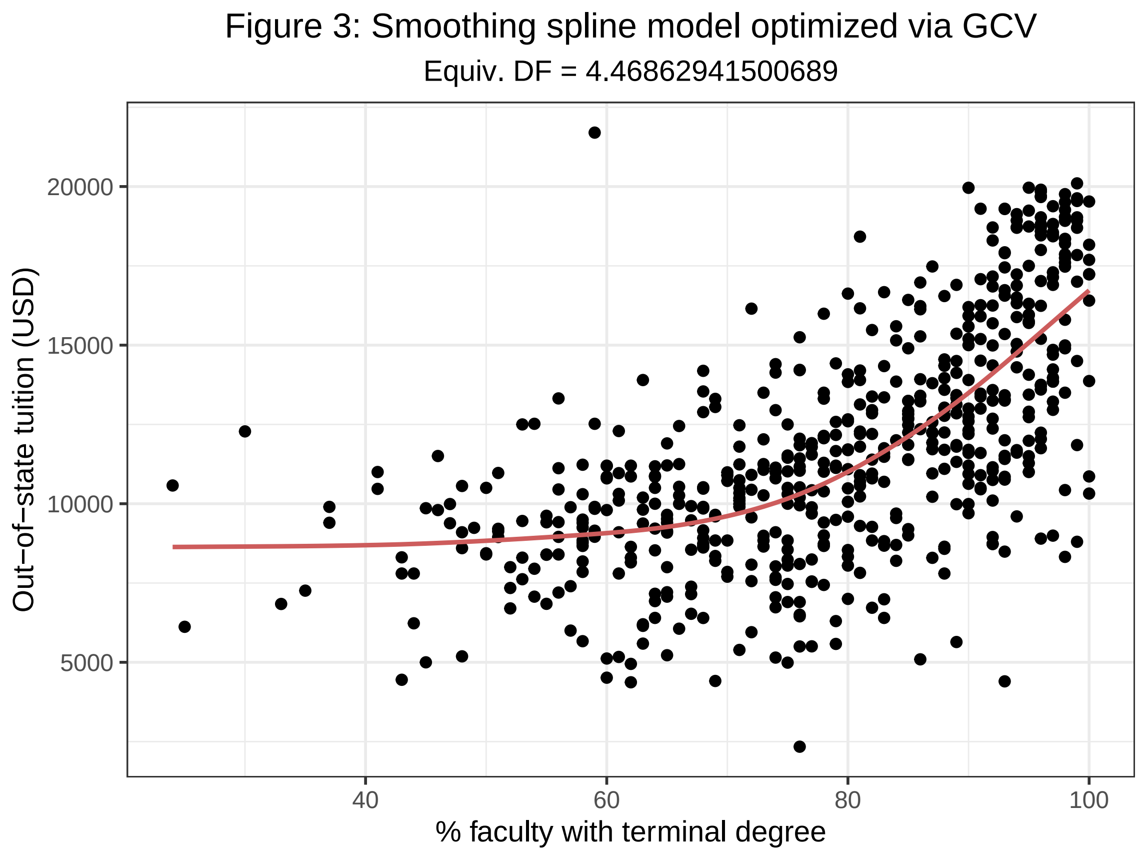


Here, we see that as the chosen DF value increased, model flexibility also increased.

2b. 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.

```
# 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)

# 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))
```



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.

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

4. 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.

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