p8106_hw2_wq2160

Wenshan Qu (wq2160)

3/5/2022

Contents

Data

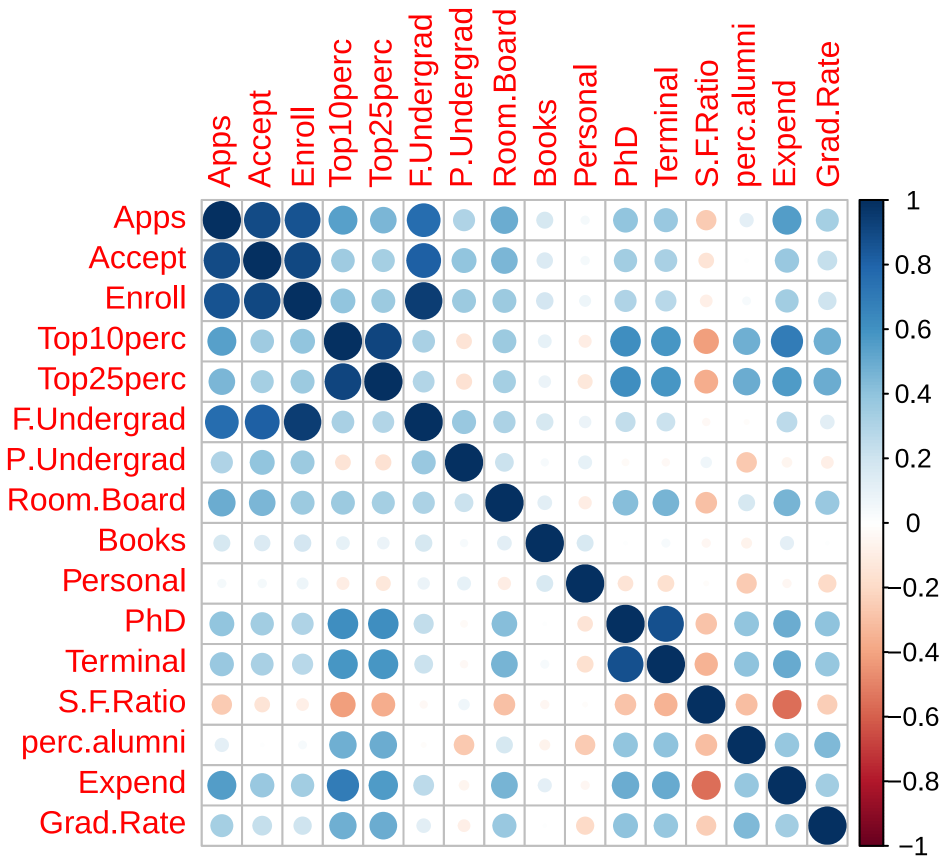
```r
college_df = read_csv("./College.csv")
college_df =
  college_df %>%
  na.omit() %>%
  select(-College)


college_df2 = model.matrix(Outstate ~ ., college_df)[ ,-1]

set.seed(33)
trainRows = createDataPartition(y = college_df$Outstate, p = 0.8, list = FALSE)

## Train Data
train_df = college_df[trainRows, ]
x = college_df2[trainRows,] ## Predictors Matrix
y = college_df$Outstate[trainRows] ## Response Vector

## Test Data
test_df = college_df[-trainRows, ]
x_test = model.matrix(Outstate ~ ., college_df)[-trainRows, -1]
y_test = college_df$Outstate[-trainRows]
```

Exploratory Data Analysis

(a) Perform exploratory data analysis using the training data (e.g., scatter plots of response vs. predictors).

Explore the correlation between predictors.

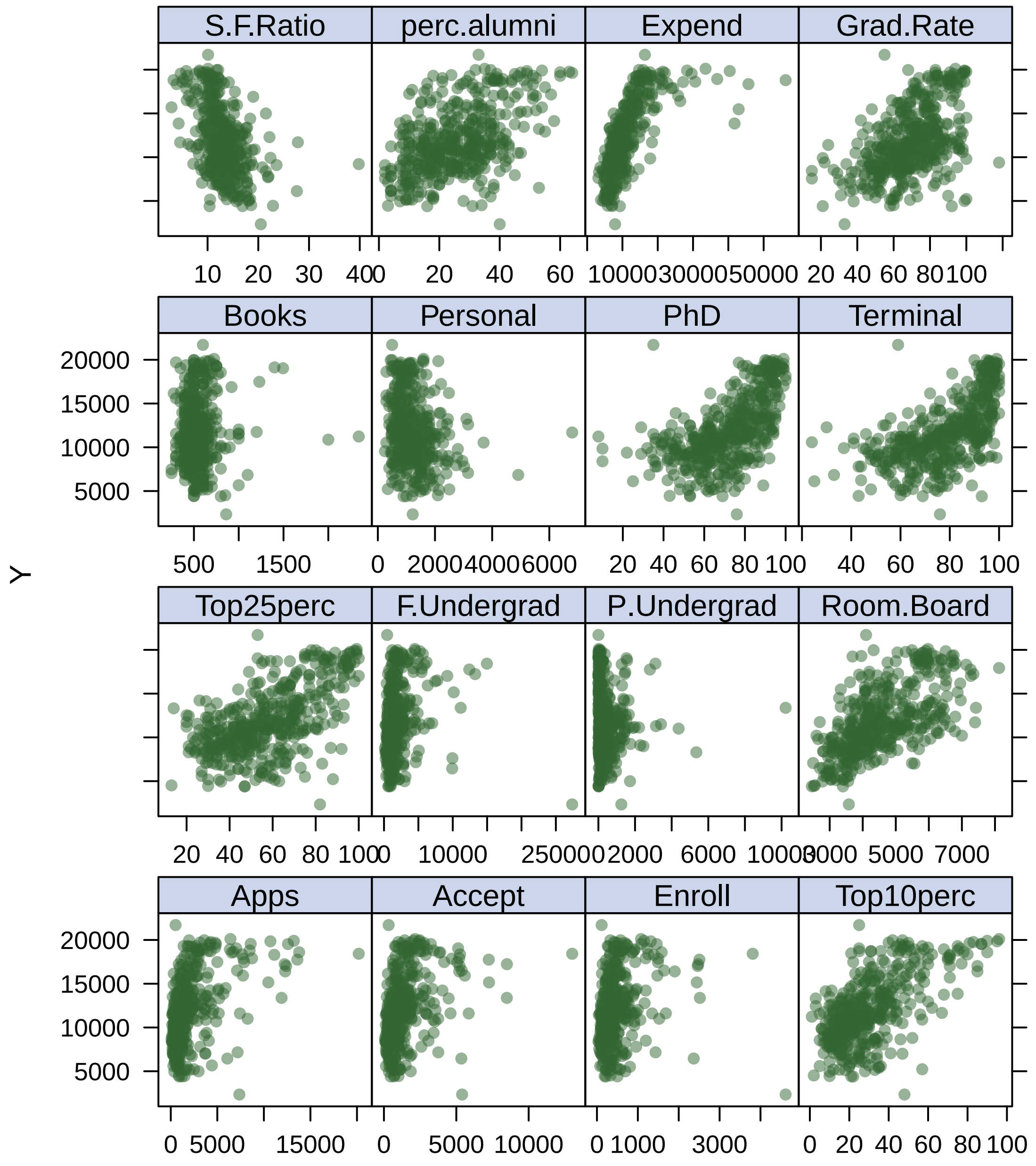
```r
corrplot(cor(x), method = "circle", type = "full")
```



Based on the result above, we can see that `Appps`, `Accept` and `Enroll` has relatively strong pairwise positive correlations, and strong correlations also exist between `Enroll` and `F.Undergrad` as well as `PhD` and `Terminal`.

Then, we use scatterplot to explore the relationship between the response out-of-state tuition `Outstate` and other variables.

```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)

## Feature Plot
featurePlot(x, y, plot = "scatter", labels = c("","Y"),
            type = c("p"), layout = c(4, 4))
```



Based on the feature plot, we can see that non-linear trend seems to appear in `Terminal`, `Top10perc`, `PhD`, `perc.alumni`, `Expend` and `Grad.Rate`, and linear trend seems to appear in `Top25perc`, `Room.Board`. While some other features, such as `Books`, `Personal` and `Apps`, have extremely large data points which make most of the data points in the plot cluster together to the left side so that the linear or non-linear trend cannot be simply observed.

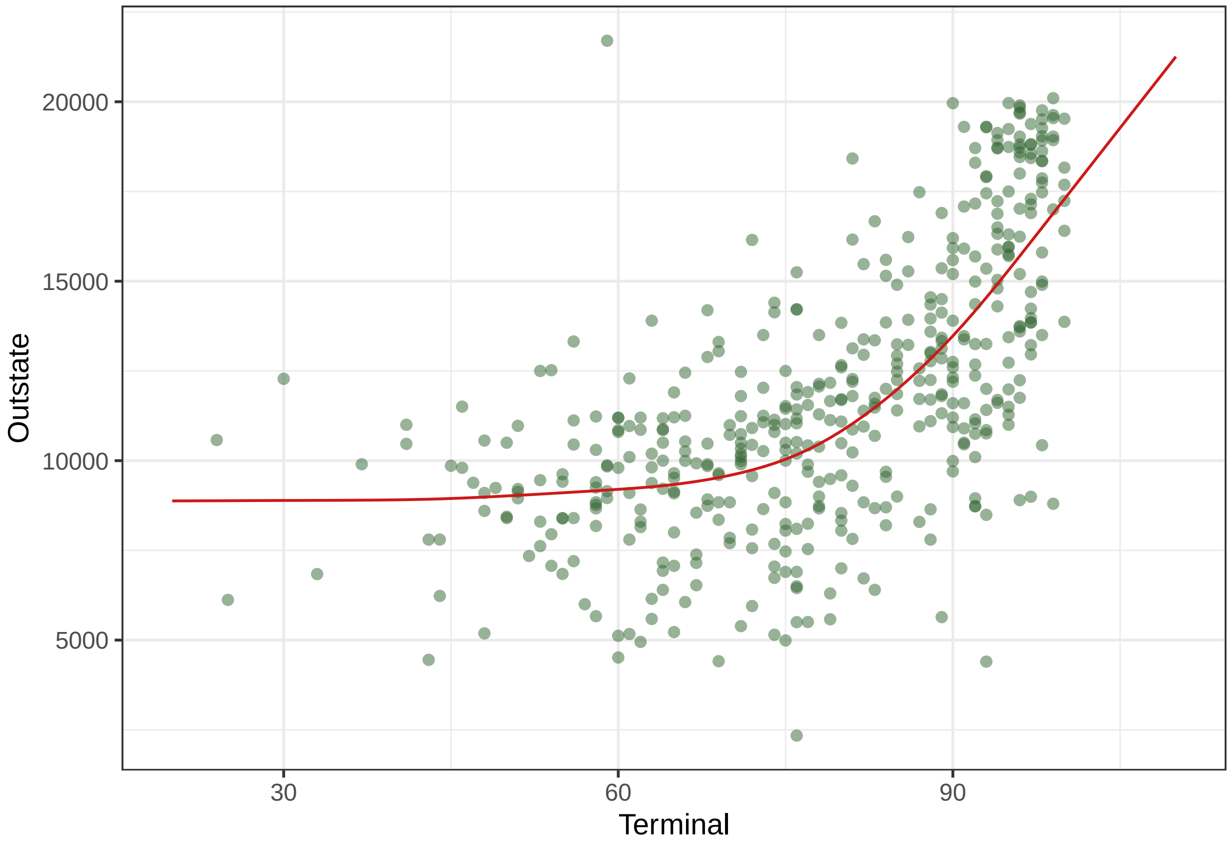
Smoothing Spline

(b) Fit smoothing spline models using `Terminal` as the only predictor of `Outstate` for a range of degrees of freedom, as well as the degree of freedom obtained by generalized cross-validation, and plot the resulting fits. Describe the results obtained.

```
terminal.grid = seq(from = 20, to = 110, by = 1) ## the range of `Terminal` is [24, 100]
p = ggplot(data = train_df, aes(x = Terminal, y = Outstate)) +
    geom_point(color = rgb(.2, .4, .2, .5))

## Obtain one optimized df by GCV (automatically optimized df)
set.seed(33)
fit.ss1 = smooth.spline(train_df$Terminal, train_df$Outstate)
fit.ss1$df
```

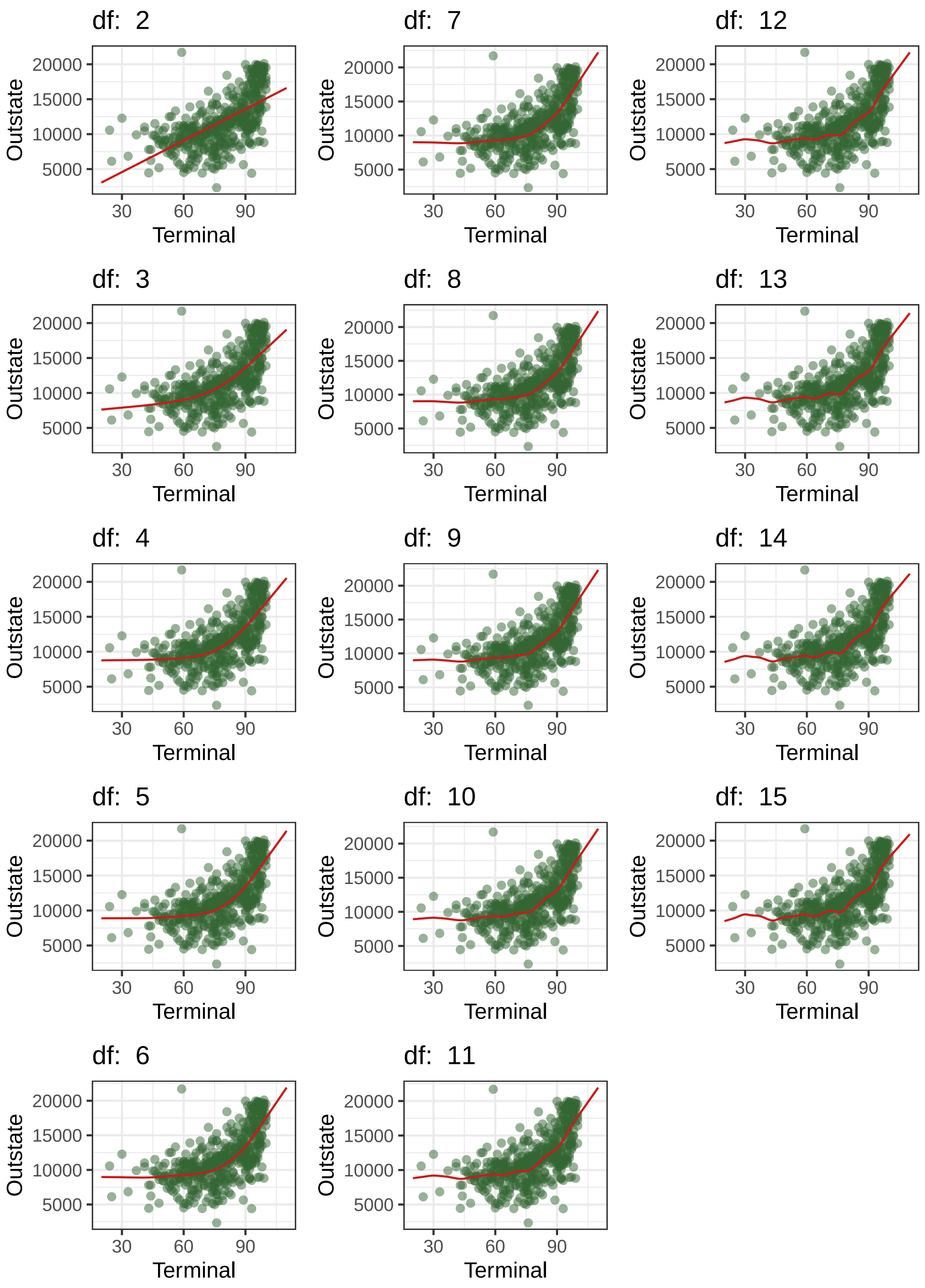
```
## [1] 4.795542
```

```
predict.ss1 = predict(fit.ss1, x = terminal.grid)
predict.ss1_df = data.frame(pred = predict.ss1$y, terminal = terminal.grid)

p +
geom_line(aes(x = terminal, y = pred), data = predict.ss1_df,
          color = rgb(.8, .1, .1, 1)) + theme_bw()
```



```
## Take a range of degree of freedom (from df=2 to df=15)
spline_function = function(i){

  fit.ss_i = smooth.spline(train_df$Terminal, train_df$Outstate, df = i)

  predict.ss_i = predict(fit.ss_i, x = terminal.grid)

  predict.ss_i_df = data.frame(pred = predict.ss_i$y, terminal = terminal.grid)
```

```r
  plot_ss_i =
    p +
    geom_line(aes(x = terminal, y = pred), data = predict.ss_i_df,
                  color = rgb(.8, .1, .1, 1)) +
    labs(title = paste("df: ", i)) +
    theme_bw()

  plot_ss_i

}

plots = list()
for (i in 2:15) {
    plots[[i]] = spline_function(i)
}
multiplot(plotlist = plots[-1], cols = 3)
```


df: 2
df: 7
df: 12
df: 3
df: 8
df: 13
df: 4
df: 9
df: 14
df: 5
df: 10
df: 15
df: 6
df: 11
Outstate
Terminal
5000
10000
15000
20000
30
60
90

Based on the analysis above, we can see that the optimized degree of freedom generated by GCV is 4.7955421, and when we take a range of df from 2 to 15, we can see that with the increasing of df, the curve will become less smooth. And specifically, when the df approaching to 2, the resulting curve will become more and more close to a linear curve. That is, actually, below our optimized degree of freedom, the fitted model tend to be less flexible and thus underfit the data points, while with df increasing, our model tend to overfit the data points.

GAM

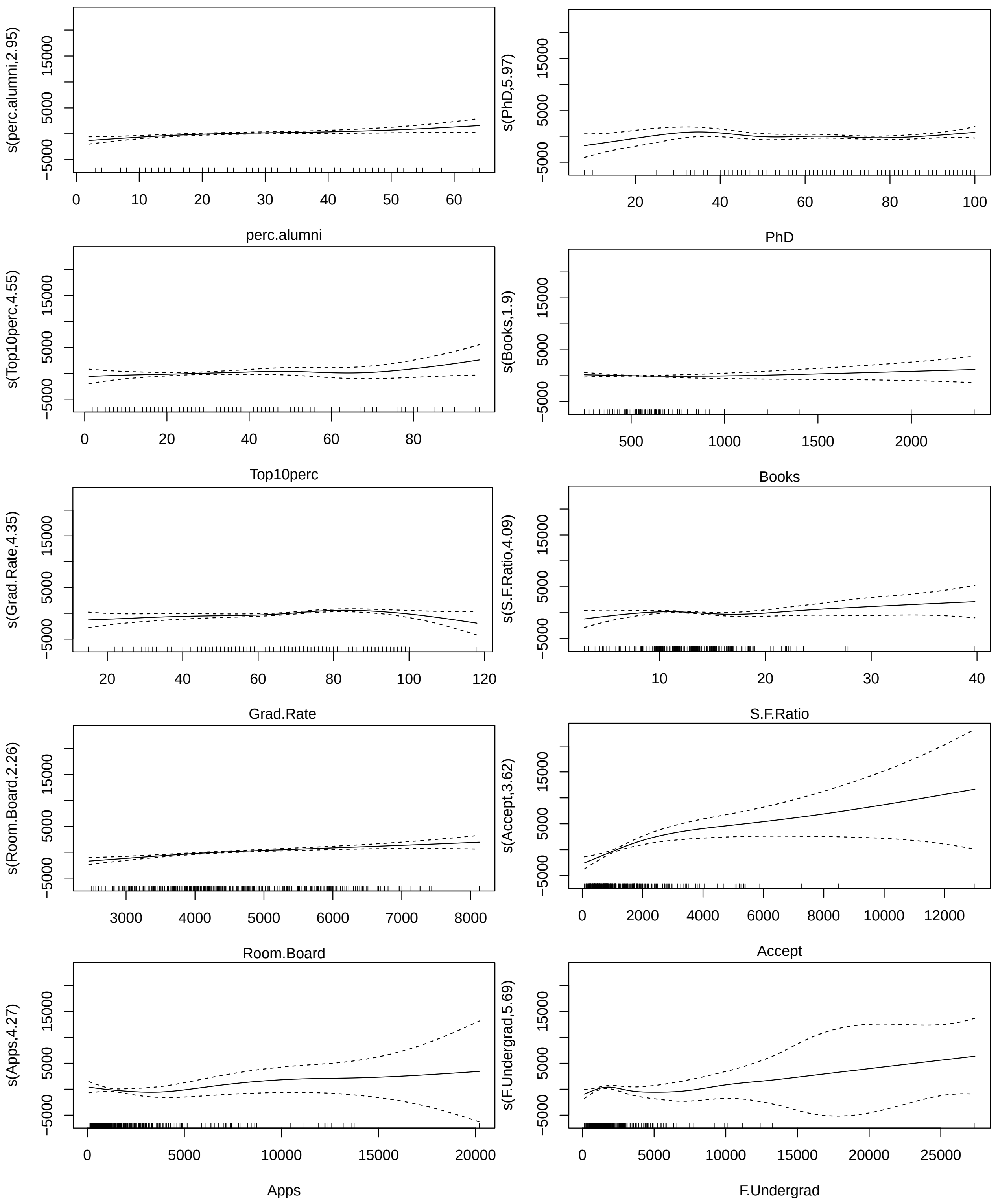
(c) Fit a generalized additive model (GAM) using all the predictors. Plot the results and explain your findings. Report the test error.

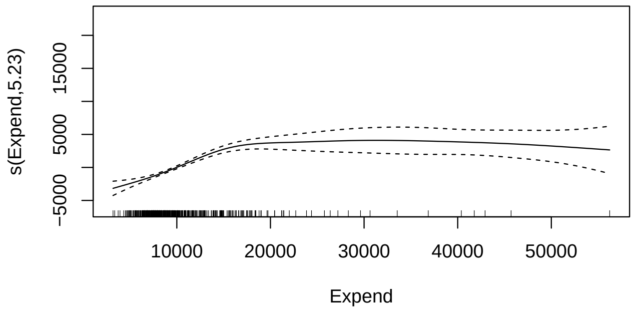
```
ctrl1 = trainControl(method = "cv", number = 10)

## Build GAM model by `caret`
set.seed(33)
gam.fit = train(x, y,
                method = "gam",
                trControl = ctrl1)
gam.fit$finalModel
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ s(perc.alumni) + s(Terminal) + s(PhD) + s(Top10perc) +
##     s(Books) + s(Grad.Rate) + s(Top25perc) + s(S.F.Ratio) + s(Personal) +
##     s(P.Undergrad) + s(Enroll) + s(Room.Board) + s(Accept) +
##     s(Apps) + s(F.Undergrad) + s(Expend)
##
## Estimated degrees of freedom:
## 2.95 1.00 5.97 4.55 1.90 4.35 1.00
## 4.09 1.00 1.00 1.00 2.26 3.62 4.27
## 5.69 5.23  total = 50.89
##
## GCV score: 2827846
```

```
## Plot the results
## Here, if we directly plot `gam.fit$finalModel`, all features including
## those linear features will be plotted.
## Thus, we manually write down the final model based on the previous fit result.
gam.m1 = gam(Outstate ~ s(perc.alumni) + Terminal + s(PhD) + s(Top10perc) + s(Books) + s(Grad.Rate) + T
```

```
plot(gam.m1)
```


s(perc.alumni,2.95)
perc.alumni
s(PhD,5.97)
PhD
s(Top10perc,4.55)
Top10perc
s(Books,1.9)
Books
s(Grad.Rate,4.35)
Grad.Rate
s(S.F.Ratio,4.09)
S.F.Ratio
s(Room.Board,2.26)
Room.Board
s(Accept,3.62)
Accept
s(Apps,4.27)
Apps
s(F.Undergrad,5.69)
F.Undergrad



```r
## Test Error
pred_gam = predict(gam.fit, x_test)
gam_test_rmse = RMSE(pred_gam, y_test)
gam_test_rmse
```

```
## [1] 1731.01
```

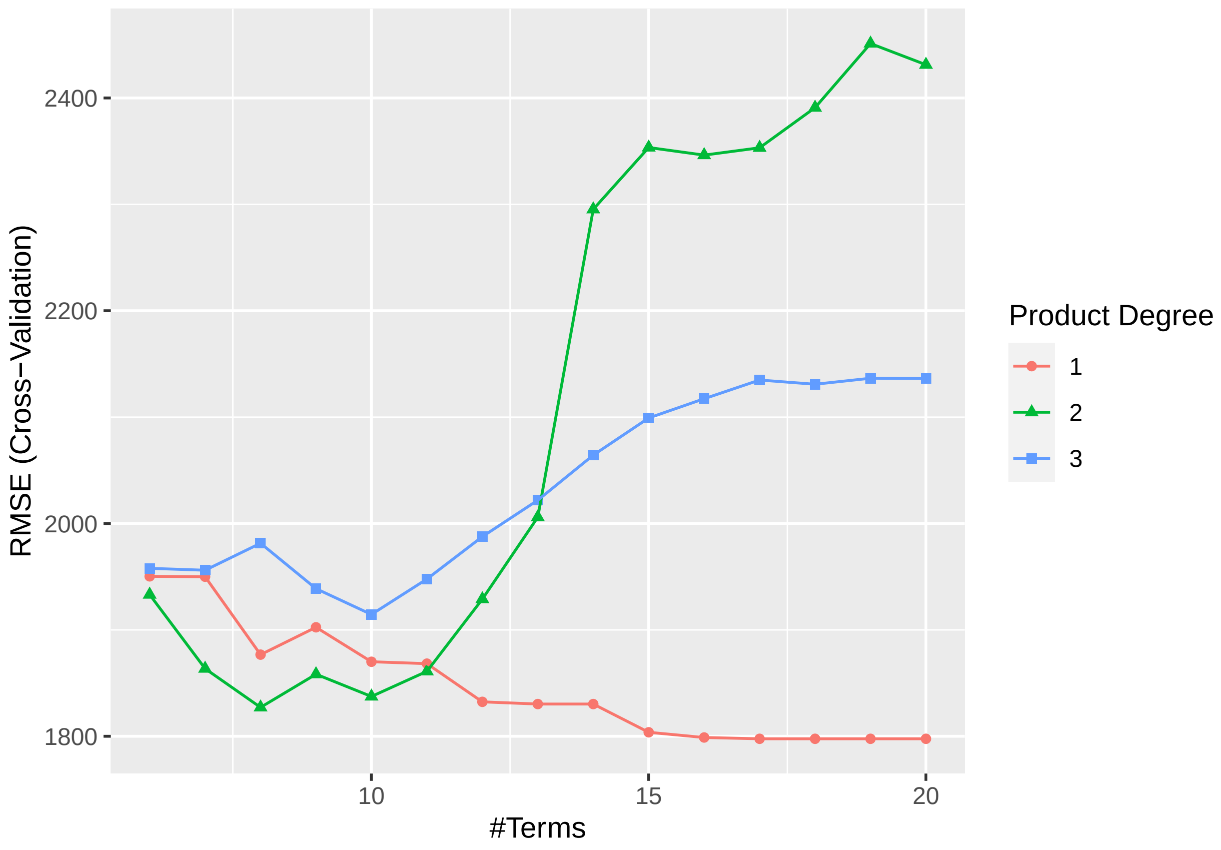
```r
# or, alternatively, by using: sqrt(mean((pred_gam - y_test)^2)), will get the same rmse.
```

Based on the fitted GAM model using all the predictors, we can see that features including `Terminal`, `Top25perc`, `Personal`, `P.Undergrad` and `Enroll` are considered as df $= 1$, which denotes the linear interaction trend between the given predictor and the response. While features such as `Books`, `Room.Board` with degree of freedom approximately 2 denote the quadratic relationship exists. For features like `per.alumni`, `Top10perc` and so forth, cubic spline or even more complicated relationships exist. The exact marginal function of each predictor vs. response could be seen in the plots above. And for the test error of the overall fit, the RMSE is 1731.010007.

MARS

(d) Train 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. Report the test error.

```r
mars_grid = expand.grid(degree = 1:3,
                        nprune = 6:20)

## Fit a MARS model
set.seed(33)
mars.fit = train(x, y,
                 method = "earth",
                 tuneGrid = mars_grid,
                 trControl = ctrl1)

ggplot(mars.fit)
```



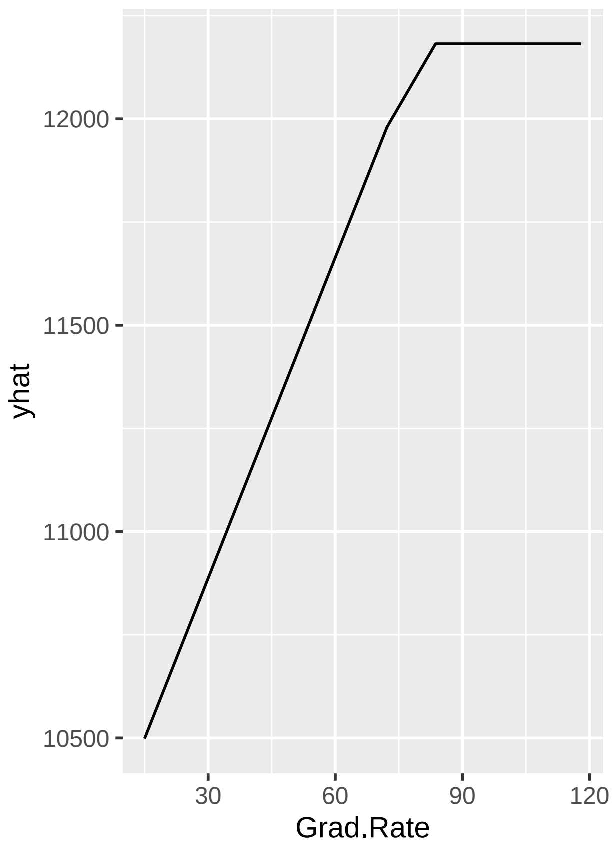
```
## Final Model
mars.fit$bestTune
```

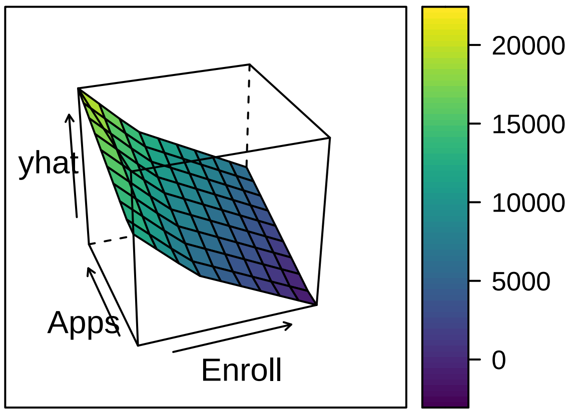
```
##    nprune degree
## 12     17      1
```

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

```
##         (Intercept)     h(15605-Expend)     h(80-Grad.Rate)  h(Room.Board-4250)
##       10696.0637132          -0.5806846         -25.9005641           0.2838710
##  h(4250-Room.Board) h(1365-F.Undergrad)   h(perc.alumni-14)   h(14-perc.alumni)
##          -1.1840794          -1.3868992          32.2294877        -117.8198469
##        h(Apps-1358)    h(1000-Personal)      h(Enroll-1499)      h(1499-Enroll)
##           0.3970981           1.3973865          -2.5482702           4.7361683
##      h(Accept-1553)      h(1553-Accept)           h(PhD-81)
##           0.5619901          -1.3818253          69.8182938
```

```
## PDP of `Grad.Rate`
p1 = pdp::partial(mars.fit, pred.var = c("Grad.Rate"), grid.resolution = 10) %>% autoplot()
## PDP of `Enroll` + `Apps`
p2 = pdp::partial(mars.fit, pred.var = c("Enroll", "Apps"),
                  grid.resolution = 10) %>%
     pdp::plotPartial(levelplot = FALSE, zlab = "yhat", drape = TRUE,
                      screen = list(z = 20, x = -60))

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





```
## Test Error
mars_pred = predict(mars.fit, x_test)
mars_test_rmse = RMSE(mars_pred, y_test)
mars_test_rmse
```

```
## [1] 1691.296
```

Based on the results above, we can see that the two optimized tuning parameters of our MARS model is product degree = 1, which denotes there should be no product of hinge function, and 17 terms in total will be included.

To better understand the relationship between features and outcome, we create partial dependence plots (PDPs) to show the contribution of feature `Grad.Rate` and the combined contribution of `Enroll` and `Apps`.

The Test Error of the final MARS model is RMSE = 1691.295667.

Model Comparison

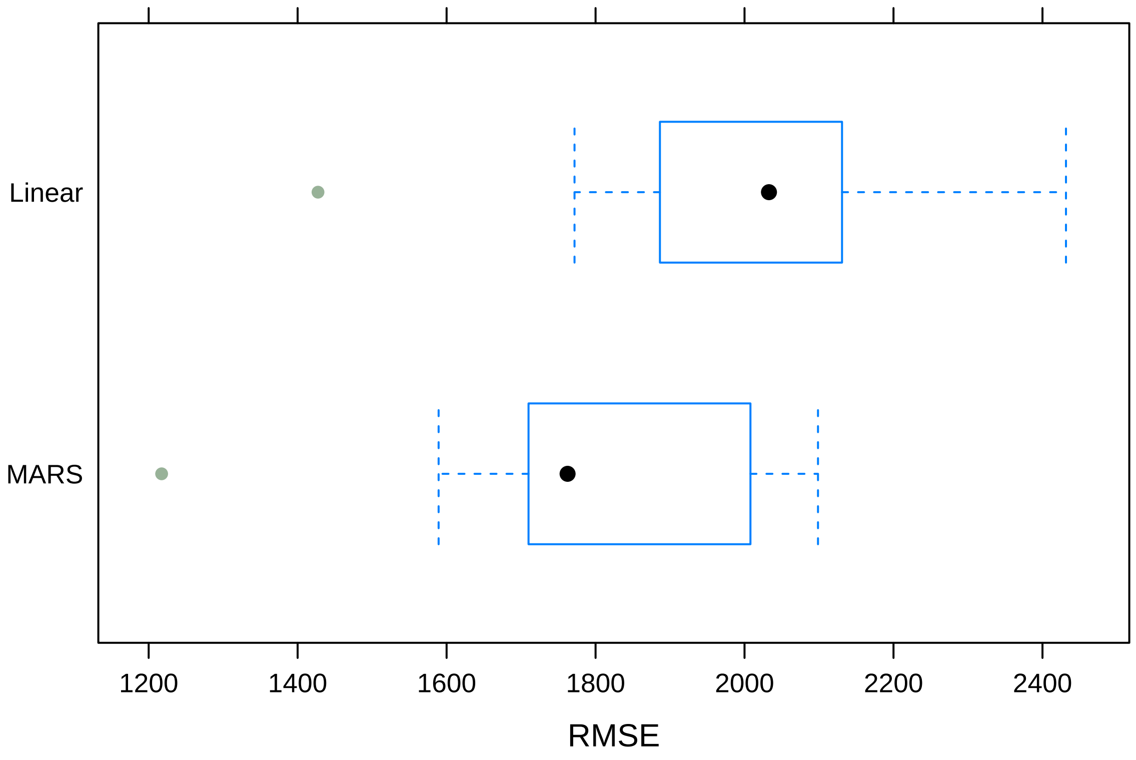
(e) In this data example, do you prefer the use of MARS model over a linear model when predicting the out-of-state tuition? Why?

Based on the summary results and the plot, MARS model is preferred compared to linear model, since the RMSE of MARS model is much more smaller than linear model, as well as the R squared of MARS is a little bit larger than linear model, which denotes more proportion of y is explained by x.

```r
## Linear Model
set.seed(33)
lm.fit = train(x, y,
               preProcess = c("center", "scale"),
               method = "lm",
               trControl = ctrl1)

## MARS vs. Linear
resamp = resamples(list(Linear = lm.fit, MARS = mars.fit))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: Linear, MARS
## Number of resamples: 10
##
## MAE
##             Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## Linear 1083.1216 1519.125 1597.762 1570.629 1706.650 1783.906    0
## MARS    930.8884 1273.507 1368.160 1383.728 1504.765 1724.034    0
##
## RMSE
##            Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## Linear 1427.355 1906.593 2032.777 2000.094 2120.096 2431.571    0
## MARS   1217.410 1716.297 1762.450 1797.639 2006.244 2098.642    0
##
## Rsquared
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Linear 0.5782226 0.7054219 0.7474281 0.7309046 0.7646330 0.8215008    0
## MARS   0.7015207 0.7349914 0.7842936 0.7808214 0.8222781 0.8733715    0
```

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



Also we could compare the test error:

```
pred_lm = predict(lm.fit, x_test)
lm_test_rmse = RMSE(pred_lm, y_test)
lm_test_rmse
```

```
## [1] 1983.44
```

Here we can see that the test error of MARS (RMSE = 1691.296) is smaller than linear model (RMSE = 1983.44), which further enforces our decision above.