P8106: Data Science II, Homework #2

Zachary Katz (UNI: zak2132)

3/8/2022

Contents

Set-Up and Data Preprocessing

```
set.seed(2132)

# Load data, clean column names, eliminate rows containing NA entries
data = read_csv("./Data/College.csv") %>%
  janitor::clean_names() %>%
  na.omit() %>%
  relocate("outstate", .after = "grad_rate") %>%
  select(-college)

# Partition data into training/test sets
indexTrain = createDataPartition(y = data$outstate,
                                 p = 0.8,
                                 list = FALSE)

training_df = data[indexTrain, ]

testing_df = data[-indexTrain, ]

# Create matrices for future analysis

# Training data
x_train = model.matrix(outstate~.,training_df)[, -1]
y_train = training_df$outstate

# Testing data
```

```r
x_test <- model.matrix(outstate~.,testing_df)[, -1]
y_test <- testing_df$outstate
```

Part (a): Exploratory Data Analysis

```r
# Summary statistics
summary(training_df)
```

```
##       apps           accept          enroll         top10perc    
##  Min.   :   81   Min.   :   72   Min.   :  35.0   Min.   : 1.00  
##  1st Qu.:  626   1st Qu.:  498   1st Qu.: 200.0   1st Qu.:16.00  
##  Median : 1132   Median :  864   Median : 336.0   Median :25.00  
##  Mean   : 2038   Mean   : 1339   Mean   : 464.1   Mean   :29.15  
##  3rd Qu.: 2227   3rd Qu.: 1598   3rd Qu.: 520.0   3rd Qu.:36.00  
##  Max.   :20192   Max.   :13007   Max.   :3810.0   Max.   :95.00  
##    top25perc       f_undergrad     p_undergrad        room_board  
##  Min.   :  9.00   Min.   :  139   Min.   :    1.0   Min.   :2370  
##  1st Qu.: 42.00   1st Qu.:  836   1st Qu.:   67.0   1st Qu.:3720  
##  Median : 55.00   Median : 1306   Median :  191.0   Median :4390  
##  Mean   : 56.72   Mean   : 1894   Mean   :  417.6   Mean   :4576  
##  3rd Qu.: 69.00   3rd Qu.: 2041   3rd Qu.:  541.0   3rd Qu.:5400  
##  Max.   :100.00   Max.   :14971   Max.   :10221.0   Max.   :8124  
##      books           personal         ph_d           terminal    
##  Min.   : 250.0   Min.   : 300   Min.   :  8.00   Min.   : 24.0  
##  1st Qu.: 450.0   1st Qu.: 800   1st Qu.: 59.00   1st Qu.: 67.0  
##  Median : 500.0   Median :1100   Median : 72.00   Median : 80.0  
##  Mean   : 550.5   Mean   :1226   Mean   : 70.28   Mean   : 77.8  
##  3rd Qu.: 600.0   3rd Qu.:1500   3rd Qu.: 84.00   3rd Qu.: 91.0  
##  Max.   :2340.0   Max.   :6800   Max.   :100.00   Max.   :100.0  
##    s_f_ratio      perc_alumni        expend        grad_rate    
##  Min.   : 2.50   Min.   : 3.00   Min.   : 3186   Min.   : 18.0  
##  1st Qu.:11.10   1st Qu.:16.00   1st Qu.: 7438   1st Qu.: 58.0  
##  Median :12.70   Median :24.00   Median : 8990   Median : 69.0  
##  Mean   :12.83   Mean   :25.62   Mean   :10558   Mean   : 69.2  
##  3rd Qu.:14.40   3rd Qu.:34.00   3rd Qu.:11625   3rd Qu.: 81.0  
##  Max.   :39.80   Max.   :64.00   Max.   :56233   Max.   :118.0  
##     outstate    
##  Min.   : 4371  
##  1st Qu.: 9100  
##  Median :11200  
##  Mean   :11788  
##  3rd Qu.:13970  
##  Max.   :19900  
```

```r
skimr::skim(training_df)
```

Table 1: Data summary

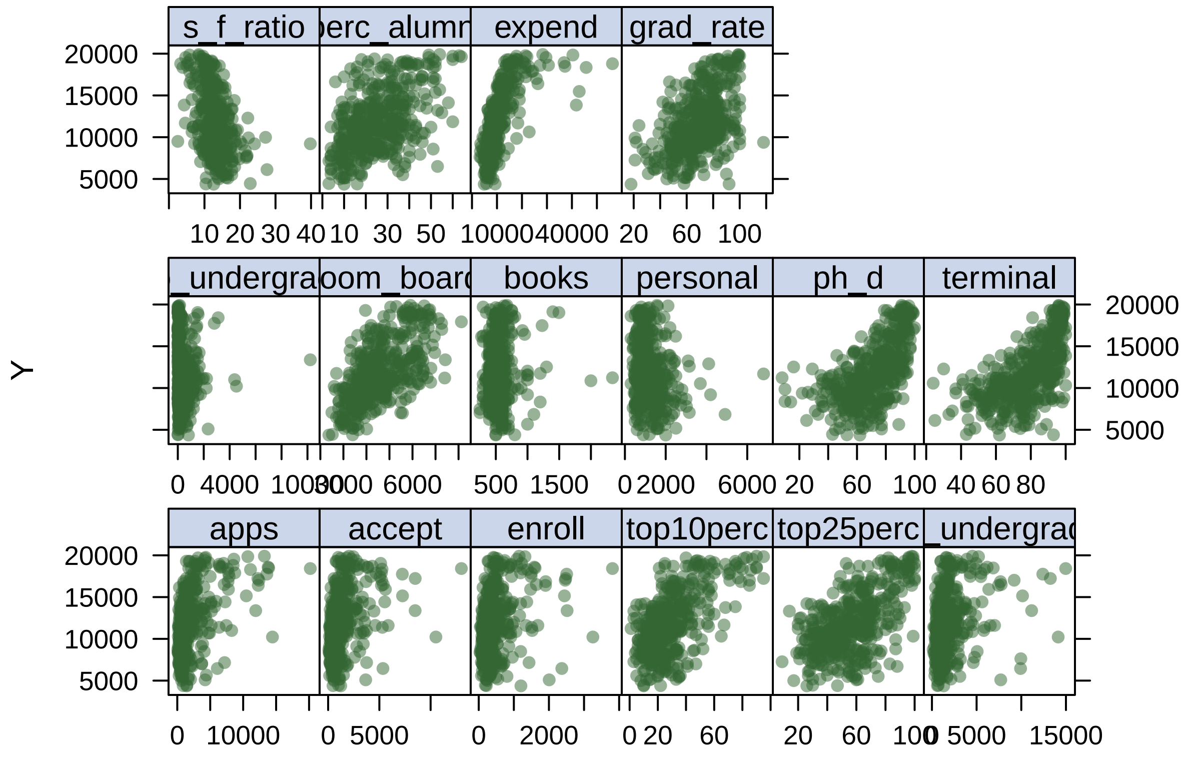
Name	training_df
Number of rows	453
Number of columns	17
Column type frequency:	
numeric	17
Group variables	None

Variable type: numeric

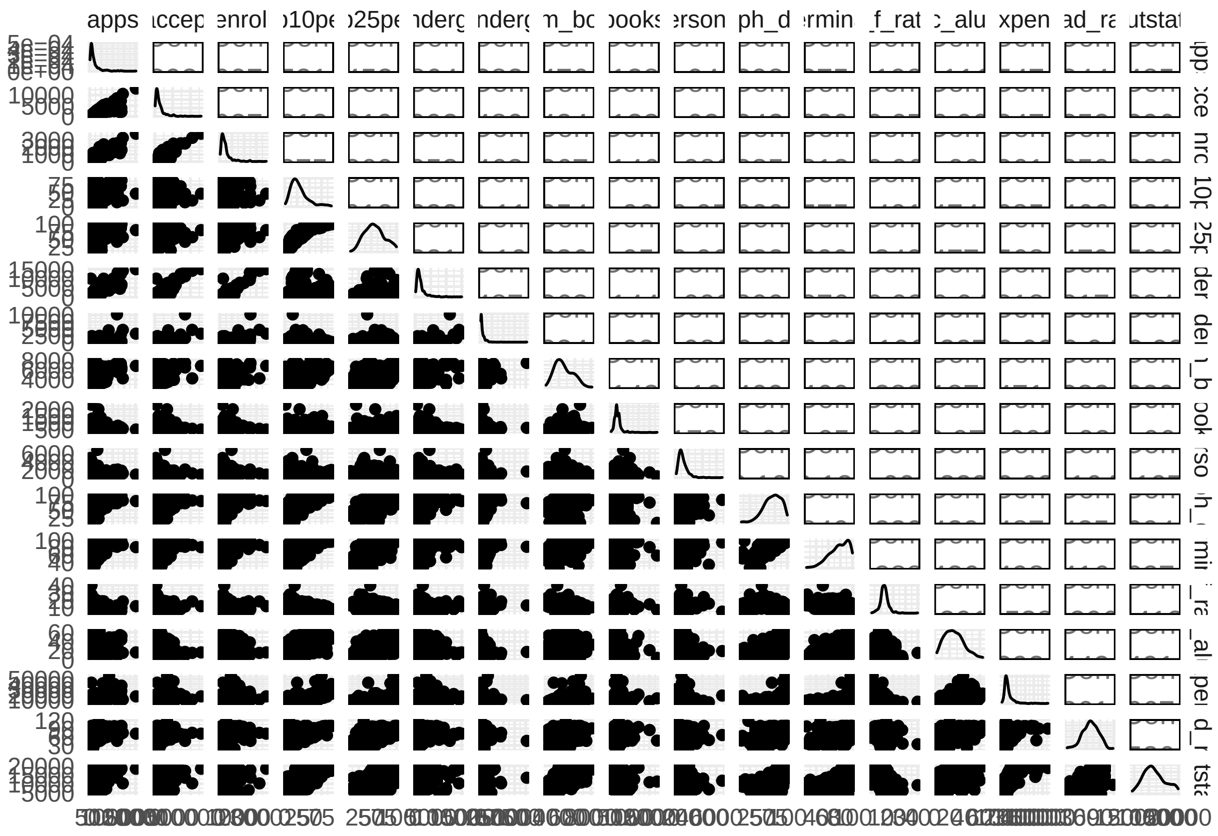
skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
apps	0	1	2038.24	2561.00	81.0	626.0	1132.0	2227.0	20192.0	
accept	0	1	1339.36	1449.87	72.0	498.0	864.0	1598.0	13007.0	
enroll	0	1	464.13	453.55	35.0	200.0	336.0	520.0	3810.0	
top10perc	0	1	29.15	17.94	1.0	16.0	25.0	36.0	95.0	
top25perc	0	1	56.72	19.58	9.0	42.0	55.0	69.0	100.0	
f_undergrad	0	1	1894.18	1951.06	139.0	836.0	1306.0	2041.0	14971.0	
p_undergrad	0	1	417.59	706.01	1.0	67.0	191.0	541.0	10221.0	
room_board	0	1	4576.34	1100.87	2370.0	3720.0	4390.0	5400.0	8124.0	
books	0	1	550.45	186.88	250.0	450.0	500.0	600.0	2340.0	
personal	0	1	1225.53	662.09	300.0	800.0	1100.0	1500.0	6800.0	
ph_d	0	1	70.28	17.59	8.0	59.0	72.0	84.0	100.0	
terminal	0	1	77.80	15.76	24.0	67.0	80.0	91.0	100.0	
s_f_ratio	0	1	12.83	3.49	2.5	11.1	12.7	14.4	39.8	
perc_alumni	0	1	25.62	12.42	3.0	16.0	24.0	34.0	64.0	
expend	0	1	10557.97	5910.08	3186.0	7438.0	8990.0	11625.0	56233.0	
grad_rate	0	1	69.20	16.49	18.0	58.0	69.0	81.0	118.0	
outstate	0	1	11787.83	3625.60	4371.0	9100.0	11200.0	13970.0	19900.0	

In total, our training data set has 453 observations on 17 variables, with no data incompleteness. Of the 17 variables, our single response (outcome) variable is `outstate`, representing out of state tuition. Our other 16 variables are continuous, numeric variables representing a range of predictors, from annual applications to cost of room and board. Notably, we have excluded college name (`college`) as a predictor given its presumed irrelevance to any kind of predictive model.

```r
# EDA scatterplots
# Set visual theme settings
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)

# All predictors are continuous; scatterplots most useful for data viz
featurePlot(x_train, y_train, plot = "scatter", labels = c("","Y"), type = c("p"))
```



```
# Pairwise relationships show numerous multicollinearities
ggpairs(training_df)
```



At first glance, the clearest linear relationships with `outstate` seem to be with predictors `perc_alumni`, `grad_rate`, `ph_d`, `terminal`, `top25perc`, `room_board`, and `top10perc`. In addition, we observe numerous multicollinearities with correlation greater than 0.90, including `apps` and `enroll`, `enroll` and `accept`, `top25perc` and `top10perc`, and several others.

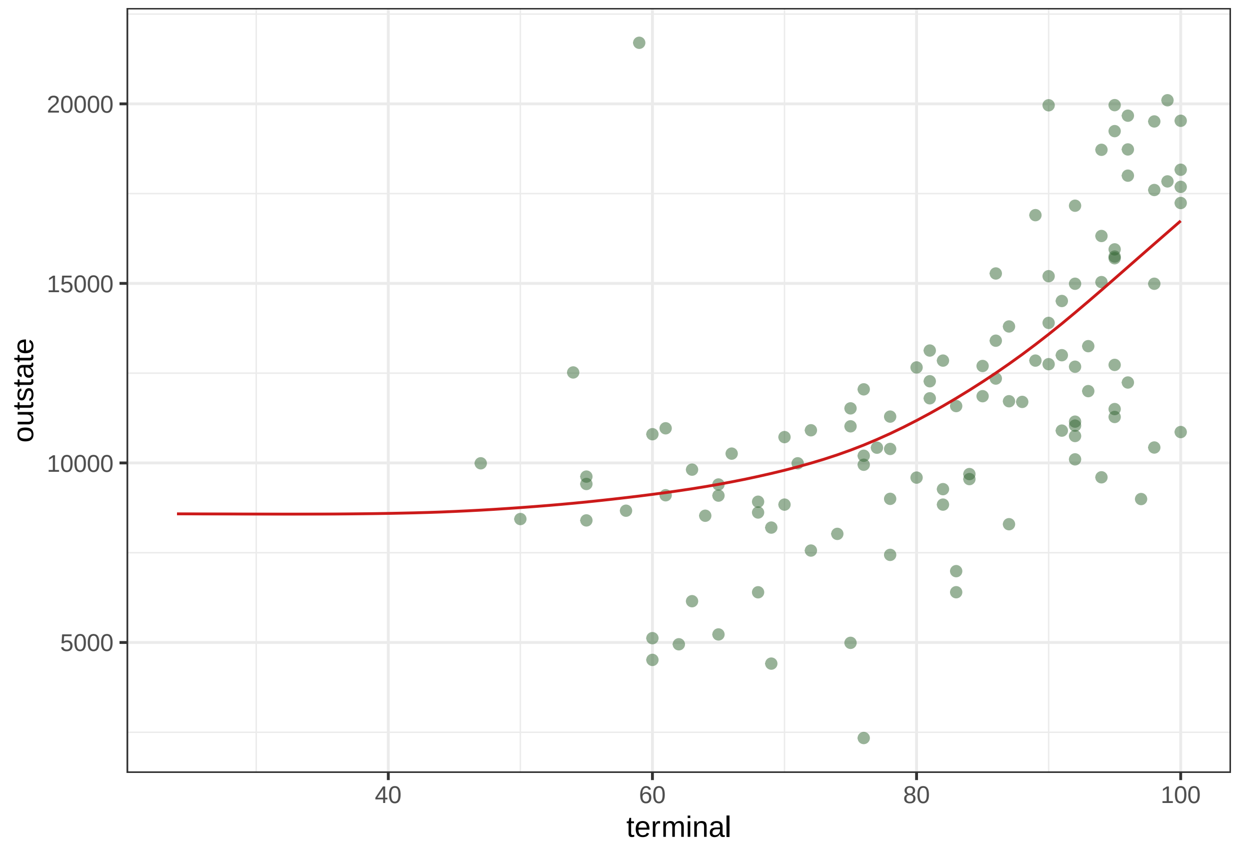
Part (b): Smoothing Spline Models

```
set.seed(2132)
# Fit smoothing spline using `terminal` as only predictor of `outstate`
# By default, uses generalized cross-validation to select lambda value (smoothing parameter)
fit_smooth_spline = smooth.spline(training_df$terminal, training_df$outstate)

# Optimal degrees of freedom based on cross-validation
fit_smooth_spline$df
```

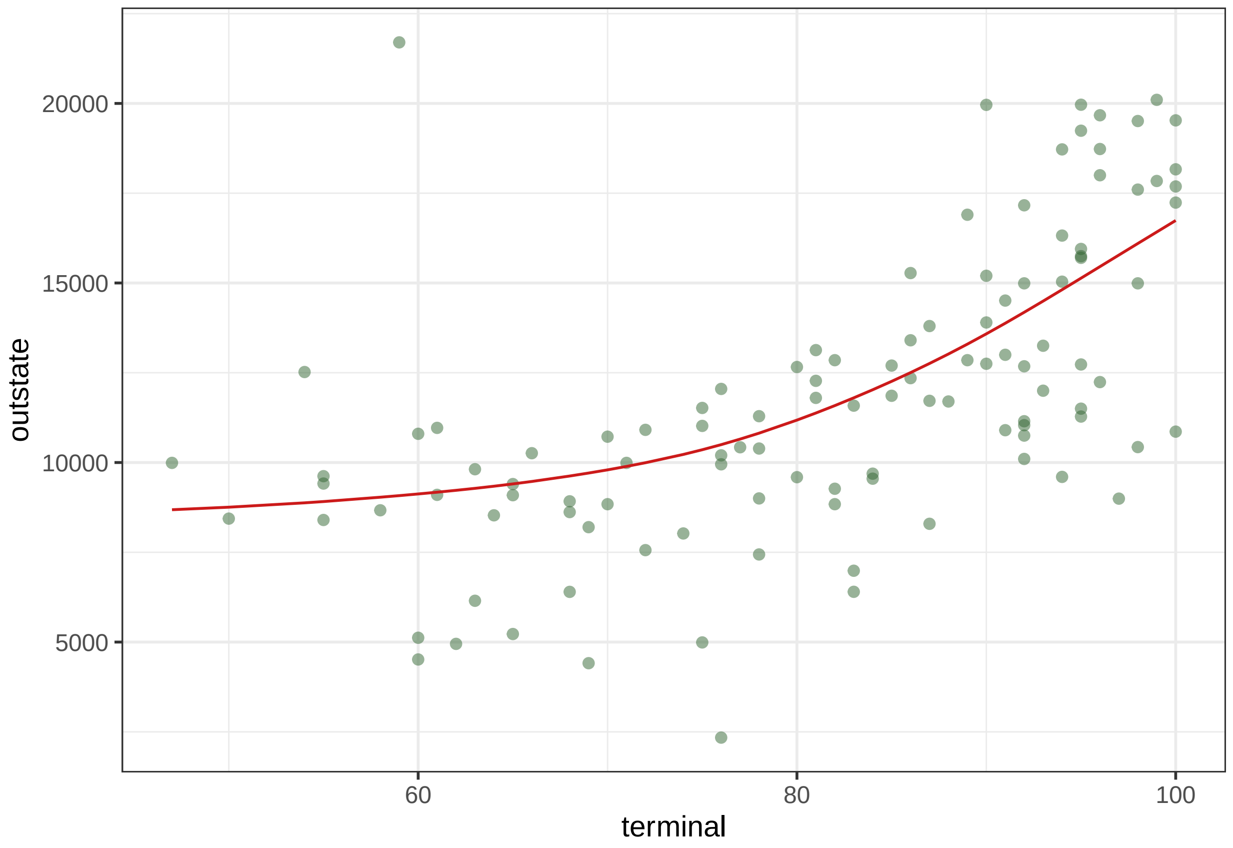
```
## [1] 4.435655
```

```
# Prediction on grid of terminal values
# Using min and max values from training and testing data
terminal_grid <- seq(from = 24, to = 100, by = 1)

pred_smooth_spline_grid = predict(fit_smooth_spline, x = terminal_grid)
```

```r
pred_smooth_spline_grid_df = data.frame(predicted = pred_smooth_spline_grid$y,
                                        terminal = terminal_grid)

p = ggplot(data = testing_df, aes(x = terminal, y = outstate)) +
     geom_point(color = rgb(.2, .4, .2, .5))

p + geom_line(aes(x = terminal, y = predicted), data = pred_smooth_spline_grid_df,
              color = rgb(.8, .1, .1, 1)) + theme_bw()
```

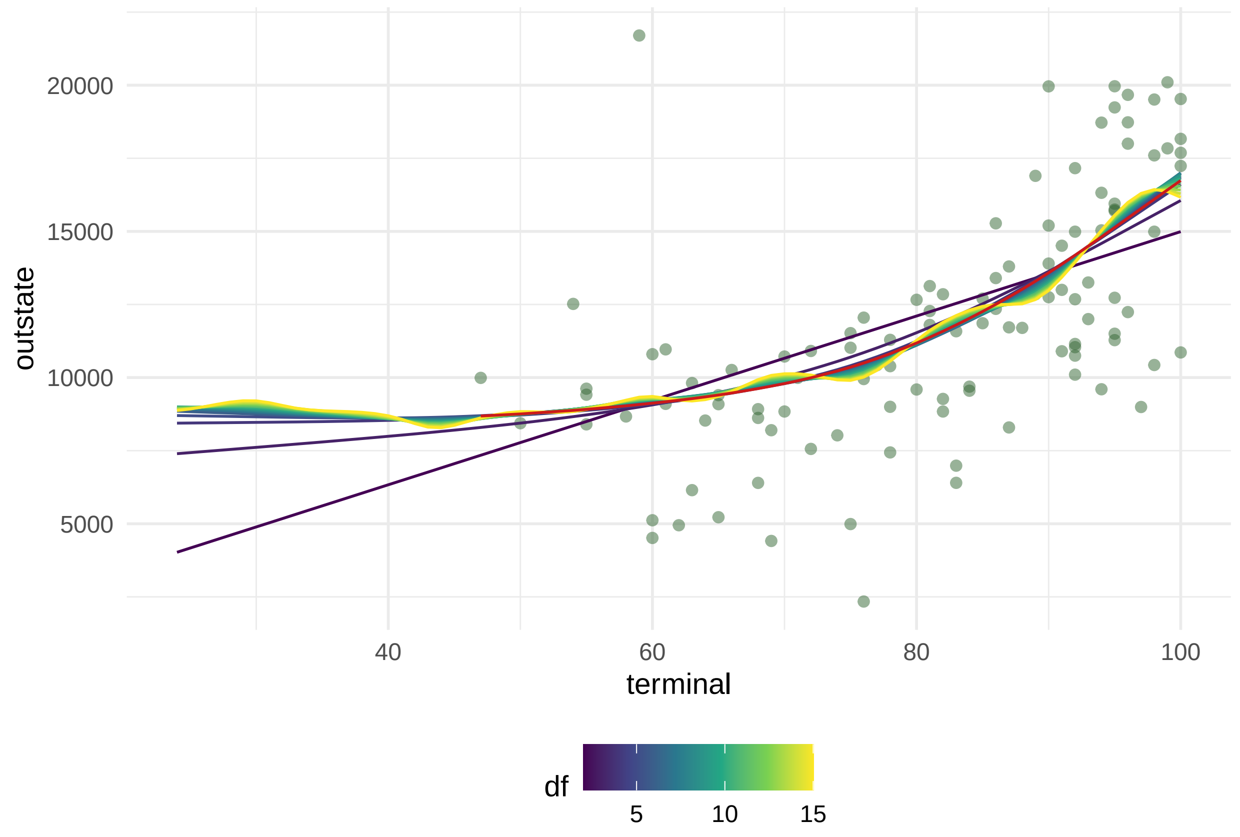


```r
# Prediction on test data
pred_smooth_spline_testing = predict(fit_smooth_spline, x = testing_df$terminal)

pred_smooth_spline_testing_df = data.frame(predicted = pred_smooth_spline_testing$y,
                                           terminal = testing_df$terminal)

p + geom_line(aes(x = terminal, y = predicted), data = pred_smooth_spline_testing_df,
          color = rgb(.8, .1, .1, 1)) + theme_bw()
```



Here, we fit a smoothing spline model using `terminal` as the only predictor of `outstate` for the optimal degrees of freedom obtained by generalized cross-validation, which is about 4.4. However, we'd also like to understand how the model fit changes with a range of degrees of freedom, which we plot below:

```r
# Try a range of degrees of freedom in function
spline_fn = function(degree){

  spline_fit = smooth.spline(training_df$terminal, training_df$outstate, df = degree)

  spline_pred = predict(spline_fit, x = terminal_grid)

  spline_df = data.frame(predicted = spline_pred$y,
                         terminal = terminal_grid,
                         df = degree)

}

# Run spline function for degrees of freedom 2 through 15
datalist = list()
for (i in 2:15) {
  datalist[[i]] = spline_fn(i)
}
all_data = do.call(rbind, datalist) %>%
  as.data.frame()

# Plot results for range of degrees of freedom
# Red line represents optimal df from base R function
p +
  geom_line(aes(x = terminal, y = predicted, group = df, color = df), data = all_data) +
  geom_line(aes(x = terminal, y = predicted), data = pred_smooth_spline_testing_df,
          color = rgb(.8, .1, .1, 1))
```



Overlaying the model with different degrees of freedom (ranging from 2 to 15), we see that with fewer than 4 degrees of freedom, our model fit is more linear. As we increase the degrees of freedom much beyond 4, there is more overfitting; our models start to “wiggle” more. We can observe that our optimized model with about 4.4 degrees of freedom optimally fits the data.

Part (c): Generalized Additive Model

```
set.seed(2132)

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

# Check whether any predictors take on fewer than 10 values
# None do, so we can use the caret function, which at times results
# in loss of flexibility when we have predictors that take on <= 10 values
sapply(x_train %>% as.data.frame(), n_distinct)
```

```
##         apps       accept       enroll    top10perc    top25perc f_undergrad
##         424         414         344          74          83         415
## p_undergrad  room_board       books    personal        ph_d    terminal
##         334         347          70         176          76          65
##   s_f_ratio perc_alumni      expend   grad_rate
##         131          57         444          75
```

```
# Run GAM in caret
# Use automatic feature selection (Cp method)
gam_fit = train(x_train, y_train,
                method = "gam",
                tuneGrid = data.frame(method = "GCV.Cp",
                                      select = c(TRUE, FALSE)),
                trControl = ctrl1)

# Parameters that fit the best model
gam_fit$bestTune
```

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

```
gam_fit$finalModel
```

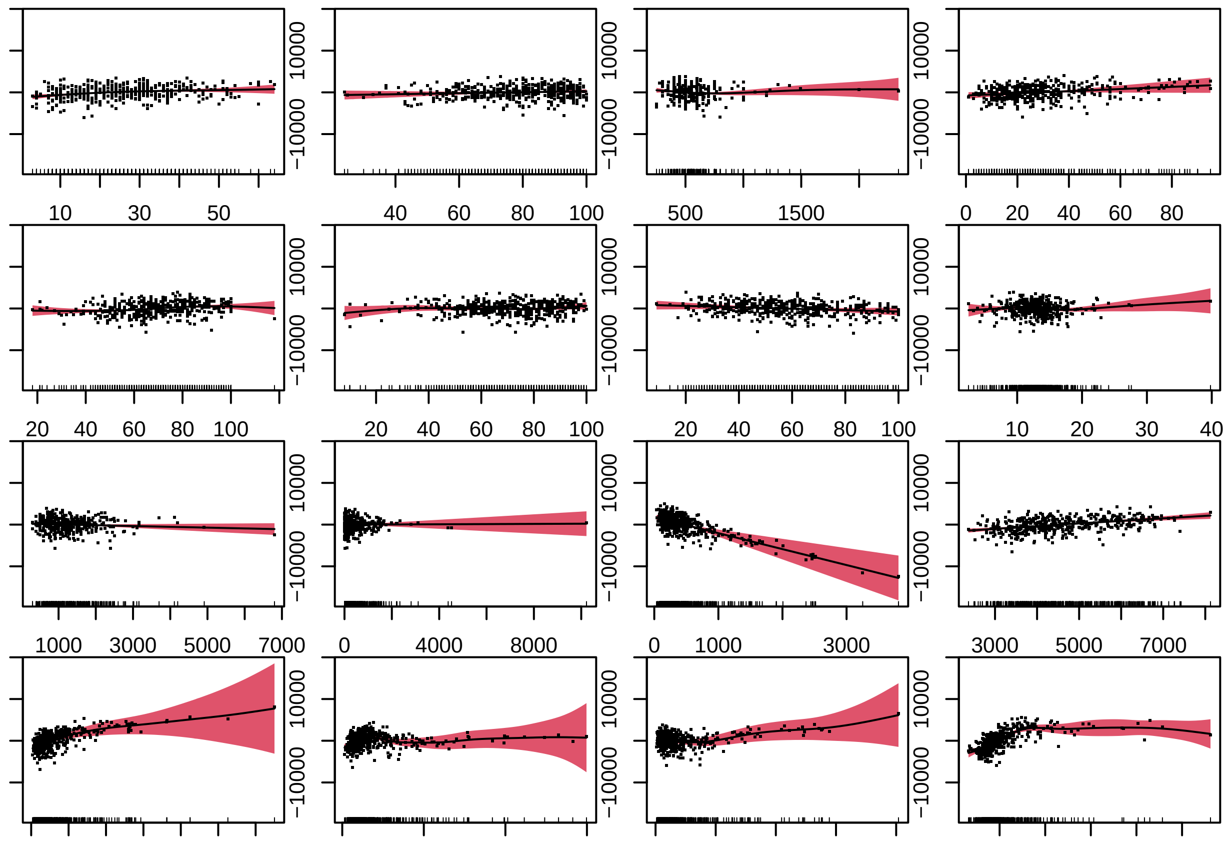
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ s(perc_alumni) + s(terminal) + s(books) + s(top10perc) +
##     s(grad_rate) + s(ph_d) + s(top25perc) + s(s_f_ratio) + s(personal) +
##     s(p_undergrad) + s(enroll) + s(room_board) + s(accept) +
##     s(f_undergrad) + s(apps) + s(expend)
##
## Estimated degrees of freedom:
## 2.24 1.00 2.73 1.00 3.16 3.70 1.00
## 3.68 1.00 1.00 1.00 1.15 3.18 6.24
## 4.21 5.70  total = 42.99
##
## GCV score: 2748289
```

```
# Summary of final model
summary(gam_fit)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ s(perc_alumni) + s(terminal) + s(books) + s(top10perc) +
##     s(grad_rate) + s(ph_d) + s(top25perc) + s(s_f_ratio) + s(personal) +
##     s(p_undergrad) + s(enroll) + s(room_board) + s(accept) +
##     s(f_undergrad) + s(apps) + s(expend)
##
```

```
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11787.8       74.1   159.1   <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) 2.237  2.838  6.626 0.000514 ***
## s(terminal)    1.000  1.000  1.434 0.231744
## s(books)       2.731  3.403  1.949 0.125752
## s(top10perc)   1.000  1.000  3.502 0.061993 .
## s(grad_rate)   3.161  4.008  4.261 0.002163 **
## s(ph_d)        3.700  4.612  1.091 0.355130
## s(top25perc)   1.000  1.000  2.417 0.120794
## s(s_f_ratio)   3.684  4.622  1.563 0.215053
## s(personal)    1.000  1.000  2.307 0.129571
## s(p_undergrad) 1.000  1.000  0.023 0.880120
## s(enroll)      1.000  1.000 22.679 2.95e-06 ***
## s(room_board)  1.151  1.284 30.300  < 2e-16 ***
## s(accept)      3.179  4.002  3.867 0.004259 **
## s(f_undergrad) 6.239  7.304  4.559 5.21e-05 ***
## s(apps)        4.213  5.184  2.040 0.075568 .
## s(expend)      5.697  6.831 17.270  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.811   Deviance explained = 82.8%
## GCV = 2.7483e+06  Scale est. = 2.4875e+06  n = 453
```

```r
# Plot of final model
par(mar=c(1,1,1,1))
par(mfrow = c(4, 4))
plot(gam_fit$finalModel, residuals = TRUE, all.terms = TRUE, shade = TRUE, shade.col = 2)
```



```
# Calculate training MSE of optimal model
gam_train_MSE = mean((y_train - predict(gam_fit))^2)
gam_train_MSE
```

```
## [1] 2251375
```

```
gam_train_RMSE = sqrt(gam_train_MSE)
gam_train_RMSE
```

```
## [1] 1500.458
```

```
# Calculate test MSE of optimal model
test_predictions = predict(gam_fit, x_test)

gam_test_MSE = mean((y_test - test_predictions)^2)
gam_test_MSE
```

```
## [1] 3364712
```

```
gam_test_RMSE = sqrt(gam_test_MSE)
gam_test_RMSE
```

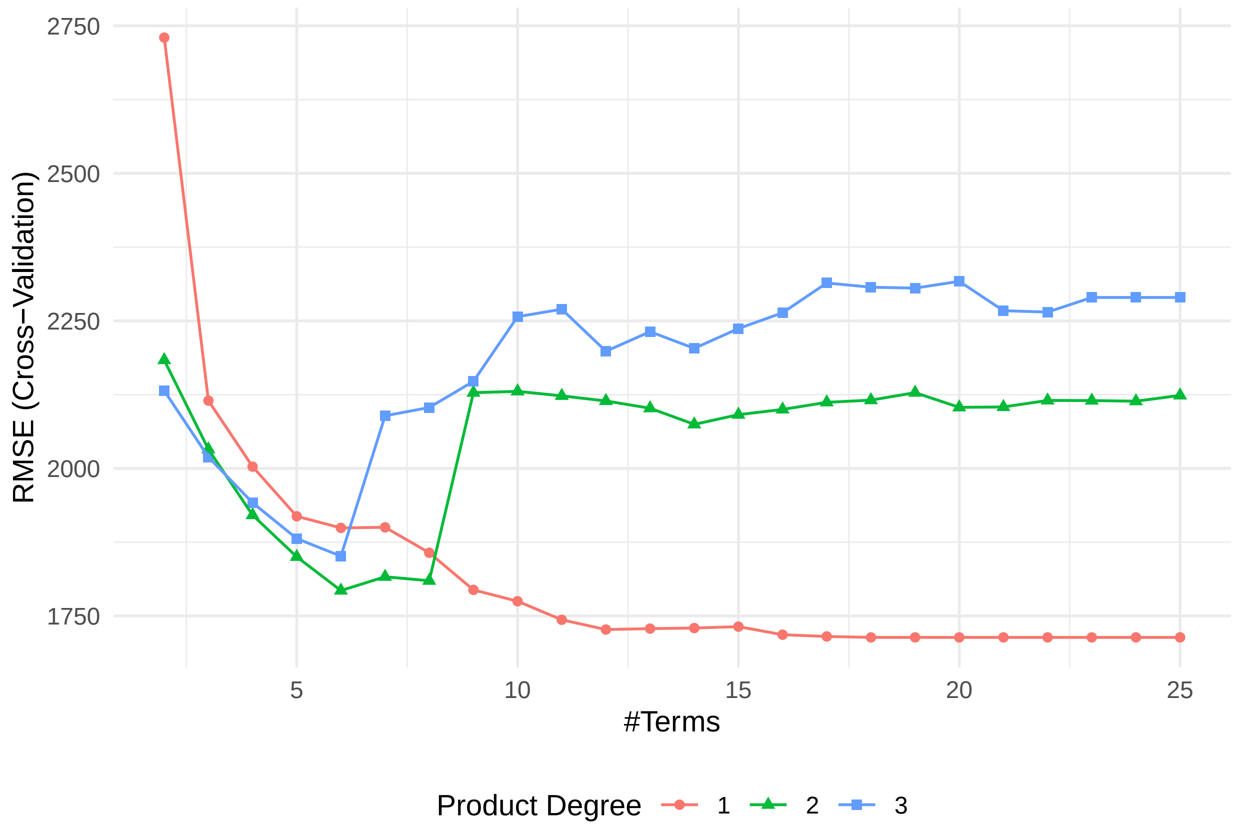
```
## [1] 1834.315
```

We may choose to fit our GAM using either MCGV or the `caret` package. Notably, the latter may result in loss of flexibility since it automatically precludes the possibility of nonlinear transformations for predictors that take fewer than 10 unique values. However, in this case, all of our predictors take more than 10 unique values, and so we do not expect loss of flexibility by using `caret`.

Using all of our predictors, our best model attains an MSE of 2251375 (RMSE 1500.5) when we apply our model to the training data and an MSE of 3364712 (RMSE 1834.3) when we apply it to the hold-out test data from our original partitioning. In our output summary, the "parametric coefficients" refers to the linear terms of the model, which in this case only includes the intercept. Coefficients are not printed for our smooth terms because each smooth term has several coefficients corresponding to different basis functions. Instead, we have effective degrees of freedom, which represent the complexity of the smooth function. `terminal`, `top10perc`, `top25perc`, `personal`, `p_undergrad`, and `enroll` all have one effective degree of freedom, corresponding to a straight line; our graphs confirm these linear relationships. Those with effective degrees of freedom around two, such as `perc_alumni` and `books`, are quadratically incorporated, whereas those with effective degrees of freedom around three, such as `grad_rate` and `accept`, are cubically incorporated, and so on. In our model, `perc_alumni`, `enroll`, `room_board`, `f_undergrad`, and `expend` are our most significant smooth terms.

Part (d): Multivariate Adaptive Regression Spline Model

```
set.seed(2132)

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

# Grid of tuning parameters
mars_grid = expand.grid(degree = 1:3,
                        nprune = 2:25)

# Fit MARS model
mars_fit = train(x_train, y_train,
                 method = "earth",
                 tuneGrid = mars_grid,
                 trControl = ctrl1)

# Choose optimal model with 1 degree and 18 hinge functions to minimize RMSE
# Note that n terms includes the intercept
ggplot(mars_fit)
```



```r
mars_fit$bestTune
```

```
##    nprune degree
## 17     18      1
```

```r
# Model summary of best fit
summary(mars_fit$finalModel)
```

```
## Call: earth(x=matrix[453,16], y=c(12280,11250,1...), keepxy=TRUE, degree=1,
##             nprune=18)
##
##                        coefficients
## (Intercept)               7538.5533
## h(apps-3767)                 0.3936
## h(2109-accept)              -1.5336
## h(accept-2109)               0.4331
## h(913-enroll)                5.4282
## h(enroll-913)               -2.9772
## h(1379-f_undergrad)         -2.2215
## h(4450-room_board)          -0.7688
## h(room_board-4450)           0.4496
## h(660-books)                 2.3720
## h(ph_d-85)                  96.5617
## h(21-perc_alumni)          -88.8273
## h(expend-5557)             0.6735
## h(expend-14773)           -0.6865
## h(grad_rate-44)           27.1928
##
## Selected 15 of 22 terms, and 10 of 16 predictors (nprune=18)
## Termination condition: RSq changed by less than 0.001 at 22 terms
## Importance: expend, grad_rate, accept, enroll, f_undergrad, room_board, ...
## Number of terms at each degree of interaction: 1 14 (additive model)
## GCV 2763911    RSS 1096876249    GRSq 0.7902001    RSq 0.8153879
```

```
# Coefficients (betas) in front of each hinge function
# Note that you can have more than 1 hinge function per predictor
# In this case, we use 10 of 16 predictors
coef(mars_fit$finalModel)
```

```
##          (Intercept)     h(expend-14773)     h(grad_rate-44)  h(room_board-4450)
##         7538.5533341          -0.6864909          27.1927519           0.4495547
##   h(4450-room_board) h(1379-f_undergrad)   h(21-perc_alumni)        h(apps-3767)
##           -0.7687973          -2.2215037         -88.8273388           0.3935794
##        h(enroll-913)       h(913-enroll)      h(accept-2109)      h(2109-accept)
##           -2.9772095           5.4282064           0.4330915          -1.5335817
##       h(expend-5557)           h(ph_d-85)        h(660-books)
##            0.6734655          96.5617402           2.3720382
```

```
# Report train MSE
mars_train_MSE = mean((y_train - predict(mars_fit))^2)
mars_train_MSE
```

```
## [1] 2421360
```

```
mars_train_RMSE = sqrt(mars_train_MSE)
mars_train_RMSE
```

```
## [1] 1556.072
```

```
# Report test MSE
test_predictions_mars = predict(mars_fit, x_test)

mars_test_MSE = mean((y_test - test_predictions_mars)^2)
mars_test_MSE
```

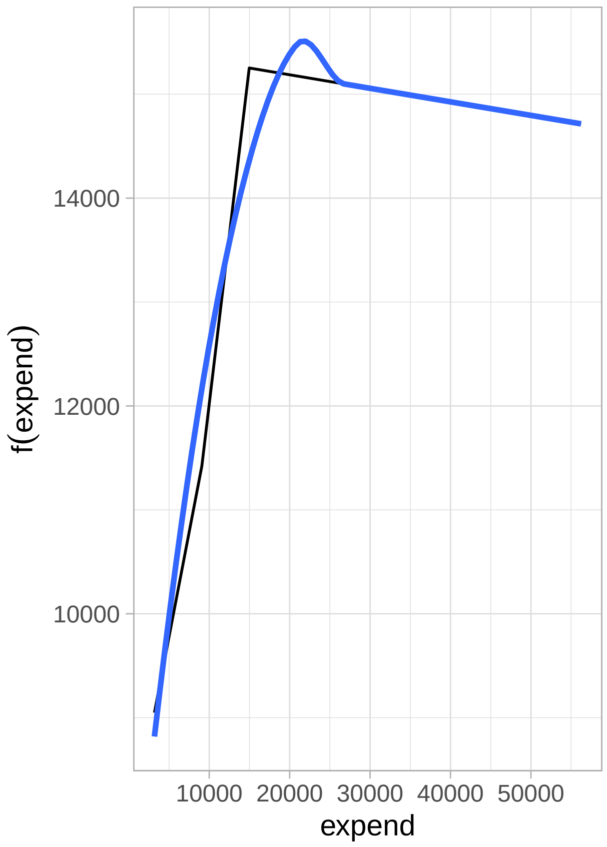
```
## [1] 3460709
```

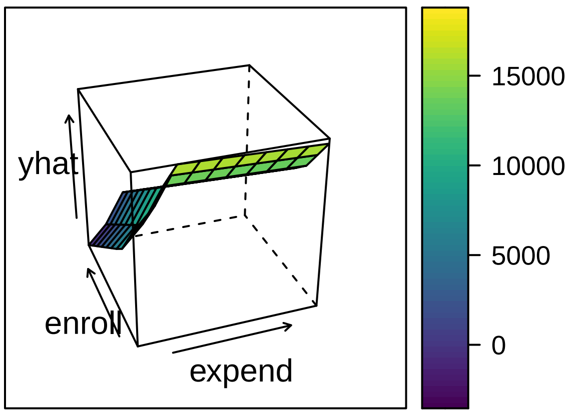
```
mars_test_RMSE = sqrt(mars_test_MSE)
mars_test_RMSE
```

```
## [1] 1860.298
```

Here, we train a MARS model using all predictors, finding that the optimal model attains an MSE of 2421360 (RMSE 1556.1) when we apply our model to the training data and an MSE of 3460709 (RMSE 1860.3) when we apply it to the hold-out test data from our original partitioning. The final model minimizes RMSE using one product degree (maximum degree of interactions, i.e. our final model is only an additive model) and 18 maximum terms, including intercept, from our `nprune` tuning parameter. 15 of 22 terms were used from 10 of the 16 original predictors. The 15 terms in our model include hinge functions and intercept. For example, looking at `apps`, we know that a knot occurs at 3767, and looking at `accept`, a knot occurs at 2109. Note that these include boundary knots. The most important predictors for our outcome appear to be `expend`, `grad_rate`, `accept`, and `enroll`.

```
# Present partial dependence plot of arbitrary predictor in final model

partial_one_pred = pdp::partial(mars_fit, pred.var = c("expend"),
                   grid.resolution = 10) %>%
  autoplot(smooth = TRUE, ylab = expression(f(expend))) +
  theme_light()

# Try with two predictors at same time, for fun

partial_two_pred = pdp::partial(mars_fit, pred.var =  c("expend", "enroll"),
                   grid.resolution = 10) %>%
  pdp::plotPartial(levelplot = FALSE, zlab = "yhat", drape = TRUE,
                    screen = list(z = 20, x = -60))

grid.arrange(partial_one_pred, partial_two_pred, ncol = 2)
```





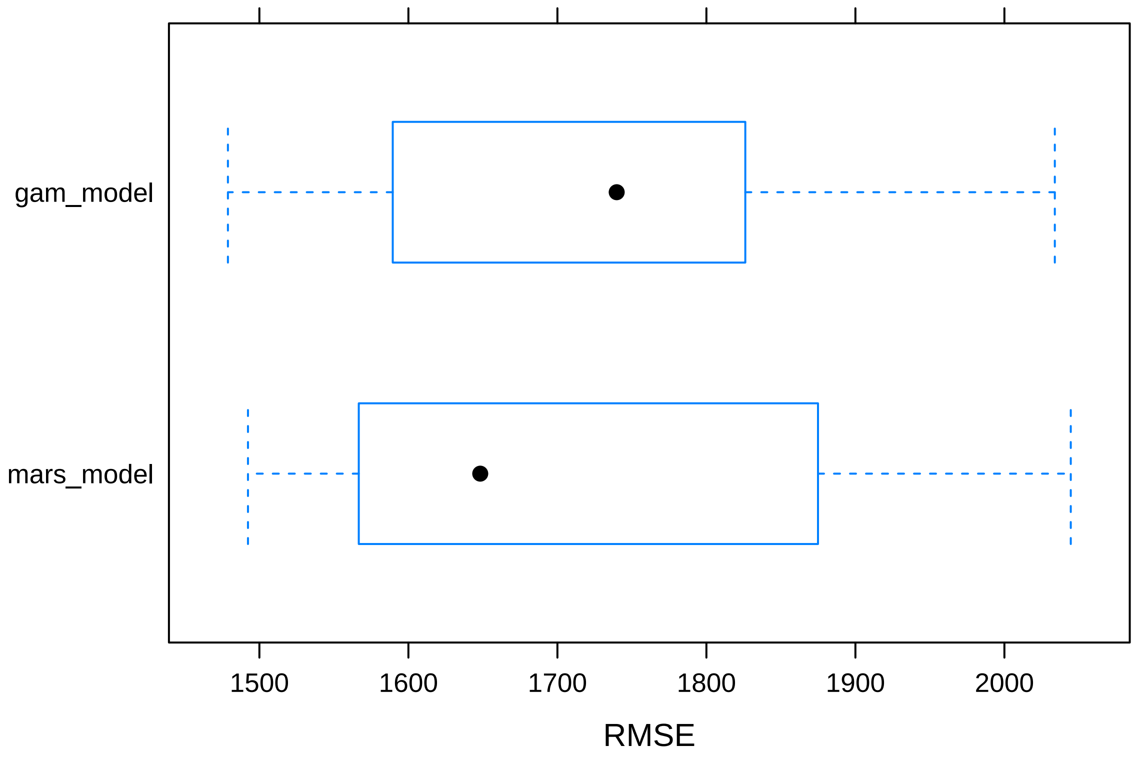
Above, we present two partial dependency plots: the first, for `expend` only, and the second, for both `expend` and `enroll`. Looking at `expend`, for example, we find a single internal knot at 14773, which corresponds to the MARS model summary printed above. As a college exceeds 14773 on the `expend` metric, each additional unit of `expend` sees a marginal decrease in `outstate` compared to colleges with less than 14773 in `expend`. The interaction plot on the right illustrates the stronger effect `expend` and `enroll` might have when combined on `outstate`, for instance.

Part (e): Selecting a Model

```
resamp = resamples(list(gam_model = gam_fit,
                        mars_model = mars_fit))

summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: gam_model, mars_model
## Number of resamples: 10
##
## MAE
##                 Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## gam_model  1150.896 1289.393 1337.376 1346.031 1413.562 1519.333    0
## mars_model 1214.381 1234.460 1298.668 1347.482 1450.206 1569.783    0
##
## RMSE
##                 Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## gam_model  1478.878 1603.574 1739.787 1735.227 1815.158 2033.835    0
## mars_model 1492.336 1569.083 1648.207 1713.669 1848.892 2044.538    0
##
## Rsquared
##                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## gam_model  0.6869804 0.7351946 0.7655835 0.7749079 0.8274491 0.8414865    0
## mars_model 0.6981001 0.7379654 0.7828860 0.7825787 0.8322636 0.8534561    0
```

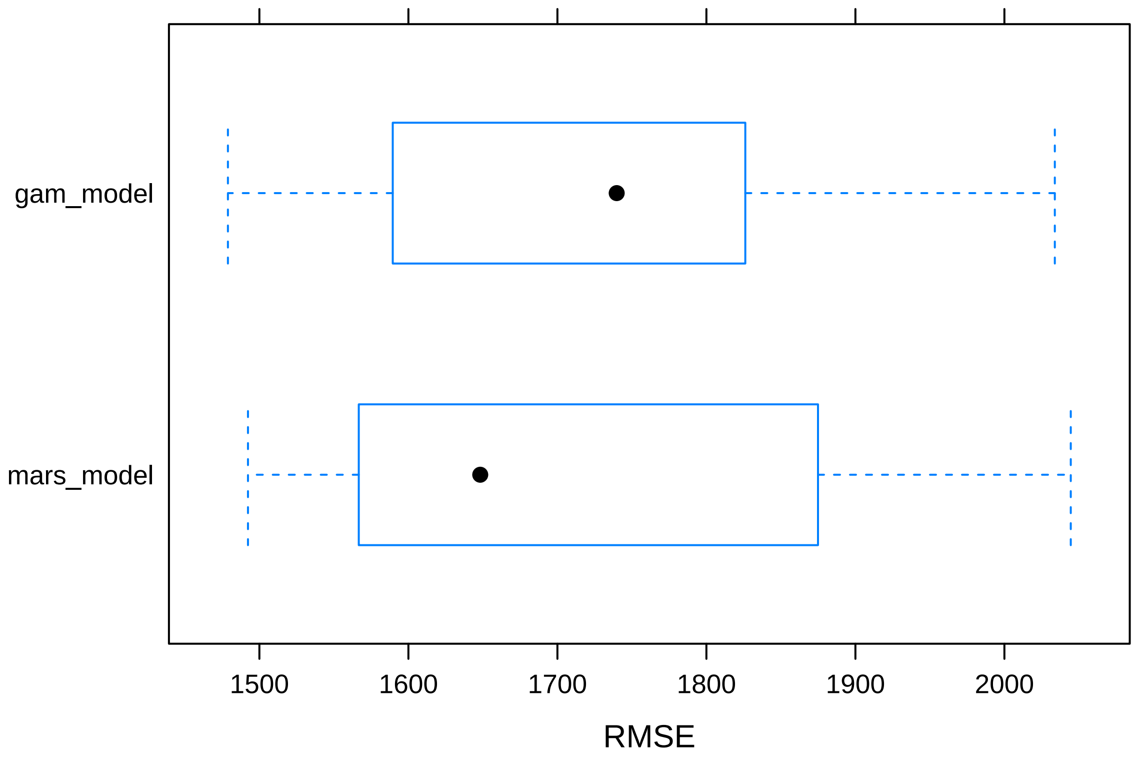
```
bwplot(resamp, metric = "RMSE")
```



```r
# Alternative method

resamp_caret = caret::resamples(list(gam_model = gam_fit,
                                     mars_model = mars_fit))

summary(resamp_caret)
```

```
##
## Call:
## summary.resamples(object = resamp_caret)
##
## Models: gam_model, mars_model
## Number of resamples: 10
##
## MAE
##                Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## gam_model  1150.896 1289.393 1337.376 1346.031 1413.562 1519.333    0
## mars_model 1214.381 1234.460 1298.668 1347.482 1450.206 1569.783    0
##
## RMSE
##                Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## gam_model  1478.878 1603.574 1739.787 1735.227 1815.158 2033.835    0
## mars_model 1492.336 1569.083 1648.207 1713.669 1848.892 2044.538    0
##
## Rsquared
##                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## gam_model  0.6869804 0.7351946 0.7655835 0.7749079 0.8274491 0.8414865    0
## mars_model 0.6981001 0.7379654 0.7828860 0.7825787 0.8322636 0.8534561    0
```

```
bwplot(resamp_caret, metric = "RMSE")
```



In this example, we prefer the MARS model over a linear model when predicting out-of-state tuition because we minimize RMSE with our fitted MARS model, and thus we see a stronger model fit.