Stats 101c Homework 5

Christy Hui

Due 11/5/2021

Problem 1

```
# prep data
college = read.csv("College Fall 2021.csv")
college = college[, -c(1, 2, 3)] # delete useless x variable and categorical predictors
table(is.na(college)) # ensure no NAs
```

```
##
## FALSE
## 34000
```

```
# library(mice)
# college = complete(mice(college)) <- fills in NAs (not needed here but good to have for later assignm
college = scale(college) # scale data set
college = data.frame(college) # ensure college is data frame after scaling
dim(college)
```

```
## [1] 2000   17
```

```
set.seed(1128)
library(caTools)
split = sample(dim(college)[1], 2000 * 0.7,replace = FALSE)
college_train = college[split,]
college_test = college[-split,]
dim(college_train) # ensure split worked
```

```
## [1] 1400   17
```

```
dim(college_test)
```

```
## [1] 600  17
```

Part A

```r
# make least squares model
c_lm = lm(Expend ~., data = college_train)
```

MSE for LM training

```r
lm_training_mse = mean(c_lm$residuals^2)
# lm_training_mse = mean((college_train$Expend - c_lm$fitted.values)^2)
# or
# lm_training_mse = mean((college_train$Expend - predict(c_lm, newdata = college_train))^2)
# all above are equivalent ways of calculating mse
lm_training_mse
```

```
## [1] 0.313138
```

MSE for LM testing

```r
lm_testing_mse = mean((college_test$Expend - predict(c_lm, newdata = college_test))^2)
lm_testing_mse
```

```
## [1] 0.3590488
```

Part B

```r
# make ridge model
x = model.matrix(Expend ~., data = college_train)
y = college_train$Expend
x_test = model.matrix(Expend ~., data = college_test) # for predicting testing
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-2
```

```r
c_ridge = cv.glmnet(x, y, alpha = 0)
c_ridge$lambda.min # best lambda
```

```
## [1] 0.06760678
```

```r
predict(c_ridge, s = c_ridge$lambda.min, type = "coefficients")
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept) -0.008135377
## (Intercept)  .
## Apps         0.253066782
## Accept      -0.117996888
## Enroll       0.013874017
```

```
## Top10perc    0.364093130
## Top25perc   -0.112185648
## F.Undergrad -0.046390090
## P.Undergrad  0.017235797
## Outstate     0.304014206
## Room.Board   0.039763093
## Books        0.032031195
## Personal     0.033556596
## PhD          0.046814305
## Terminal     0.054457345
## S.F.Ratio   -0.257546239
## perc.alumni  0.027585111
## Grad.Rate   -0.057677283
```

Interestingly, ridge regression did not take away any coefficients (and we are left with 18).

MSE for Ridge Training

```
ridge_training_mse = mean((college_train$Expend - predict(c_ridge, s = c_ridge$lambda.min, newx = x))^2)
ridge_training_mse
```

```
## [1] 0.3245715
```

MSE for Ridge Testing

```
ridge_testing_mse = mean((college_test$Expend - predict(c_ridge, s = c_ridge$lambda.min, newx = x_test))
ridge_testing_mse
```

```
## [1] 0.3762057
```

Part C

```
# make lasso model
c_lasso = cv.glmnet(x, y, alpha = 1)
c_lasso$lambda.min # best lambda
```

```
## [1] 0.0003959743
```

```
predict(c_lasso, s = c_lasso$lambda.min, type = "coefficients")
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) -0.006833507
## (Intercept)  .
## Apps         0.495154095
## Accept      -0.409216442
## Enroll       0.135209107
## Top10perc    0.440495852
## Top25perc   -0.228102668
## F.Undergrad -0.093724954
```

```
## P.Undergrad  0.015358318
## Outstate     0.362146449
## Room.Board   0.011749462
## Books        0.027617851
## Personal     0.031343997
## PhD          0.037751457
## Terminal     0.072264214
## S.F.Ratio   -0.254801337
## perc.alumni  0.009086860
## Grad.Rate   -0.074491386
```

Interestingly, lasso regression also did not take away any coefficients (and we are left with 18).

MSE for Lasso Training

```
lasso_training_mse = mean((college_train$Expend - predict(c_lasso, s = c_lasso$lambda.min, newx = x))^2
lasso_training_mse
```

```
## [1] 0.3131833
```

MSE for Lasso Testing

```
lasso_testing_mse = mean((college_test$Expend - predict(c_lasso, s = c_lasso$lambda.min, newx = x_test)
lasso_testing_mse
```

```
## [1] 0.3590919
```

Problem 2

Part A

```
library(pls)
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
c_pcr = pcr(Expend ~., data = college_train, scale = TRUE, validation = "CV")
summary(c_pcr)
```

```
## Data:    X dimension: 1400 16
##  Y dimension: 1400 1
## Fit method: svdpc
## Number of components considered: 16
##
## VALIDATION: RMSEP
```

```
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.9939   0.8253   0.6818   0.6656   0.6650   0.6436   0.6358
## adjCV       0.9939   0.8251   0.6817   0.6654   0.6653   0.6433   0.6356
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.6363   0.6192   0.6199    0.6190    0.6072    0.6044    0.6067
## adjCV   0.6361   0.6189   0.6196    0.6186    0.6068    0.6040    0.6065
##        14 comps  15 comps  16 comps
## CV       0.5763    0.5732    0.5706
## adjCV    0.5758    0.5727    0.5700
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         32.10    59.47    66.61    72.39    77.80    82.81    86.64    90.05
## Expend    31.35    53.14    55.51    55.54    58.28    59.50    59.55    61.70
##         9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X         93.06     95.65     97.16     98.16     99.03     99.66     99.88
## Expend    61.79     62.02     63.57     63.99     64.07     67.43     67.91
##         16 comps
## X         100.00
## Expend     68.26
```

We use 85% as our threshold in variation. Because 7 principal components explains 86.64% of the variance, but 6 principal components explain 82.21%, we conclude that M = 7. Thus, we have reduced the dimensionality of predictors from 16 to 7.

MSE for PCR training using 7 principal components

```
pcr_training_mse = mean((college_train$Expend - predict(c_pcr, newdata = college_train, ncomp = 7))^2)
pcr_training_mse
```

```
## [1] 0.3990467
```

MSE for PCR testing using 7 principal components

```
pcr_testing_mse = mean((college_test$Expend - predict(c_pcr, newdata = college_test, ncomp = 7))^2)
pcr_testing_mse
```

```
## [1] 0.4622925
```

Part B

```
library(pls)
c_pls = plsr(Expend ~., data = college_train, scale = TRUE, validation = "CV")
summary(c_pls)
```

```
## Data:    X dimension: 1400 16
##  Y dimension: 1400 1
## Fit method: kernelpls
## Number of components considered: 16
```

```
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.9939   0.6628   0.6296   0.6034   0.5892   0.5792   0.5750
## adjCV       0.9939   0.6626   0.6294   0.6030   0.5887   0.5786   0.5744
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.5729   0.5728   0.5722    0.5714    0.5709    0.5709    0.5709
## adjCV   0.5723   0.5722   0.5716    0.5707    0.5703    0.5703    0.5703
##        14 comps  15 comps  16 comps
## CV       0.5711    0.5709    0.5708
## adjCV    0.5705    0.5702    0.5702
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         30.38    50.25    64.89    69.15    72.77    75.99    78.11    82.18
## Expend    55.97    60.55    63.97    65.97    67.22    67.73    67.97    68.03
##         9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X         84.23     86.64     90.96     92.93     95.78     97.52     99.13
## Expend    68.13     68.21     68.23     68.24     68.25     68.25     68.26
##         16 comps
## X         100.00
## Expend     68.26
```

We use 85% as our threshold in variation. Because 10 principal components explains 86.64% of the variance, but 9 principal components explain 84.23%, we conclude that M = 10. Thus, we have reduced the dimensionality of predictors from 16 to 10.

MSE for PLS training using 10 principal components

```
pls_training_mse = mean((college_train$Expend - predict(c_pls, newdata = college_train, ncomp = 10))^2)
pls_training_mse
```

```
## [1] 0.3135909
```

MSE for PLS testing using 10 principal components

```
pls_testing_mse = mean((college_test$Expend - predict(c_pls, newdata = college_test, ncomp = 10))^2)
pls_testing_mse
```

```
## [1] 0.3581315
```

Problem 3

Part A

```
c_step = step(c_lm,
              direction = "backward",
              k = log(nrow(college_train)))
```

```
## Start:  AIC=-1502.4
## Expend ~ Apps + Accept + Enroll + Top10perc + Top25perc + F.Undergrad +
##     P.Undergrad + Outstate + Room.Board + Books + Personal +
##     PhD + Terminal + S.F.Ratio + perc.alumni + Grad.Rate
##
##               Df Sum of Sq    RSS     AIC
## - perc.alumni  1     0.061 438.45 -1509.5
## - Room.Board   1     0.106 438.50 -1509.3
## - P.Undergrad  1     0.222 438.61 -1508.9
## - PhD          1     0.508 438.90 -1508.0
## - Books        1     0.837 439.23 -1507.0
## - F.Undergrad  1     0.873 439.27 -1506.9
## - Personal     1     1.116 439.51 -1506.1
## - Enroll       1     1.411 439.80 -1505.2
## - Terminal     1     1.950 440.34 -1503.4
## <none>                     438.39 -1502.4
## - Grad.Rate    1     4.228 442.62 -1496.2
## - Accept       1    12.448 450.84 -1470.5
## - Top25perc    1    12.575 450.97 -1470.1
## - Apps         1    29.173 467.57 -1419.5
## - Top10perc    1    37.176 475.57 -1395.7
## - Outstate     1    53.166 491.56 -1349.4
## - S.F.Ratio    1    54.269 492.66 -1346.3
##
## Step:  AIC=-1509.45
## Expend ~ Apps + Accept + Enroll + Top10perc + Top25perc + F.Undergrad +
##     P.Undergrad + Outstate + Room.Board + Books + Personal +
##     PhD + Terminal + S.F.Ratio + Grad.Rate
##
##               Df Sum of Sq    RSS     AIC
## - Room.Board   1     0.094 438.55 -1516.4
## - P.Undergrad  1     0.227 438.68 -1516.0
## - PhD          1     0.506 438.96 -1515.1
## - Books        1     0.831 439.29 -1514.0
## - F.Undergrad  1     0.916 439.37 -1513.8
## - Personal     1     1.078 439.53 -1513.3
## - Enroll       1     1.494 439.95 -1511.9
## - Terminal     1     1.994 440.45 -1510.3
## <none>                     438.45 -1509.5
## - Grad.Rate    1     4.183 442.64 -1503.4
## - Top25perc    1    12.549 451.00 -1477.2
## - Accept       1    12.636 451.09 -1476.9
## - Apps         1    29.113 467.57 -1426.7
## - Top10perc    1    37.721 476.17 -1401.2
## - S.F.Ratio    1    54.814 493.27 -1351.8
## - Outstate     1    57.288 495.74 -1344.8
##
## Step:  AIC=-1516.4
## Expend ~ Apps + Accept + Enroll + Top10perc + Top25perc + F.Undergrad +
##     P.Undergrad + Outstate + Books + Personal + PhD + Terminal +
##     S.F.Ratio + Grad.Rate
##
##               Df Sum of Sq    RSS     AIC
## - P.Undergrad  1     0.257 438.80 -1522.8
```

```
## - PhD          1     0.489 439.04 -1522.1
## - Books        1     0.856 439.40 -1520.9
## - F.Undergrad  1     0.906 439.45 -1520.8
## - Personal     1     1.053 439.60 -1520.3
## - Enroll       1     1.435 439.98 -1519.1
## - Terminal     1     2.152 440.70 -1516.8
## <none>                    438.55 -1516.4
## - Grad.Rate    1     4.101 442.65 -1510.6
## - Accept       1    12.585 451.13 -1484.0
## - Top25perc    1    12.691 451.24 -1483.7
## - Apps         1    29.741 468.29 -1431.8
## - Top10perc    1    37.789 476.34 -1407.9
## - S.F.Ratio    1    54.906 493.45 -1358.5
## - Outstate     1    70.372 508.92 -1315.3
##
## Step:  AIC=-1522.82
## Expend ~ Apps + Accept + Enroll + Top10perc + Top25perc + F.Undergrad +
##     Outstate + Books + Personal + PhD + Terminal + S.F.Ratio +
##     Grad.Rate
##
##               Df Sum of Sq    RSS     AIC
## - PhD          1     0.559 439.36 -1528.3
## - F.Undergrad  1     0.743 439.55 -1527.7
## - Books        1     0.924 439.73 -1527.1
## - Personal     1     1.206 440.01 -1526.2
## - Enroll       1     1.430 440.23 -1525.5
## - Terminal     1     2.139 440.94 -1523.3
## <none>                    438.80 -1522.8
## - Grad.Rate    1     4.512 443.32 -1515.8
## - Top25perc    1    12.694 451.50 -1490.1
## - Accept       1    13.287 452.09 -1488.3
## - Apps         1    30.624 469.43 -1435.6
## - Top10perc    1    37.650 476.45 -1414.8
## - S.F.Ratio    1    55.230 494.03 -1364.1
## - Outstate     1    71.792 510.60 -1317.9
##
## Step:  AIC=-1528.29
## Expend ~ Apps + Accept + Enroll + Top10perc + Top25perc + F.Undergrad +
##     Outstate + Books + Personal + Terminal + S.F.Ratio + Grad.Rate
##
##               Df Sum of Sq    RSS     AIC
## - Books        1     0.723 440.09 -1533.2
## - F.Undergrad  1     0.737 440.10 -1533.2
## - Personal     1     1.208 440.57 -1531.7
## - Enroll       1     1.413 440.78 -1531.0
## <none>                    439.36 -1528.3
## - Grad.Rate    1     4.422 443.79 -1521.5
## - Terminal     1     8.973 448.34 -1507.2
## - Top25perc    1    12.730 452.09 -1495.5
## - Accept       1    12.990 452.35 -1494.7
## - Apps         1    30.406 469.77 -1441.8
## - Top10perc    1    40.042 479.40 -1413.4
## - S.F.Ratio    1    54.697 494.06 -1371.3
## - Outstate     1    72.256 511.62 -1322.4
```

```
##
## Step:  AIC=-1533.23
## Expend ~ Apps + Accept + Enroll + Top10perc + Top25perc + F.Undergrad +
##     Outstate + Personal + Terminal + S.F.Ratio + Grad.Rate
##
##               Df Sum of Sq    RSS     AIC
## - F.Undergrad  1     0.729 440.81 -1538.2
## - Enroll       1     1.401 441.49 -1536.0
## - Personal     1     1.598 441.68 -1535.4
## <none>                     440.09 -1533.2
## - Grad.Rate    1     4.584 444.67 -1526.0
## - Terminal     1     9.235 449.32 -1511.4
## - Top25perc    1    12.497 452.58 -1501.3
## - Accept       1    13.152 453.24 -1499.2
## - Apps         1    30.844 470.93 -1445.6
## - Top10perc    1    40.516 480.60 -1417.2
## - S.F.Ratio    1    54.307 494.39 -1377.6
## - Outstate     1    72.676 512.76 -1326.5
##
## Step:  AIC=-1538.16
## Expend ~ Apps + Accept + Enroll + Top10perc + Top25perc + Outstate +
##     Personal + Terminal + S.F.Ratio + Grad.Rate
##
##             Df Sum of Sq    RSS     AIC
## - Enroll     1     0.707 441.52 -1543.2
## - Personal   1     1.356 442.17 -1541.1
## <none>                   440.81 -1538.2
## - Grad.Rate  1     4.241 445.06 -1532.0
## - Terminal   1     8.777 449.59 -1517.8
## - Accept     1    12.758 453.57 -1505.5
## - Top25perc  1    13.518 454.33 -1503.1
## - Apps       1    30.142 470.96 -1452.8
## - Top10perc  1    42.770 483.59 -1415.8
## - S.F.Ratio  1    55.255 496.07 -1380.1
## - Outstate   1    74.937 515.75 -1325.6
##
## Step:  AIC=-1543.16
## Expend ~ Apps + Accept + Top10perc + Top25perc + Outstate + Personal +
##     Terminal + S.F.Ratio + Grad.Rate
##
##             Df Sum of Sq    RSS     AIC
## - Personal   1     1.641 443.16 -1545.2
## <none>                   441.52 -1543.2
## - Grad.Rate  1     4.440 445.96 -1536.4
## - Terminal   1     9.102 450.62 -1521.8
## - Top25perc  1    14.143 455.66 -1506.3
## - Accept     1    16.282 457.80 -1499.7
## - Apps       1    29.451 470.97 -1460.0
## - Top10perc  1    48.628 490.15 -1404.1
## - S.F.Ratio  1    54.696 496.22 -1386.9
## - Outstate   1    77.948 519.47 -1322.8
##
## Step:  AIC=-1545.21
## Expend ~ Apps + Accept + Top10perc + Top25perc + Outstate + Terminal +
```

```
##     S.F.Ratio + Grad.Rate
##
##             Df Sum of Sq    RSS     AIC
## <none>                   443.16 -1545.2
## - Grad.Rate  1     5.501 448.66 -1535.2
## - Terminal   1     9.247 452.41 -1523.5
## - Top25perc  1    14.533 457.70 -1507.3
## - Accept     1    15.785 458.95 -1503.5
## - Apps       1    29.757 472.92 -1461.5
## - Top10perc  1    50.087 493.25 -1402.5
## - S.F.Ratio  1    57.890 501.05 -1380.6
## - Outstate   1    76.680 519.84 -1329.0
```

According to BIC and backwards stepwise regression, the satisfactory number of predictors is 8. These predictors are:

Grad.Rate

Terminal

Top25perc

Accept

Apps

Top10perc

S.F.Ratio

Outstate

We now want to create a lm model with only these 8 predictors in order to see how well the step() function's predictor reduction does.

```
c_step_lm = lm(Expend ~ Grad.Rate + Terminal + Top25perc + Accept + Apps + Top10perc + S.F.Ratio + Outs
```

MSE for Step Function LM Training

```
step_training_mse = mean(c_step_lm$residuals^2)
step_training_mse
```

```
## [1] 0.3165448
```

MSE for Step Function LM Testing

```
step_testing_mse = mean((college_test$Expend - predict(c_step_lm, newdata = college_test))^2)
step_testing_mse
```

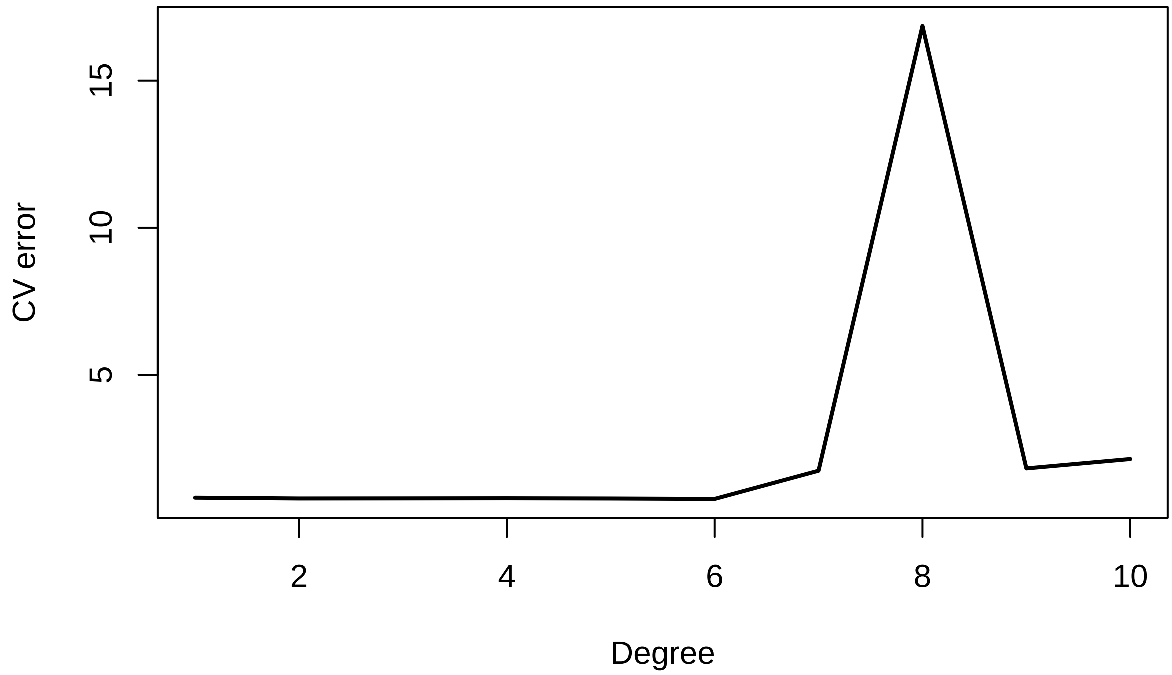
```
## [1] 0.3606349
```

Part B

Now that we have our 8 predictors, our mission is to now fit a GAM model (spline) using the bs() function in R. But to find the "best" or "close to best" bs(), we need to make sure we have the polynomials that give the best MSE. In this case, we will find the polynomails with the best MSE with 10-folds cross validation.

For Grad.Rate:

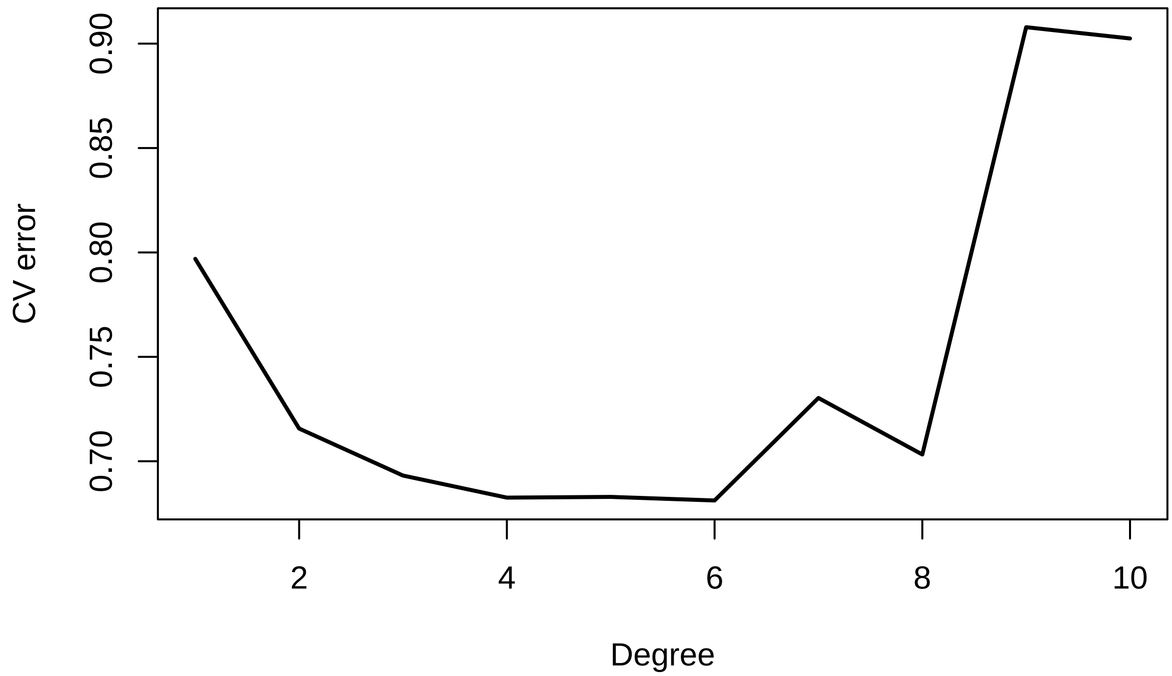
```r
library(boot)
all.deltas = rep(NA, 10)
for (i in 1:10) {
 glm.fit = glm(Expend ~ poly(Grad.Rate, i), data = college_train)
 all.deltas[i] = cv.glm(college_train, glm.fit, K = 10)$delta[2]
}
plot(1:10, all.deltas, xlab = "Degree", ylab = "CV error", type = "l", pch = 20, lwd = 2)
```



Grad.Rate seems to do well with only a function of only 1 degree. This means that when doing our Spline function, we will choose the degree of the function to be 1.

For Terminal:

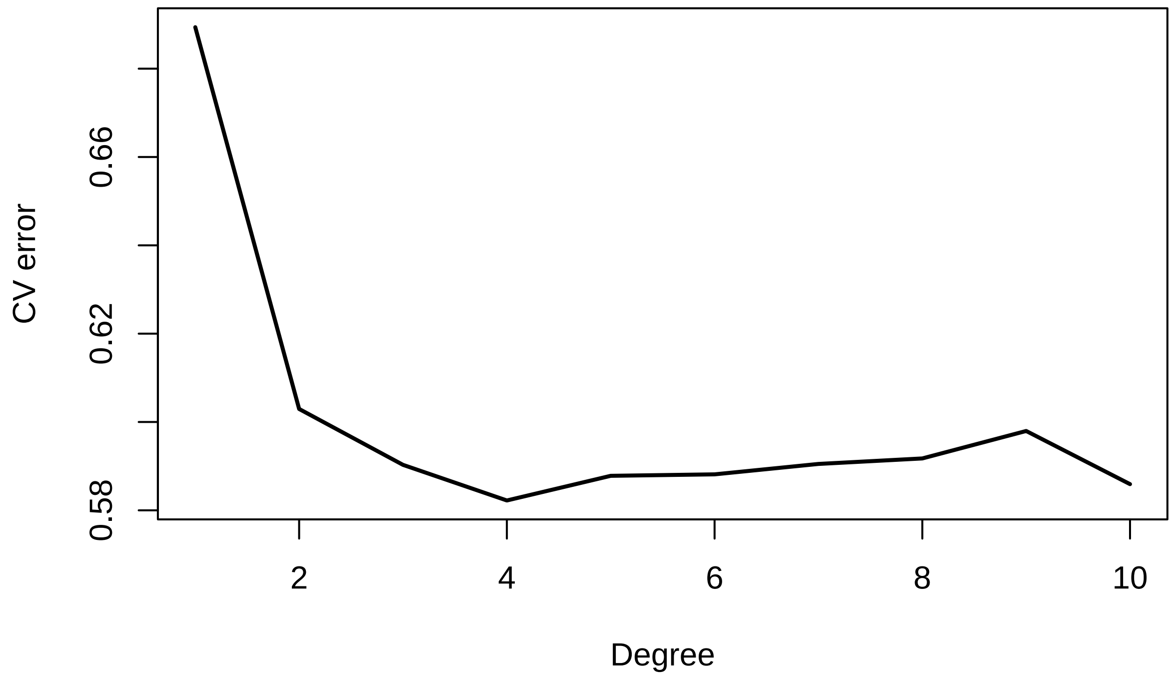
```r
library(boot)
all.deltas = rep(NA, 10)
for (i in 1:10) {
 glm.fit = glm(Expend ~ poly(Terminal, i), data = college_train)
 all.deltas[i] = cv.glm(college_train, glm.fit, K = 10)$delta[2]
}
plot(1:10, all.deltas, xlab = "Degree", ylab = "CV error", type = "l", pch = 20, lwd = 2)
```



For the "Terminal" predictor, we can see that the lowest MSE comes from a degree of around 4. Thus, when doing our Spline function for "Terminal," we will choose the degree of the function to be 4.

For Top25perc

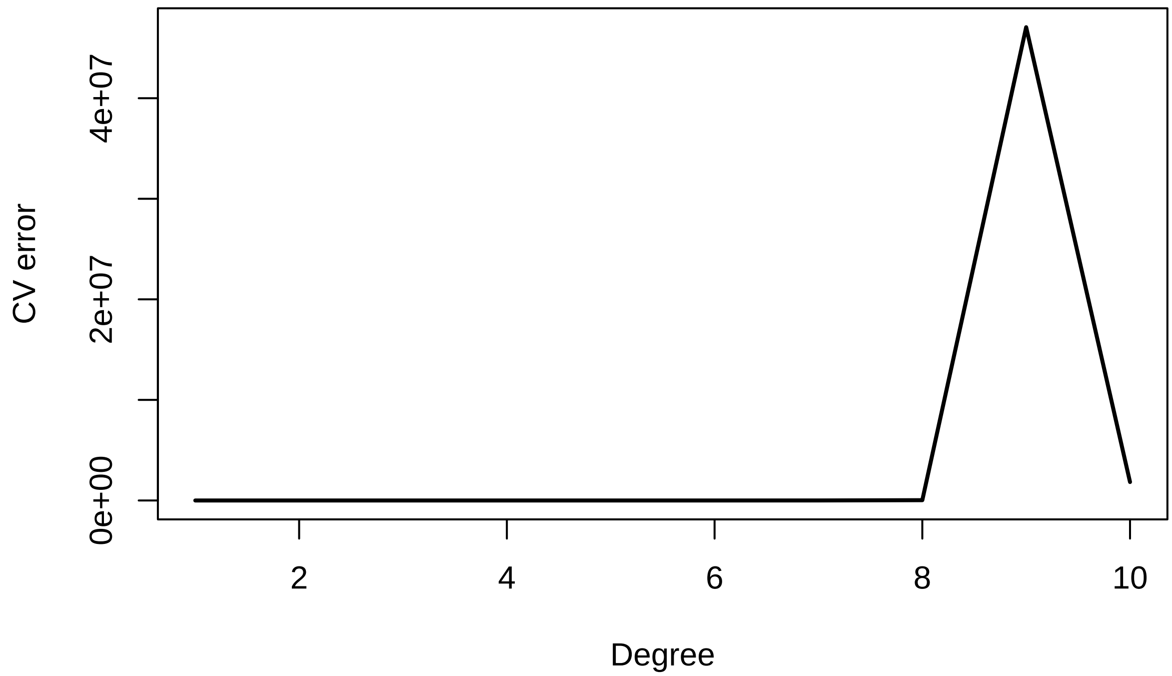
```
library(boot)
all.deltas = rep(NA, 10)
for (i in 1:10) {
 glm.fit = glm(Expend ~ poly(Top25perc, i), data = college_train)
 all.deltas[i] = cv.glm(college_train, glm.fit, K = 10)$delta[2]
}
plot(1:10, all.deltas, xlab = "Degree", ylab = "CV error", type = "l", pch = 20, lwd = 2)
```



When plotting the MSE for the different polynomials, we see that a degree of 5 seems to be the best for the "Top25perc" variable. Thus, we will choose a degree of 5.

For Accept:

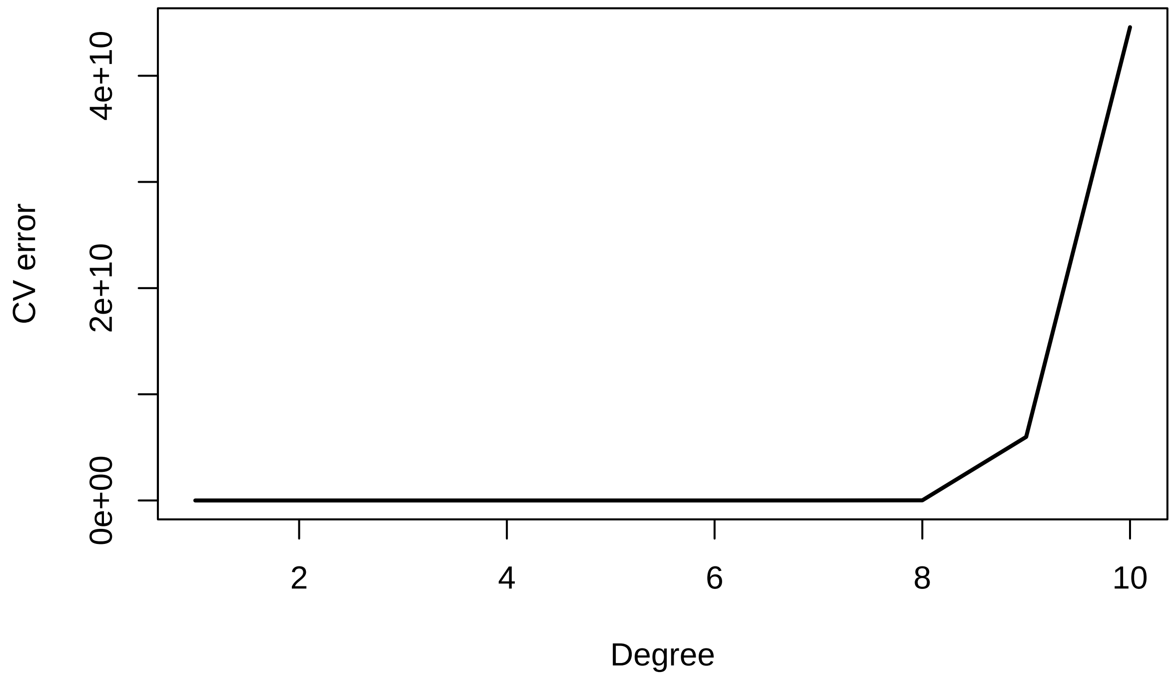
```
library(boot)
all.deltas = rep(NA, 10)
for (i in 1:10) {
 glm.fit = glm(Expend ~ poly(Accept, i), data = college_train)
 all.deltas[i] = cv.glm(college_train, glm.fit, K = 10)$delta[2]
}
plot(1:10, all.deltas, xlab = "Degree", ylab = "CV error", type = "l", pch = 20, lwd = 2)
```



Interestingly, it seems as though a degree of 1 yields the best MSE for the "Accept" predictor (just like Grad.Rate). Thus, we will choose the degree for the "Accept" predictor to be 1.

For Apps:

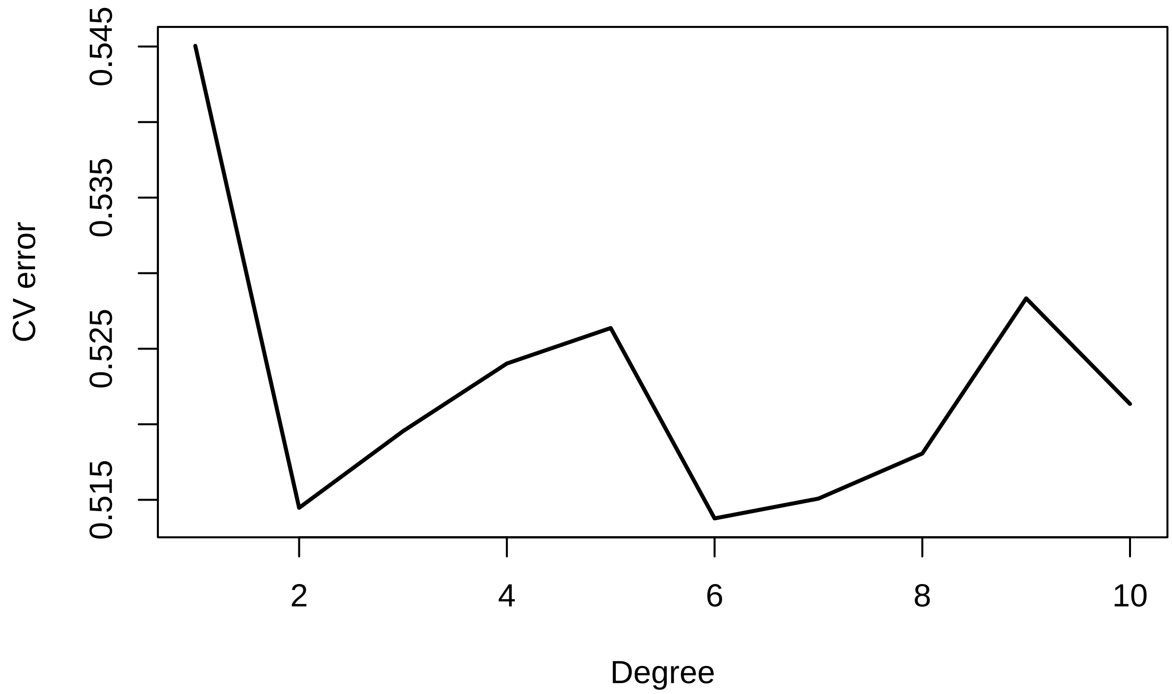
```
library(boot)
all.deltas = rep(NA, 10)
for (i in 1:10) {
 glm.fit = glm(Expend ~ poly(Apps, i), data = college_train)
 all.deltas[i] = cv.glm(college_train, glm.fit, K = 10)$delta[2]
}
plot(1:10, all.deltas, xlab = "Degree", ylab = "CV error", type = "l", pch = 20, lwd = 2)
```



Similarly to the "Grad.Rate" and "Accept" predictors, the "Apps" predictor also seems to yield the best MSE when it has a polynomial of degree 1. We will choose the degree equal to 1.

For Top10perc:

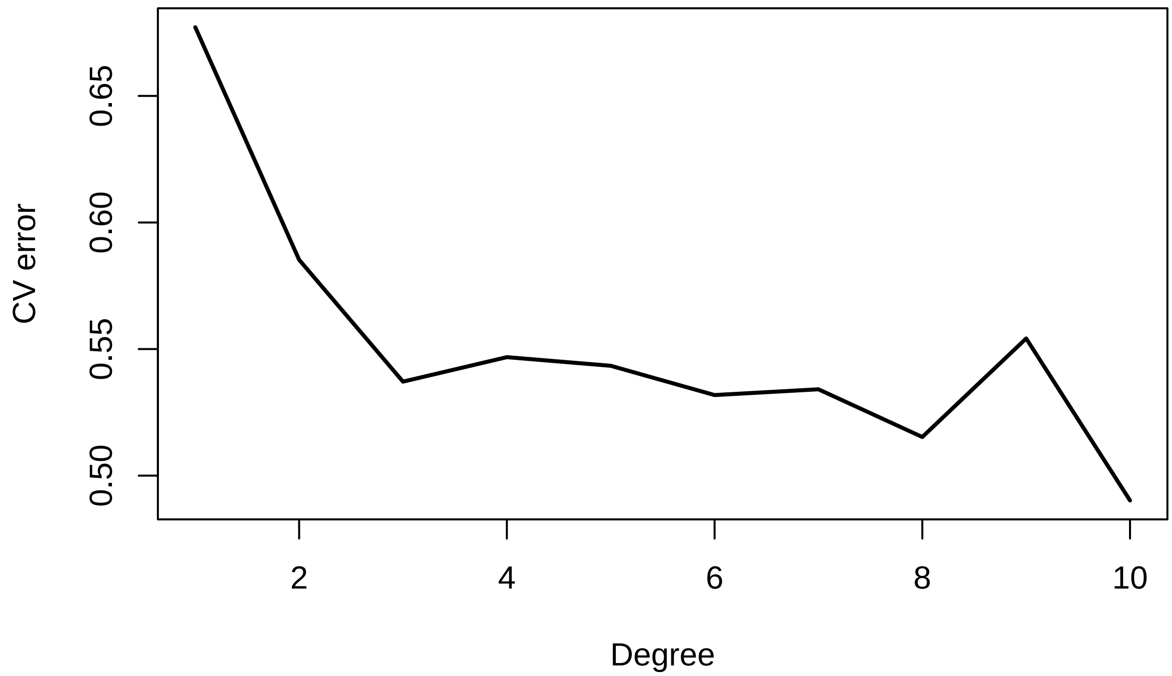
```
library(boot)
all.deltas = rep(NA, 10)
for (i in 1:10) {
 glm.fit = glm(Expend ~ poly(Top10perc, i), data = college_train)
 all.deltas[i] = cv.glm(college_train, glm.fit, K = 10)$delta[2]
}
plot(1:10, all.deltas, xlab = "Degree", ylab = "CV error", type = "l", pch = 20, lwd = 2)
```



The MSE for Top10perc seems to be lowest when the degree of the polynomial is 9. Thus, we will choose the degree to be 9 when doing GAM.

For S.F.Ratio:

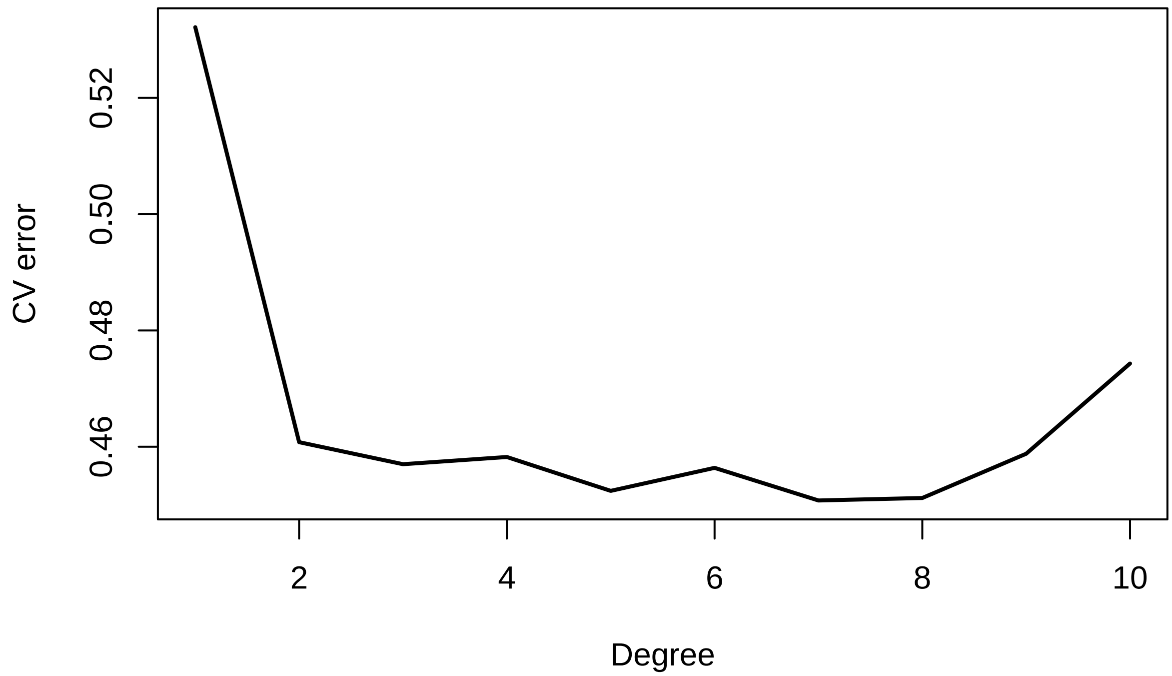
```
library(boot)
all.deltas = rep(NA, 10)
for (i in 1:10) {
 glm.fit = glm(Expend ~ poly(S.F.Ratio, i), data = college_train)
 all.deltas[i] = cv.glm(college_train, glm.fit, K = 10)$delta[2]
}
plot(1:10, all.deltas, xlab = "Degree", ylab = "CV error", type = "l", pch = 20, lwd = 2)
```



For the S.F.Ratio, the lowest MSE seems to be when the degree equals to 9. Thus, we will choose the 9th degree polynomial.

For Outstate:

```
library(boot)
all.deltas = rep(NA, 10)
for (i in 1:10) {
 glm.fit = glm(Expend ~ poly(Outstate, i), data = college_train)
 all.deltas[i] = cv.glm(college_train, glm.fit, K = 10)$delta[2]
}
plot(1:10, all.deltas, xlab = "Degree", ylab = "CV error", type = "l", pch = 20, lwd = 2)
```



A degree of 1 (like Grad.Rate and other predictors) has the lowest MSE for the Outstate predictor. So we will choose a degree of 1.

Now that we have all of the best degrees for each of the 8 strongest predictors, we will create the Spline function.

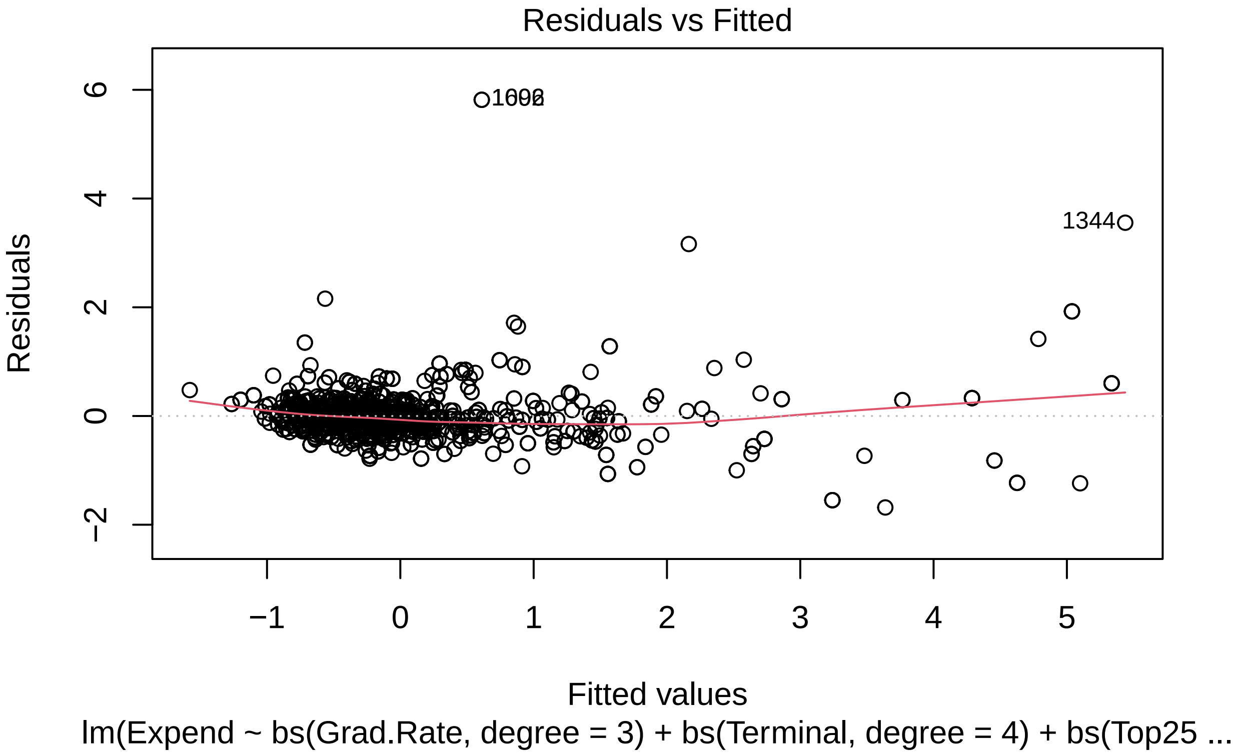
Note: In bs(), we can specify the degree. Since bs() does only cubic splines, the degrees of the predictors that we agreed upon to only be 1 will be 3.

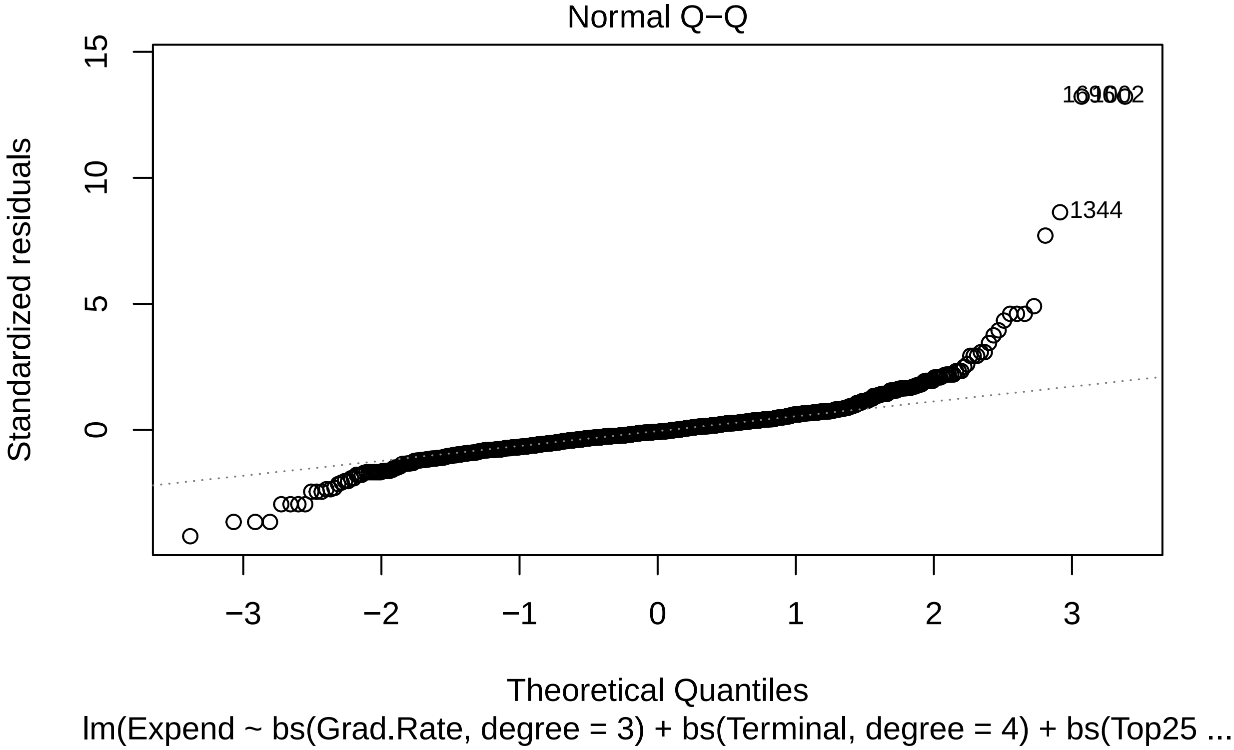
```
library(splines)
c_spline = lm(Expend ~ bs(Grad.Rate, degree = 3) + bs(Terminal, degree = 4) + bs(Top25perc, degree = 5)
              data = college_train)
summary(c_spline)
```

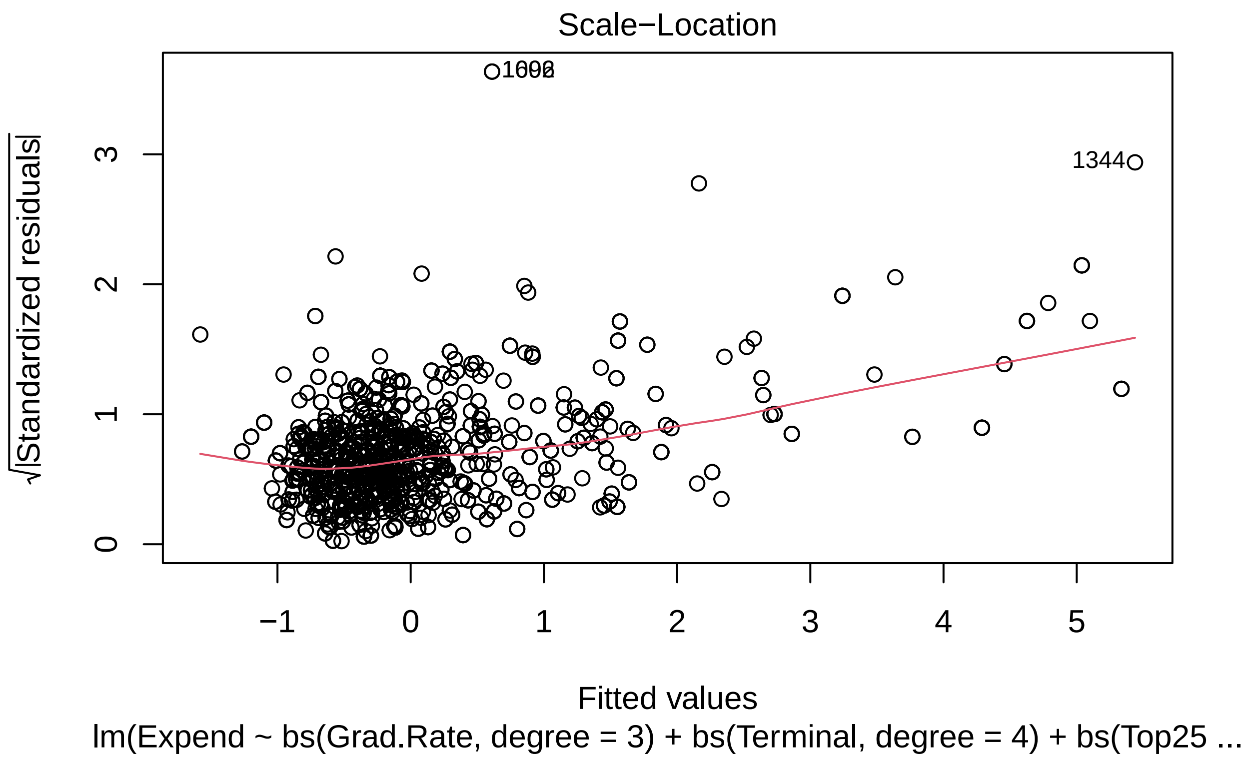
```
##
## Call:
## lm(formula = Expend ~ bs(Grad.Rate, degree = 3) + bs(Terminal,
##     degree = 4) + bs(Top25perc, degree = 5) + bs(Accept, degree = 3) +
##     bs(Apps, degree = 3) + bs(Top10perc, degree = 9) + bs(S.F.Ratio,
##     degree = 9) + bs(Outstate, degree = 3), data = college_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.6824 -0.1970 -0.0357  0.1478  5.8164
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     0.27280    0.47109   0.579 0.562624
## bs(Grad.Rate, degree = 3)1      0.39329    0.29222   1.346 0.178577
## bs(Grad.Rate, degree = 3)2     -0.16938    0.19097  -0.887 0.375277
## bs(Grad.Rate, degree = 3)3     -0.20388    0.28951  -0.704 0.481410
## bs(Terminal, degree = 4)1      -0.31104    0.65195  -0.477 0.633372
## bs(Terminal, degree = 4)2      -0.16868    0.26875  -0.628 0.530342
```

```
## bs(Terminal, degree = 4)3    -0.06023    0.40516  -0.149 0.881850
## bs(Terminal, degree = 4)4     0.15101    0.29166   0.518 0.604711
## bs(Top25perc, degree = 5)1   -0.89285    0.69314  -1.288 0.197917
## bs(Top25perc, degree = 5)2    0.70858    0.55778   1.270 0.204173
## bs(Top25perc, degree = 5)3   -1.92819    0.80405  -2.398 0.016615 *
## bs(Top25perc, degree = 5)4    0.65032    0.45728   1.422 0.155207
## bs(Top25perc, degree = 5)5   -0.45384    0.34995  -1.297 0.194902
## bs(Accept, degree = 3)1      -1.92656    0.61298  -3.143 0.001709 **
## bs(Accept, degree = 3)2      -0.25793    1.05632  -0.244 0.807126
## bs(Accept, degree = 3)3      -1.03424    2.10031  -0.492 0.622502
## bs(Apps, degree = 3)1         3.19347    0.69571   4.590 4.84e-06 ***
## bs(Apps, degree = 3)2        -0.80563    1.23607  -0.652 0.514664
## bs(Apps, degree = 3)3         1.64430    2.15156   0.764 0.444859
## bs(Top10perc, degree = 9)1    1.29028    0.98090   1.315 0.188597
## bs(Top10perc, degree = 9)2   -3.13773    2.41323  -1.300 0.193746
## bs(Top10perc, degree = 9)3    8.27142    5.81403   1.423 0.155062
## bs(Top10perc, degree = 9)4  -12.23115    9.28972  -1.317 0.188183
## bs(Top10perc, degree = 9)5   16.59192   11.15838   1.487 0.137260
## bs(Top10perc, degree = 9)6  -17.04990    9.04016  -1.886 0.059505 .
## bs(Top10perc, degree = 9)7   14.46730    4.86896   2.971 0.003017 **
## bs(Top10perc, degree = 9)8   -5.69586    1.40761  -4.046 5.49e-05 ***
## bs(Top10perc, degree = 9)9    2.88164    0.34427   8.370  < 2e-16 ***
## bs(S.F.Ratio, degree = 9)1   23.90077    1.93616  12.344  < 2e-16 ***
## bs(S.F.Ratio, degree = 9)2  -95.05903    7.11482 -13.361  < 2e-16 ***
## bs(S.F.Ratio, degree = 9)3  240.97470   21.73407  11.087  < 2e-16 ***
## bs(S.F.Ratio, degree = 9)4 -475.18167   49.38067  -9.623  < 2e-16 ***
## bs(S.F.Ratio, degree = 9)5  742.14988   89.90122   8.255 3.55e-16 ***
## bs(S.F.Ratio, degree = 9)6 -938.87149  129.56858  -7.246 7.16e-13 ***
## bs(S.F.Ratio, degree = 9)7  915.05740  143.23273   6.389 2.29e-10 ***
## bs(S.F.Ratio, degree = 9)8 -591.84085  102.67160  -5.764 1.01e-08 ***
## bs(S.F.Ratio, degree = 9)9   -0.87379    0.38243  -2.285 0.022475 *
## bs(Outstate, degree = 3)1    -0.25401    0.23538  -1.079 0.280708
## bs(Outstate, degree = 3)2     0.54465    0.16149   3.373 0.000766 ***
## bs(Outstate, degree = 3)3     1.76077    0.20092   8.764  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4433 on 1360 degrees of freedom
## Multiple R-squared:  0.8065, Adjusted R-squared:  0.8009
## F-statistic: 145.3 on 39 and 1360 DF,  p-value: < 2.2e-16
```

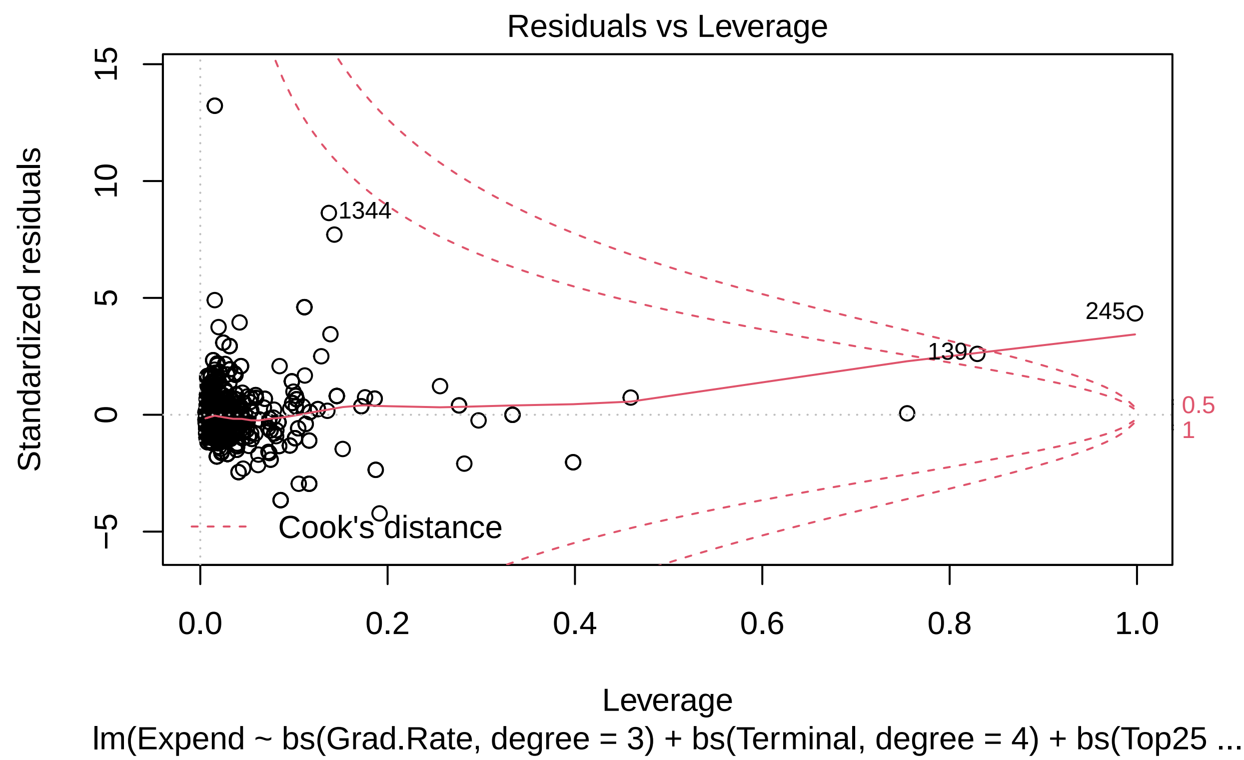
```
plot(c_spline)
```


Residuals vs Fitted
Residuals
Fitted values
lm(Expend ~ bs(Grad.Rate, degree = 3) + bs(Terminal, degree = 4) + bs(Top25 ...
1344
6
4
2
0
−2
−1
0
1
2
3
4
5


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Expend ~ bs(Grad.Rate, degree = 3) + bs(Terminal, degree = 4) + bs(Top25 ...
1344
15
10
5
0
−3
−2
−1
0
1
2
3




```
## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced

## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced
```



From our plots, we see some deviation from residual and error points; overall, however, our plot does a decent job. Looking at the first plot between the fitted values vs the residuals, we can see that the plot bounces around the red line, indicating a good relationship between our line and points. In the second plot, a good proportion of points lie on the line, indicating a well-behaved plot. Also, the third plot is well-behaved (or a little bit well-behaved) because it demonstrates some sort of linearity and relationship between the points. Thus, conclude that our model did a decent job in fitting.

Part C

MSE for GAM Training

```
gam_training_mse = mean((college_train$Expend - predict(c_spline, newdata = college_train))^2)
gam_training_mse
```

```
## [1] 0.1909127
```

MSE for GAM Testing

```
gam_testing_mse = mean((college_test$Expend - predict(c_spline, newdata = college_test))^2)
```

```
## Warning in bs(Grad.Rate, degree = 3L, knots = numeric(0), Boundary.knots =
## c(-2.81019868692296, : some 'x' values beyond boundary knots may cause ill-
## conditioned bases
```

```
gam_testing_mse
```

```
## [1] 0.3156828
```

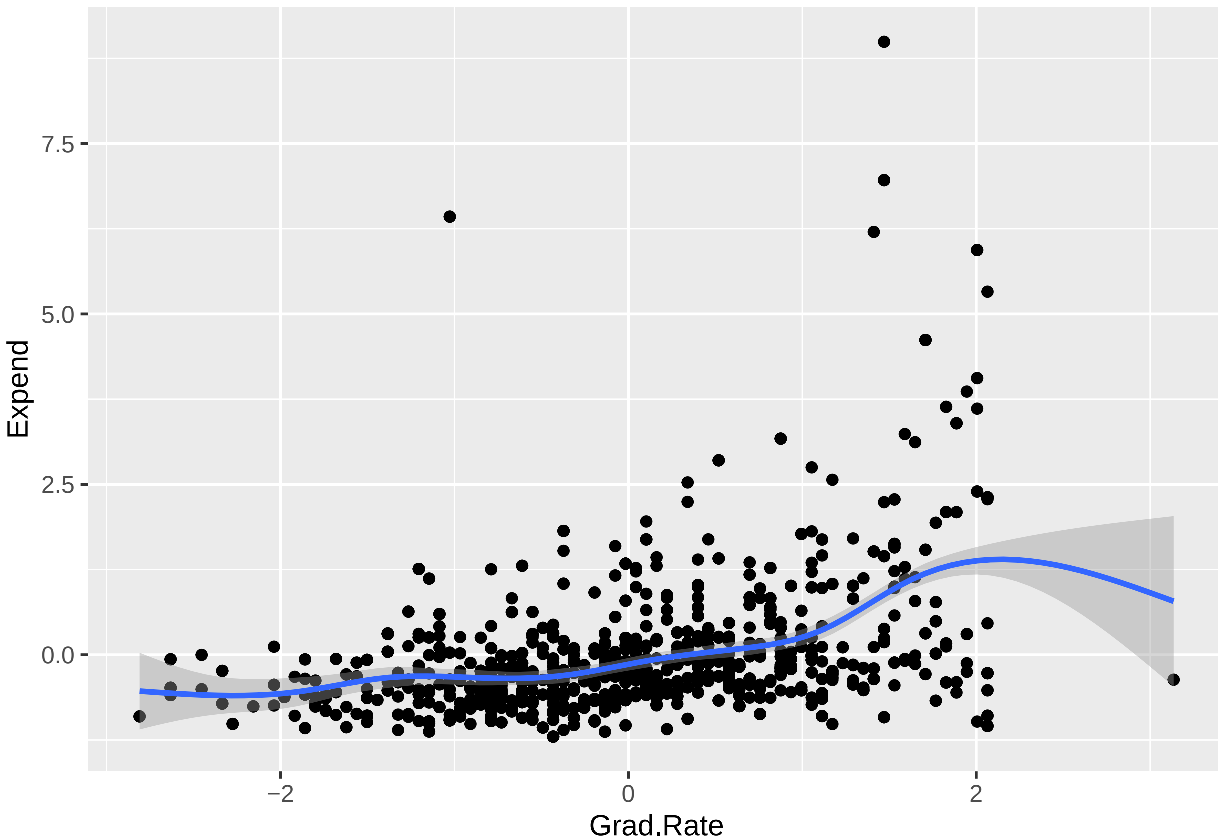
Above, we see that the MSE for the testing data is 0.3156828, which is higher than the training (0.1909127). This makes sense, since the model fits under the training data set (quite well, in fact, compared to our other models).

Part D

To answer this question, it may be more clear if we look at the plots.

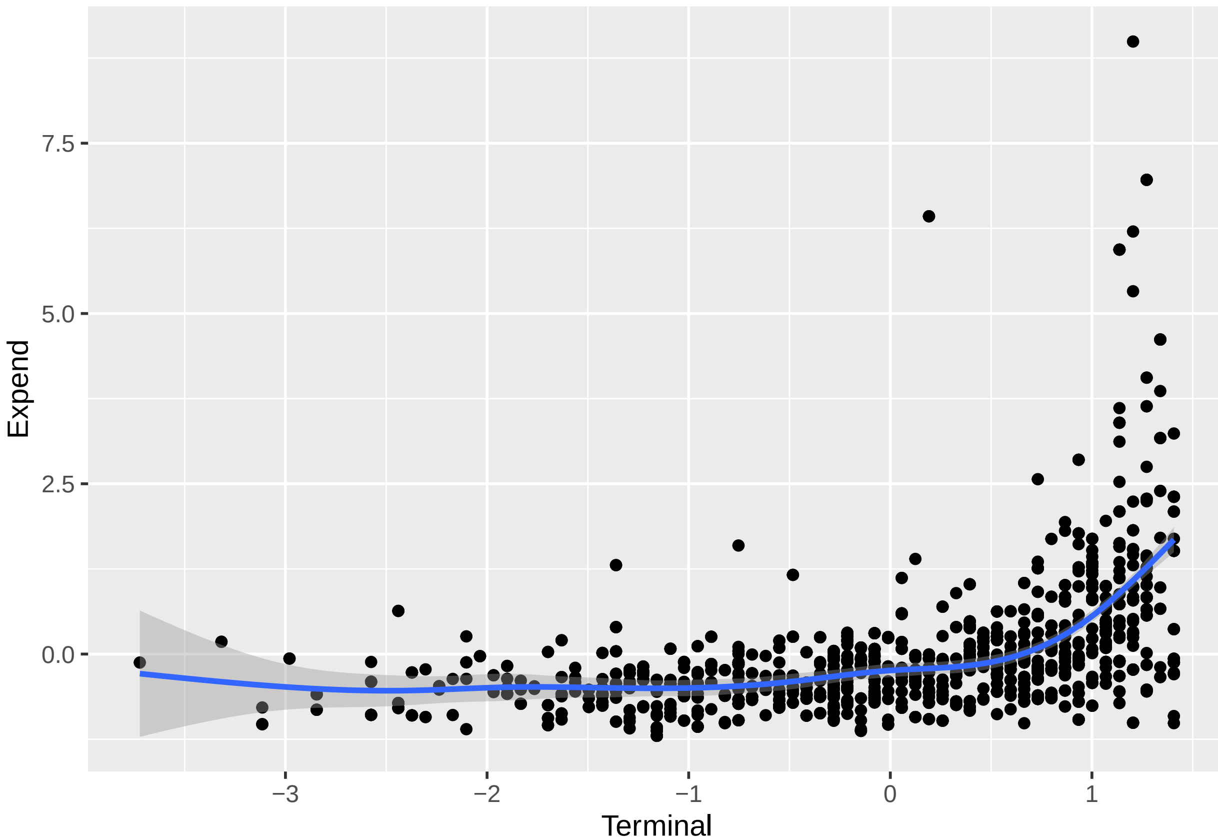
```
library(ggplot2)
ggplot(data = college_train, aes(Grad.Rate, Expend)) + geom_point() + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



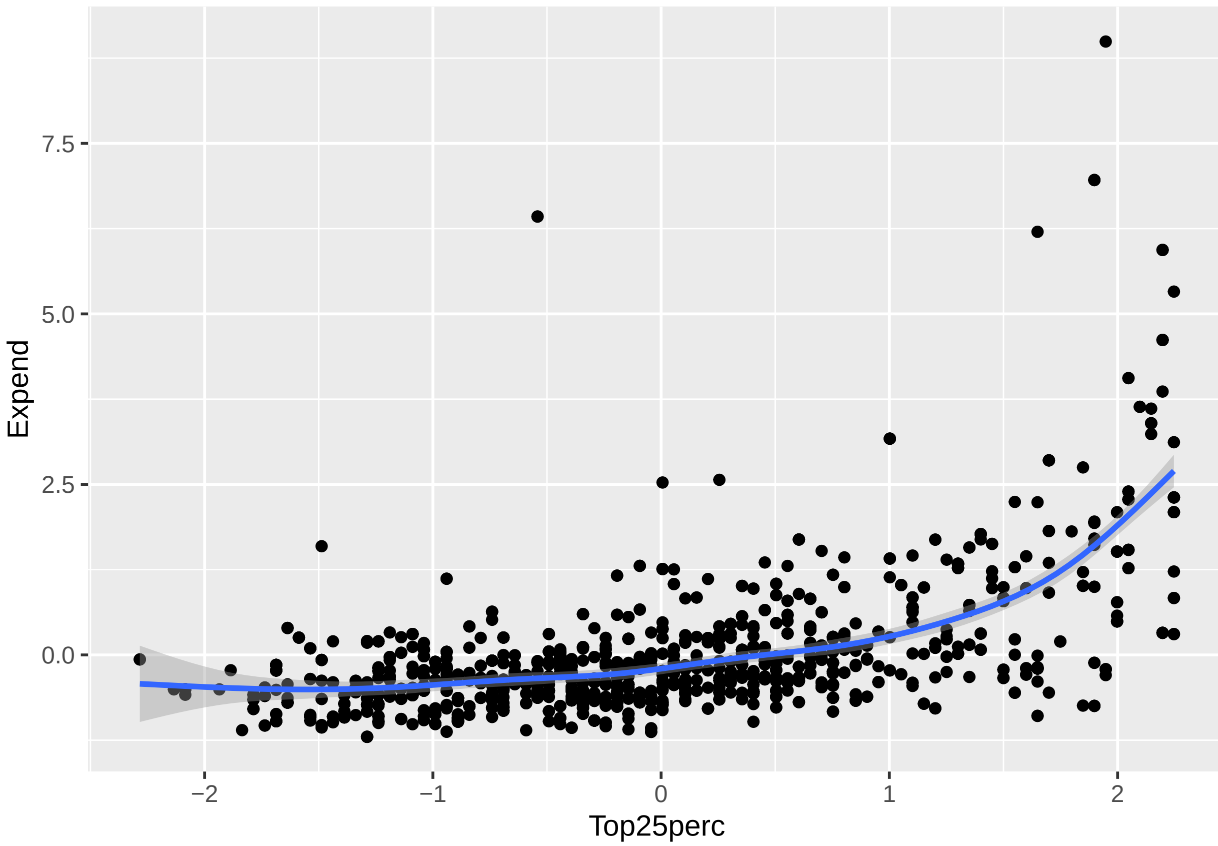

```
ggplot(data = college_train, aes(Terminal, Expend)) + geom_point() + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



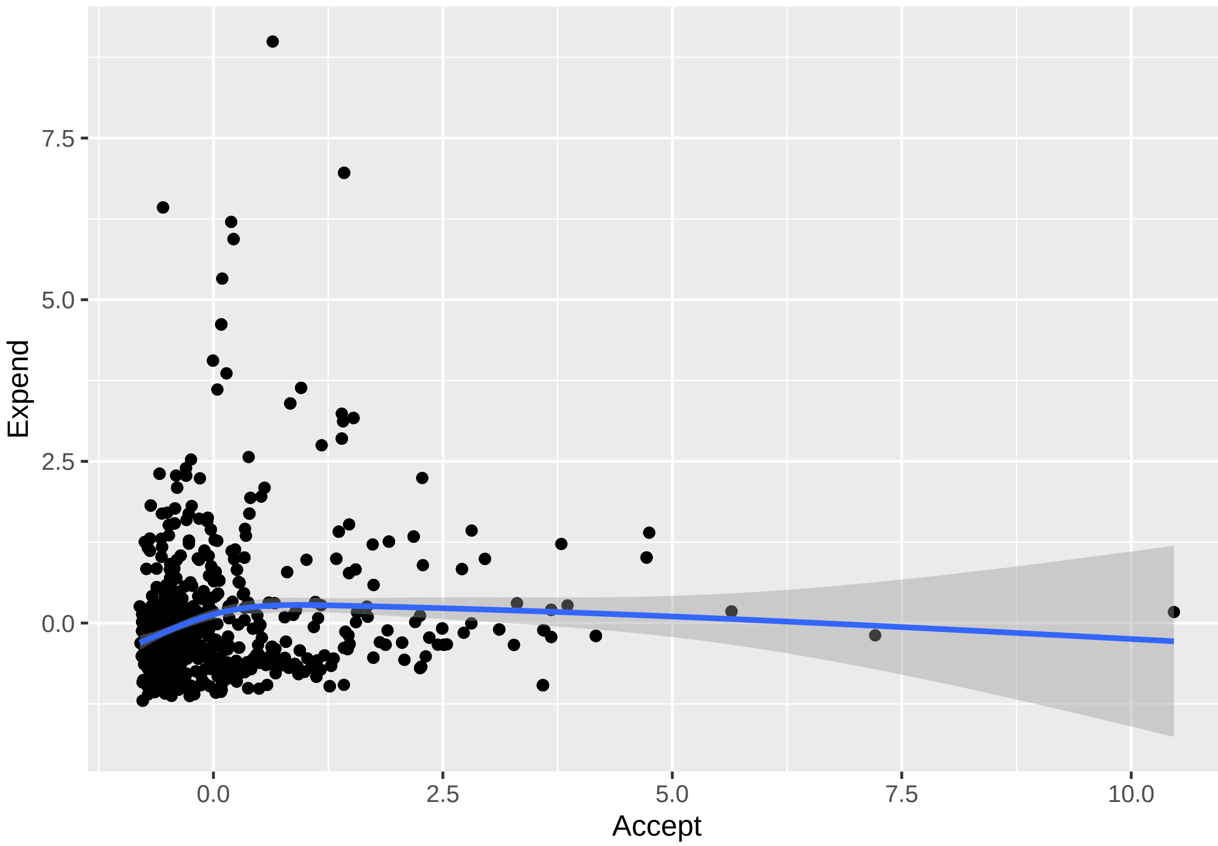

```
ggplot(data = college_train, aes(Top25perc, Expend)) + geom_point() + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



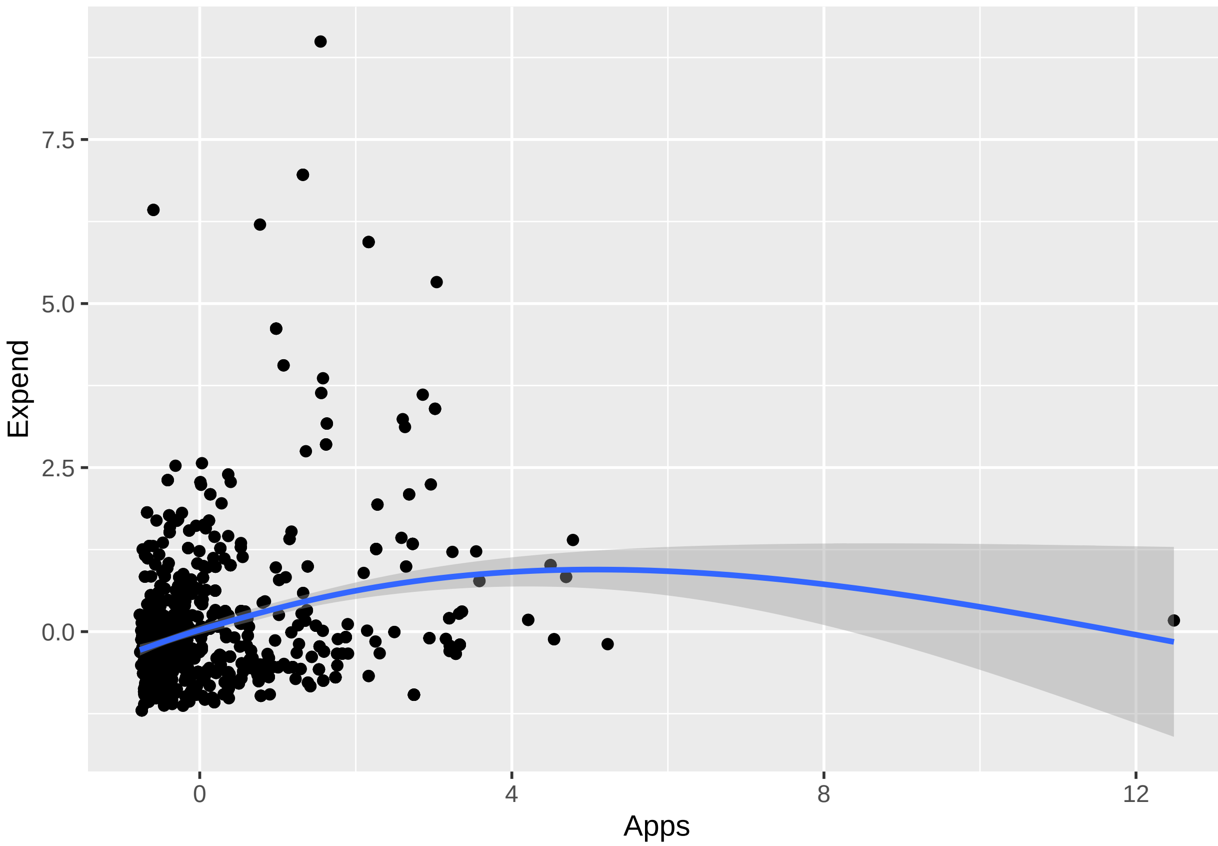

```
ggplot(data = college_train, aes(Accept, Expend)) + geom_point() + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



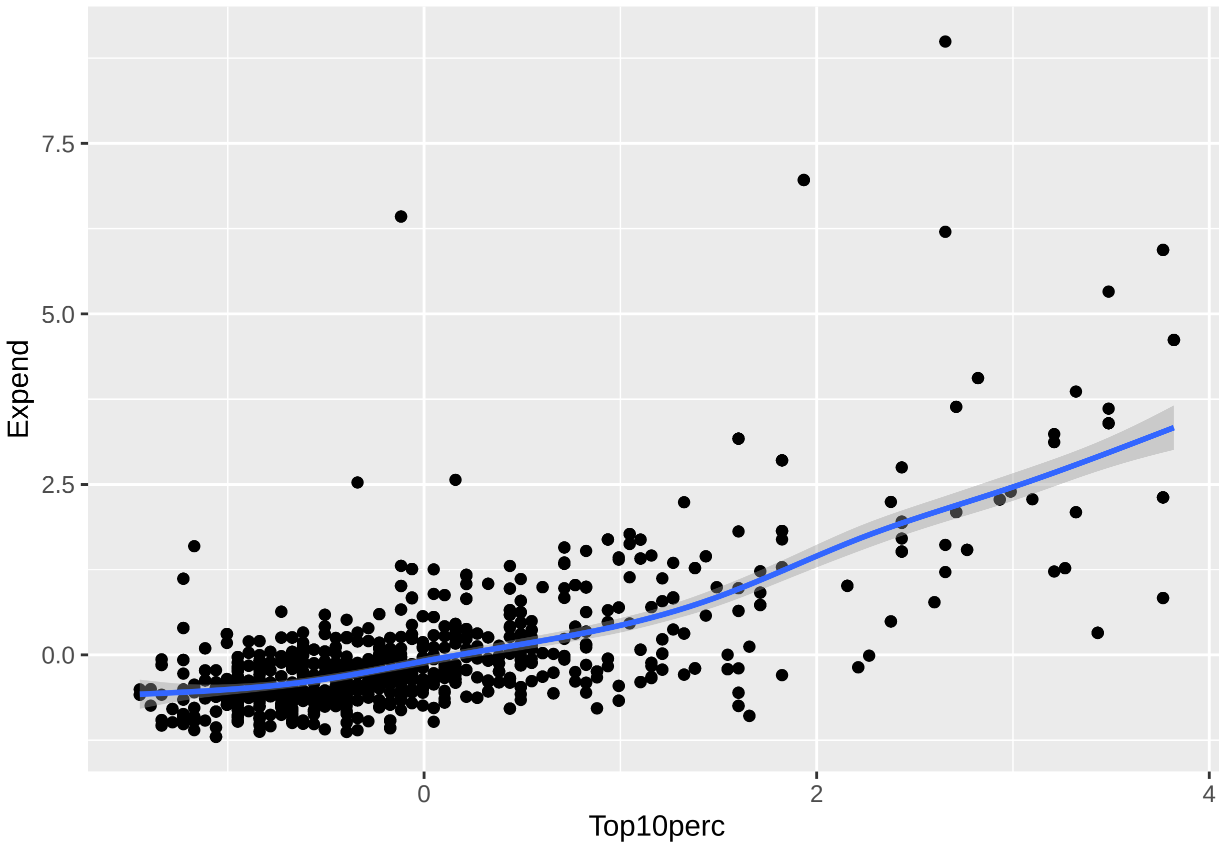

```
ggplot(data = college_train, aes(Apps, Expend)) + geom_point() + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



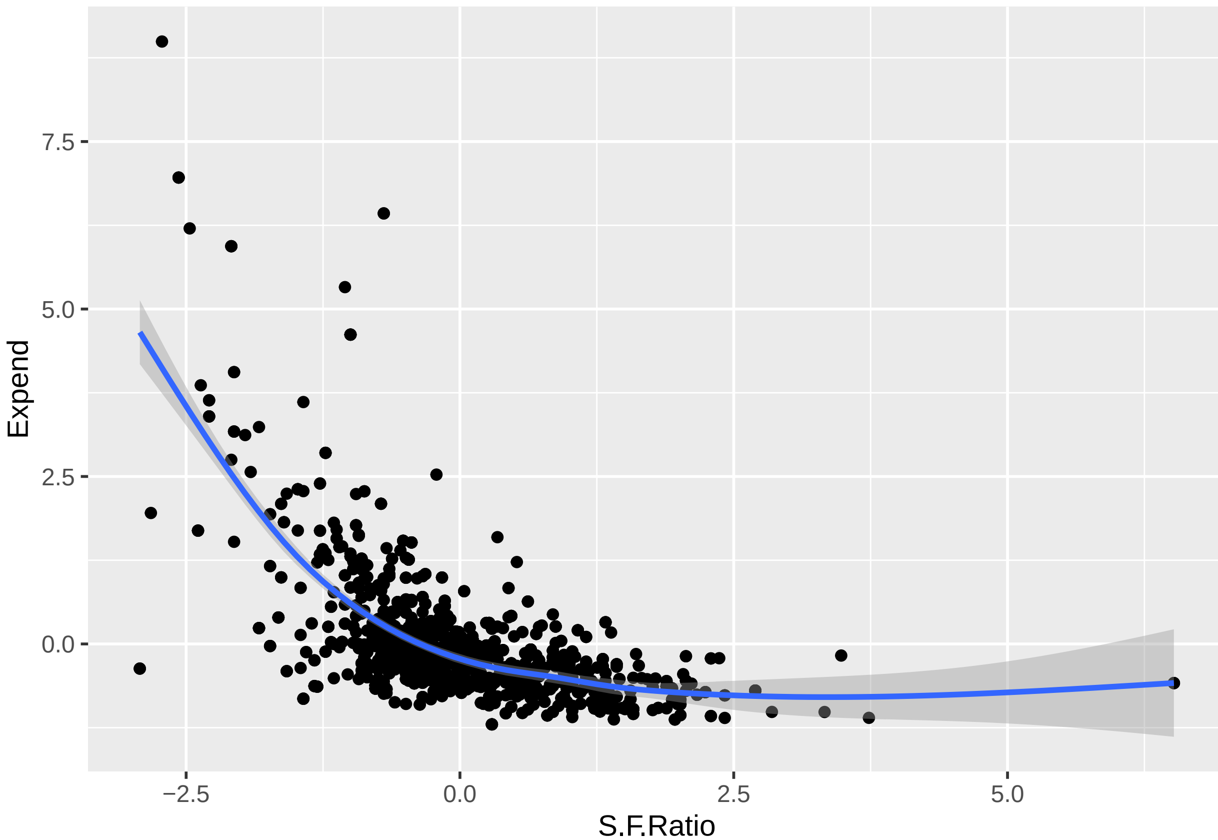

```
ggplot(data = college_train, aes(Top10perc, Expend)) + geom_point() + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



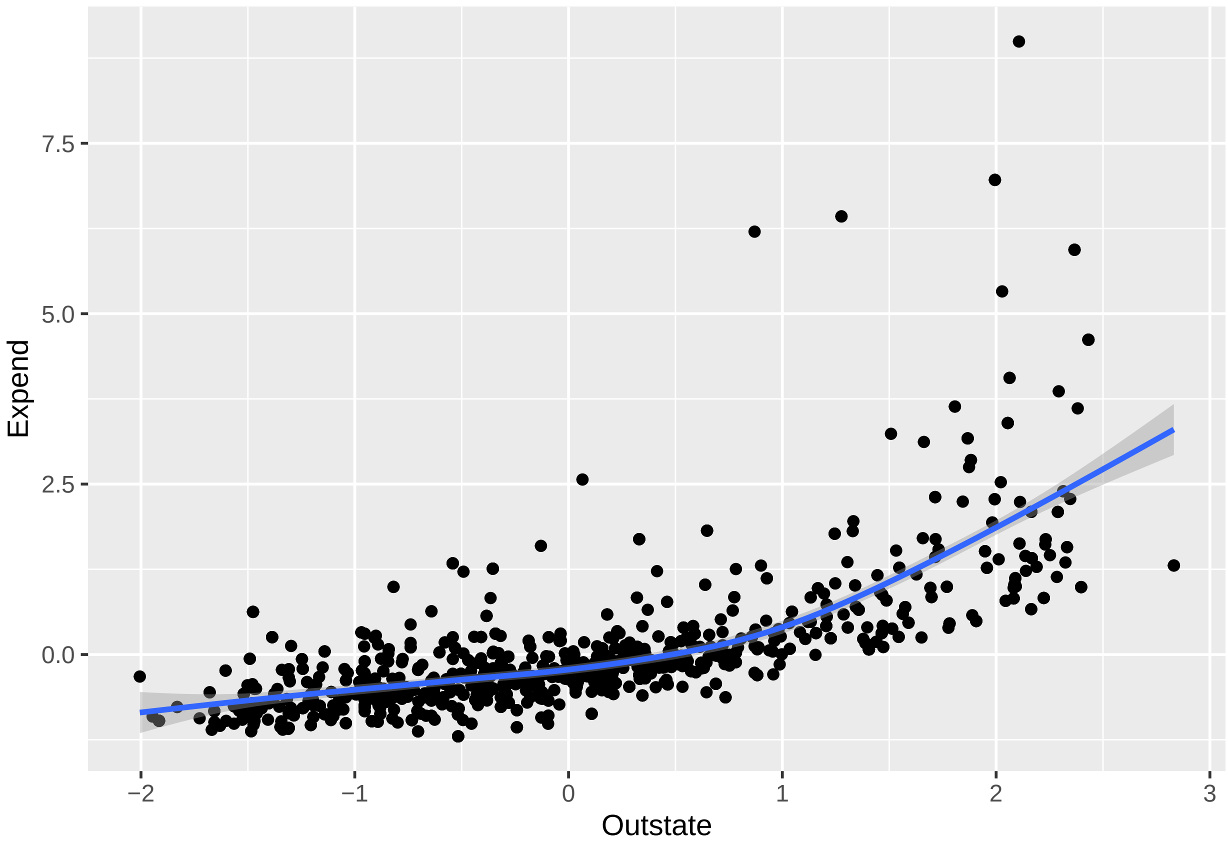

```
ggplot(data = college_train, aes(S.F.Ratio, Expend)) + geom_point() + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
ggplot(data = college_train, aes(Outstate, Expend)) + geom_point() + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Looking at these plots, it seems as though all 8 predictors need some sort of non-linear relationship. None of the predictors seem to have a linear relationship (which may indicate that we chose incorrectly when doing our GAM function). The lines of predictors that seem the closest to a linear relationship are Grad.Rate, Accept, Top10perc, and Outstate. However, the points seem to be highly uncorrelated (and thus hard to plot). Thus, it is probably more accurate to say that all 8 predictors do not have a linear-relationship.

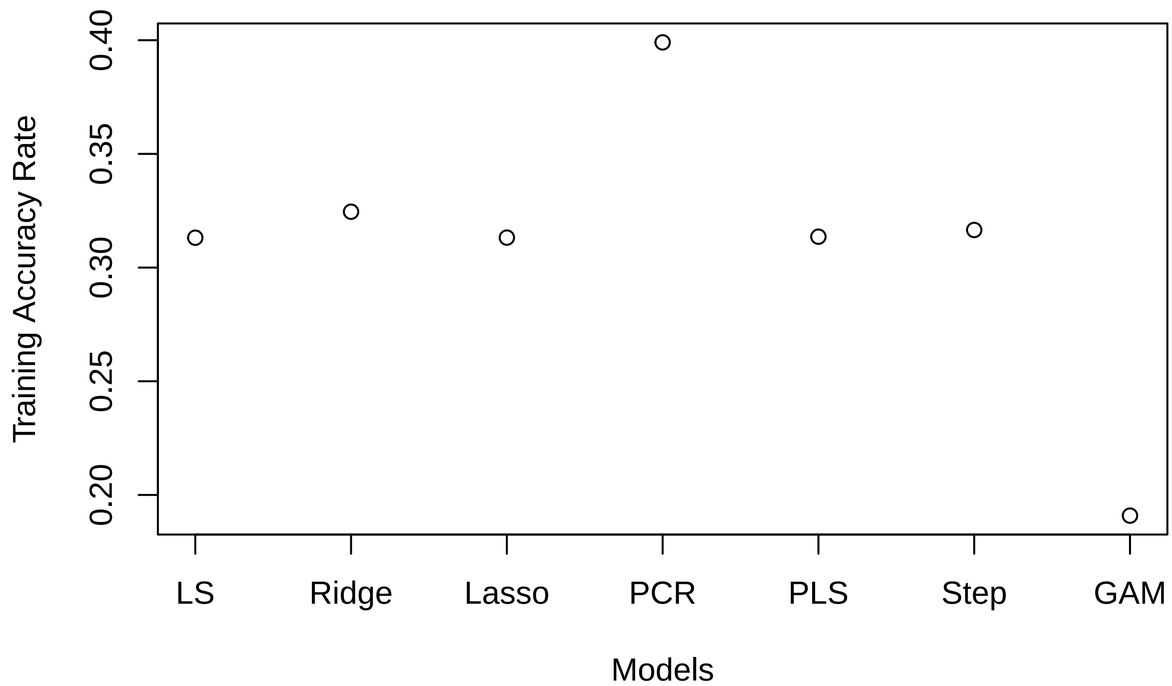
Problem 4

In order to better picture our MSEs, let us plot the MSEs all in one graph.

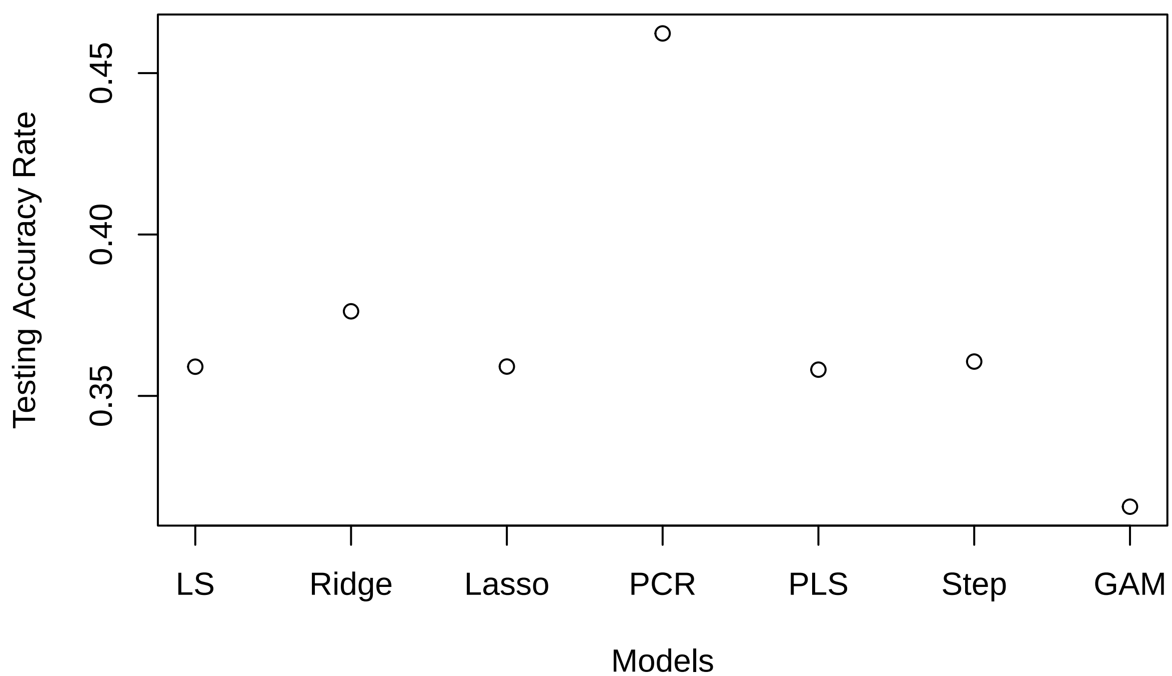
```
library(reshape)
```

```
##
## Attaching package: 'reshape'

## The following object is masked from 'package:Matrix':
##
##     expand
```

```
Approach = c("LS", "Ridge", "Lasso", "PCR", "PLS", "Step", "GAM")
Training_MSEs = c(lm_training_mse, ridge_training_mse, lasso_training_mse, pcr_training_mse, pls_trainin
Testing_MSEs = c(lm_testing_mse, ridge_testing_mse, lasso_testing_mse, pcr_testing_mse, pls_testing_mse

plot(Training_MSEs, xlab = "Models", ylab = "Training Accuracy Rate", xaxt = "n")
axis(1, at = 1:7, labels = Approach)
```




```r
plot(Testing_MSEs, xlab = "Models", ylab = "Testing Accuracy Rate", xaxt = "n")
axis(1, at = 1:7, labels = Approach)
```



Notice how GAM did the best out of all of the functions in both the training and testing data sets. We can accurately say that ,out of all of our models, GAM did the best (in terms of the lowest MSE). Least squares (the LM model) and Step model both seemed to do the second best. This may imply that the a linear model with all predictors and 8 best predictors may also be a good fit for the model. In regards to the difference in MSE, we can accurately see that PCR did the worst in both the training and testing data sets. The difference between PCR and GAM is quite high (around 0.20 in the training!), which allows us to conclude that in terms of MSE and our other models, PCR does not do well.