STT 481 HW 4

Derien Weatherspoon

2023-04-03

Libraries

```
library(MASS)
library(ISLR)
library(gam)
```

```
## Loading required package: splines

## Loading required package: foreach

## Loaded gam 1.22-2
```

```
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-7
```

```
library(boot)
library(leaps)
library(splines)
```

Question 1

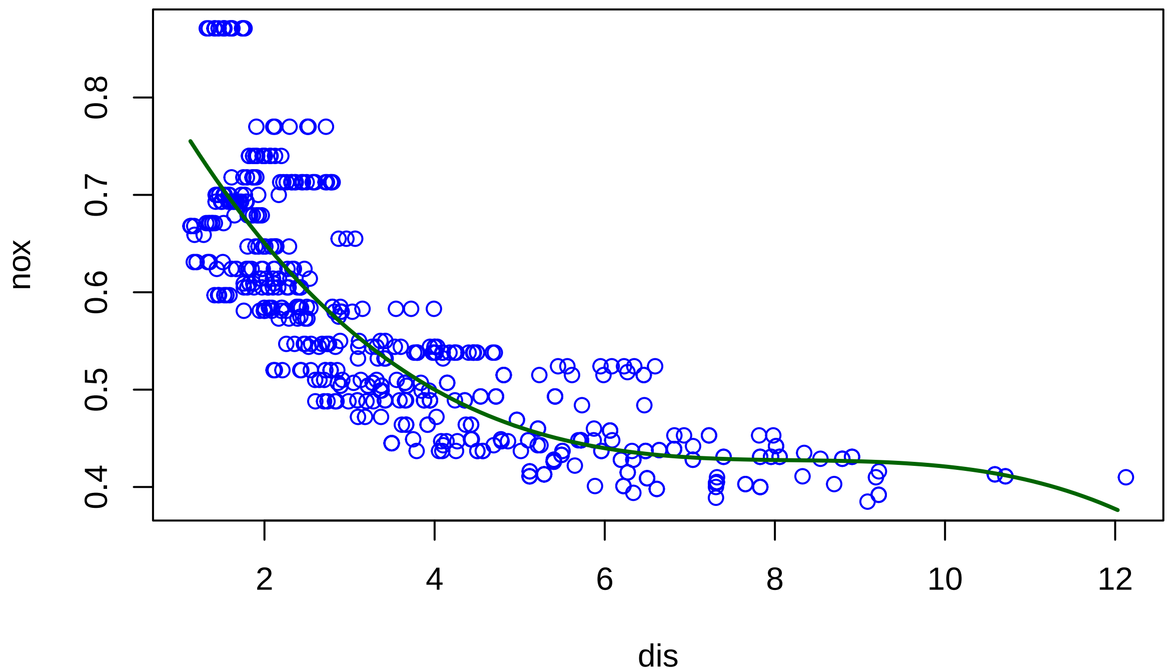
This question uses the variables dis (the weighted mean of distances to five Boston employment centers) and nox (nitrogen oxides concentration in parts per 10 million) from the Boston data (load the data from MASS package). We will treat dis as the predictor and nox as the response.

a) Use the poly() function to fit a cubic polynomial regression to predict nox using dis. Plot the resulting data and polynomial fits

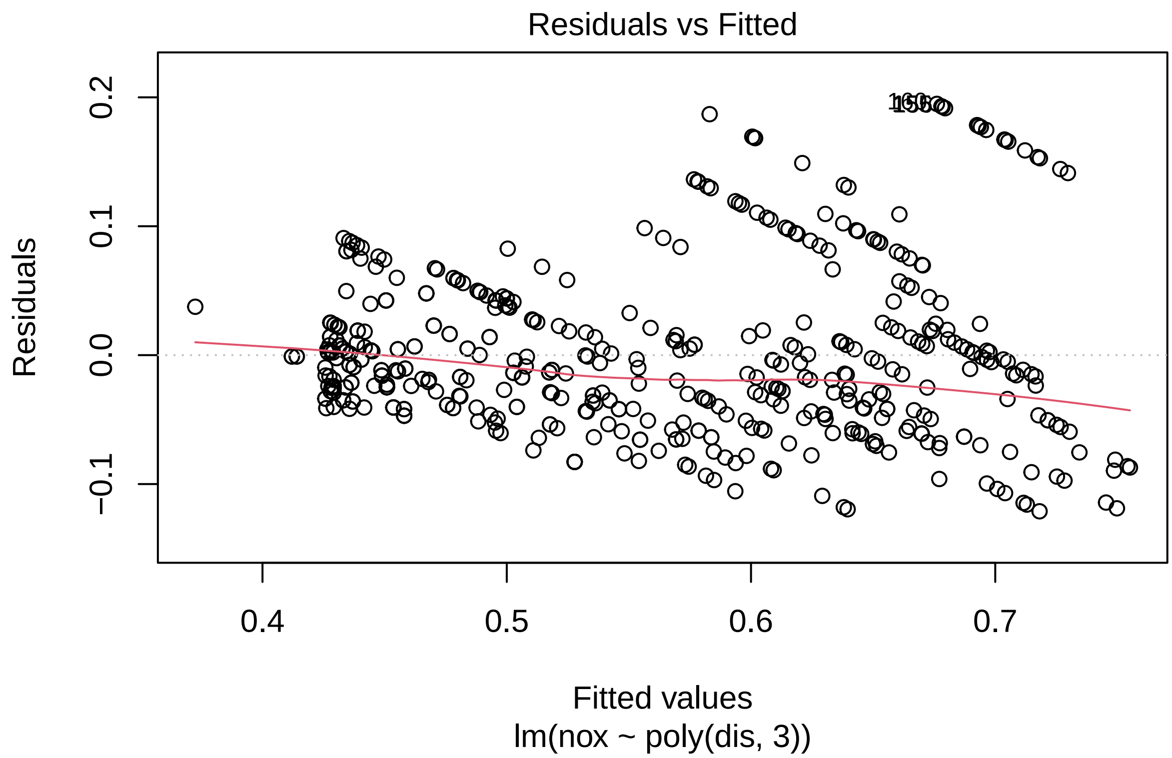
```
bos.poly <- lm(nox ~ poly(dis,3), data = Boston)
summary(bos.poly)
```

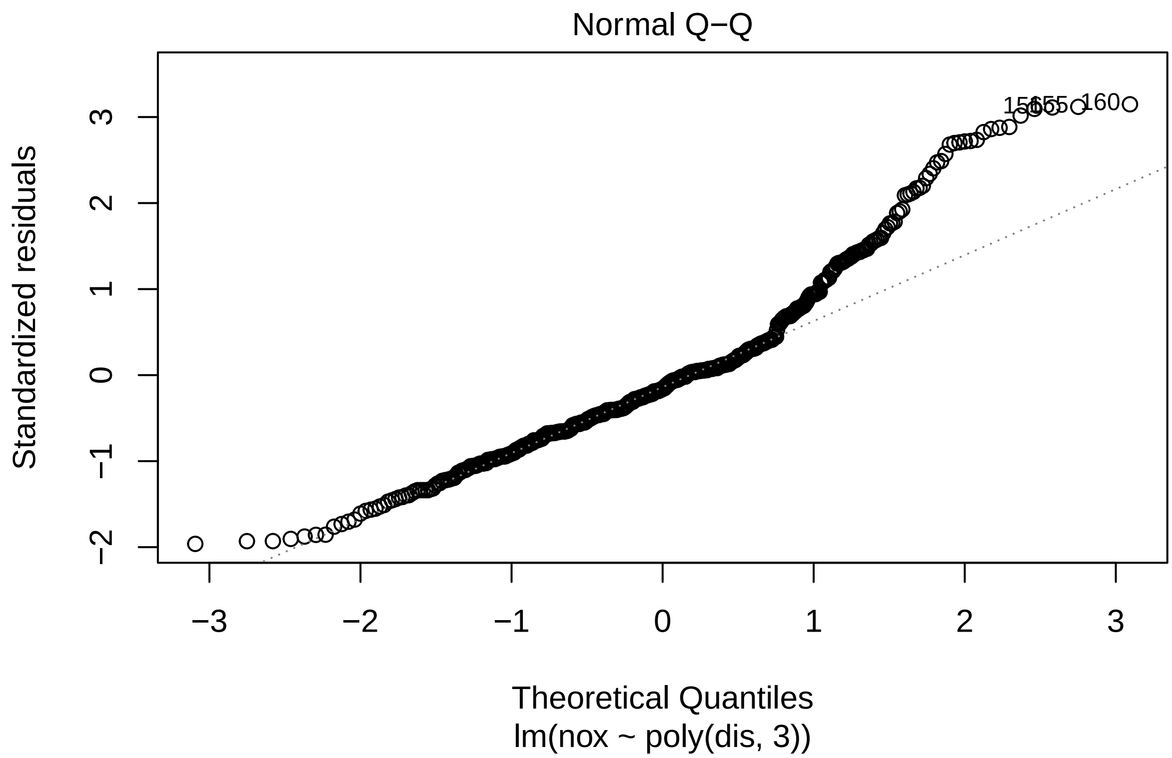
```
##
## Call:
## lm(formula = nox ~ poly(dis, 3), data = Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.121130 -0.040619 -0.009738  0.023385  0.194904
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.554695   0.002759 201.021  < 2e-16 ***
## poly(dis, 3)1 -2.003096   0.062071 -32.271  < 2e-16 ***
## poly(dis, 3)2  0.856330   0.062071  13.796  < 2e-16 ***
## poly(dis, 3)3 -0.318049   0.062071  -5.124 4.27e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06207 on 502 degrees of freedom
## Multiple R-squared:  0.7148, Adjusted R-squared:  0.7131
## F-statistic: 419.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

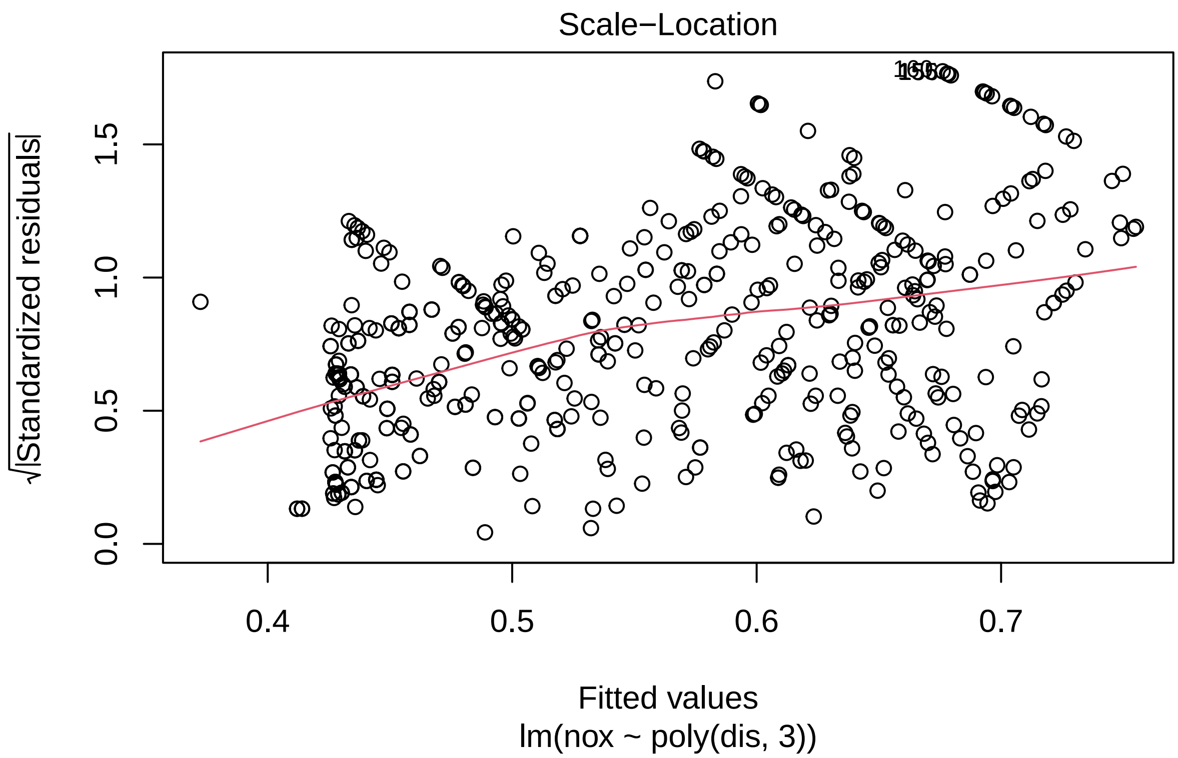
```r
# using outline from Rlab

dislims <- range(Boston$dis)
dis.grid <- seq(from = dislims[1], to = dislims[2], by = 0.1)
preds <- predict(bos.poly, list(dis = dis.grid))
plot(nox ~ dis, data = Boston, col = "blue")
lines(dis.grid, preds, col = "darkgreen", lwd = 2)
```

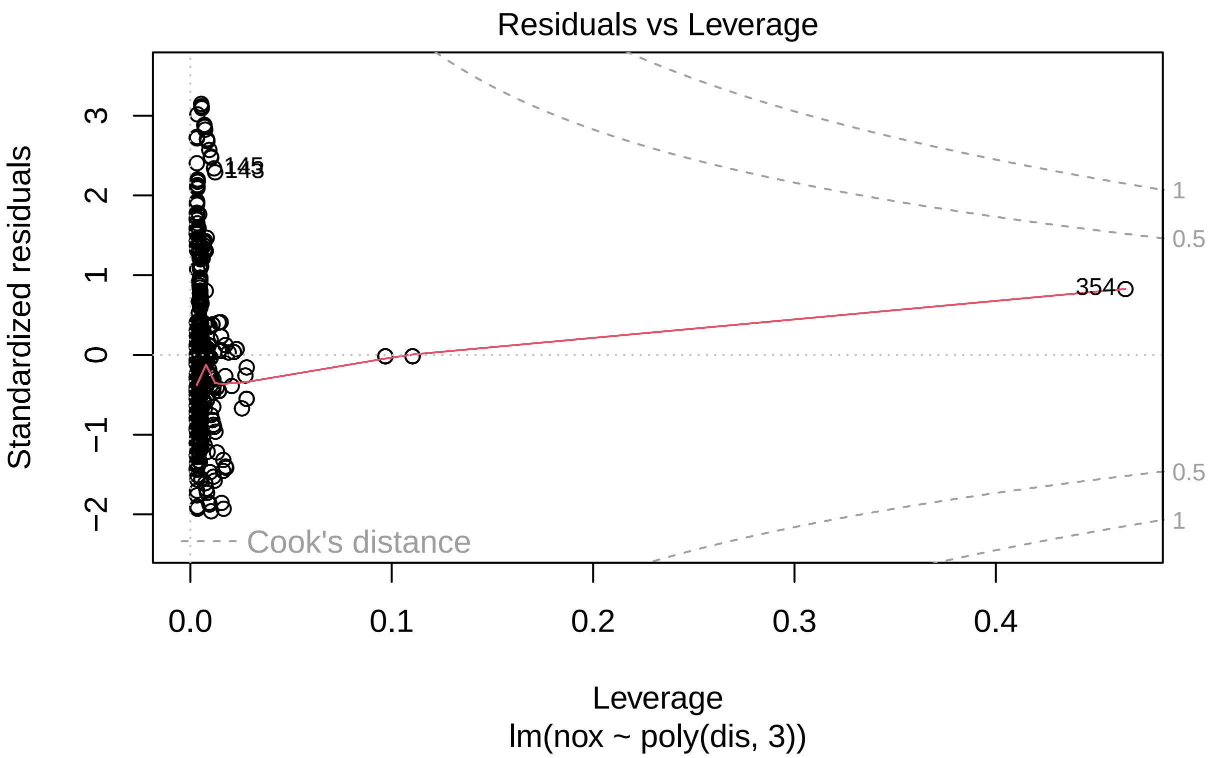



```
plot(bos.poly) # residuals
```


Residuals vs Fitted
Residuals
0.2
0.1
0.0
−0.1
0.4
0.5
0.6
0.7
Fitted values
lm(nox ~ poly(dis, 3))

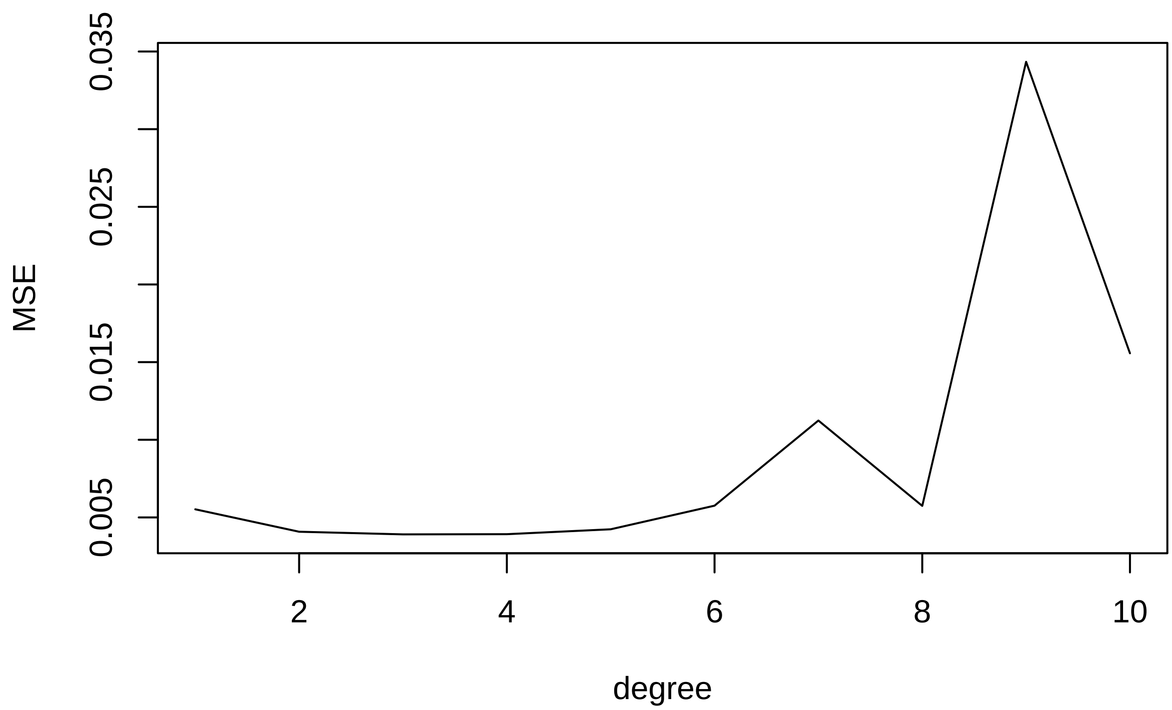

Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(nox ~ poly(dis, 3))
−3
−2
−1
0
1
2
3
160
155


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
0.4
0.5
0.6
0.7
Fitted values
lm(nox ~ poly(dis, 3))



b) Perform cross-validation to select the optimal degree for the polynomial. Plot the resulting polynomial fits with the optimal degree.

```
errors <- rep(0, 10)
for (i in 1:10) {
    bos.cv <- glm(nox ~ poly(dis, i), data = Boston)
    errors[i] <- cv.glm(Boston, bos.cv, K = 10)$delta[1]
}
plot(1:10, errors, xlab = "degree", ylab = "MSE", type = "l")
```




```r
which.min(errors)
```

```
## [1] 3
```

Using cross-validation, it seems like 3 is the best degree for the model.

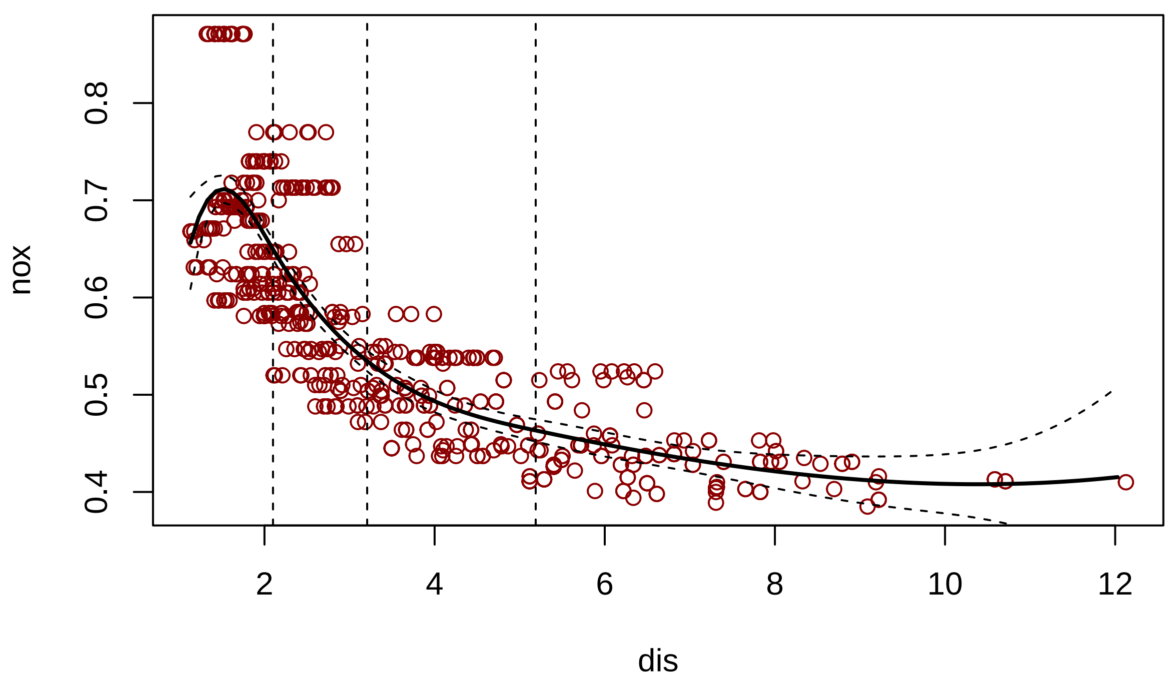
c) Use the bs() function to fit a cubic spline with six degrees of freedom to predict nox using dis. Report the knot locations. Plot the resulting fit.

```r
dis <- Boston$dis
nox <- Boston$nox
bs.fit <- lm(data = Boston, nox ~ bs(dis, 6))
summary(bs.fit)
```

```
##
## Call:
## lm(formula = nox ~ bs(dis, 6), data = Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.128538 -0.037813 -0.009987  0.022644  0.195494
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

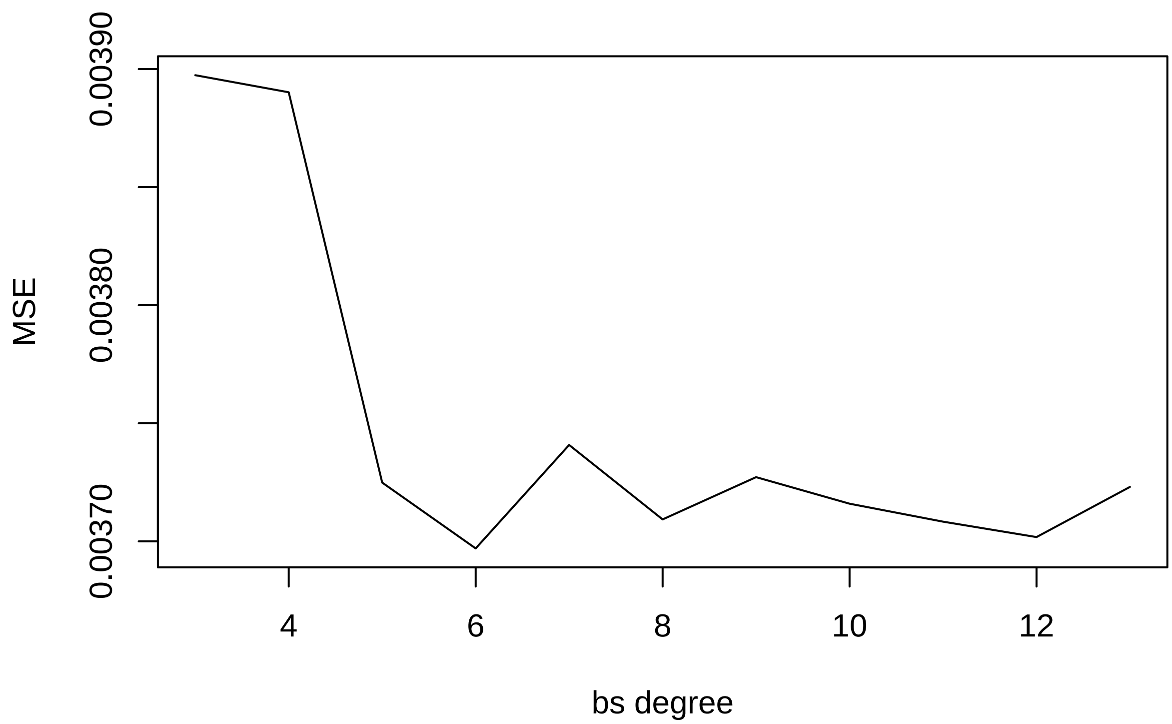
```
## (Intercept)  0.65622    0.02370  27.689  < 2e-16 ***
## bs(dis, 6)1  0.10222    0.03516   2.907  0.00381 **
## bs(dis, 6)2 -0.02963    0.02338  -1.267  0.20571
## bs(dis, 6)3 -0.15959    0.02791  -5.718 1.86e-08 ***
## bs(dis, 6)4 -0.22815    0.03324  -6.864 1.99e-11 ***
## bs(dis, 6)5 -0.26272    0.04930  -5.329 1.50e-07 ***
## bs(dis, 6)6 -0.24002    0.05434  -4.417 1.23e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06062 on 499 degrees of freedom
## Multiple R-squared:  0.7295, Adjusted R-squared:  0.7263
## F-statistic: 224.3 on 6 and 499 DF,  p-value: < 2.2e-16
```

```r
pred <- predict(bs.fit, newdata = list(dis = dis.grid), se = T)
plot(dis, nox, col = "darkred")
lines(dis.grid, pred$fit, lwd = 2)
lines(dis.grid, pred$fit + 2*pred$se, lty = "dashed")
lines(dis.grid, pred$fit - 2*pred$se, lty = "dashed")
abline(v = attr(bs(dis, df = 6), "knots"), lty = 2) # knots placement
```



d) Perform cross-validation in order to select the best degrees of freedom for a cubic spline on this data. Plot the resulting cubic spline fits with the best degrees of freedom.

```r
set.seed(1)
errors_2 <- rep(3, 13)
for (i in 3:13) {
    bs.cv <- glm(nox ~ bs(dis, df = i), data = Boston)
    errors_2[i] <- cv.glm(Boston, bs.cv, K = 10)$delta[1]
}
plot(3:13, errors_2[-c(1,2)], xlab = "bs degree", ylab = "MSE", type = "l")
```




```r
which.min(errors_2)
```

```
## [1] 6
```

Using cross-validation, it looks like 6 is the best degree of freedom for the cubic spline model.

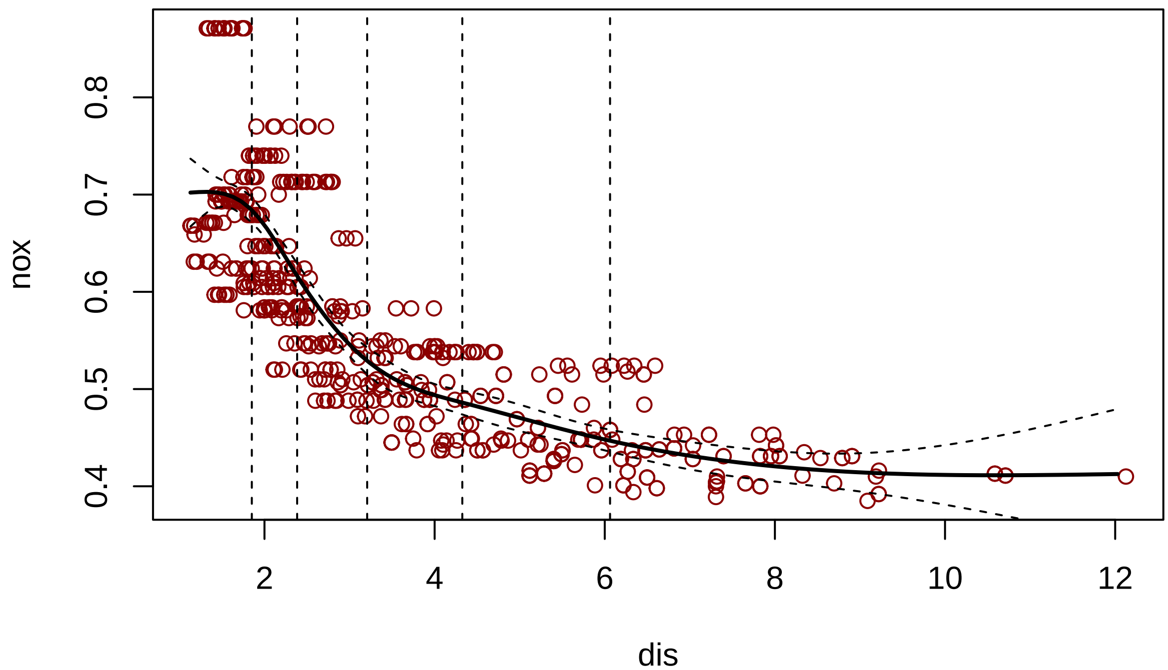
e) Use the ns() function to fit a natural cubic spline with six degrees of freedom to predict nox using dis. Report the knot locations. Plot the resulting fit.

```r
ns.fit <- lm(data = Boston, nox ~ ns(dis, 6))
summary(ns.fit)
```

```
##
## Call:
## lm(formula = nox ~ ns(dis, 6), data = Boston)
```

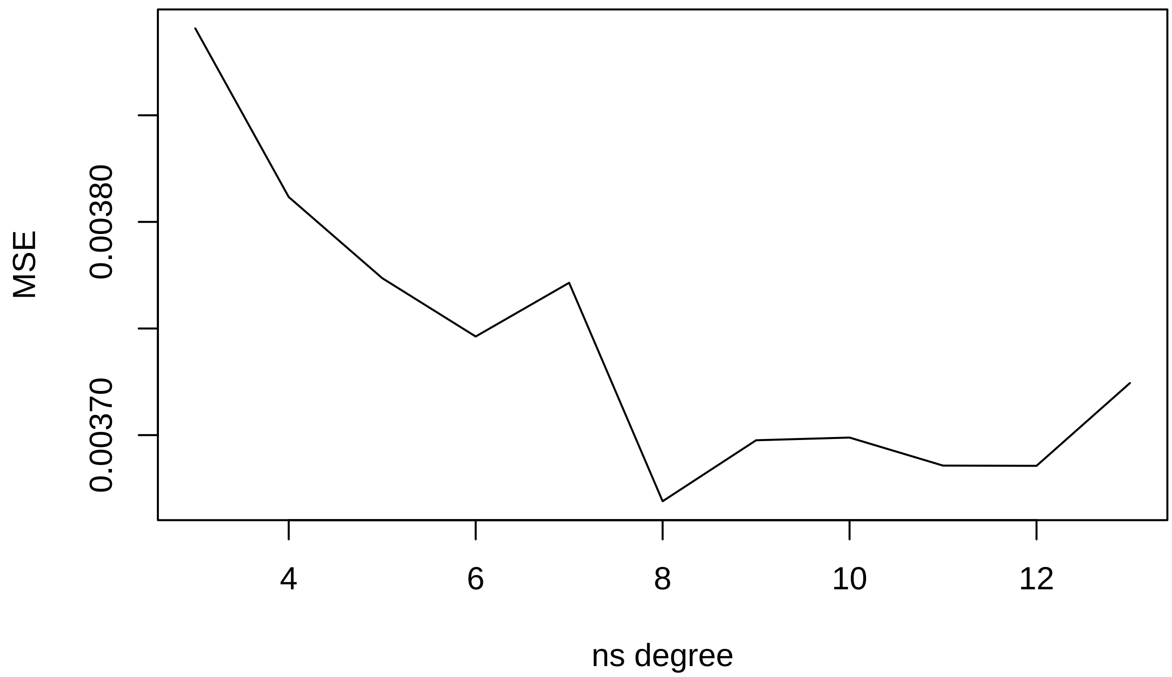
```
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.135007 -0.039356 -0.006573  0.024535  0.196108
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.70205    0.01744  40.263  < 2e-16 ***
## ns(dis, 6)1 -0.10413    0.01807  -5.764 1.44e-08 ***
## ns(dis, 6)2 -0.19401    0.02354  -8.242 1.50e-15 ***
## ns(dis, 6)3 -0.21955    0.02053 -10.694  < 2e-16 ***
## ns(dis, 6)4 -0.29707    0.02070 -14.352  < 2e-16 ***
## ns(dis, 6)5 -0.28848    0.04286  -6.731 4.63e-11 ***
## ns(dis, 6)6 -0.29134    0.03528  -8.258 1.33e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06096 on 499 degrees of freedom
## Multiple R-squared:  0.7266, Adjusted R-squared:  0.7233
## F-statistic:   221 on 6 and 499 DF,  p-value: < 2.2e-16
```

```r
pred <- predict(ns.fit, newdata = list(dis = dis.grid), se = T)
plot(dis, nox, col = "darkred")
lines(dis.grid, pred$fit, lwd = 2)
lines(dis.grid, pred$fit + 2*pred$se, lty = "dashed")
lines(dis.grid, pred$fit - 2*pred$se, lty = "dashed")
abline(v = attr(ns(dis, df = 6), "knots"), lty = 2) # knots placement
```



f) Perform cross-validation in order to select the best degrees of freedom for a natural cubic spline on this data. Plot the resulting natural cubic spline fits with the best degrees of freedom.

```
set.seed(1)
errors_3 <- rep(3, 13)
for (i in 3:13) {
    ns.cv <- glm(nox ~ ns(dis, df = i), data = Boston)
    errors_3[i] <- cv.glm(Boston, ns.cv, K = 10)$delta[1]
}
plot(3:13, errors_3[-c(1,2)], xlab = "ns degree", ylab = "MSE", type = "l")
```




```
which.min(errors_3)
```

```
## [1] 8
```

Using cross-validation, it looks like the best degree of freedom is 8 for this model.

Question 2

```
data("College")
set.seed(123)
train <- sample(nrow(College), 600)
College.train <- College[train,]
College.test <- College[-train,]
```

This question relates to the College data set.

a) Using out-of-state tuition as the response and the other variables as the predictors, perform forward stepwise selection on the training set in order to identify a satisfactory model that uses just a subset of the predictors.

```r
fwd.college <- regsubsets(Outstate ~ ., data = College.train, method = 'forward', nvmax = ncol(College.
fwd.summary <- summary(fwd.college)

which.min(fwd.summary$bic) # lowest point on bic
```

```
## [1] 10
```

```r
which.min(fwd.summary$cp) # lowest point on cp
```

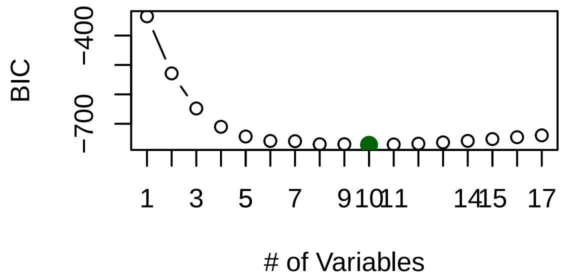
```
## [1] 12
```

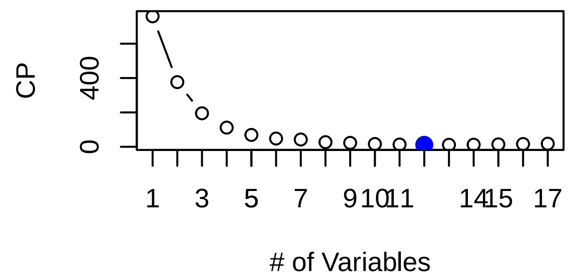
```r
which.min(fwd.summary$adjr2) # lowest point on adj r^2
```

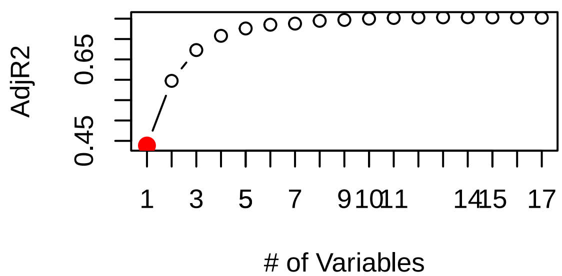
```
## [1] 1
```

```r
par(mfrow=c(2, 2))

plot(fwd.summary$bic, type='b', xlab='# of Variables', ylab='BIC') # Plot BIC
axis(1, at = seq(1,17,by=1))
points(10, fwd.summary$bic[10], col = "darkgreen", cex = 2, pch = 20)

plot(fwd.summary$cp, type='b', xlab='# of Variables', ylab='CP') # Plot CP
axis(1, at = seq(1,17,by=1))
points(12, fwd.summary$cp[12], col = "blue", cex = 2, pch = 20)

plot(fwd.summary$adjr2, type='b', xlab='# of Variables', ylab='AdjR2') # Plot AdjR^2
axis(1, at = seq(1,17,by=1))
points(1, fwd.summary$adjr2[1], col = "red", cex = 2, pch = 20)

# Find the 10 variables used for the GAM model

coefs <- coef(fwd.college, id = 10)
names(coefs)
```

```
##  [1] "(Intercept)" "PrivateYes"   "Apps"         "Accept"      "Enroll"
##  [6] "Top10perc"   "Room.Board"   "PhD"          "perc.alumni" "Expend"
## [11] "Grad.Rate"
```



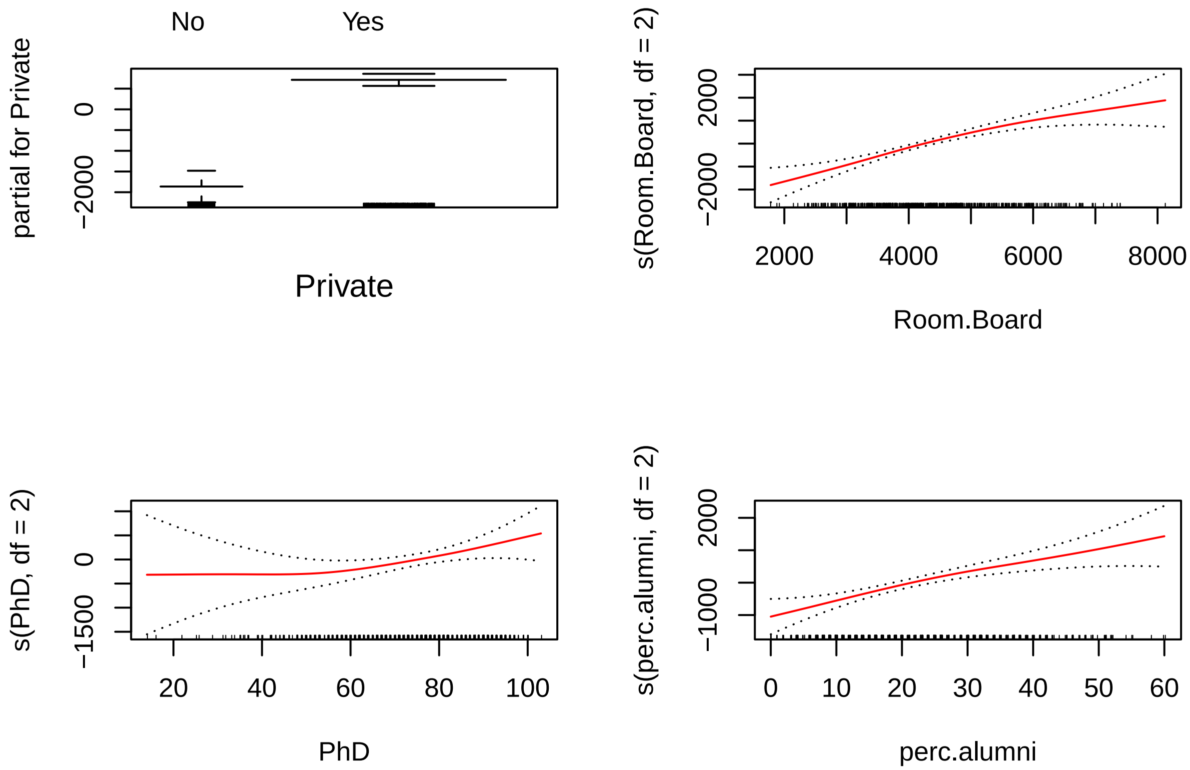


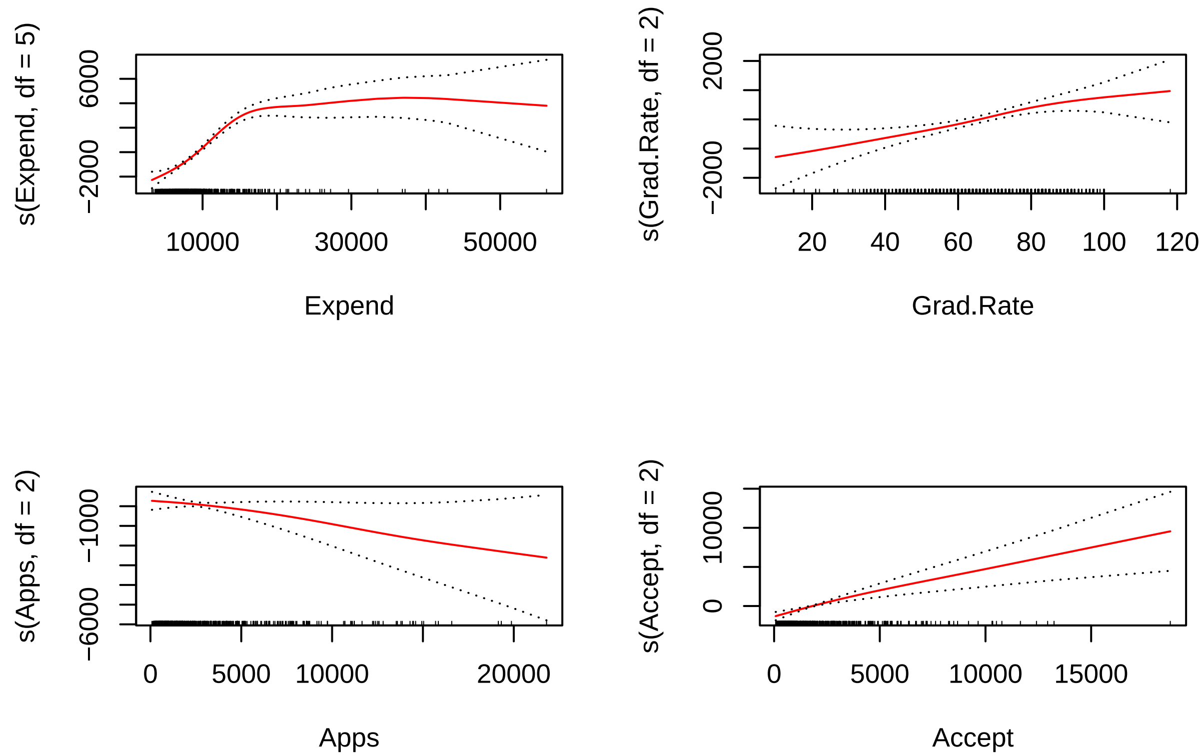


According to BIC, the number of variables best fit for the model is 10, while for cp the number of variables is 12, and for Adjusted R^2 the number of variables is only 1. In this case, I believe I am going with BIC is the best. It is not too many variables, while at the same time not excluding too many variables. The selected predictors are: "Private", "Apps", "Accept", "Enroll", "Top10perc", "Room.Board", "PhD", "perc.alumni", "Expend", "Grad.Rate"

b) Fit a GAM on the training data, using out-of-state tuition as the response and the features selected in the previous step as the predictors.

```
fit.gam <- gam(Outstate ~ Private + s(Room.Board, df = 2) + s(PhD, df = 2) + s(perc.alumni, df = 2) + s
# I use s() to indicate I want a smoothing spline
par(mfrow = c(2, 2))
plot(fit.gam, se = T, col = "red")
```


No
Yes
partial for Private
0
−2000
Private
s(Room.Board, df = 2)
2000
−2000
2000
4000
6000
8000
Room.Board
s(PhD, df = 2)
0
−1500
20
40
60
80
100
PhD
s(perc.alumni, df = 2)
2000
−1000
0
10
20
30
40
50
60
perc.alumni



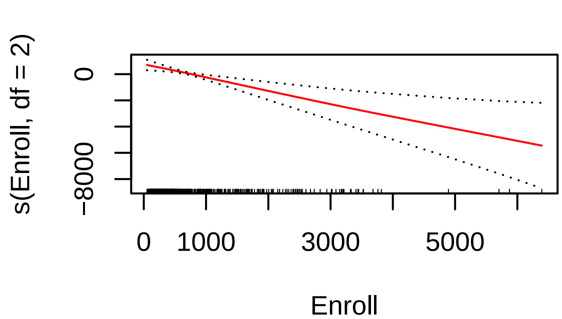

```
summary(fit.gam)
```

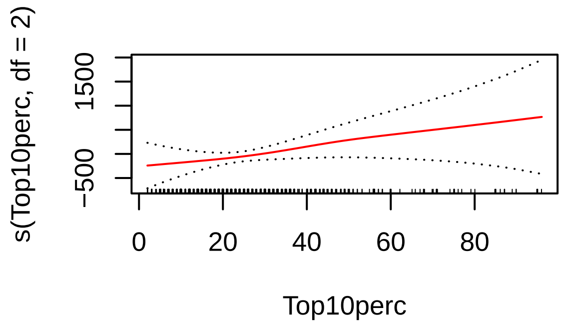
```
##
## Call: gam(formula = Outstate ~ Private + s(Room.Board, df = 2) + s(PhD,
##     df = 2) + s(perc.alumni, df = 2) + s(Expend, df = 5) + s(Grad.Rate,
##     df = 2) + s(Apps, df = 2) + s(Accept, df = 2) + s(Enroll,
##     df = 2) + s(Top10perc, df = 2), data = College.train)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -6804.14 -1072.08    83.49  1236.38  7831.31
##
## (Dispersion Parameter for gaussian family taken to be 3455825)
##
##     Null Deviance: 9694792870 on 599 degrees of freedom
## Residual Deviance: 1994010747 on 576.9999 degrees of freedom
## AIC: 10760.62
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##                         Df     Sum Sq    Mean Sq  F value    Pr(>F)
## Private                  1 2732131662 2732131662 790.5874 < 2.2e-16 ***
## s(Room.Board, df = 2)    1 1796653320 1796653320 519.8913 < 2.2e-16 ***
## s(PhD, df = 2)           1  624690191  624690191 180.7644 < 2.2e-16 ***
## s(perc.alumni, df = 2)   1  390893264  390893264 113.1114 < 2.2e-16 ***
```

```
## s(Expend, df = 5)        1  735768872  735768872 212.9069 < 2.2e-16 ***
## s(Grad.Rate, df = 2)     1   95393137   95393137  27.6036 2.098e-07 ***
## s(Apps, df = 2)          1    8586354    8586354   2.4846 0.1155134
## s(Accept, df = 2)        1   21644616   21644616   6.2632 0.0126024 *
## s(Enroll, df = 2)        1   45696898   45696898  13.2232 0.0003012 ***
## s(Top10perc, df = 2)     1    9236598    9236598   2.6728 0.1026241
## Residuals              577 1994010747    3455825
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                        Npar Df  Npar F  Pr(F)
## (Intercept)
## Private
## s(Room.Board, df = 2)        1  2.8824 0.0901 .
## s(PhD, df = 2)               1  1.8082 0.1792
## s(perc.alumni, df = 2)       1  1.0829 0.2985
## s(Expend, df = 5)            4 22.8533 <2e-16 ***
## s(Grad.Rate, df = 2)         1  2.2233 0.1365
## s(Apps, df = 2)              1  2.5250 0.1126
## s(Accept, df = 2)            1  5.5050 0.0193 *
## s(Enroll, df = 2)            1  0.6237 0.4300
## s(Top10perc, df = 2)         1  0.6516 0.4199
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
fit.gam$aic
```

```
## [1] 10760.62
```





With this model, the produced aic is 10760.62. In order to lower the AIC score, I could attempt to change around the degrees of freedom for some variables. Either decreasing some, increasing some, or removing some all together. I can do this by looking at what variables are and aren't significant.

c) Evaluate the model obtain on the test set, and explain the results obtained.

```
test.pred <- predict(fit.gam, College.test)
errors_4 <- mean((College.test$Outstate - test.pred)^2)
errors_4
```

```
## [1] 2997372
```

```
rsq <- 1- errors_4/mean(((College.test$Outstate - mean(College.test$Outstate)))^2)
rsq
```

```
## [1] 0.8143664
```

With 10 predictors, the rsquared for the test set is 0.8033426 using GAM. This means that roughly 80% of the variance within the data can be explained within the model of 10 predictors.

Question 3

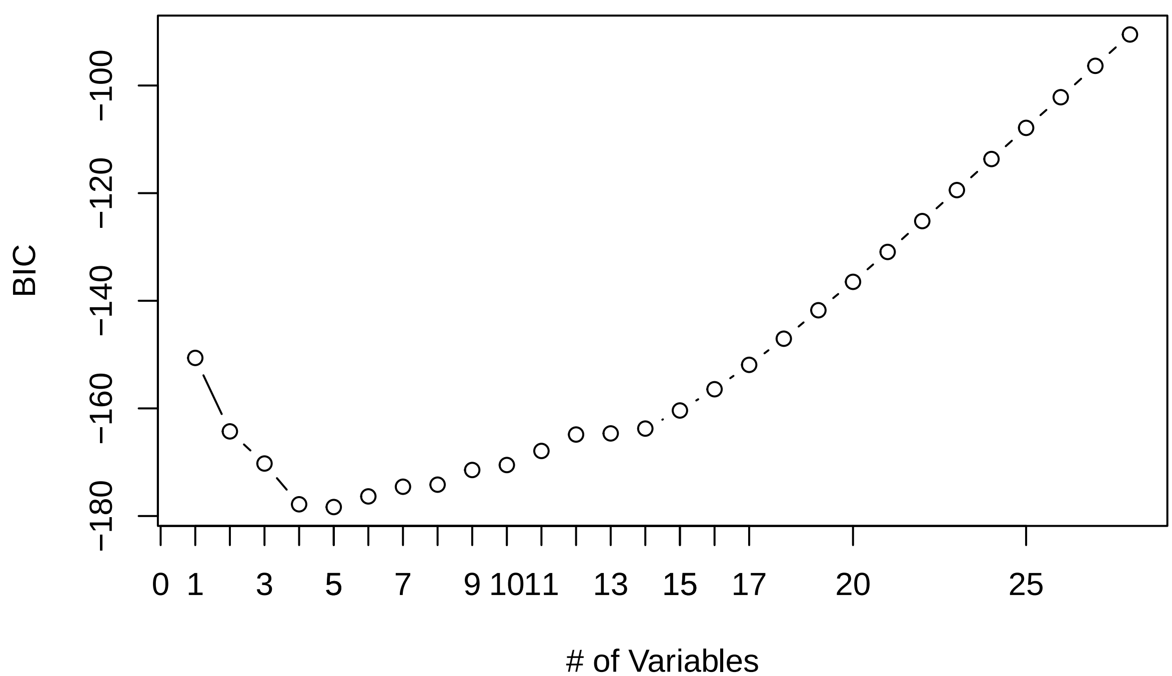
In this question, we wish to predict the 2019-20 salaries of five NBA players, who were free agents in 2019 summer, on the basis of various statistics associated with performance in the 2018-19 season.

a) Run the following command lines to load the two data sets. Notice that here **row.names = 1** because by doing so, we can see those players' names.

```
train.dat <- read.csv("nbadata17.csv", row.names = 1)
test.dat <- read.csv("nbadata18.csv", row.names = 1)
```

b) Using salary as the response and the other variables as the predicts, perform forward stepwise selection on the training set, and use BIC as the selection criterion. Report the non-zero coefficient estimates.

```
fwd.nba <- regsubsets(salary ~ ., data = train.dat, method = 'forward', nvmax = ncol(train.dat))
fwd.nbasum <- summary(fwd.nba)
plot(fwd.nbasum$bic, type='b', xlab='# of Variables', ylab='BIC') # Plot BIC
axis(1, at = seq(1,17,by=1))
```




```
which.min(fwd.nbasum$bic)
```

```
## [1] 5
```

```
coef(fwd.nba, id = 5)
```

```
##  (Intercept)          age            g           gs    x2ppercent          pts
## -10003.77541    360.81836    -69.17732     79.48794    8599.43163    644.44076
```

I use 5 predictors in the models moving forward, according to BIC. The coefficients are: "age", "g", "gs", "x2ppercent", and "pts".

c) Predict the salaries of the five players in the test.dat using the fitted model in (b)

```
predict.regsubsets <- function (object, newdata , id, ...){
  form <- as.formula(object$call[[2]])  # formula of null model
  mat <- model.matrix(form, newdata)    # building an "X" matrix from newdata
  coefi <- coef(object, id = id)        # coefficient estimates associated with the object model contai
  xvars <- names(coefi)            # names of the non-zero coefficient estimates
  return(mat[,xvars] %*% coefi)    # X[,non-zero variables] %*% Coefficients[non-zero variables]
}
fwd.pred <- predict.regsubsets(fwd.nba, newdata = test.dat, id = 5)
fwd.pred
```

```
##                       [,1]
## Khris Middleton 16496.228
## Terrence Ross    8028.138
## Ish Smith        6595.110
## Myles Turner    11748.581
## Thaddeus Young  14625.969
```

d) Fit a ridge regression model on the training set, and predict the salaries in the test set.

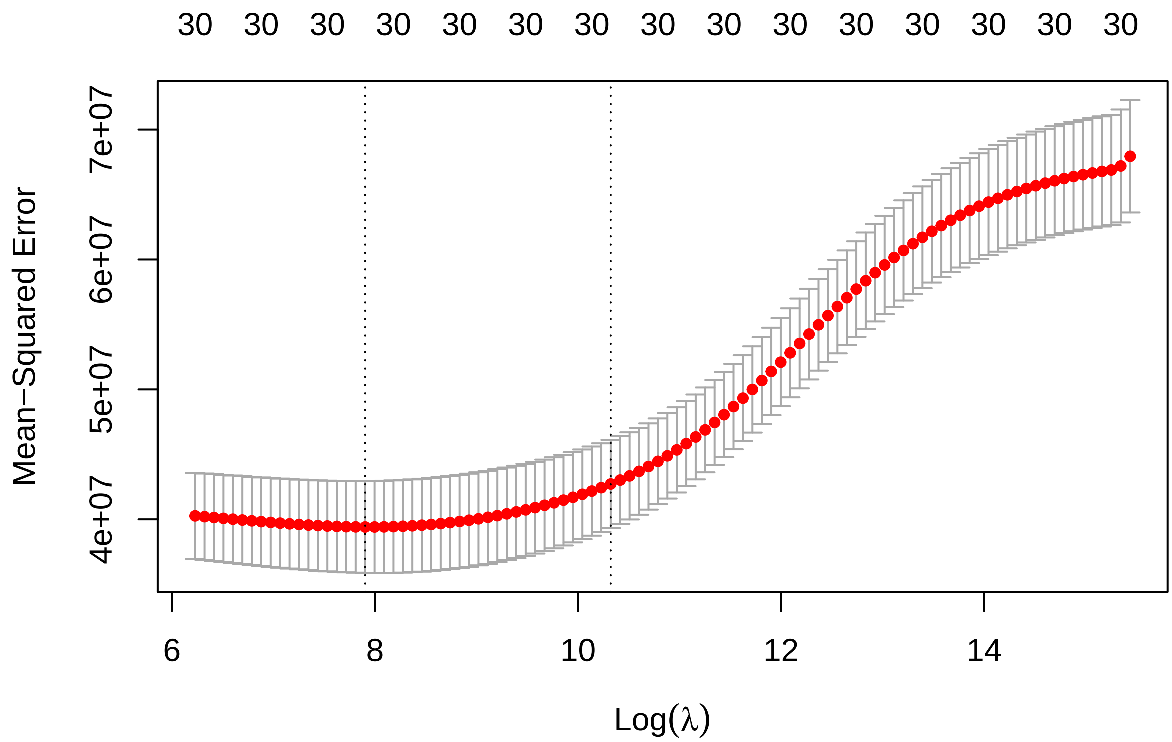
```
x.fit <- model.matrix(salary ~., data = train.dat)[,-1]
y.fit <-  model.matrix(salary ~., data = test.dat)[,-1] # the model matrix for test to make predictions
trs <- train.dat$salary


set.seed(1)
ridge.fit <- glmnet(x.fit, trs, alpha = 0)
names(ridge.fit)
```

```
##  [1] "a0"        "beta"      "df"        "dim"       "lambda"    "dev.ratio"
##  [7] "nulldev"   "npasses"   "jerr"      "offset"    "call"      "nobs"
```

```
cv.ridge <- cv.glmnet(x.fit, trs, alpha = 0, nfolds = 10)
bestridge_lambda <- cv.ridge$lambda.min # best tuning parameter
bestridge_lambda
```

```
## [1] 2705.659
```

```
plot(cv.ridge)
```




```r
# Predict

ridge.pred <- predict(ridge.fit, s = bestridge_lambda, newx = y.fit)
head(ridge.pred)
```

```
##                        s1
## Khris Middleton 16234.727
## Terrence Ross    9405.972
## Ish Smith        6576.191
## Myles Turner    12234.336
## Thaddeus Young  12683.456
```

```r
# Coefficients

coef(ridge.fit, s = bestridge_lambda)
```

```
## 31 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept) -5357.349462
## posPF         -35.771352
## posPG        -525.723930
## posSF         331.935355
## posSG        -433.625881
## age           265.637195
## g             -29.479467
```

```
## gs              39.150786
## mp              38.846573
## fg             192.520019
## fga             74.079614
## fgpercent   -1820.651544
## x3p            516.270084
## x3pa           312.439160
## x3ppercent  -1227.033143
## x2p            136.134341
## x2pa             3.452693
## x2ppercent   6498.212628
## efgpercent    390.182411
## ft             599.639512
## fta            414.380551
## ftpercent   -2973.031826
## orb            262.456020
## drb            186.074215
## trb            128.808577
## ast            446.806723
## stl            285.781085
## blk            696.780007
## tov           -327.040226
## pf            -786.604472
## pts             89.432425
```

Keep note that there are no zero coefficients in this model, this is important when comparing the predictions made by lasso.

e) Fit a lasso regression model on the training set, and predict the salaries in the test set. Report the non-zero coefficient estimates.

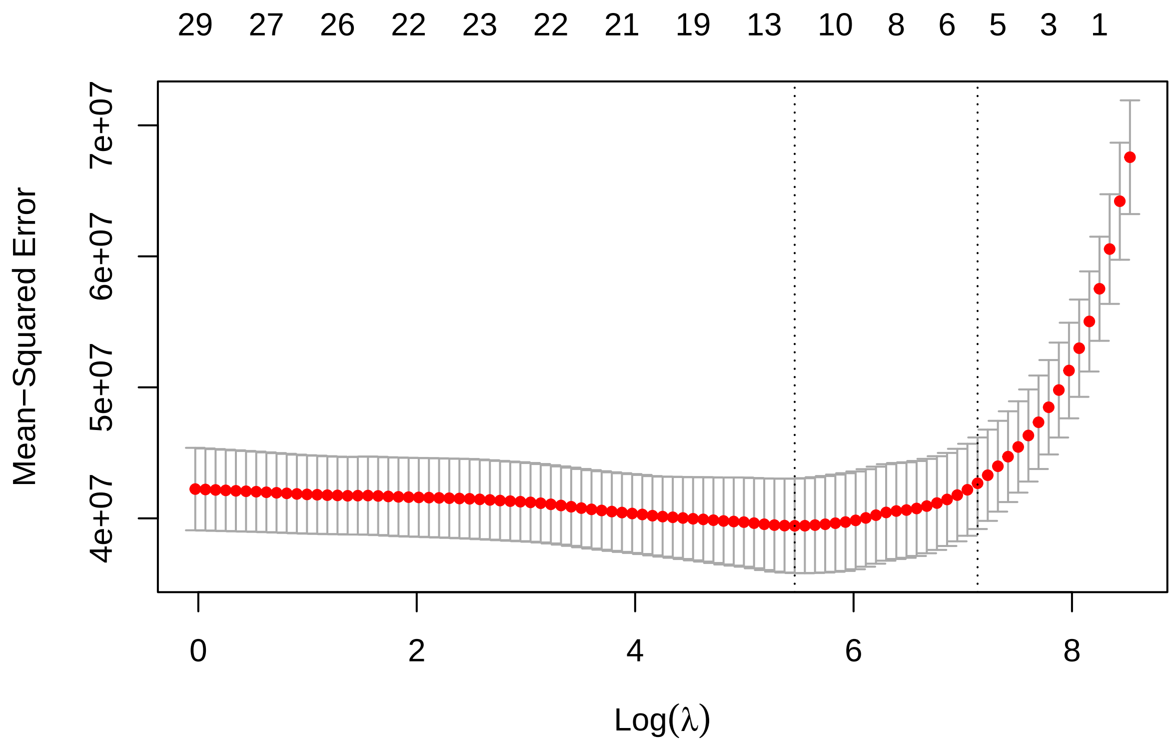
```r
set.seed(1)
lasso.fit <- glmnet(x.fit, trs, alpha = 1)
names(lasso.fit)
```

```
##  [1] "a0"        "beta"      "df"        "dim"       "lambda"    "dev.ratio"
##  [7] "nulldev"   "npasses"   "jerr"      "offset"    "call"      "nobs"
```

```r
cv.lasso <- cv.glmnet(x.fit, trs, alpha = 1, nfolds = 10)
bestlasso_lambda <- cv.lasso$lambda.min # best tuning parameter
bestlasso_lambda
```

```
## [1] 235.3244
```

```r
plot(cv.lasso)
```




```
# Predict

lasso.pred <- predict(lasso.fit, s = bestridge_lambda, newx = y.fit)
head(lasso.pred)
```

```
##                           s1
## Khris Middleton 11988.298
## Terrence Ross   10353.186
## Ish Smith        8051.272
## Myles Turner    10114.499
## Thaddeus Young   9895.244
```

```
# Lasso coefficients
coef(lasso.fit, s = bestlasso_lambda)
```

```
## 31 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) -6436.81525
## posPF             .
## posPG             .
## posSF             .
## posSG             .
## age            293.38081
## g              -31.12952
## gs              65.49790
```

```
## mp                .
## fg                .
## fga               .
## fgpercent         .
## x3p               .
## x3pa           398.96029
## x3ppercent        .
## x2p               .
## x2pa              .
## x2ppercent    7037.20394
## efgpercent        .
## ft            1356.26682
## fta               .
## ftpercent    -3272.60106
## orb               .
## drb            146.58135
## trb            114.87792
## ast            321.99443
## stl               .
## blk            322.10480
## tov               .
## pf            -295.41155
## pts            156.00788
```

Lasso predictions are a lot lower than the ridge predictions. This could be because the way lasso works as a whole. Looking at the coefficients, it is obvious to tell that this is due to all the coefficients not lasso'd in. There are a lot less non-zero coefficients than in ridge.

f) Fit a GAM model on the training set using the features selected in model (b) as the predictors, and plot the results and explain your findings, and then predict the salaries in the test set.

```r
nba.gam <- gam(salary ~ age + s(g, df = 2) + s(gs, df = 2) + x2ppercent + s(pts, df = 5), data = train.d
# I use s() to indicate I want a smoothing spline
par(mfrow = c(2, 3))
plot(nba.gam, se = T, col = "red")
summary(nba.gam)
```

```
##
## Call: gam(formula = salary ~ age + s(g, df = 2) + s(gs, df = 2) + x2ppercent +
##     s(pts, df = 5), data = train.dat)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -17235.1  -3237.3   -658.2   2911.9  21290.7
##
## (Dispersion Parameter for gaussian family taken to be 35810703)
##
##     Null Deviance: 23146145548 on 340 degrees of freedom
## Residual Deviance: 11781715666 on 328.9998 degrees of freedom
## AIC: 6912.771
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
```

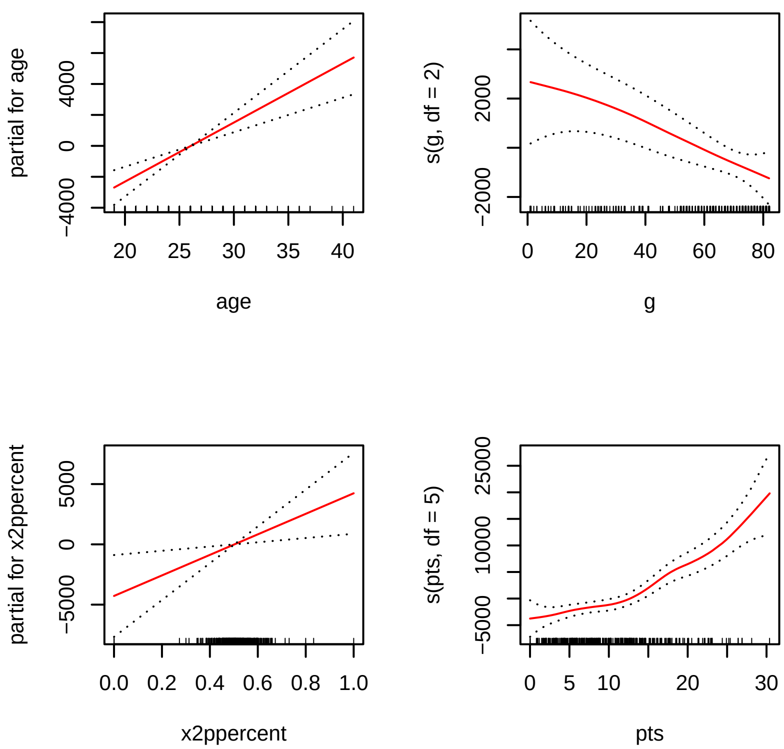
```
##                  Df     Sum Sq    Mean Sq  F value    Pr(>F)
## age               1 1.4671e+09 1467117652  40.9687 5.327e-10 ***
## s(g, df = 2)      1 1.0354e+09 1035385408  28.9127 1.439e-07 ***
## s(gs, df = 2)     1 6.0612e+09 6061185258 169.2562 < 2.2e-16 ***
## x2ppercent        1 2.4646e+08  246461171   6.8823  0.009111 **
## s(pts, df = 5)    1 2.1746e+09 2174625501  60.7256 8.685e-14 ***
## Residuals       329 1.1782e+10   35810703
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                 Npar Df Npar F    Pr(F)
## (Intercept)
## age
## s(g, df = 2)         1 0.2879 0.591950
## s(gs, df = 2)        1 0.6359 0.425770
## x2ppercent
## s(pts, df = 5)       4 3.8712 0.004347 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
nba.gam$aic
```

```
## [1] 6912.771
```

```
test.predict <- predict(nba.gam, test.dat)
test.predict
```

```
## Khris Middleton   Terrence Ross       Ish Smith    Myles Turner  Thaddeus Young
##       17487.701        7018.941        5746.448       10796.790       14151.335
```



Using the 5 predictors selected in the earlier fitted BIC model, I have produced a model with an AIC of 6912.771, which is a pretty good number for this. The predictions look fairly accurate too, this model seems to have produced the best predictions.