STT481 HW4

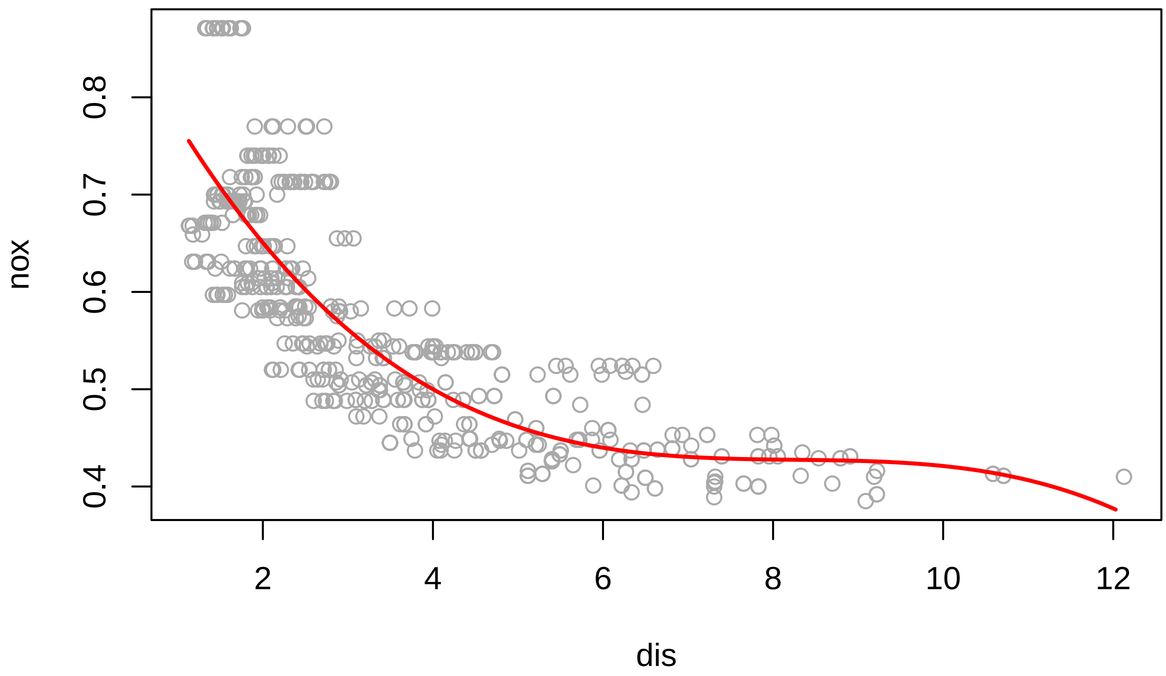
100 points total

Q1. Load the Boston dataset

```r
set.seed(1)
library(MASS)
attach(Boston)
```

(a)

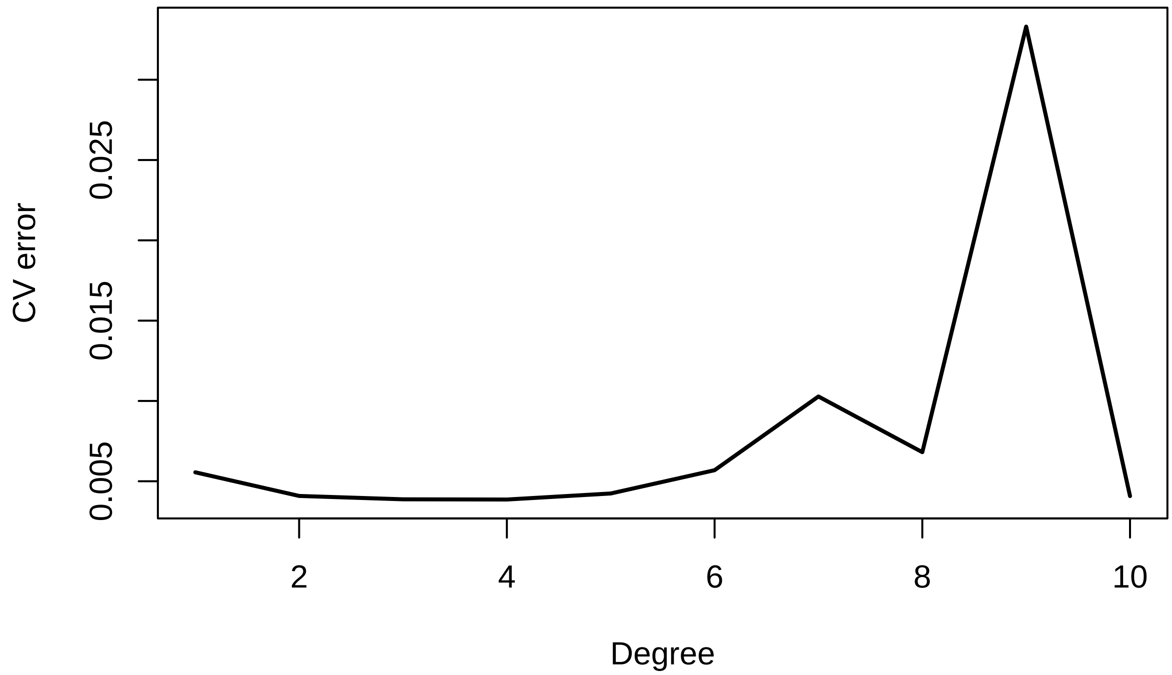
```r
lm.fit = lm(nox ~ poly(dis, 3), data = Boston)

dislim = range(dis)
dis.grid = seq(from = dislim[1], to = dislim[2], by = 0.1)
lm.pred = predict(lm.fit, list(dis = dis.grid))
plot(nox ~ dis, data = Boston, col = "darkgrey")
lines(dis.grid, lm.pred, col = "red", lwd = 2)
```



(b) When the degree is 4, it results in the minimum test error.

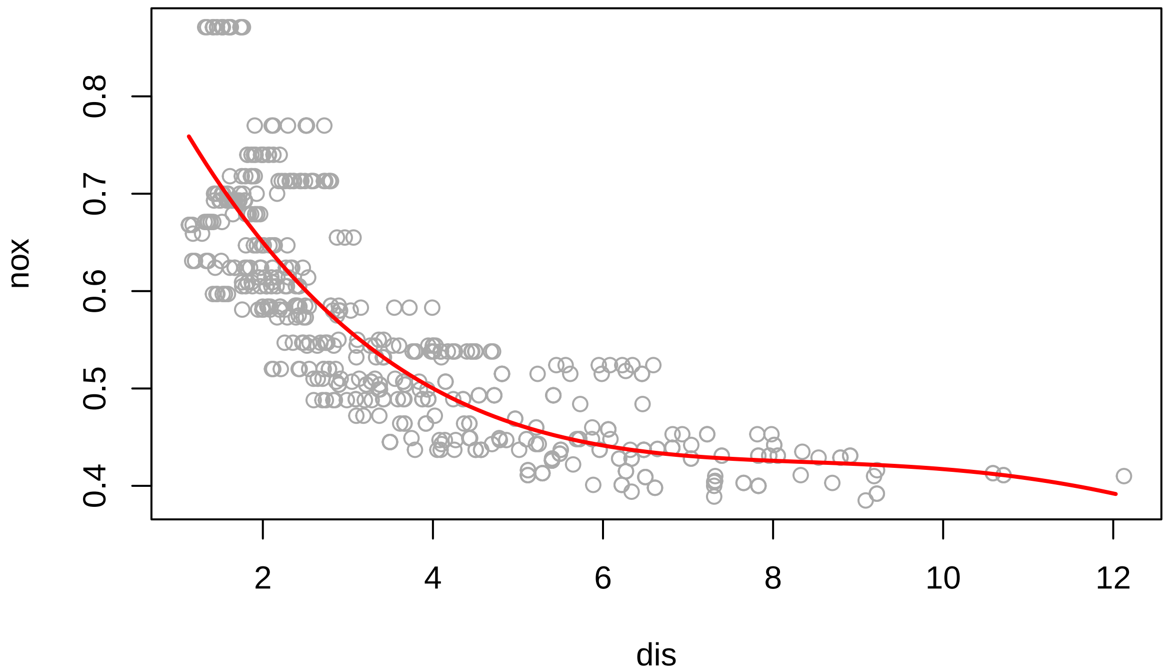
```r
library(boot)
all.deltas = rep(NA, 10)
for (i in 1:10) {
```

```
    glm.fit = glm(nox ~ poly(dis, i), data = Boston)
    all.deltas[i] = cv.glm(Boston, glm.fit, K = 10)$delta[1]
}
plot(1:10, all.deltas, xlab = "Degree", ylab = "CV error", type = "l", pch = 20, lwd = 2)
```



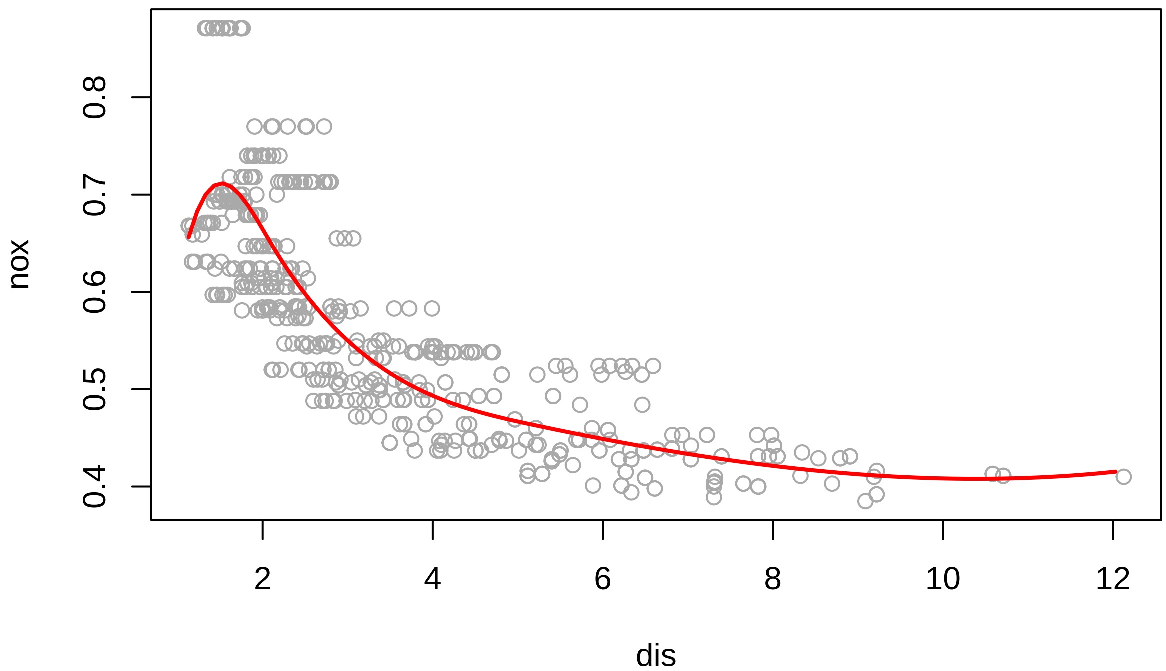
The polynomial regression with the best degree is given below.

```
lm.fit = lm(nox ~ poly(dis, which.min(all.deltas)), data = Boston)
lm.pred = predict(lm.fit, list(dis = dis.grid))
plot(nox ~ dis, data = Boston, col = "darkgrey")
lines(dis.grid, lm.pred, col = "red", lwd = 2)
```



(c) We fit a cubic spline with 6 degrees of freedom.

```
library(splines)
sp.fit = lm(nox ~ bs(dis, df = 6), data = Boston)

sp.pred = predict(sp.fit, list(dis = dis.grid))
plot(nox ~ dis, data = Boston, col = "darkgrey")
lines(dis.grid, sp.pred, col = "red", lwd = 2)
```



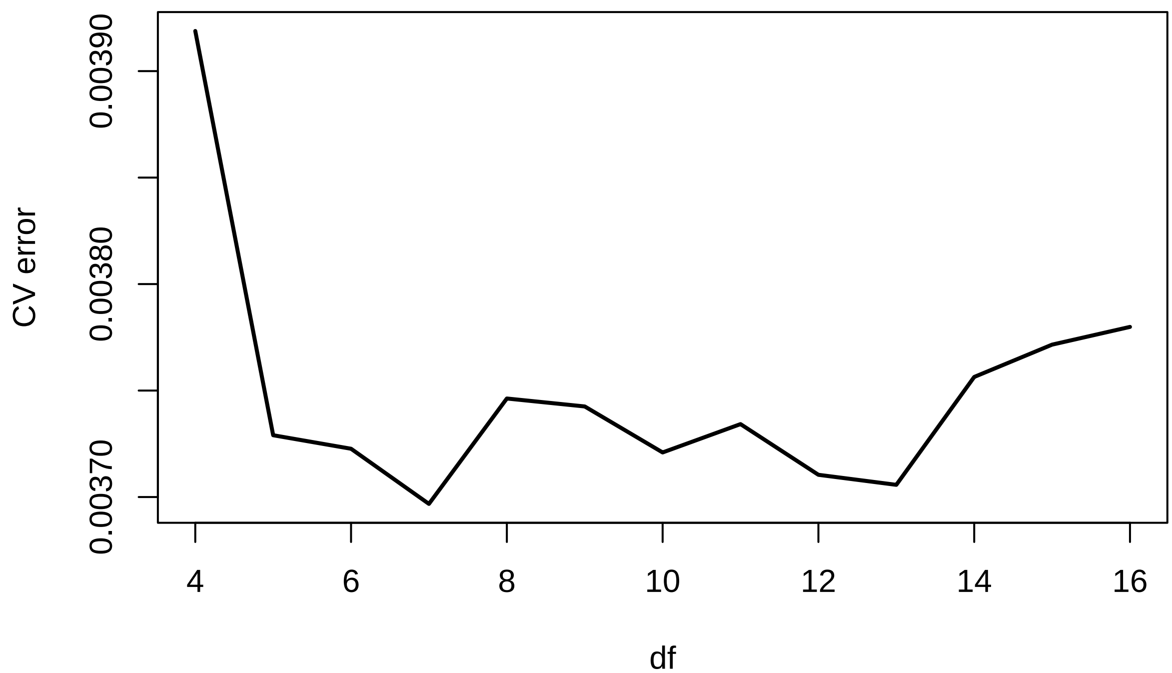
```r
attr(bs(dis, df = 6), "knots")
```

```
##      25%      50%      75%
## 2.100175 3.207450 5.188425
```

The knot locations are 2.100175, 3.207450, and 5.188425.

(d) We use a 10-fold cross-validation to find best df. We try all integer values of df between 4 and 16.

```r
set.seed(1)
all.df <- 4:16
all.cv <- rep(0, length(all.df))
counter <- 0
for (df in all.df) {
  counter <- counter + 1
  lm.fit <- glm(nox ~ bs(dis, df = df), data = Boston)
  all.cv[counter] <- cv.glm(Boston, lm.fit, K = 10)$delta[1]
}

plot(all.df, all.cv, lwd = 2, type = "l", xlab = "df", ylab = "CV error")
```



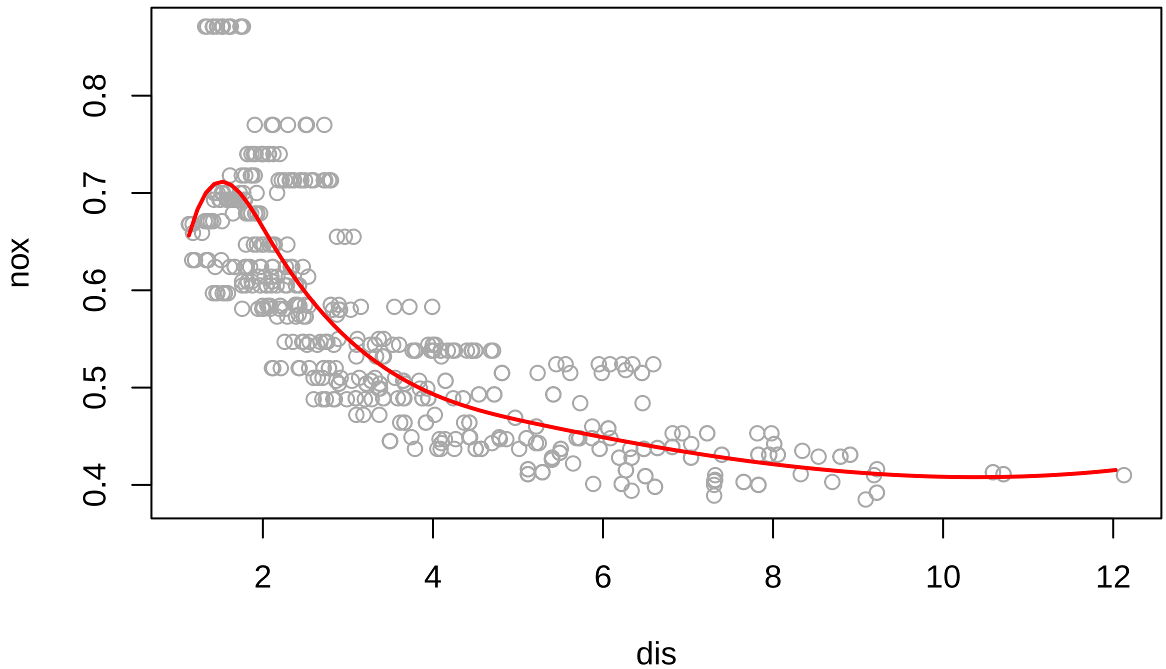

```r
print(all.df[which.min(all.cv)])
```

```
## [1] 7
```

CV error is more jumpy in this case, but attains minimum at df=7. We pick 7 as the optimal degrees of freedom.

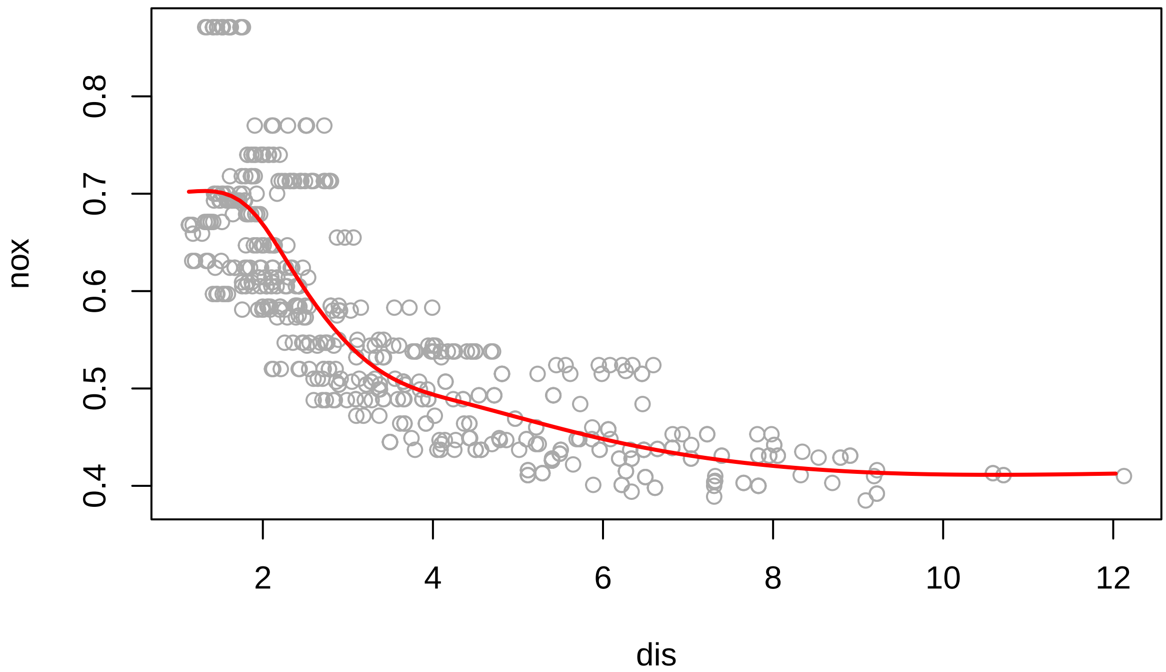
The cubic spline with the best degrees of freedom is given below.

```r
lm.fit <- glm(nox ~ bs(dis, df = all.df[which.min(all.cv)]), data = Boston)
sp.pred <- predict(sp.fit, list(dis = dis.grid))
plot(nox ~ dis, data = Boston, col = "darkgrey")
lines(dis.grid, sp.pred, col = "red", lwd = 2)
```



(e) We fit a natural cubic spline with 6 degrees of freedom.

```r
library(splines)
sp.fit = lm(nox ~ ns(dis, df = 6), data = Boston)

sp.pred = predict(sp.fit, list(dis = dis.grid))
plot(nox ~ dis, data = Boston, col = "darkgrey")
lines(dis.grid, sp.pred, col = "red", lwd = 2)
```



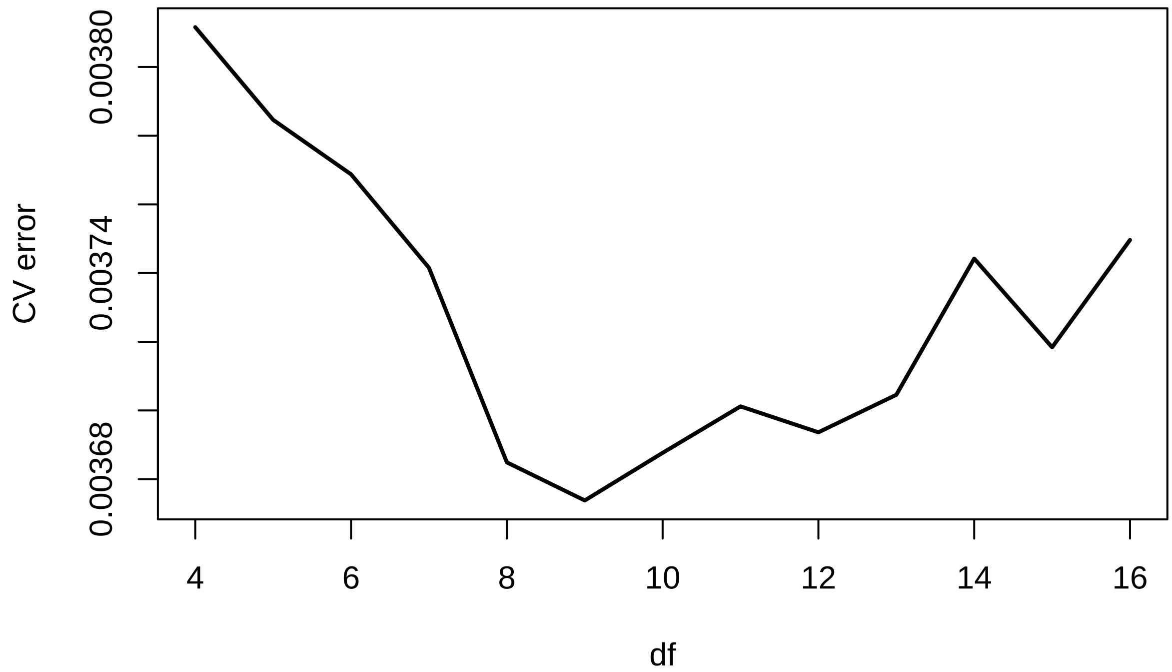

```
attr(ns(dis, df = 6), "knots")
```

```
## 16.66667% 33.33333%       50% 66.66667% 83.33333%
##  1.851317  2.384033  3.207450  4.325700  6.062200
```

The knot locations are 1.851317, 2.384033, 3.207450, 4.325700, and 6.062200.

(f) We use a 10-fold cross-validation to find best df. We try all integer values of df between 4 and 16.

```
set.seed(1)
all.df <- 4:16
all.cv <- rep(0, length(all.df))
counter <- 0
for (df in all.df) {
  counter <- counter + 1
  lm.fit <- glm(nox ~ ns(dis, df = df), data = Boston)
  all.cv[counter] <- cv.glm(Boston, lm.fit, K = 10)$delta[1]
}

plot(all.df, all.cv, lwd = 2, type = "l", xlab = "df", ylab = "CV error")
```



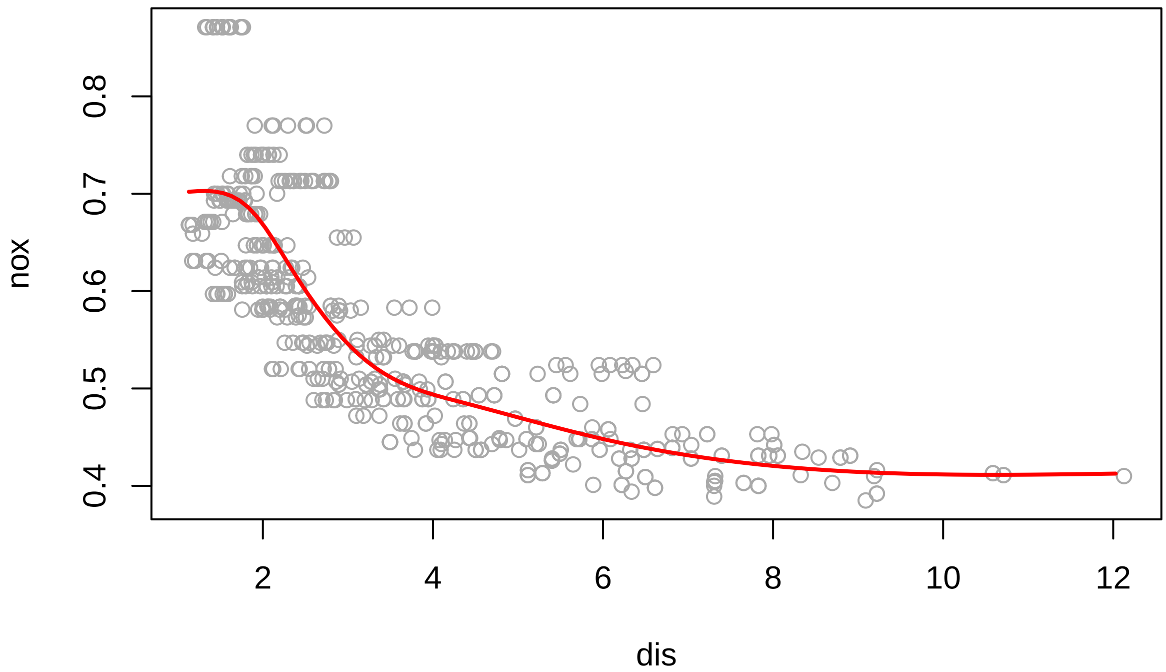

```
print(all.df[which.min(all.cv)])
```

```
## [1] 9
```

CV error is more jumpy in this case, but attains minimum at df=9. We pick 9 as the optimal degrees of freedom.

The natural cubic spline with the best degrees of freedom is given below.

```
lm.fit <- glm(nox ~ ns(dis, df = all.df[which.min(all.cv)]), data = Boston)
sp.pred <- predict(sp.fit, list(dis = dis.grid))
plot(nox ~ dis, data = Boston, col = "darkgrey")
lines(dis.grid, sp.pred, col = "red", lwd = 2)
```



Q2

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

(a)

```
library(leaps)
reg.fit = regsubsets(Outstate ~ ., data = College.train, nvmax = 17, method = "forward")
reg.summary = summary(reg.fit)
par(mfrow = c(1, 3))
min.cp = which.min(reg.summary$cp)
print(min.cp)
```

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

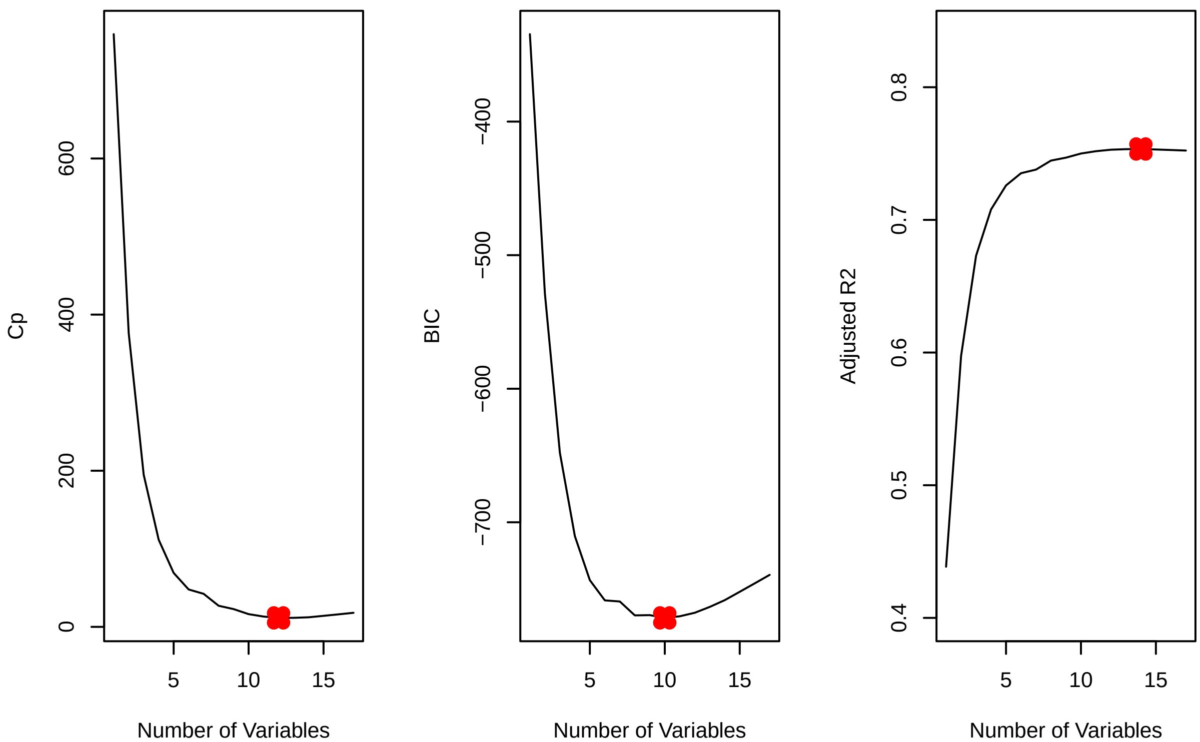
```
plot(reg.summary$cp, xlab = "Number of Variables", ylab = "Cp", type = "l")
points(min.cp, reg.summary$cp[min.cp], pch = 4, col = "red", lwd = 7)

min.bic = which.min(reg.summary$bic)
print(min.bic)
```

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

```r
plot(reg.summary$bic, xlab = "Number of Variables", ylab = "BIC", type = "l")
points(min.bic, reg.summary$bic[min.bic], pch = 4, col = "red", lwd = 7)

max.adjr2 = which.max(reg.summary$adjr2)
print(max.adjr2)
```

```
## [1] 14
```

```r
plot(reg.summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted R2",
    type = "l", ylim = c(0.4, 0.84))
points(max.adjr2, reg.summary$adjr2[max.adjr2], pch = 4, col = "red", lwd = 7)
```




```r
coefi = coef(reg.fit, id = min.bic)
names(coefi)
```

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

Cp and Adj R2 show that size 12 and 14 are the minimum sizes for the subset, respectively, while BIC chooses the size = 10. We pick 10 as the best subset size based on BIC criterion but you can also use Cp or Adj R2 as your crierion.

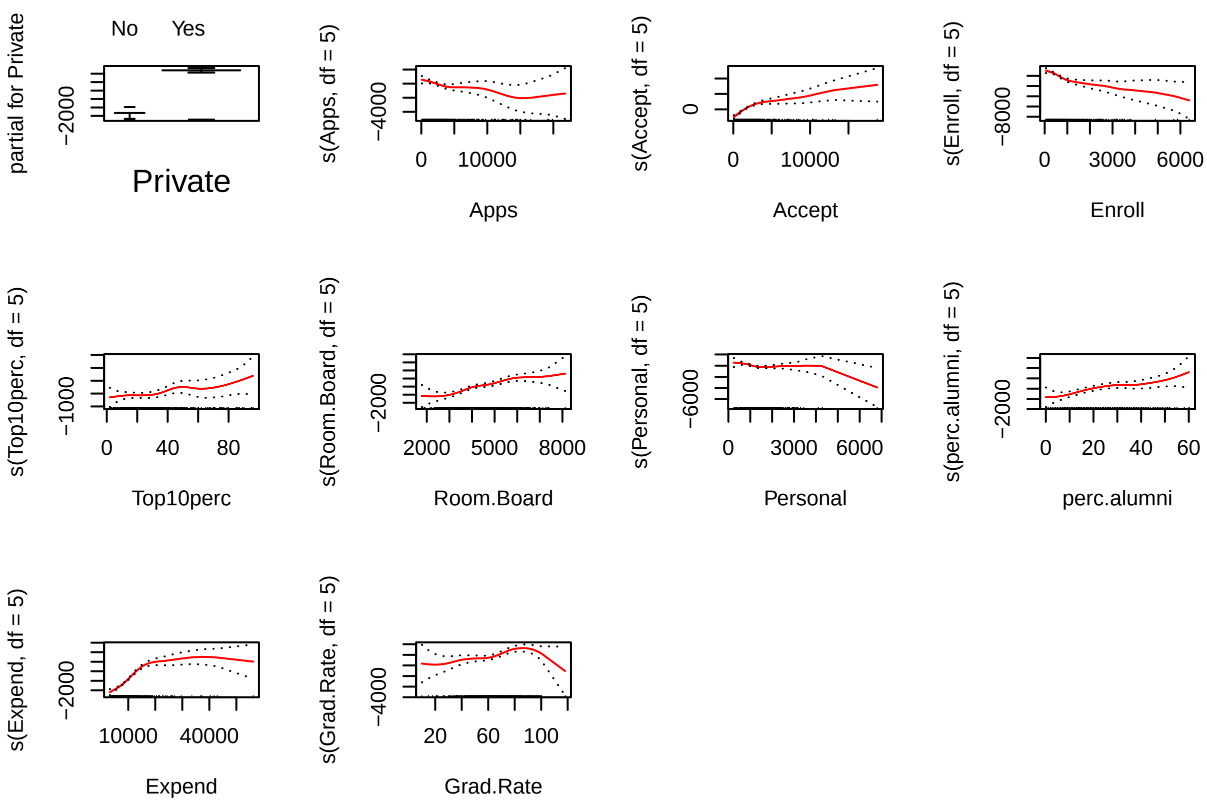
(b)

```
library(gam)
```

```
## Loading required package: foreach
```

```
## Loaded gam 1.20.2
```

```
gam.fit = gam(Outstate ~ Private + s(Apps, df = 5) + s(Accept, df = 5) + s(Enroll, df = 5) +
                s(Top10perc, df = 5) + s(Room.Board, df = 5) + s(Personal, df = 5) +
                s(perc.alumni, df = 5) + s(Expend, df = 5) + s(Grad.Rate, df = 5), data = College.train
par(mfrow = c(3, 4))
plot(gam.fit, se = T, col = "red")
```



Findings: From the plots, it appears that higher enroll numbers result in lower out-of-state tuition fee. It also appears that "Accept", "Enroll", "Room.Board", and "perc.alumni" have a linear relationship with out-of-state tuition. "Private" predictor indicates private and public universities have significantly different out-of-state tuition by holding other predictors. For the predictors "Apps", "Top10perc", "Personal", "Expend" and "Grad.Rate", it seems there exists a quadratic or higher degrees effect. Note here I chose `df = 5` but you can use other dfs. In general, `df` =3-6 would be recommended.

(c)

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

```
## [1] 3183501
```

We obtain a test mean squared error of 3183501 using GAM with 10 predictors.

We can use **summary** to see whether the non-linear effects are significant or not.

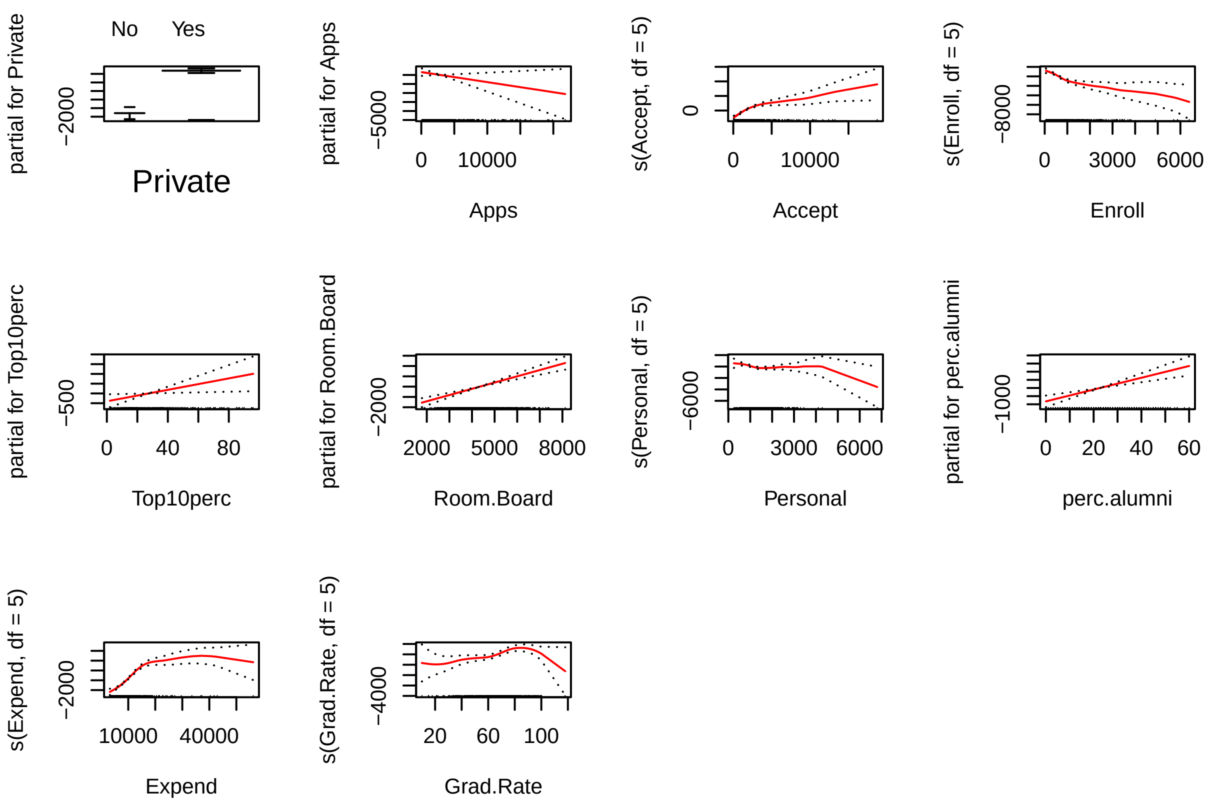
```
summary(gam.fit)
```

```
##
## Call: gam(formula = Outstate ~ Private + s(Apps, df = 5) + s(Accept,
##     df = 5) + s(Enroll, df = 5) + s(Top10perc, df = 5) + s(Room.Board,
##     df = 5) + s(Personal, df = 5) + s(perc.alumni, df = 5) +
##     s(Expend, df = 5) + s(Grad.Rate, df = 5), data = College.train)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -6275.98 -1069.78    85.95  1147.84  7446.96
##
## (Dispersion Parameter for gaussian family taken to be 3210667)
##
##     Null Deviance: 9694792870 on 599 degrees of freedom
## Residual Deviance: 1775499257 on 553.0002 degrees of freedom
## AIC: 10738.98
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##                         Df     Sum Sq    Mean Sq F value    Pr(>F)
## Private                  1 2742410325 2742410325 854.156 < 2.2e-16 ***
## s(Apps, df = 5)          1 1195413597 1195413597 372.326 < 2.2e-16 ***
## s(Accept, df = 5)        1   97976877   97976877  30.516 5.103e-08 ***
## s(Enroll, df = 5)        1  171618769  171618769  53.453 9.337e-13 ***
## s(Top10perc, df = 5)     1  815049317  815049317 253.857 < 2.2e-16 ***
## s(Room.Board, df = 5)    1  491831182  491831182 153.187 < 2.2e-16 ***
## s(Personal, df = 5)      1   39418544   39418544  12.277 0.0004956 ***
## s(perc.alumni, df = 5)   1  219360932  219360932  68.323 1.035e-15 ***
## s(Expend, df = 5)        1  574736351  574736351 179.008 < 2.2e-16 ***
## s(Grad.Rate, df = 5)     1   43959989   43959989  13.692 0.0002369 ***
## Residuals              553 1775499257    3210667
## ---
## 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(Apps, df = 5)              4  2.1759 0.0704230 .
## s(Accept, df = 5)            4 21.4399  2.22e-16 ***
## s(Enroll, df = 5)            4  5.2582 0.0003665 ***
## s(Top10perc, df = 5)         4  1.2001 0.3096976
## s(Room.Board, df = 5)        4  1.9962 0.0937235 .
## s(Personal, df = 5)          4  2.8825 0.0221169 *
## s(perc.alumni, df = 5)       4  1.0929 0.3591952
## s(Expend, df = 5)            4 25.3298 < 2.2e-16 ***
## s(Grad.Rate, df = 5)         4  3.3488 0.0100772 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

To avoid overfitting issue, you can also remove the insignificant nonlinear effects, such as `perc.alumni`, where the p-value is 0.3591952. For example, you can do so by replacing `s(perc.alumni, df=5)` with `perc.alumni`. See below.

```
gam.fit = gam(Outstate ~ Private + Apps + s(Accept, df = 5) + s(Enroll, df = 5) +
                Top10perc + Room.Board + s(Personal, df = 5) + perc.alumni +
                s(Expend, df = 5) + s(Grad.Rate, df = 5), data = College.train)
# I leave Grad.Rate as it is since it's not too insignificant
par(mfrow = c(3, 4))
plot(gam.fit, se = T, col = "red")
gam.pred = predict(gam.fit, College.test)
gam.err = mean((College.test$Outstate - gam.pred)^2)
gam.err
```

```
## [1] 2846868
```



We obtain a lower test mean squared error of 2846868 by removing the insignificant nonlinear effects.

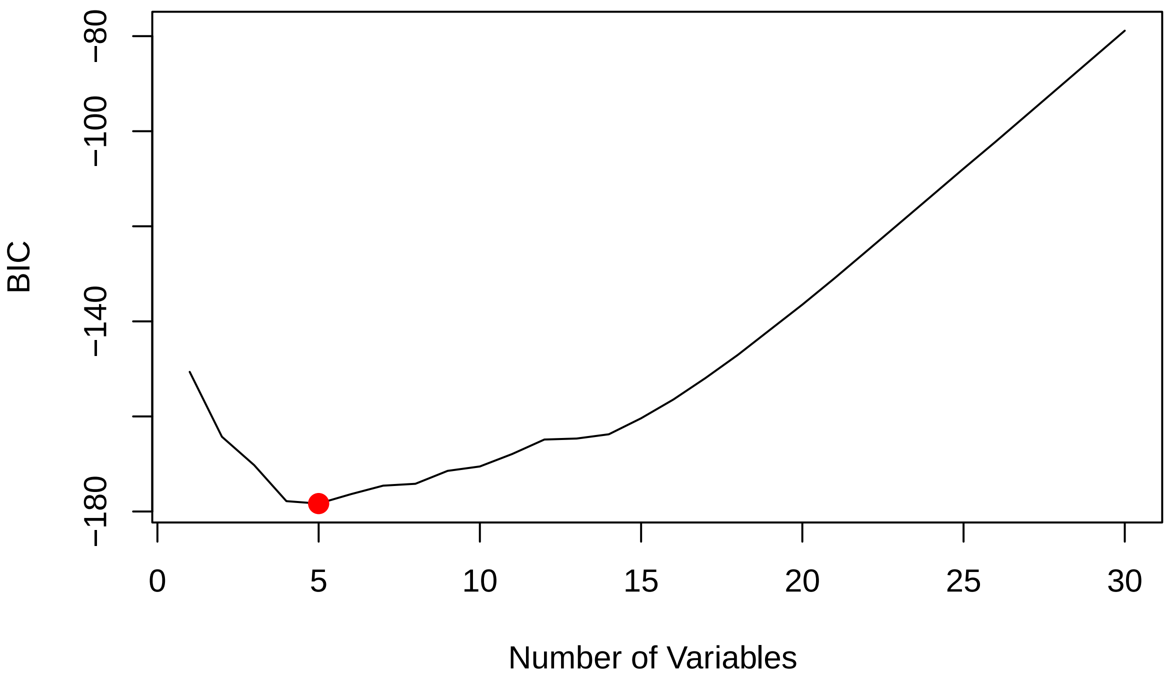
Q3 (a)

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

(b)

```
library(leaps)
regfit.fwd <- regsubsets(salary ~ . , data = train.dat, nvmax = 32, method = "forward")
```

```r
reg.fwd.summary <- summary(regfit.fwd)
plot(reg.fwd.summary$bic, xlab = "Number of Variables ", ylab = "BIC", type = "l")
points(which.min(reg.fwd.summary$bic),reg.fwd.summary$bic[which.min(reg.fwd.summary$bic)], col = "red",
```




```r
coef(regfit.fwd, 5)
```

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

BIC criterion indicates it selects 5 predictors, and they are `age`, `g`, `gs`, `x2ppercent`, and `pts`.

(c) The prediction of their salaries are below.

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

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

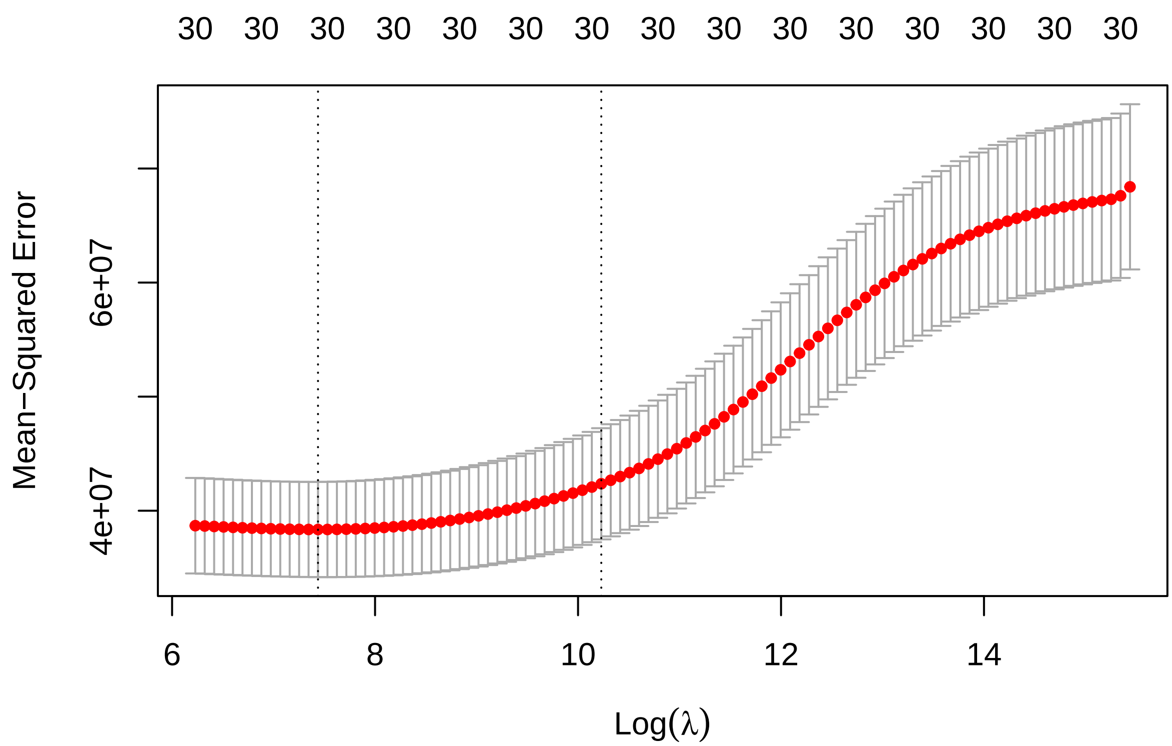
(d) We fit a ridge regression model on the training set and predict the salaries using the fitted model.

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-4
```

```
X <- model.matrix(salary ~ . , train.dat)[,-1]  # the first column is the intercept
y <- train.dat$salary
ridge.mod <- glmnet(X, y, alpha = 0)  # 0 indicates ridge regression
cv.out <- cv.glmnet(X, y, alpha = 0, nfolds = 10)
plot(cv.out)
```




```
bestlam <- cv.out$lambda.min
bestlam
```

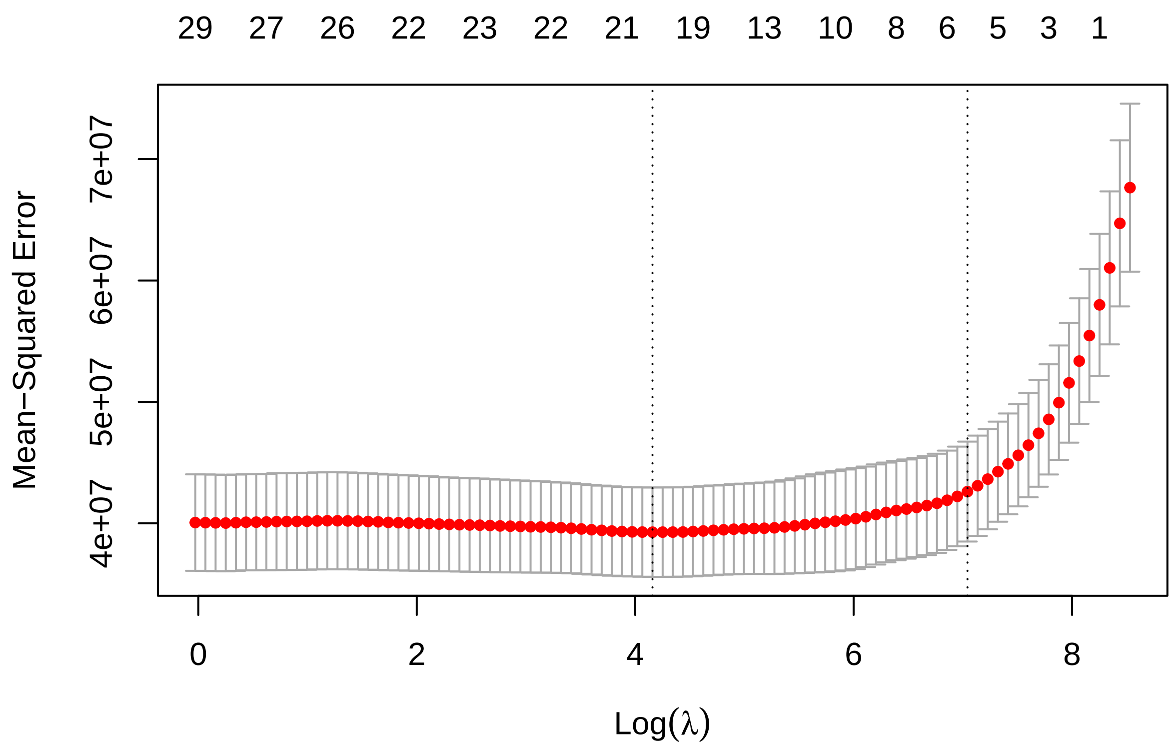
```
## [1] 1699.232
```

```
test.X <- model.matrix(salary ~., test.dat)[,-1] # same form as X
ridge.pred <- predict(ridge.mod, s = bestlam, newx = test.X)
ridge.pred
```

```
##                         s1
## Khris Middleton 16369.088
## Terrence Ross    9126.686
## Ish Smith        6365.419
## Myles Turner    12378.610
## Thaddeus Young  12827.164
```

(e) We fit a lasso model on the training set and predict the salaries using the fitted model, and report the non-zero coefficient estimates.

```
lasso.mod <- glmnet(X, y, alpha = 1)
cv.out <- cv.glmnet(X, y, alpha = 1)
plot(cv.out)
```




```
bestlam <- cv.out$lambda.min
coef(lasso.mod, s = bestlam)
```

```
## 31 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept) -5887.8085261
## posPF            .
## posPG        -955.1017882
## posSF         129.0623908
## posSG        -172.5341187
## age           316.9460181
## g             -48.6773725
## gs             69.4018629
## mp               .
## fg            367.0638893
## fga              .
## fgpercent   -1716.0581083
## x3p              .
## x3pa          684.2299547
## x3ppercent   -154.8122753
## x2p             0.4944476
## x2pa             .
```

```
## x2ppercent   9281.5959872
## efgpercent      .
## ft           1747.9818535
## fta             .
## ftpercent   -4429.0627815
## orb           564.5035918
## drb             .
## trb           252.6887510
## ast          1064.3383195
## stl             .
## blk           982.4325082
## tov         -1933.3153465
## pf           -703.1011589
## pts             .
```

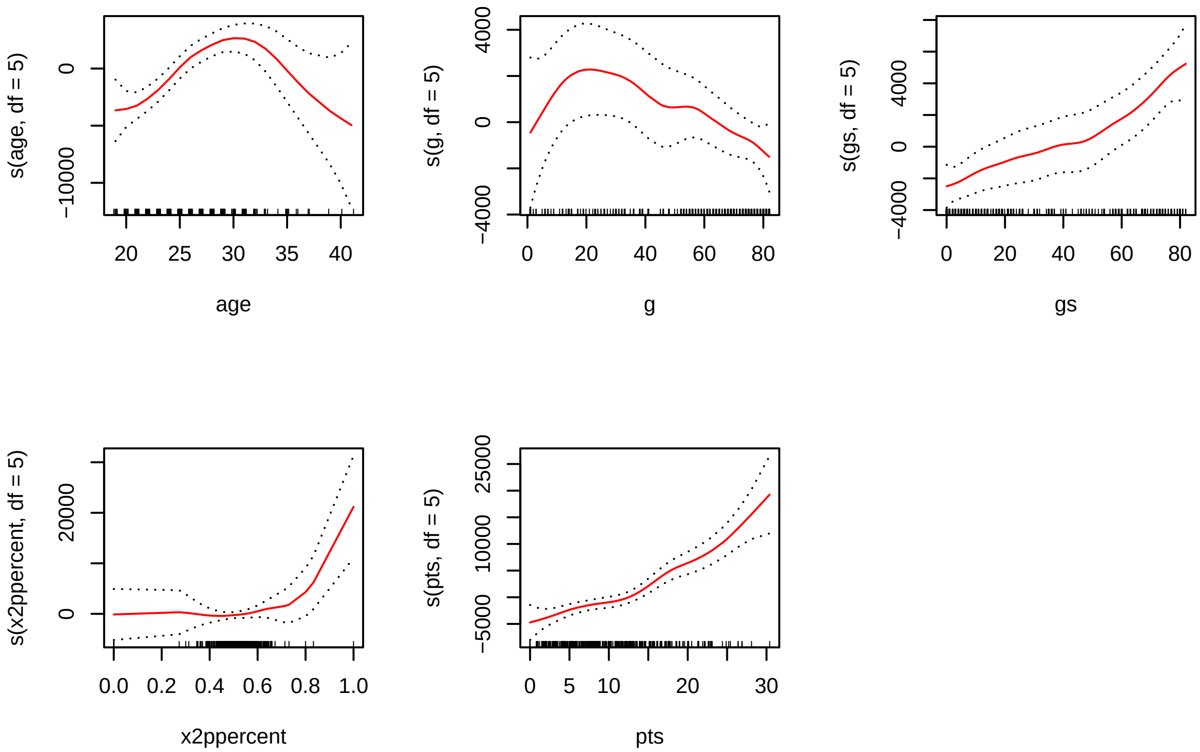
```
lasso.pred <- predict(lasso.mod, s = bestlam, newx = test.X)
lasso.pred
```

```
##                        s1
## Khris Middleton 16471.288
## Terrence Ross    8022.151
## Ish Smith        5884.402
## Myles Turner    12878.233
## Thaddeus Young  13362.675
```

(f) We fit a GAM on the training set using the five predictors: `age`, `g`, `gs`, `x2ppercent`, and `pts`, and predict the salaries using the fitted model.

```
library(gam)
gam.fit <- gam(salary ~ s(age, df = 5) + s(g, df = 5) + s(gs, df = 5) +
                 s(x2ppercent, df = 5) + s(pts, df = 5),
               data = train.dat)
par(mfrow = c(2, 3))
plot(gam.fit, se = T, col = "red")
predict(gam.fit, test.dat)
```

```
## Khris Middleton   Terrence Ross       Ish Smith    Myles Turner  Thaddeus Young
##       18961.822        8257.568        7890.952        9736.555       15201.548
```



Findings: The `age` appears to have a quadratic effect: if the player is young or old, the salary is low, and if the player is at middle age, the salary is high. The `gs` have a linear effect: if the player plays more games as a starter, then salary is high. Similarly `g` and `x2ppercent` appears to have non-linear effects, and `pts` has a linear effect (which is not too surprising: if a player scores more, then the salary is higher).

Similarly, we can use `summary` to see whether the non-linear effects are significant or not.

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

```
##
## Call: gam(formula = salary ~ s(age, df = 5) + s(g, df = 5) + s(gs,
##     df = 5) + s(x2ppercent, df = 5) + s(pts, df = 5), data = train.dat)
## Deviance Residuals:
##    Min      1Q Median     3Q    Max
## -17058  -3134   -643   2841  17099
##
## (Dispersion Parameter for gaussian family taken to be 32577645)
##
##     Null Deviance: 23146145548 on 340 degrees of freedom
## Residual Deviance: 10261941621 on 314.9995 degrees of freedom
## AIC: 6893.678
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##                        Df     Sum Sq    Mean Sq  F value    Pr(>F)
## s(age, df = 5)          1 1.5450e+09 1544968977  47.4242 3.109e-11 ***
## s(g, df = 5)            1 1.2687e+09 1268653996  38.9425 1.407e-09 ***
## s(gs, df = 5)           1 5.8595e+09 5859484135 179.8621 < 2.2e-16 ***
## s(x2ppercent, df = 5)   1 1.4296e+08  142959048   4.3883   0.03699 *
```

```
## s(pts, df = 5)          1 2.2529e+09 2252932342  69.1558 2.772e-15 ***
## Residuals             315 1.0262e+10   32577645
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                       Npar Df Npar F     Pr(F)
## (Intercept)
## s(age, df = 5)              4 6.8175 2.837e-05 ***
## s(g, df = 5)                4 1.5708  0.181864
## s(gs, df = 5)               4 0.8509  0.493884
## s(x2ppercent, df = 5)       4 4.0478  0.003244 **
## s(pts, df = 5)              4 3.5543  0.007456 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

It appears that `g` and `gs`'s non-linear effects are very insignificant. If we remove the non-linear effects of `g` and `gs`, then the prediction is below.

```
gam.fit <- gam(salary ~ s(age, df = 5) + g + gs +
                 s(x2ppercent, df = 5) + s(pts, df = 5),
               data = train.dat)
gam.pred <- predict(gam.fit, test.dat)
gam.pred
```

```
## Khris Middleton   Terrence Ross       Ish Smith    Myles Turner  Thaddeus Young
##       18319.679        8471.934        7070.722        9352.009       14714.949
```