Bruce Campbell ST-617 Homework 3

Wed Jul 13 09:12:16 2016

Chapter 6

Problem 8

In this exercise, we will generate simulated data, and will then use this data to perform best subset selection.

a)

Use the rnorm() function to generate a predictor X of length n = 100, as well as a noise vector ϵ of length n = 100.

```
rm(list = ls())
set.seed(123)
X <- rnorm(100, mean = 0, sd = 1)

epsilon <- rnorm(100, mean = 0, sd = 1)
```

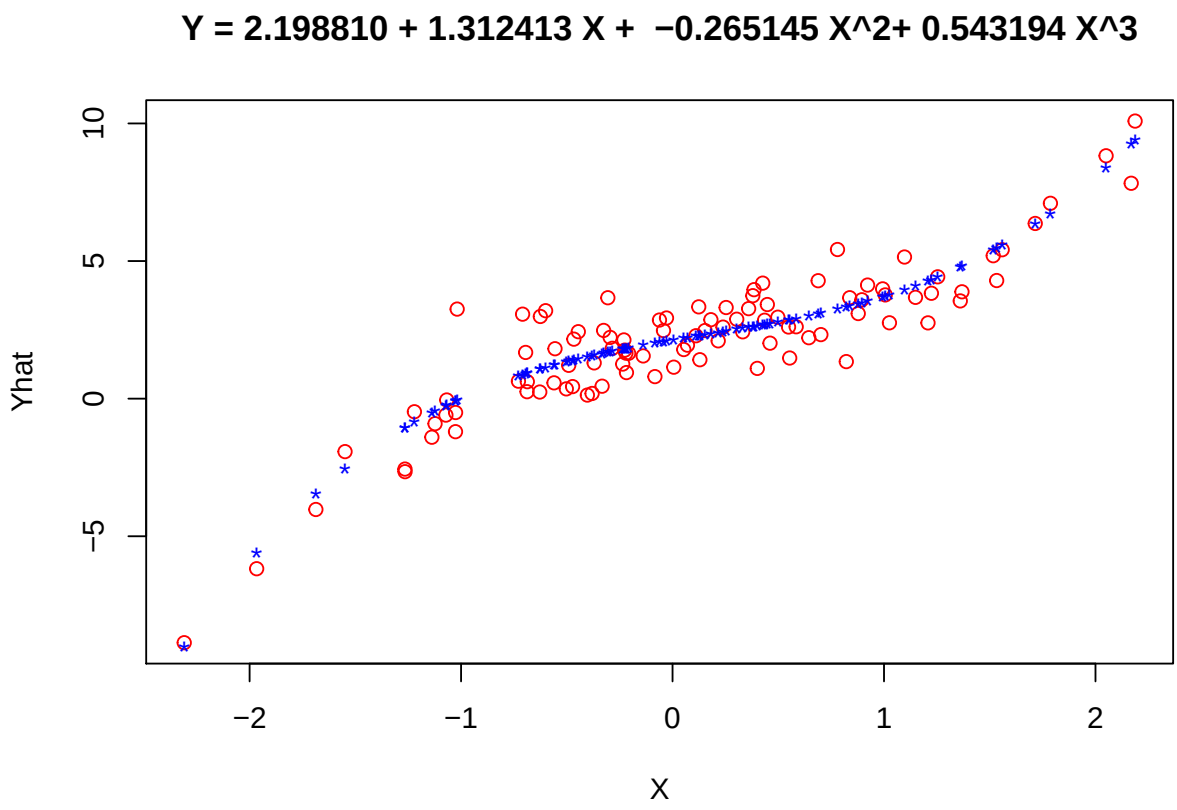
b)

Generate a response vector Y of length n = 100 according to the model

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$$

, where $\beta_0, \beta_1, \beta_2, \beta_3$ are constants of your choice.

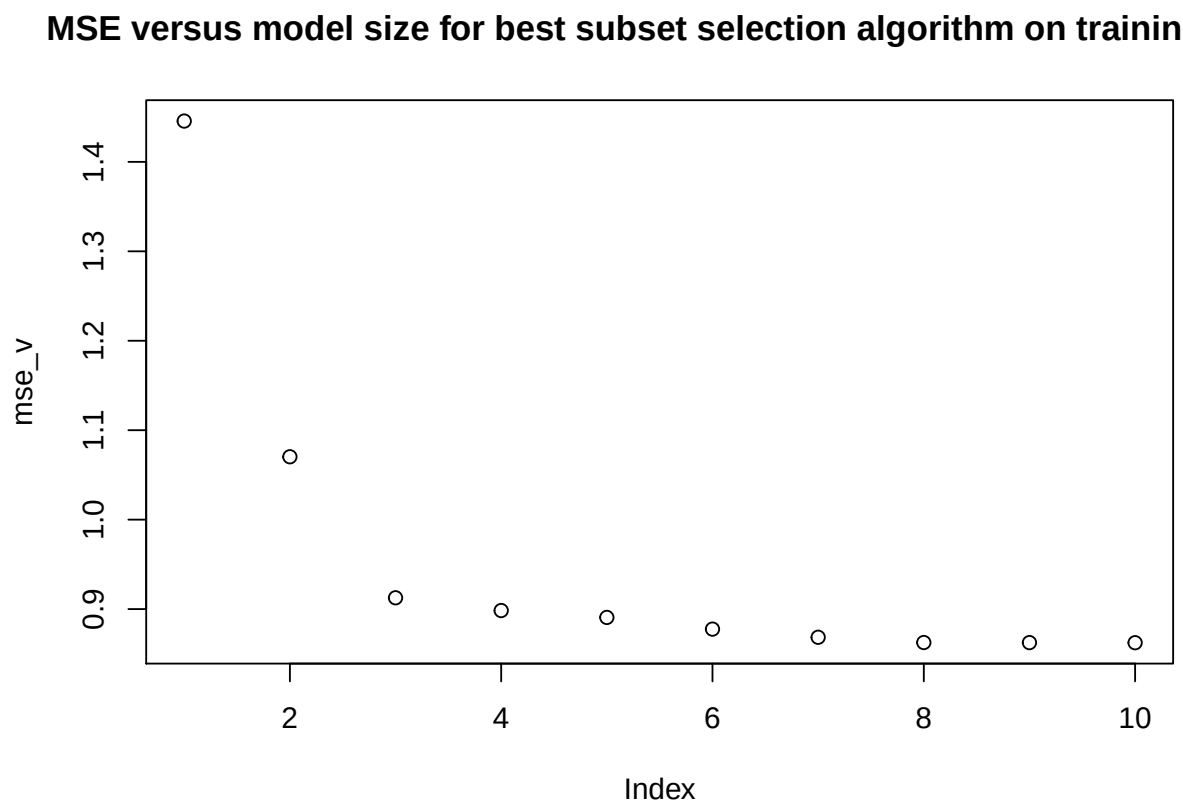
```
beta = rnorm(4, mean = 0, sd = 1)
Y <- matrix(NA, nrow = 100, ncol = 1)
Yhat <- matrix(NA, nrow = 100, ncol = 1)

for (i in 1:100) {
    Y[i] = beta[1] + beta[2] * X[i] + beta[3] * X[i]^2 + beta[4] * X[i]^3

    Yhat[i] = beta[1] + beta[2] * X[i] + beta[3] * X[i]^2 + beta[4] * X[i]^3 +
        epsilon[i]
}

plot(X, Yhat, col = "red")
points(X, Y, pch = "*", col = "blue")
title(main = sprintf("Y = %f + %f X +  %f X^2+ %f X^3", beta[1], beta[2], +beta[3],
    beta[4]), cex = 4.6)
```



c)

Use the regsubsets() function to perform best subset selection in order to choose the best model containing the predictors $X, X^2, ..., X^10$. What is the best model obtained according to Cp, BIC, and adjusted R2? Show some plots to provide evidence for your answer, and report the coefficients of the best model obtained. Note you will need to use the data.frame() function to create a single data set containing both X and Y .

```
DF <- as.data.frame(X)
DF <- cbind(DF, Yhat)
names(DF) = c("X", "Y")
library(leaps)

regfit.full <- regsubsets(Y ~ X + I(X^2) + I(X^3) + I(X^4) + I(X^5) + I(X^6) +
    I(X^7) + I(X^8) + I(X^9) + I(X^10), data = DF, nvmax = 10)

reg.summary <- summary(regfit.full)

mse_v <- reg.summary$rss/nrow(DF)

plot(mse_v)
title("MSE versus model size for best subset selection algorithm on training set.")
```



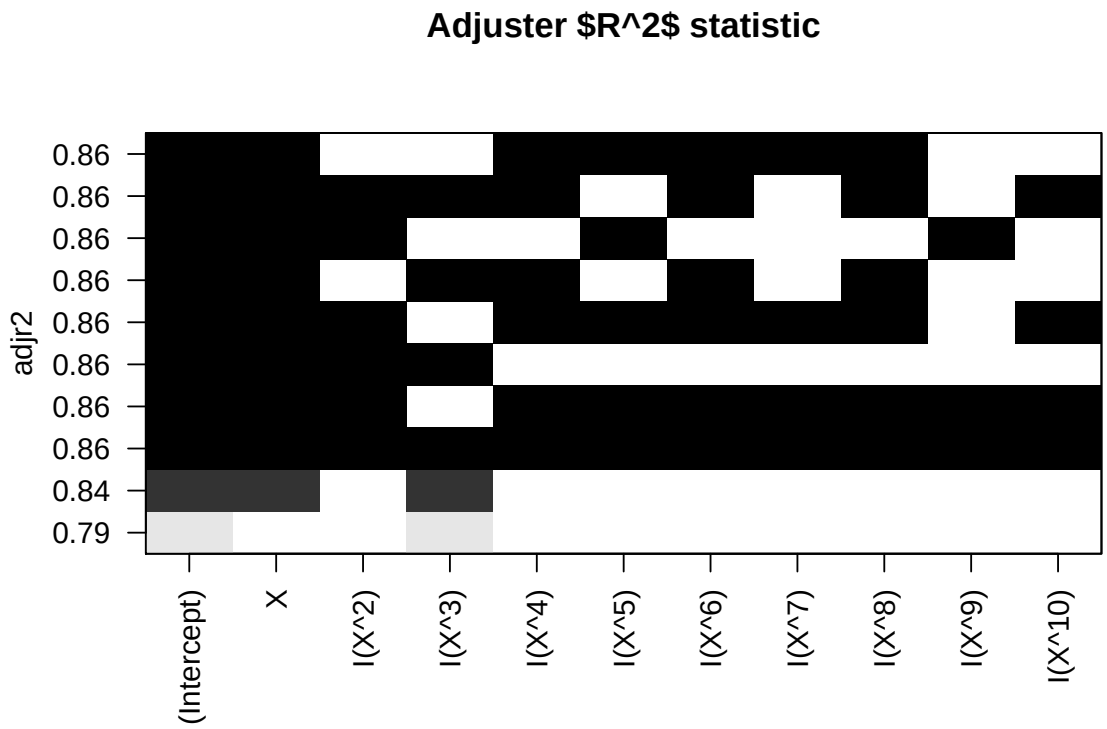
We see that there is a sharp drop in the training set MSE until 3 or 4 predictors are included and that there is a steady decrease as additional polynomial terms are included. This is attributed to over fitting to the training data.

```
summary(regfit.full)
```

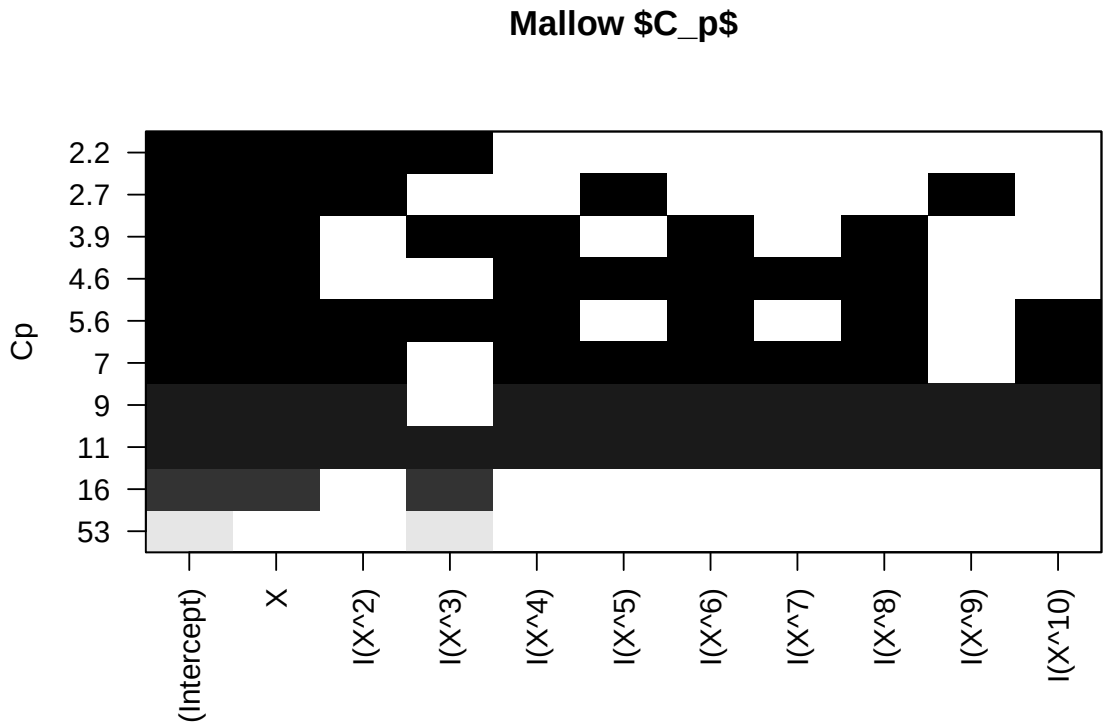
```
## Subset selection object
## Call: regsubsets.formula(Y ~ X + I(X^2) + I(X^3) + I(X^4) + I(X^5) +
##     I(X^6) + I(X^7) + I(X^8) + I(X^9) + I(X^10), data = DF, nvmax = 10)
## 10 Variables  (and intercept)
##          Forced in Forced out
## X            FALSE      FALSE
## I(X^2)       FALSE      FALSE
## I(X^3)       FALSE      FALSE
## I(X^4)       FALSE      FALSE
## I(X^5)       FALSE      FALSE
## I(X^6)       FALSE      FALSE
## I(X^7)       FALSE      FALSE
## I(X^8)       FALSE      FALSE
## I(X^9)       FALSE      FALSE
## I(X^10)      FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: exhaustive
##           X   I(X^2) I(X^3) I(X^4) I(X^5) I(X^6) I(X^7) I(X^8) I(X^9)
## 1  ( 1 )  " " " "    "*"    " "    " "    " "    " "    " "    " "
```

```
## 2  ( 1 )  "*" " "    "*"    " "    " "    " "    " "    " "    " "
## 3  ( 1 )  "*" "*"    "*"    " "    " "    " "    " "    " "    " "
## 4  ( 1 )  "*" "*"    " "    " "    "*"    " "    " "    " "    "*"
## 5  ( 1 )  "*" " "    "*"    "*"    " "    "*"    " "    "*"    " "
## 6  ( 1 )  "*" " "    " "    "*"    "*"    "*"    "*"    "*"    " "
## 7  ( 1 )  "*" "*"    "*"    "*"    " "    "*"    " "    "*"    " "
## 8  ( 1 )  "*" "*"    " "    "*"    "*"    "*"    "*"    "*"    " "
## 9  ( 1 )  "*" "*"    " "    "*"    "*"    "*"    "*"    "*"    "*"
## 10  ( 1 ) "*" "*"    "*"    "*"    "*"    "*"    "*"    "*"    "*"
##           I(X^10)
## 1  ( 1 )  " "
## 2  ( 1 )  " "
## 3  ( 1 )  " "
## 4  ( 1 )  " "
## 5  ( 1 )  " "
## 6  ( 1 )  " "
## 7  ( 1 )  "*"
## 8  ( 1 )  "*"
## 9  ( 1 )  "*"
## 10  ( 1 ) "*"
```

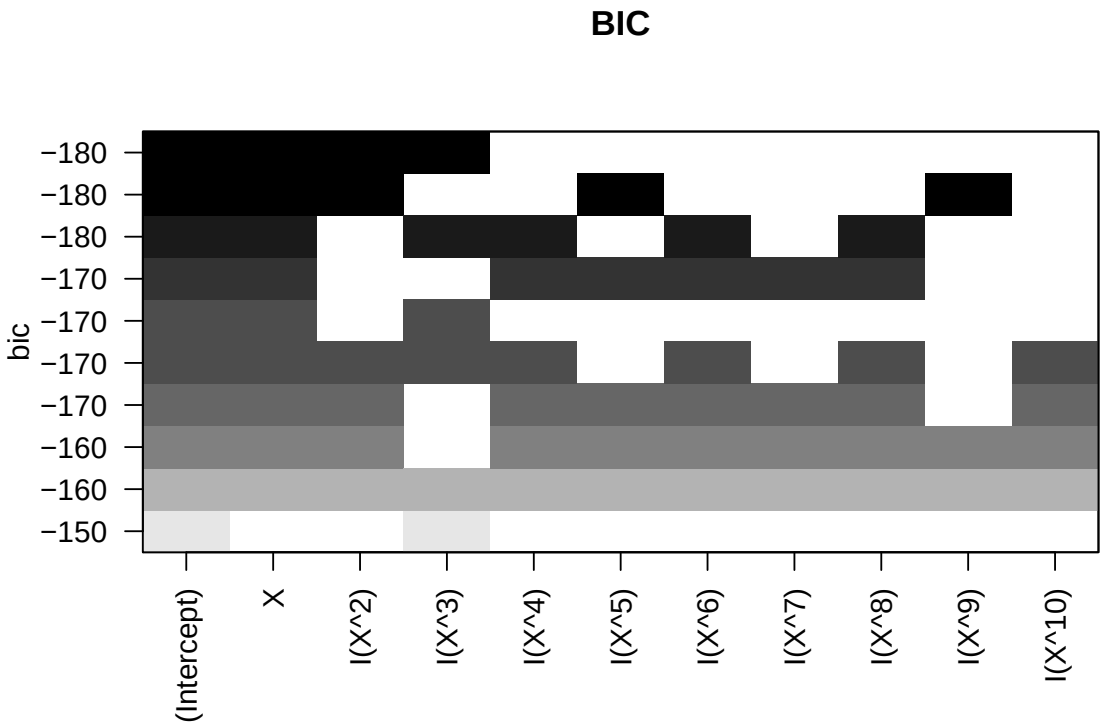
We notice the best subset algorithm has correctly included the proper terms in the third sets of predictors.



We need to remember that with R^2 statistic the values closer to 1 are a better fit. Among those with a value of 0.86 we see that the model with $Intercept, X, X^2, X^3$ is selected.



The C_p statistic indicates that the best model contains $Intercept, X, X^2, X^3$

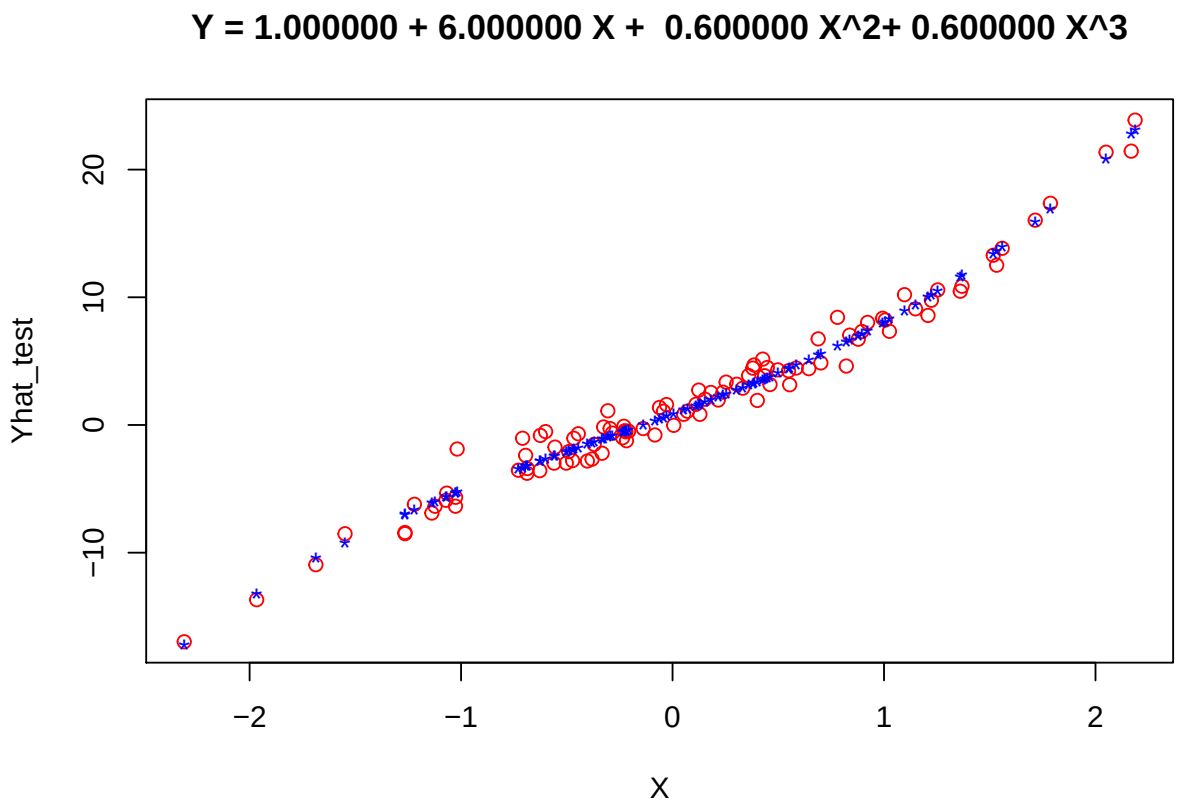


The BIC statistic indicates that the best model contains $Intercept, X, X^2, X^3$

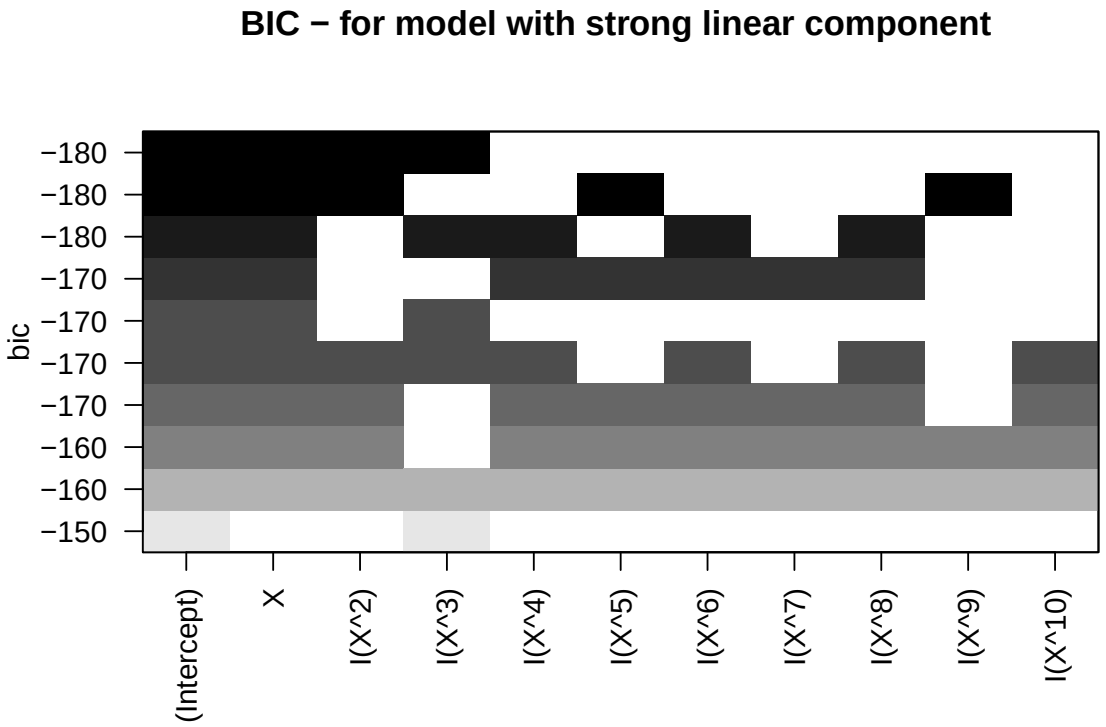
There was a problem with knitr where the cache was corrupted and the plots from old models were included. The section below is retained for that purpose.

```
beta_test <- matrix(NA, nrow = 4, ncol = 1)
beta_test[1] = 1
beta_test[2] = 6
beta_test[3] = 0.6
beta_test[4] = 0.6

Y_test <- matrix(NA, nrow = 100, ncol = 1)
Yhat_test <- matrix(NA, nrow = 100, ncol = 1)

for (i in 1:100) {
    Y_test[i] = beta_test[1] + beta_test[2] * X[i] + beta_test[3] * X[i]^2 +
        beta_test[4] * X[i]^3

    Yhat_test[i] = beta_test[1] + beta_test[2] * X[i] + beta_test[3] * X[i]^2 +
        beta_test[4] * X[i]^3 + epsilon[i]
}
DF_test <- as.data.frame(X)
DF_test <- cbind(DF, Yhat_test)
names(DF) = c("X", "Y")

plot(X, Yhat_test, col = "red")
```

```
points(X, Y_test, pch = "*", col = "blue")
title(main = sprintf("Y = %f + %f X +  %f X^2+ %f X^3", beta_test[1], beta_test[2],
    +beta_test[3], beta_test[4]), cex = 4.6)
```



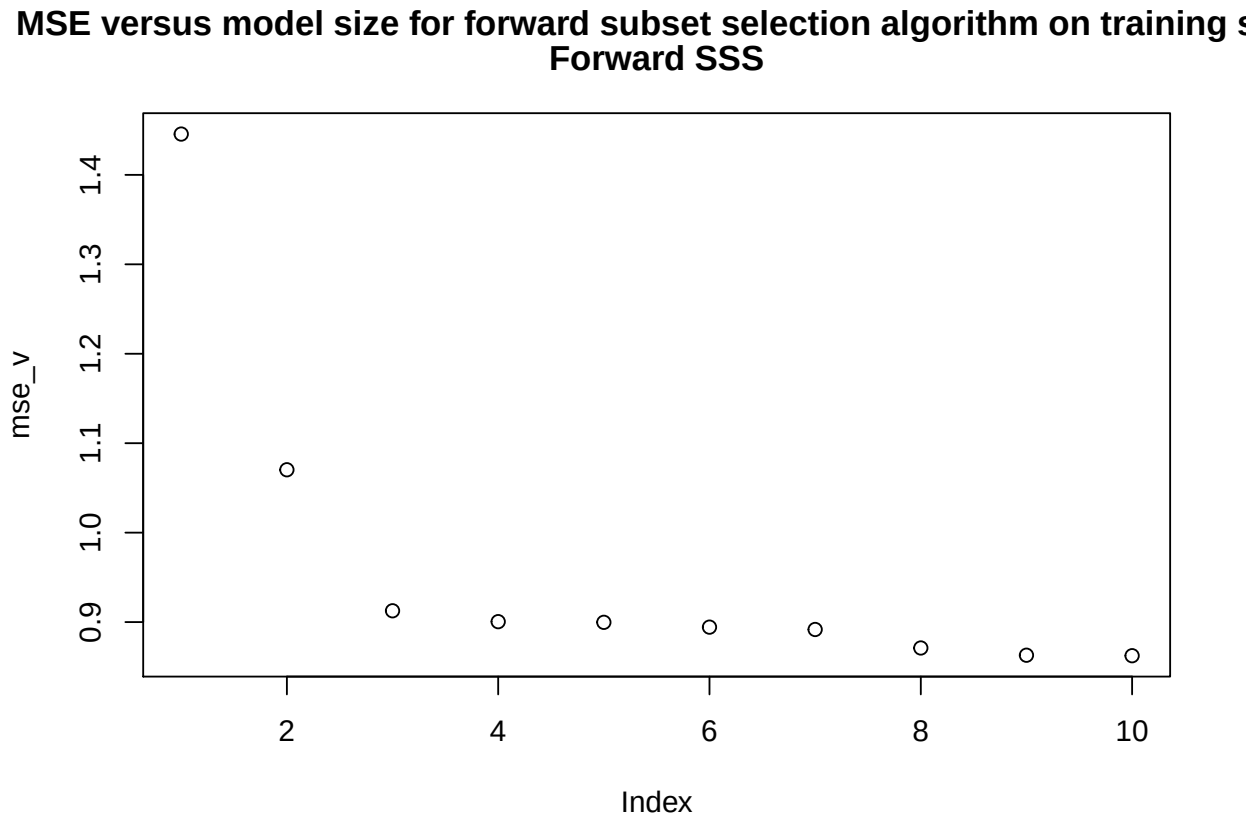

```
regfit.full <- regsubsets(Y ~ X + I(X^2) + I(X^3) + I(X^4) + I(X^5) + I(X^6) +
    I(X^7) + I(X^8) + I(X^9) + I(X^10), data = DF_test, nvmax = 10)
plot(regfit.full, scale = "bic")
title("BIC - for model with strong linear component ")
```



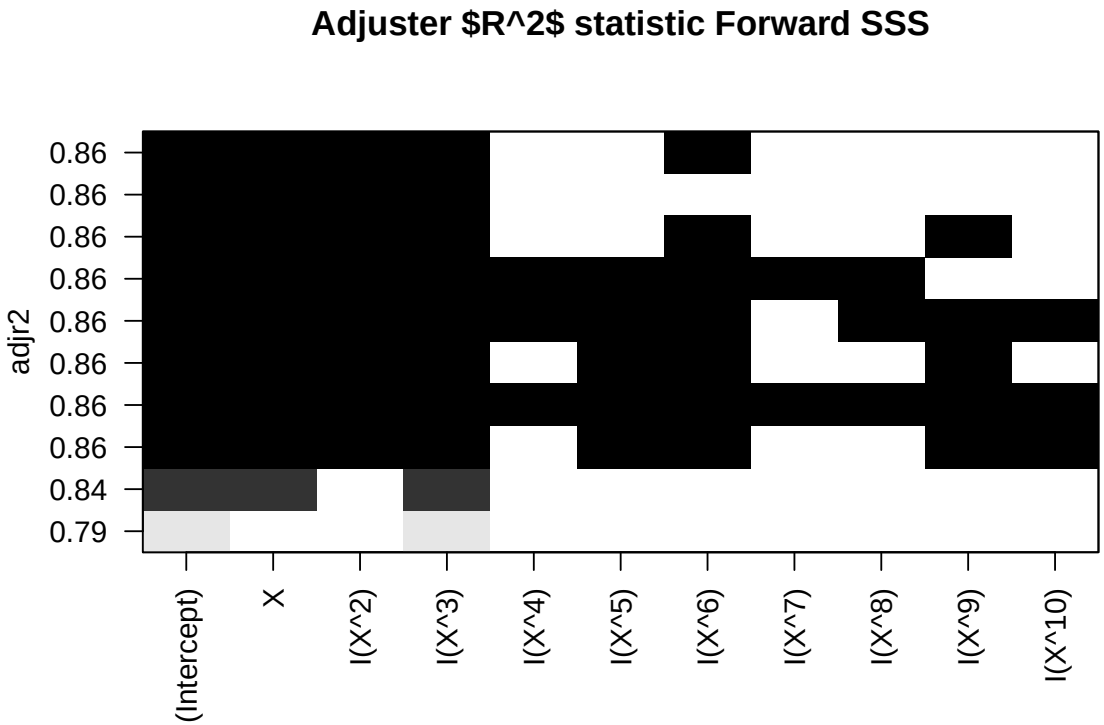
###d) Repeat (c), using forward stepwise selection and also using backwards stepwise selection. How does your answer compare to the results in (c)?

FORWARD SSS

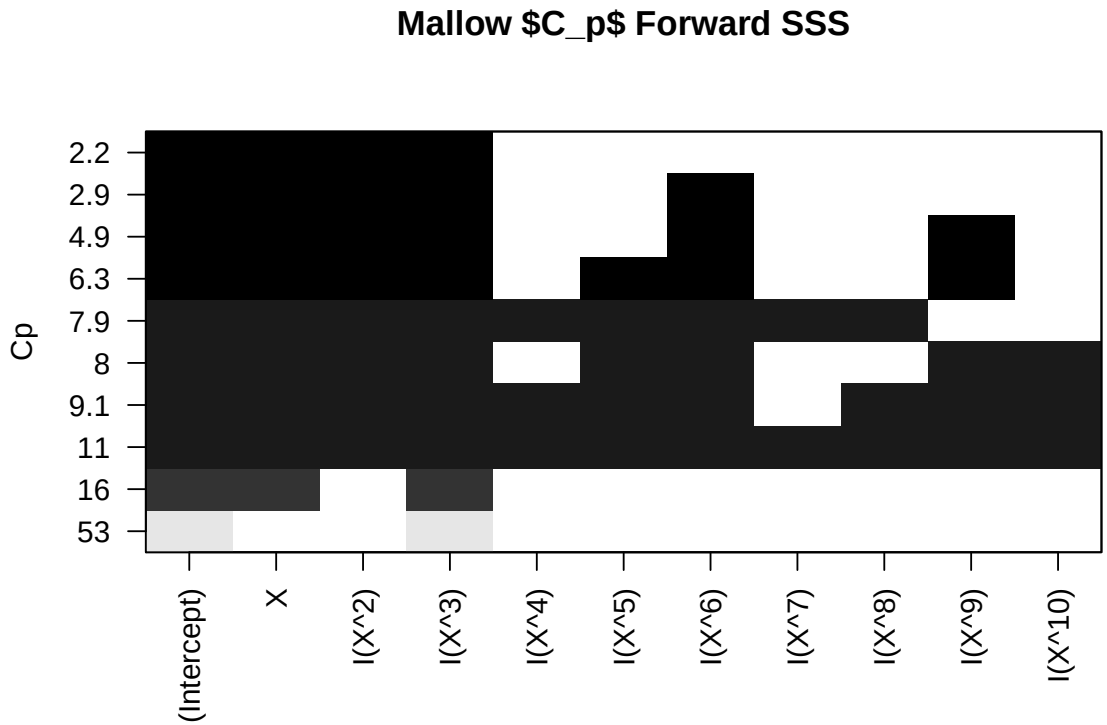
```
regfit.full <- regsubsets(Y ~ X + I(X^2) + I(X^3) + I(X^4) + I(X^5) + I(X^6) +
    I(X^7) + I(X^8) + I(X^9) + I(X^10), data = DF, nvmax = 10, method = "forward")

reg.summary <- summary(regfit.full)

mse_v <- reg.summary$rss/nrow(DF)

plot(mse_v)
title(c("MSE versus model size for forward subset selection algorithm on training set.",
    "Forward SSS"))
```



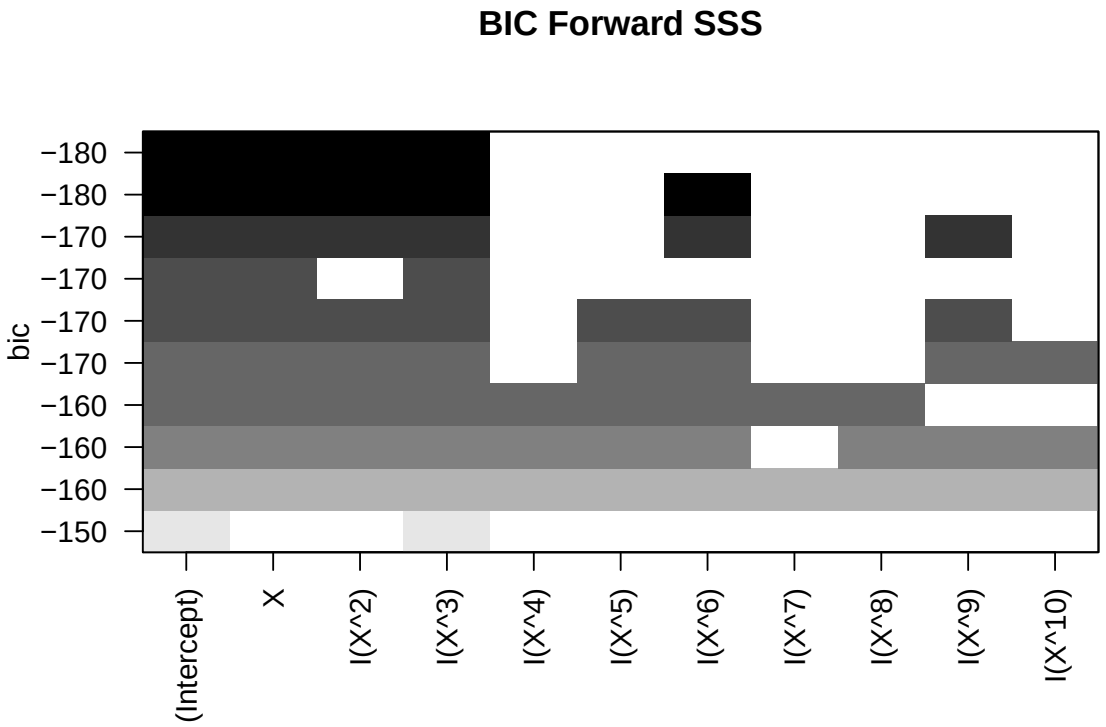
We see that there is a sharp drop in the training set MSE until 3 or 4 predictors are included and there is a steady decrease as additional polynomial terms are included. This is attributed to over fitting to the training data.



Among those with a value of 0.86 we see that the model with $Intercept, X, X^2, X^3$ is selected.



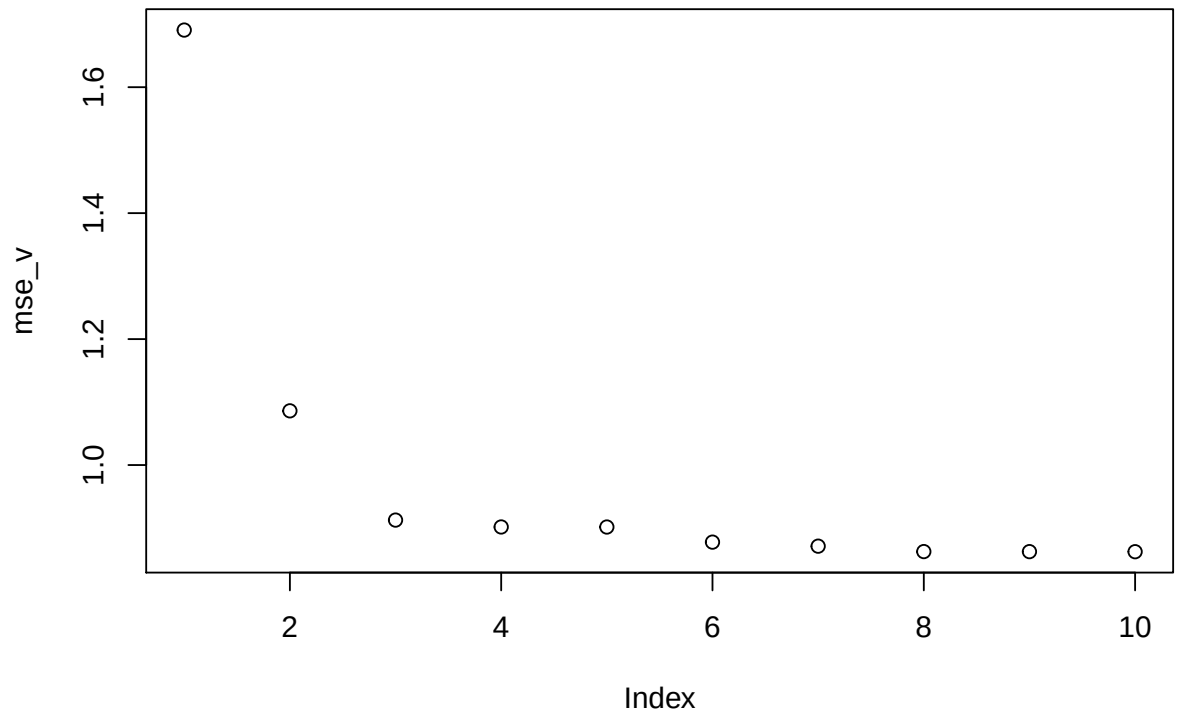
The C_p statistic indicates that the best model contains $Intercept, X, X^2, X^3$



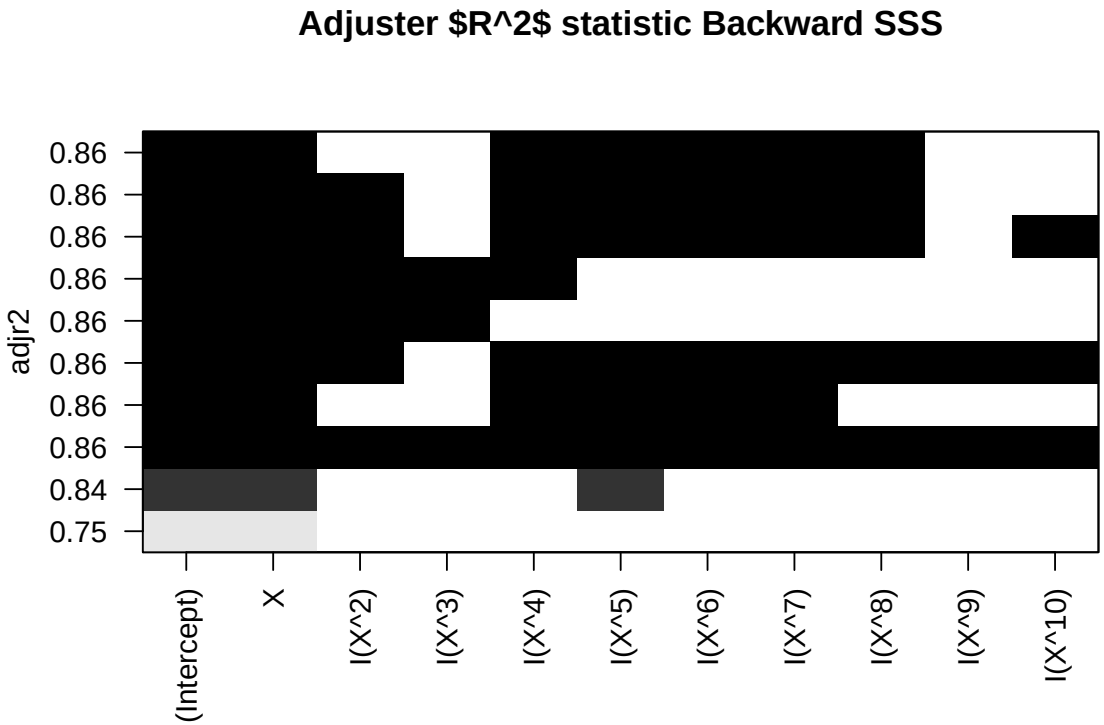
The BIC statistic indicates that the best model contains $Intercept, X, X^2, X^3$

BACKWARD SSS

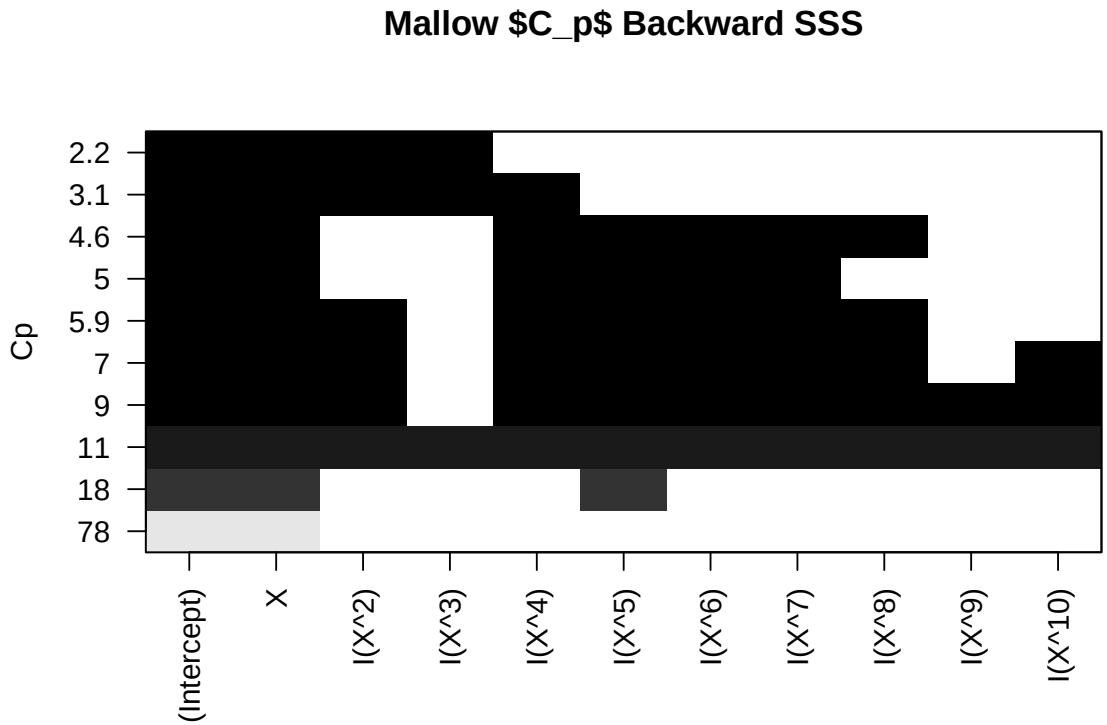
```
regfit.full <- regsubsets(Y ~ X + I(X^2) + I(X^3) + I(X^4) + I(X^5) + I(X^6) +
    I(X^7) + I(X^8) + I(X^9) + I(X^10), data = DF, nvmax = 10, method = "backward")

reg.summary <- summary(regfit.full)

mse_v <- reg.summary$rss/nrow(DF)

plot(mse_v)
title(c("MSE versus model size for backward subset selection algorithm on training set.",
    "Backward SSS"))
```



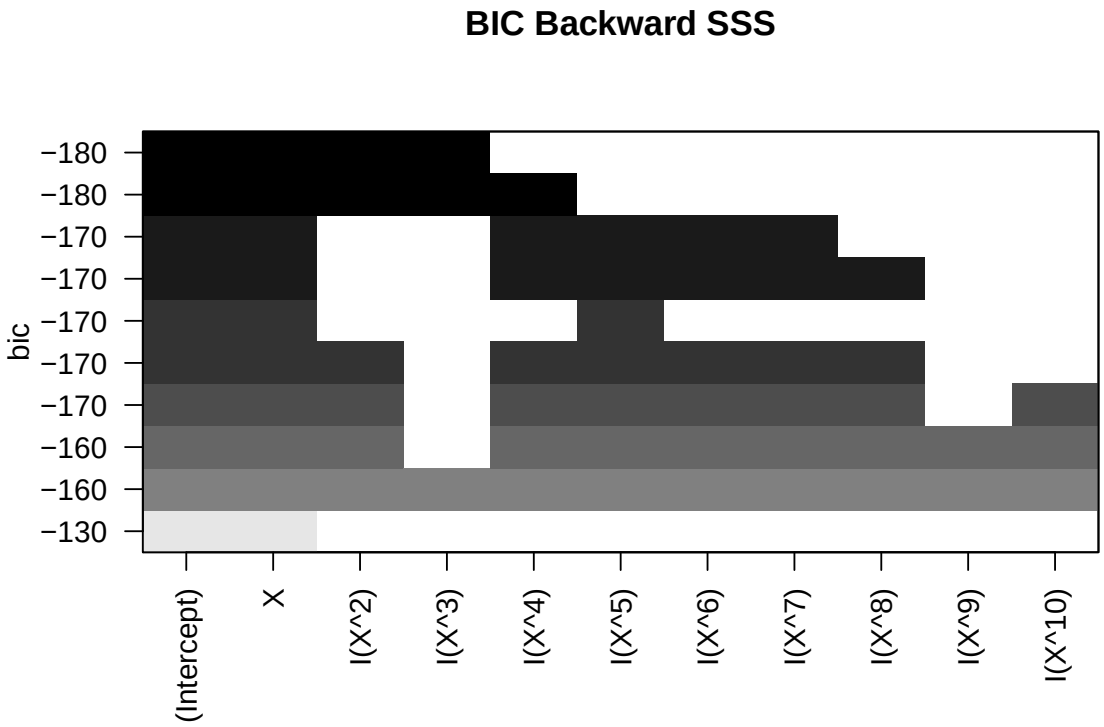
We see that there is a sharp drop in the training set MSE until 2 predictors are included and there is a steady decrease as additional polynomial terms are included. This is attributed to over fitting to the training data.



We need to remember that with R^2 statistic the values closer to 1 are a better fit. Among those with a value of 0.86 we see the full model $(Intercept), X, X^2, X^3$ has been selected.



The C_p statistic indicates that the best model contains $(Intercept), X, X^2, X^3$

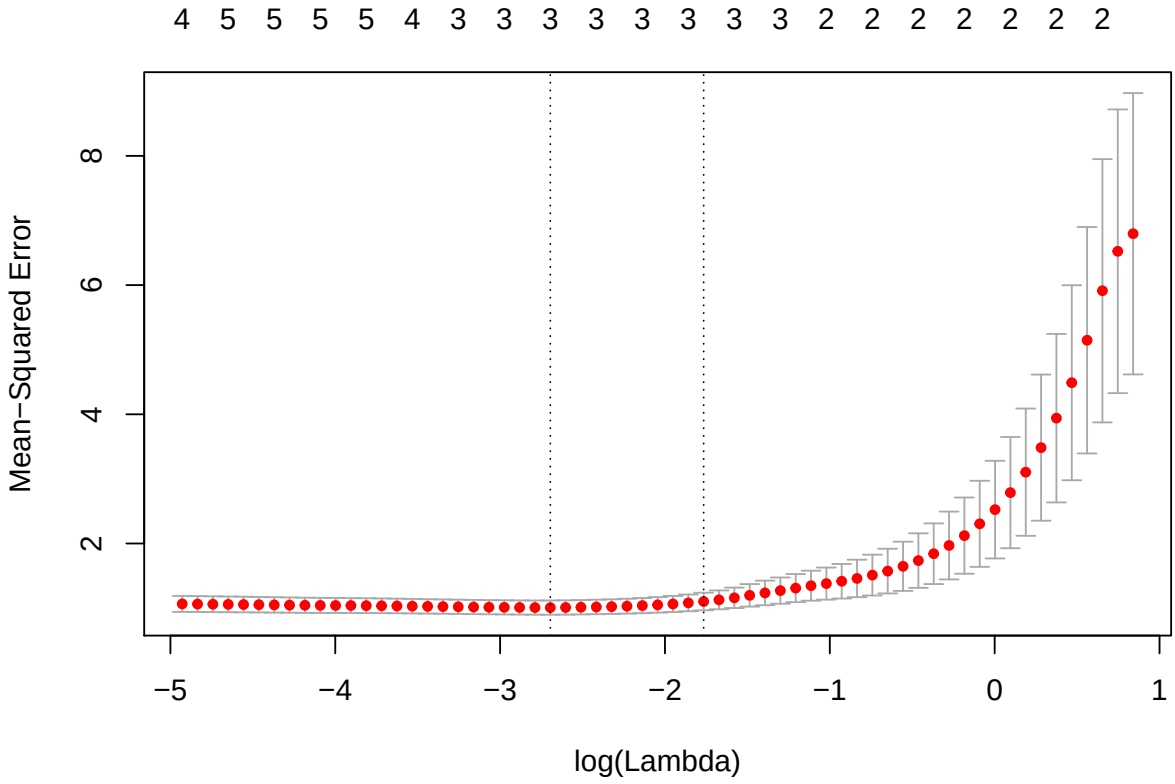


The BIC statistic indicates that the best model contains $(Intercept), X, X^2, X^3$

e)

Now fit a lasso model to the simulated data, again using $X, X^2, ..., X^10$ as predictors. Use cross-validation to select the optimal value of λ. Create plots of the cross-validation error as a function of λ. Report the resulting coefficient estimates, and discuss the results obtained.

```
library(glmnet)
x_lasso = model.matrix(Y ~ X + I(X^2) + I(X^3) + I(X^4) + I(X^5) + I(X^6) +
    I(X^7) + I(X^8) + I(X^9) + I(X^10), DF)[, -1]
y_lasso = DF$Y
cv.out = cv.glmnet(x_lasso, y_lasso, alpha = 1)
plot(cv.out)
```




```
bestlam = cv.out$lambda.min
bestlam
```

```
## [1] 0.06750938
```

```
best_lasso = glmnet(x_lasso, y_lasso, alpha = 1, lambda = bestlam)
predict(best_lasso, type = "coefficients", s = bestlam)[1:10, ]
```

```
## (Intercept)           X      I(X^2)      I(X^3)      I(X^4)      I(X^5)
##   2.1157969   1.1876459  -0.2813453   0.5440925   0.0000000   0.0000000
##      I(X^6)      I(X^7)      I(X^8)      I(X^9)
##   0.0000000   0.0000000   0.0000000   0.0000000
```

We see that the lasso for a value of lambda given by cross validation has driven all the extra model coedfficeints to 0 as expected.

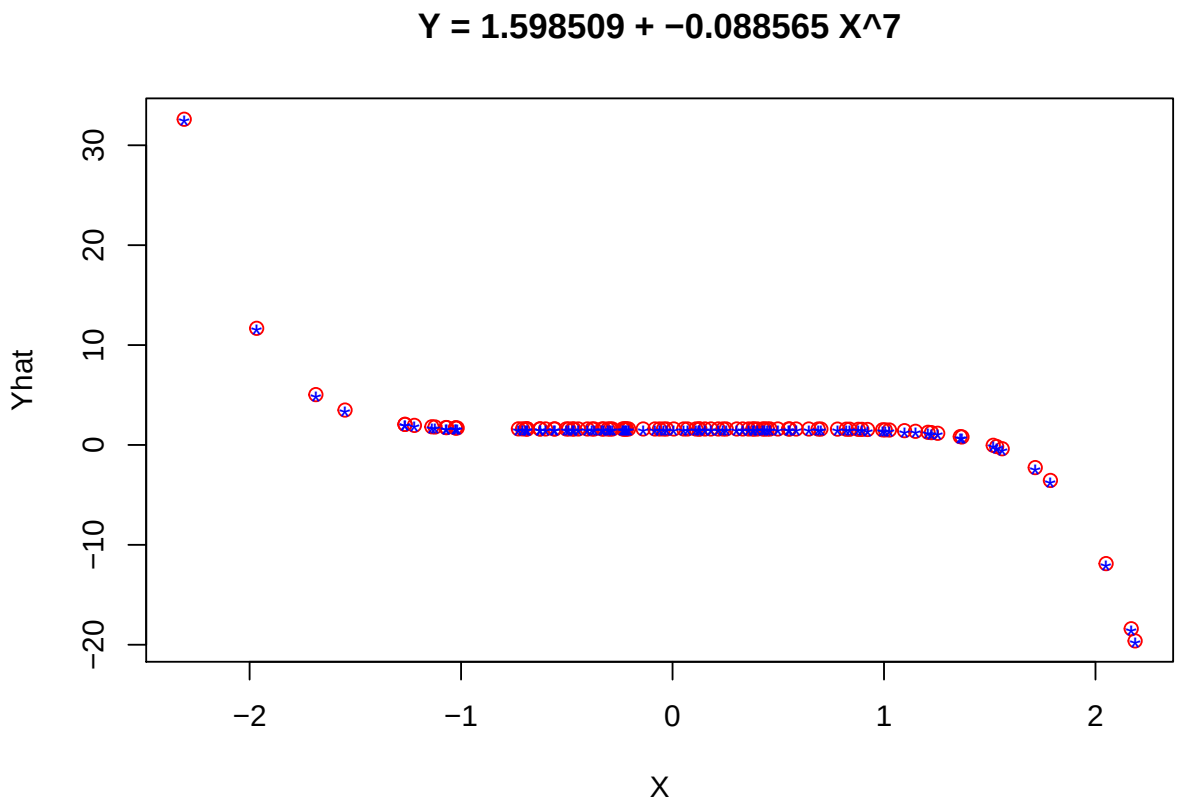
f)

Now generate a response vector Y according to the model

$$Y = \beta_0 + \beta_7 X_7 + \epsilon$$

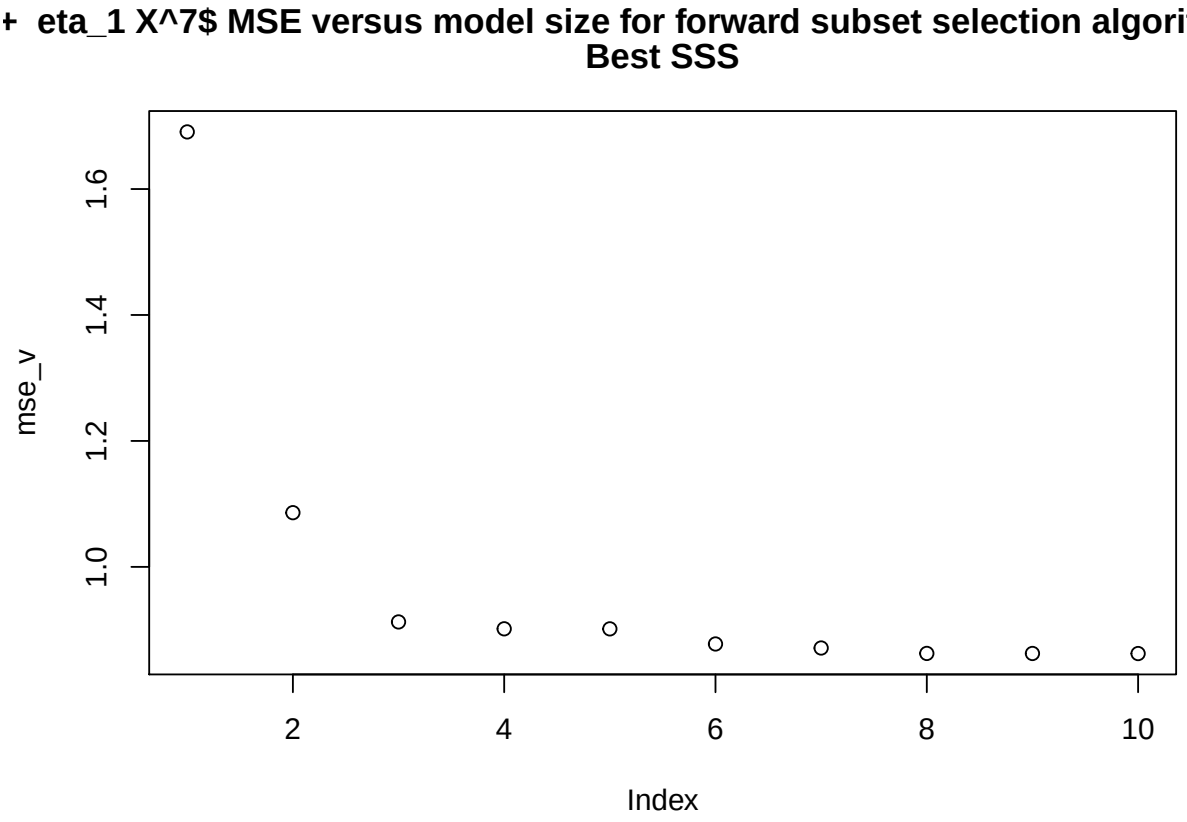
, and perform best subset selection and the lasso. Discuss the results obtained.

```r
beta = rnorm(2, mean = 0, sd = 1)
Y <- matrix(NA, nrow = 100, ncol = 1)
Yhat <- matrix(NA, nrow = 100, ncol = 1)

for (i in 1:100) {
    Y[i] = beta[1] + beta[2] * X[i]^7

    Yhat[i] = beta[1] + beta[2] * X[i]^7
}

plot(X, Yhat, col = "red")
points(X, Y, pch = "*", col = "blue")
title(main = sprintf("Y = %f + %f X^7", beta[1], beta[2]), cex = 4.6)
```

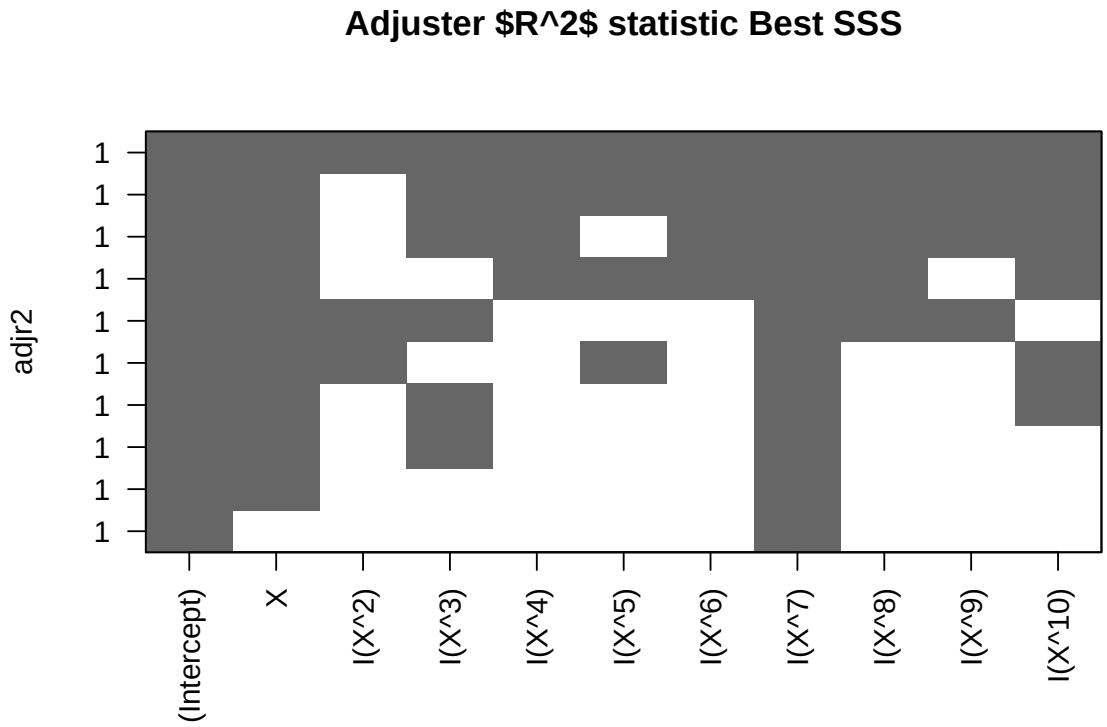



```r
DF <- as.data.frame(X)
DF <- cbind(DF, Yhat)
names(DF) = c("X", "Y")
library(leaps)

regfit.full <- regsubsets(Y ~ X + I(X^2) + I(X^3) + I(X^4) + I(X^5) + I(X^6) +
    I(X^7) + I(X^8) + I(X^9) + I(X^10), data = DF, nvmax = 10)

reg.summary <- summary(regfit.full)
```

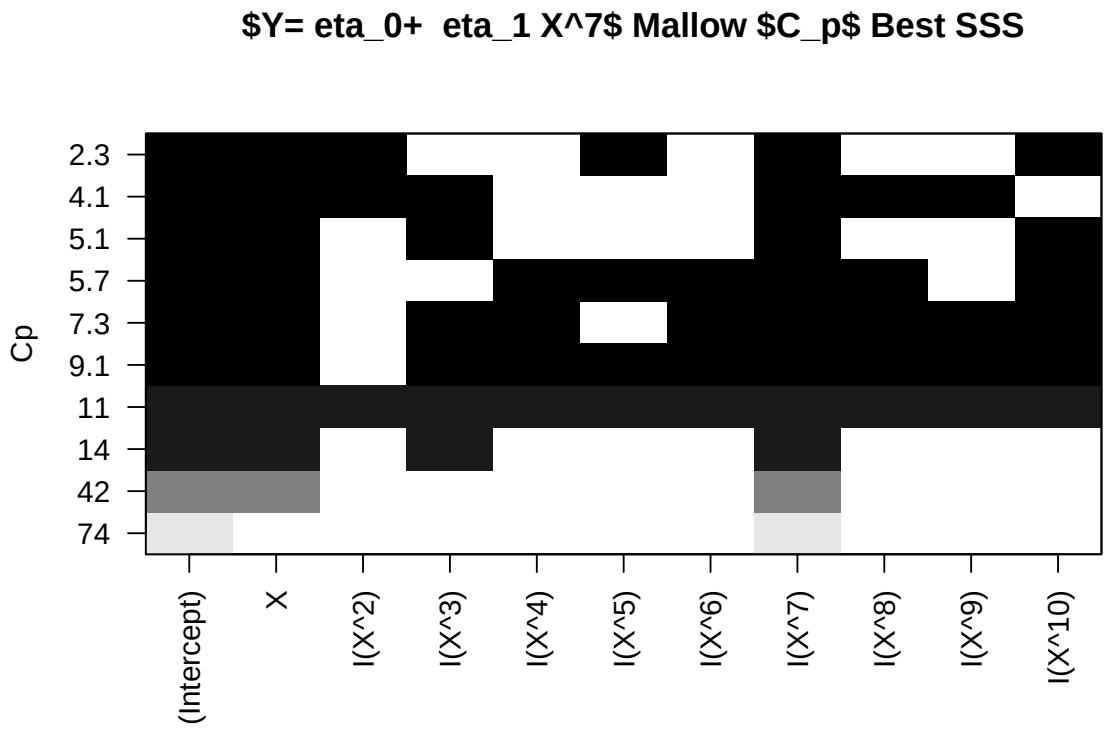
```
plot(mse_v)
title(c("$Y=\beta_0+ \beta_1 X^7$ MSE versus model size for forward subset selection algorithm on traini
    "Best SSS"))
```



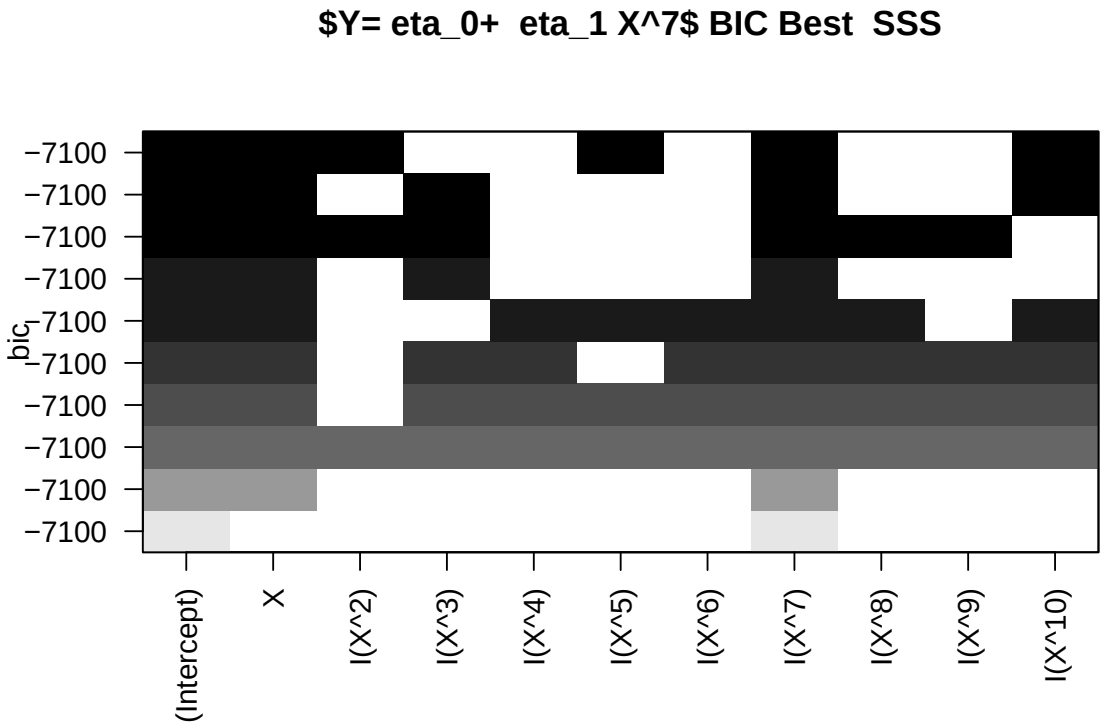

```
plot(regfit.full, scale = "adjr2")
title("Adjuster $R^2$ statistic Best SSS")
```



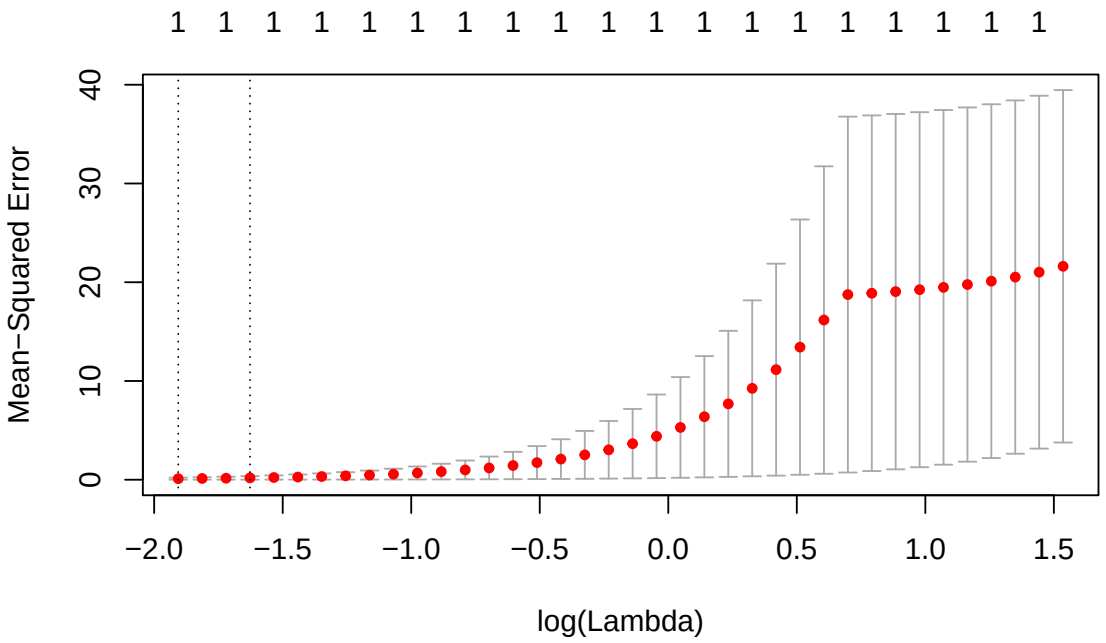

```
plot(regfit.full, scale = "Cp")
title("$Y=\beta_0+ \beta_1 X^7$ Mallow $C_p$ Best SSS")
```




```
plot(regfit.full, scale = "bic")
title("$Y=\beta_0+ \beta_1 X^7$ BIC Best  SSS")
```




```
x_lasso = model.matrix(Y ~ X + I(X^2) + I(X^3) + I(X^4) + I(X^5) + I(X^6) +
    I(X^7) + I(X^8) + I(X^9) + I(X^10), DF)[, -1]
y_lasso = DF$Y
cv.out = cv.glmnet(x_lasso, y_lasso, alpha = 1)
plot(cv.out)
```




```
bestlam = cv.out$lambda.min
bestlam
```

```
## [1] 0.1486221
```

```
best_lasso = glmnet(x_lasso, y_lasso, alpha = 1, lambda = bestlam)
predict(best_lasso, type = "coefficients", s = bestlam)[1:10, ]
```

```
##   (Intercept)             X        I(X^2)        I(X^3)        I(X^4)
##  1.590678e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
##        I(X^5)        I(X^6)        I(X^7)        I(X^8)        I(X^9)
##  0.000000e+00  0.000000e+00 -8.557615e-02  0.000000e+00 -2.777786e-05
```

All subset selection methods select the 7th term, but none have just that term and the intercept. The adjusted RSS statistic is confused, we suspect because the coefficient for X^7 (randomly generated) was small. The other statistics for Best subset models include additional terms. The lasso with a regularization parameter chose by cross validation comes very close to correctly selecting the model $Y = \beta_0 + \beta_1 X^7$. There is a X^9 term with a very small coefficient.

We note that this model may be difficult to fit since the range of the predictor is close to [-1,1] where a high order term like x^7 is relatively constant.

Chapter 6

Problem 9

In this exercise, we will predict the number of applications received using the other variables in the College data set.

a)

Split the data set into a training set and a test set.

```
rm(list = ls())
library(ISLR)
DF = College
train = sample(nrow(DF), floor(nrow(DF) * 2/3))
DFTrain <- DF[train, ]
DFTest <- DF[-train, ]
```

b)

Fit a linear model using least squares on the training set, and report the test error obtained.

```
names(DF)
```

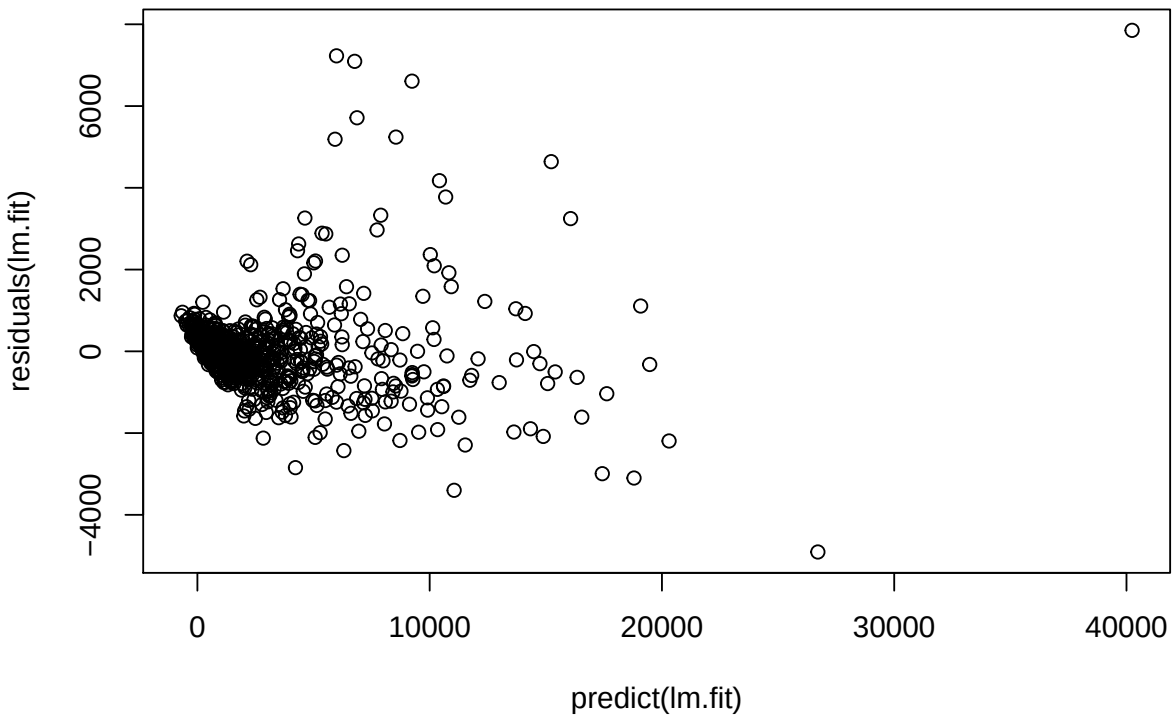
```
##  [1] "Private"     "Apps"        "Accept"      "Enroll"      "Top10perc"
##  [6] "Top25perc"   "F.Undergrad" "P.Undergrad" "Outstate"    "Room.Board"
## [11] "Books"       "Personal"    "PhD"         "Terminal"    "S.F.Ratio"
## [16] "perc.alumni" "Expend"      "Grad.Rate"
```

```
lm.fit <- lm(Apps ~ ., data = DF)
summary(lm.fit)
```

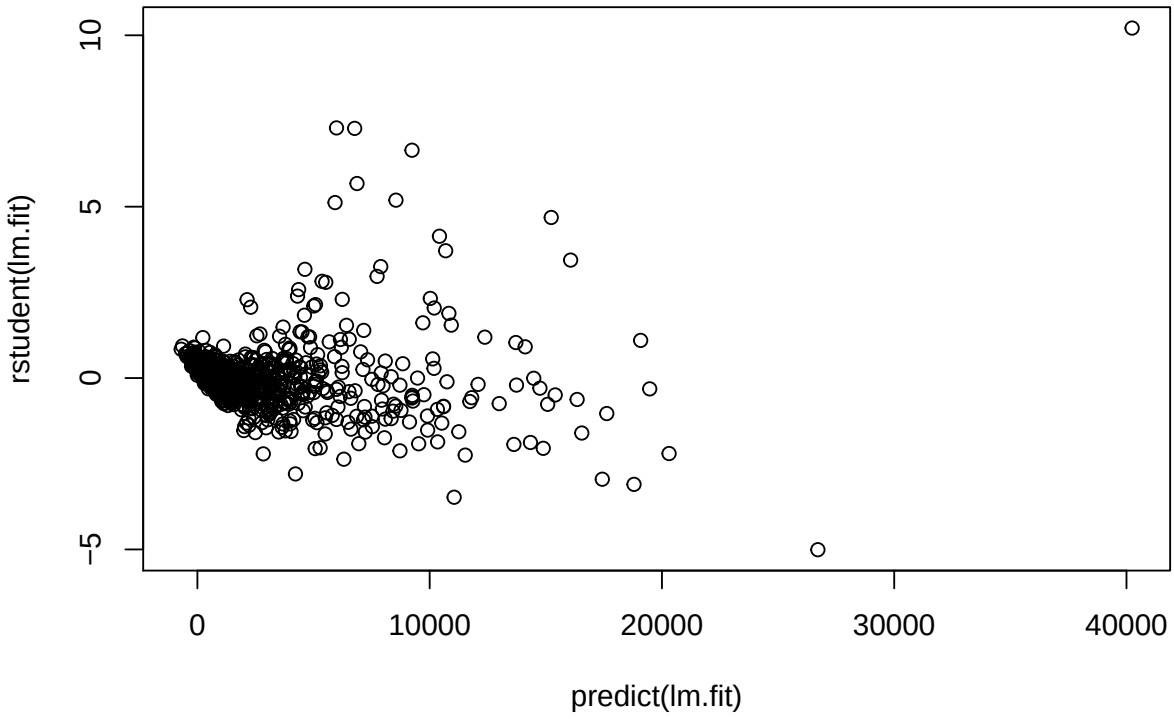
```
##
## Call:
## lm(formula = Apps ~ ., data = DF)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4908.8  -430.2   -29.5   322.3  7852.5
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -445.08413  408.32855  -1.090 0.276053
## PrivateYes  -494.14897  137.81191  -3.586 0.000358 ***
## Accept         1.58581    0.04074  38.924  < 2e-16 ***
## Enroll        -0.88069    0.18596  -4.736 2.60e-06 ***
## Top10perc     49.92628    5.57824   8.950  < 2e-16 ***
## Top25perc    -14.23448    4.47914  -3.178 0.001543 **
## F.Undergrad    0.05739    0.03271   1.754 0.079785 .
## P.Undergrad    0.04445    0.03214   1.383 0.167114
```

```
## Outstate      -0.08587    0.01906  -4.506 7.64e-06 ***
## Room.Board     0.15103    0.04829   3.127 0.001832 **
## Books          0.02090    0.23841   0.088 0.930175
## Personal       0.03110    0.06308   0.493 0.622060
## PhD           -8.67850    4.63814  -1.871 0.061714 .
## Terminal      -3.33066    5.09494  -0.654 0.513492
## S.F.Ratio     15.38961   13.00622   1.183 0.237081
## perc.alumni    0.17867    4.10230   0.044 0.965273
## Expend         0.07790    0.01235   6.308 4.79e-10 ***
## Grad.Rate      8.66763    2.94893   2.939 0.003390 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1041 on 759 degrees of freedom
## Multiple R-squared:  0.9292, Adjusted R-squared:  0.9276
## F-statistic: 585.9 on 17 and 759 DF,  p-value: < 2.2e-16
```

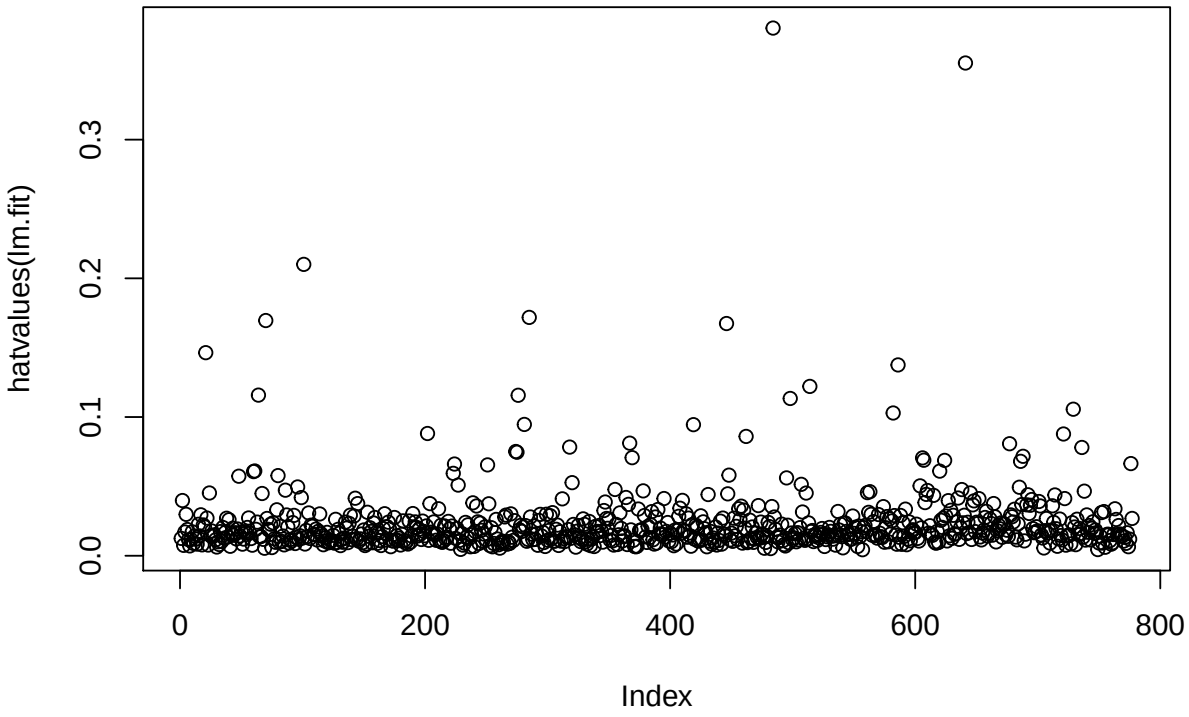
```
plot(predict(lm.fit), residuals(lm.fit))
```



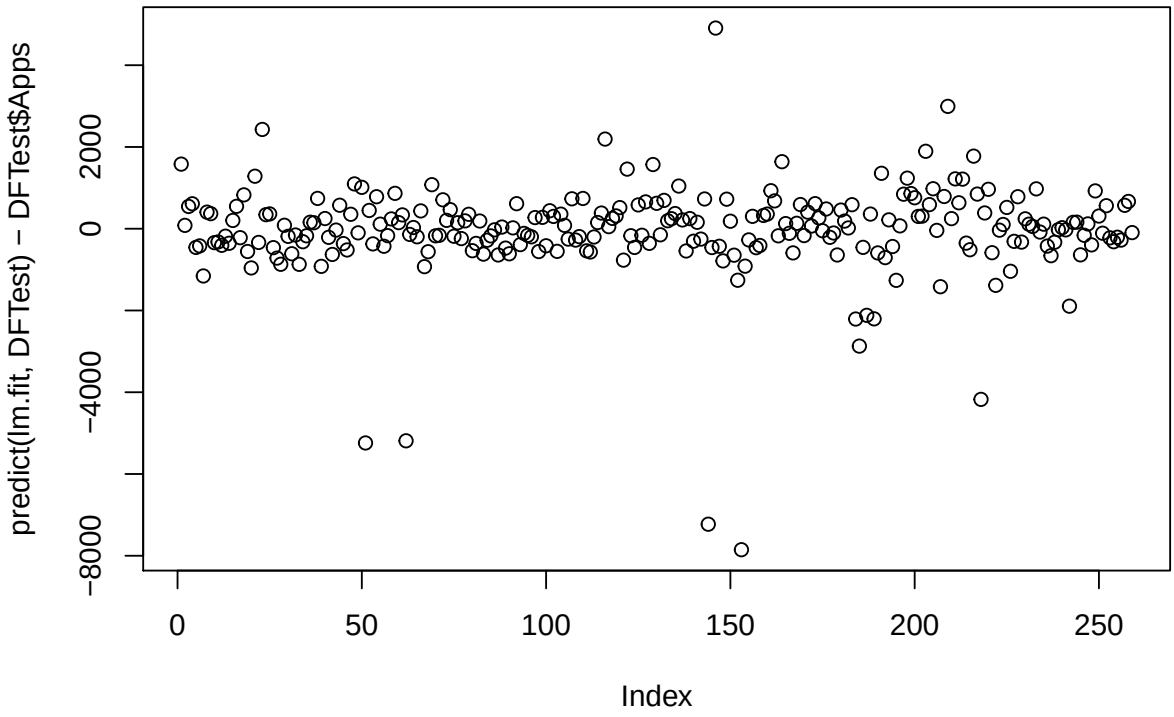

```
plot(predict(lm.fit), rstudent(lm.fit))
```




```
plot(hatvalues(lm.fit))
```




```
plot(predict(lm.fit, DFTest) - DFTest$Apps)
```




```
lm.test_mse <- mean((predict(lm.fit, DFTest) - DFTest$Apps)^2)

mse_summary <- data.frame(method = "lm", MSE = lm.test_mse)
```

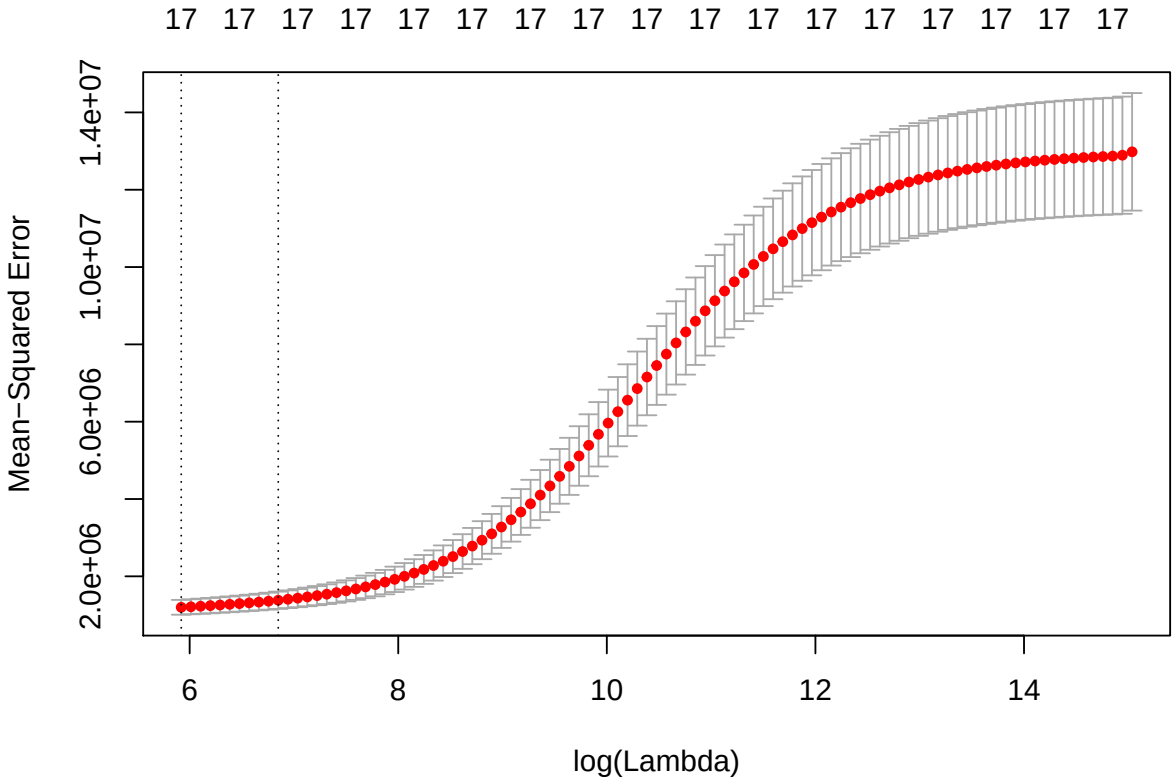
The test set for a linear model is

MSE = `1.3182593\times 10^{6}`

c)

Fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test error obtained.

```
library(glmnet)
x_ridge = model.matrix(Apps ~ ., DFTrain)[, -1]
y_ridge = DFTrain$Apps
cv.out = cv.glmnet(x_ridge, y_ridge, alpha = 0)
plot(cv.out)
```




```
bestlam = cv.out$lambda.min
bestlam
```

```
## [1] 372.1049
```

```
best_ridge = glmnet(x_ridge, y_ridge, alpha = 0, lambda = bestlam)
predict(best_ridge, type = "coefficients", s = bestlam)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                           1
## (Intercept) -9.734382e+02
## PrivateYes  -6.662649e+02
## Accept       7.942530e-01
## Enroll       6.827494e-01
## Top10perc    2.535093e+01
## Top25perc    2.782845e+00
## F.Undergrad  1.051378e-01
## P.Undergrad  1.988279e-02
## Outstate    -1.789992e-03
## Room.Board   2.277420e-01
## Books       -2.638279e-01
## Personal    -6.871798e-02
## PhD         -4.790712e+00
## Terminal    -2.667357e+00
```

```
## S.F.Ratio   -5.078747e+00
## perc.alumni -1.692403e+01
## Expend       6.582733e-02
## Grad.Rate    1.068775e+01
```

```
x_ridge_test = model.matrix(Apps ~ ., DFTest)[, -1]
y_ridge_test = DFTest$Apps

ridge.pred = predict(best_ridge, newx = x_ridge_test)
ridge.test_mse <- mean((ridge.pred - y_ridge_test)^2)



mse_summary <- rbind(mse_summary, data.frame(method = "ridge", MSE = ridge.test_mse))
```

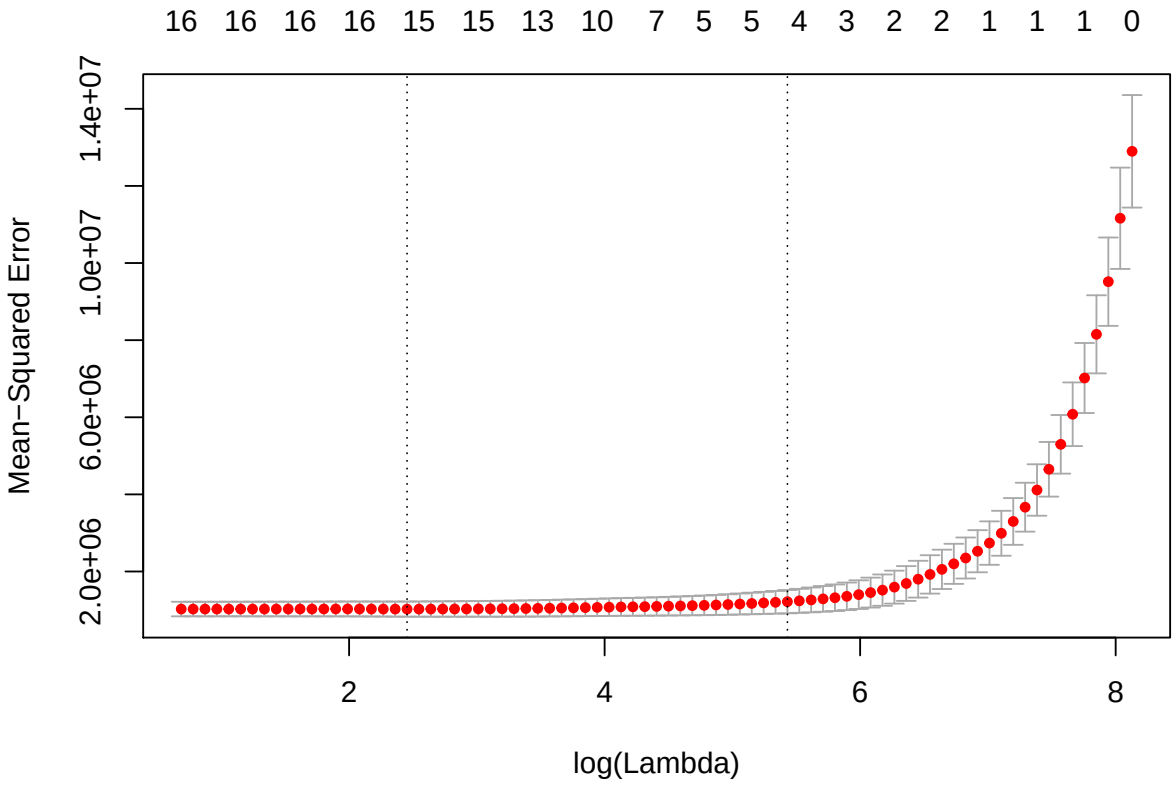
The test set MSE for a ridge regression model where the regularization parameter is set by cross validation is

MSE = `2.7335804\times 10^{6}`

d) Fit a lasso model on the training set, with λ chosen by cross-validation.

Report the test error obtained, along with the number of non-zero coefficient estimates.

```
x_lasso = model.matrix(Apps ~ ., DFTrain)[, -1]
y_lasso = DFTrain$Apps
cv.out = cv.glmnet(x_lasso, y_lasso, alpha = 1)
plot(cv.out)
```




```
bestlam = cv.out$lambda.min
bestlam
```

```
## [1] 11.63094
```

```
best_lasso = glmnet(x_lasso, y_lasso, alpha = 1, lambda = bestlam)
predict(best_lasso, type = "coefficients", s = bestlam)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept) -403.31800045
## PrivateYes  -528.07297696
## Accept         1.28999500
## Enroll         .
## Top10perc     38.34236306
## Top25perc     -5.35638436
## F.Undergrad    0.02985515
## P.Undergrad    0.03030955
## Outstate      -0.04317690
## Room.Board     0.16154560
## Books         -0.25910845
## Personal      -0.02724122
## PhD           -7.80564839
## Terminal      -1.33956095
```

```
## S.F.Ratio      .
## perc.alumni   -8.12716882
## Expend         0.06406385
## Grad.Rate      6.80804787
```

```
x_lasso_test = model.matrix(Apps ~ ., DFTest)[, -1]
y_lasso_test = DFTest$Apps

lasso.pred = predict(best_lasso, newx = x_lasso_test)
lasso.test_mse <- mean((lasso.pred - y_lasso_test)^2)

coeff_lasso <- predict(best_lasso, type = "coefficients", s = bestlam)[1:18,
    ]
library(pander)
pander(coeff_lasso)
```

Table 1: Table continues below

(Intercept)	PrivateYes	Accept	Enroll	Top10perc	Top25perc
-403.3	-528.1	1.29	0	38.34	-5.356

Table 2: Table continues below

F.Undergrad	P.Undergrad	Outstate	Room.Board	Books	Personal	PhD
0.02986	0.03031	-0.04318	0.1615	-0.2591	-0.02724	-7.806

Terminal	S.F.Ratio	perc.alumni	Expend	Grad.Rate
-1.34	0	-8.127	0.06406	6.808

```
mse_summary <- rbind(mse_summary, data.frame(method = "lasso", MSE = lasso.test_mse))
```

The test set MSE for a lasso regression model where the regularization parameter is set by cross validation is

MSE $=$ `1.6356686\times 10^{6}`

All but one of the predictors was included in the lasso model with the best lambda selected by cross validation. A more parsimonious model may help with inference so using the cross validation MSE chart we below we bump up lambda to $e^4.5$ to get a model with fewer predictors

```
bestlam = exp(4.2)

best_lasso = glmnet(x_lasso, y_lasso, alpha = 1, lambda = bestlam)
predict(best_lasso, type = "coefficients", s = bestlam)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                          1
```

```
## (Intercept) -818.94759419
## PrivateYes  -382.26488676
## Accept         1.29478097
## Enroll         .
## Top10perc     27.12951298
## Top25perc      .
## F.Undergrad    0.02810284
## P.Undergrad    .
## Outstate       .
## Room.Board     0.05382040
## Books          .
## Personal       .
## PhD            .
## Terminal       .
## S.F.Ratio      .
## perc.alumni   -2.15889650
## Expend         0.04578136
## Grad.Rate      .
```

```
x_lasso_test = model.matrix(Apps ~ ., DFTest)[, -1]
y_lasso_test = DFTest$Apps

lasso.pred = predict(best_lasso, newx = x_lasso_test)
lasso.test_mse <- mean((lasso.pred - y_lasso_test)^2)

mse_summary <- rbind(mse_summary, data.frame(method = "lasso-reduced", MSE = lasso.test_mse))

coeff_lasso <- predict(best_lasso, type = "coefficients", s = bestlam)[1:18,
    ]
library(pander)
pander(coeff_lasso)
```

Table 4: Table continues below

(Intercept)	PrivateYes	Accept	Enroll	Top10perc	Top25perc
-818.9	-382.3	1.295	0	27.13	0

Table 5: Table continues below

F.Undergrad	P.Undergrad	Outstate	Room.Board	Books	Personal	PhD
0.0281	0	0	0.05382	0	0	0

Terminal	S.F.Ratio	perc.alumni	Expend	Grad.Rate
0	0	-2.159	0.04578	0

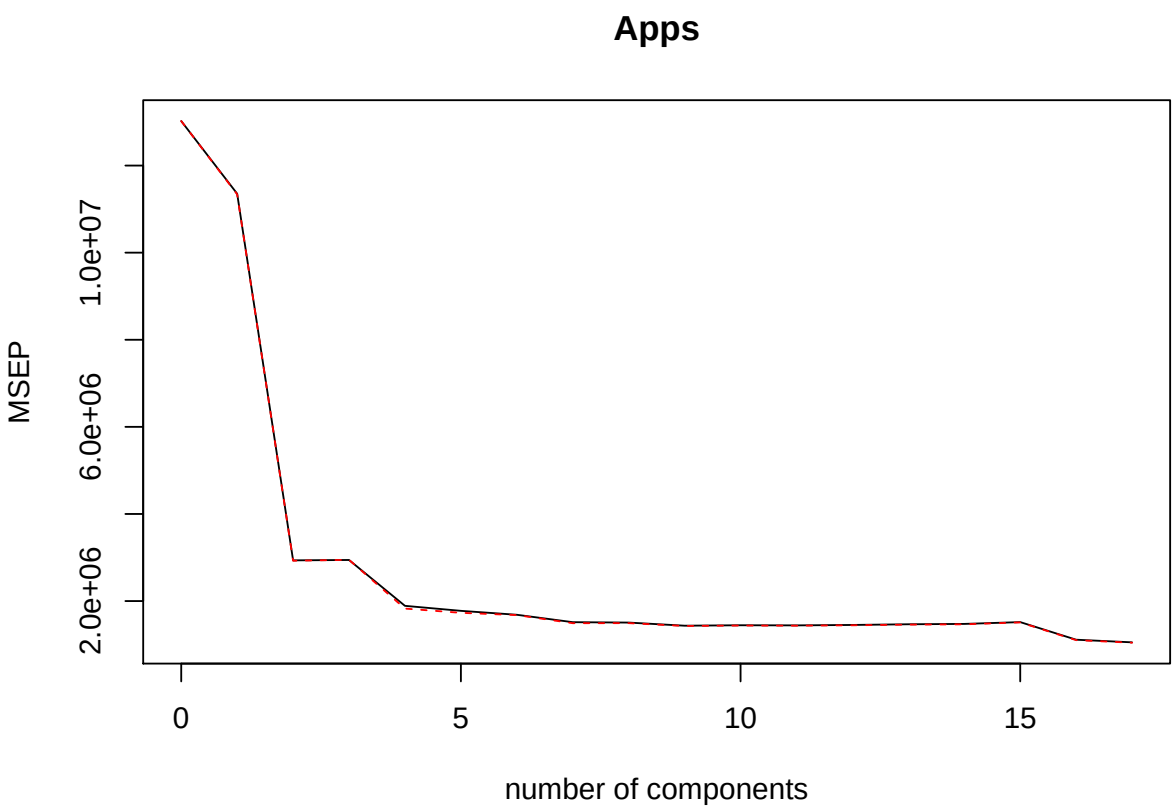
e)

Fit a PCR model on the training set, with M chosen by cross-validation. Report the test error obtained, along with the value of M selected by cross-validation.

```
pcr.test_mse = 1
library(pls)
pcr.fit = pcr(Apps ~ ., data = DFTrain, scale = TRUE, validation = "CV")
summary(pcr.fit)
```

```
## Data:    X dimension: 518 17
##  Y dimension: 518 1
## Fit method: svdpc
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            3609     3369     1713     1716     1375     1332     1298
## adjCV         3609     3367     1711     1717     1352     1315     1295
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1231     1227     1198      1202      1201      1205      1211
## adjCV     1221     1222     1195      1199      1198      1202      1208
##        14 comps  15 comps  16 comps  17 comps
## CV         1215      1232      1055      1026
## adjCV      1211      1229      1050      1022
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       31.40    56.67    63.90    69.77    75.26    80.36    84.02
## Apps    13.49    78.03    78.06    87.10    87.62    87.80    89.31
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X       87.54    90.66     93.14     95.28     97.05     98.13     98.93
## Apps    89.32    89.84     89.85     89.89     89.92     89.92     89.92
##       15 comps  16 comps  17 comps
## X        99.42     99.85    100.00
## Apps     90.12     92.67     93.11
```

```
validationplot(pcr.fit, val.type = "MSEP")
```




```
pcr.pred = predict(pcr.fit, DFTest, ncomp = 8)
pcr.test_mse <- mean((pcr.pred - DFTest$Apps)^2)

mse_summary <- rbind(mse_summary, data.frame(method = "pcr", MSE = pcr.test_mse))
```

The test set MSE for a principal components regression is

$$\mathbf{MSE} = 3.9222017\times 10^{6}$$

f)

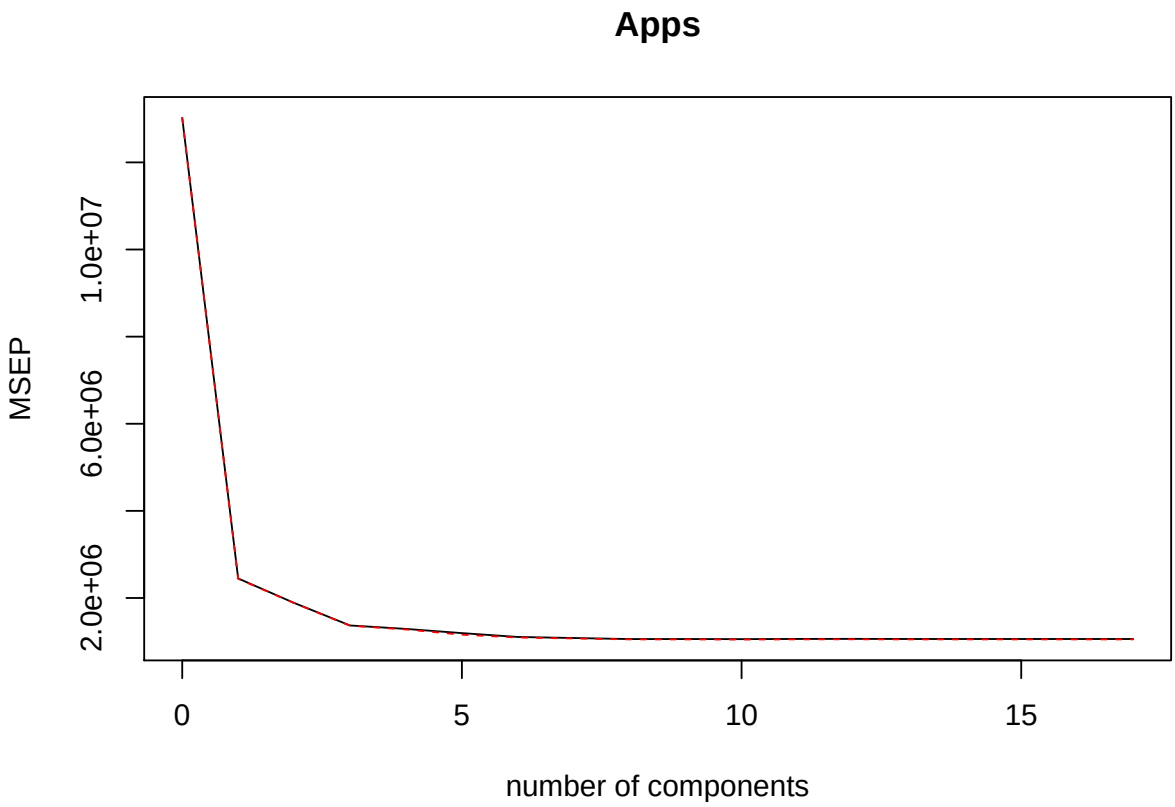
Fit a PLS model on the training set, with M chosen by cross-validation. Report the test error obtained, along with the value of M selected by cross-validation.

```
plsr.fit = plsr(Apps ~ ., data = DFTrain, scale = TRUE, validation = "CV")
summary(plsr.fit)
```

```
## Data:    X dimension: 518 17
##  Y dimension: 518 1
## Fit method: kernelpls
## Number of components considered: 17
##
## VALIDATION: RMSEP
```

```
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            3609     1565     1374     1170     1136     1093     1051
## adjCV         3609     1562     1372     1167     1130     1076     1045
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1040     1028     1029      1027      1030      1030      1029
## adjCV     1035     1025     1024      1023      1025      1026      1024
##        14 comps  15 comps  16 comps  17 comps
## CV         1029      1029      1029      1029
## adjCV      1025      1024      1025      1025
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       26.63    49.56    62.25    65.58    67.47    73.01    77.08
## Apps    81.92    86.57    90.32    91.31    92.69    92.91    92.97
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X       80.93    83.29     85.29     89.42     91.12     92.48     94.15
## Apps    93.00    93.04     93.09     93.10     93.11     93.11     93.11
##       15 comps  16 comps  17 comps
## X        96.52     98.96    100.00
## Apps     93.11     93.11     93.11
```

```
validationplot(plsr.fit, val.type = "MSEP")
```



```
plsr.pred = predict(plsr.fit, DFTest, ncomp = 8)
plsr.test_mse <- mean((plsr.pred - DFTest$Apps)^2)


mse_summary <- rbind(mse_summary, data.frame(method = "plsr", MSE = plsr.test_mse))
```

The test set MSE for a partial least quares regression is

MSE = `1.6091296\times 10^{6}`

g)

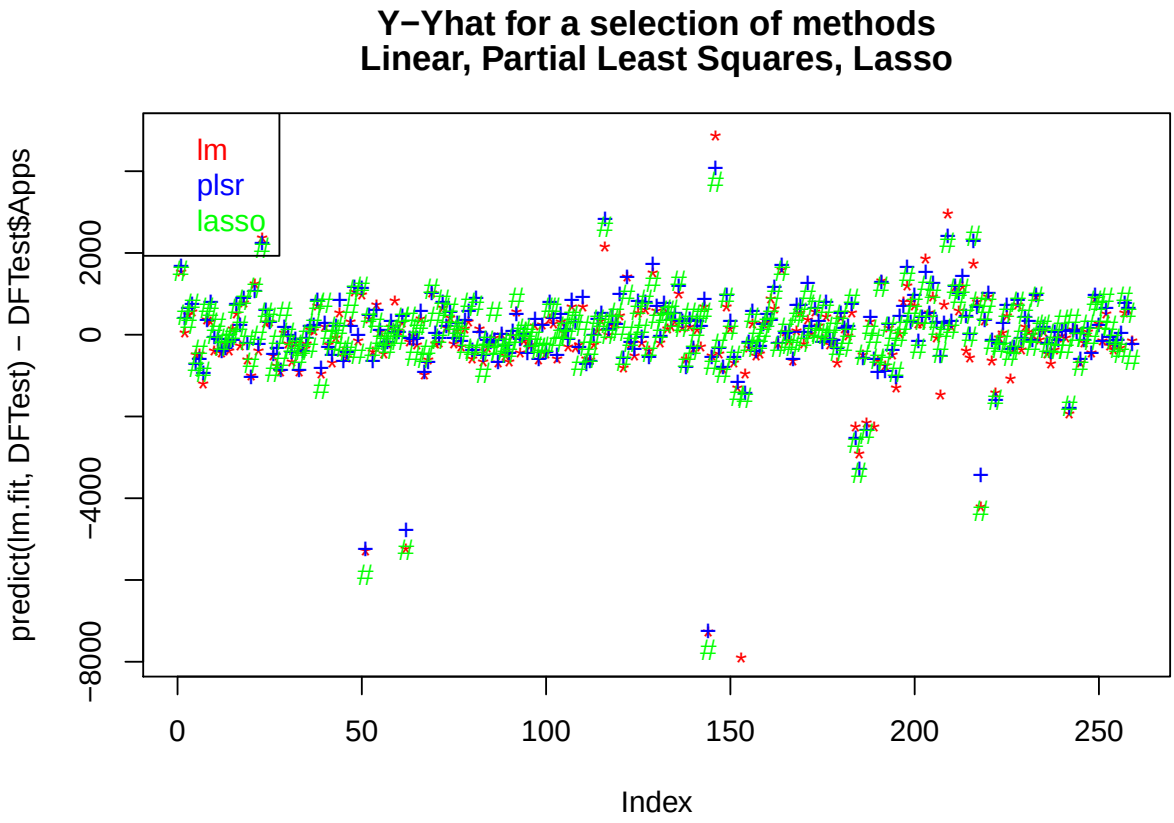
`Comment on the results obtained. How accurately can we predict the number of college applications received? Is there much difference among the test errors resulting from these five approaches?`

```
pander(mse_summary)
```

method	MSE
lm	1318259
ridge	2733580
lasso	1635669
lasso-reduced	1699330
pcr	3922202
plsr	1609130

We see the linear model is the best but that the lasso is competitive

```
plot(predict(lm.fit, DFTest) - DFTest$Apps, pch = "*", col = "red")
points(plsr.pred - DFTest$Apps, pch = "+", col = "blue")
points(lasso.pred - y_lasso_test, pch = "#", col = "green")
legend("topleft", title.col = "black", c("lm", "plsr", "lasso"), text.col = c("red",
    "blue", "green"), text.font = 1, cex = 1)
title(c("Y-Yhat for a selection of methods", "Linear, Partial Least Squares, Lasso"))
```



Chapter 7

Problem 6

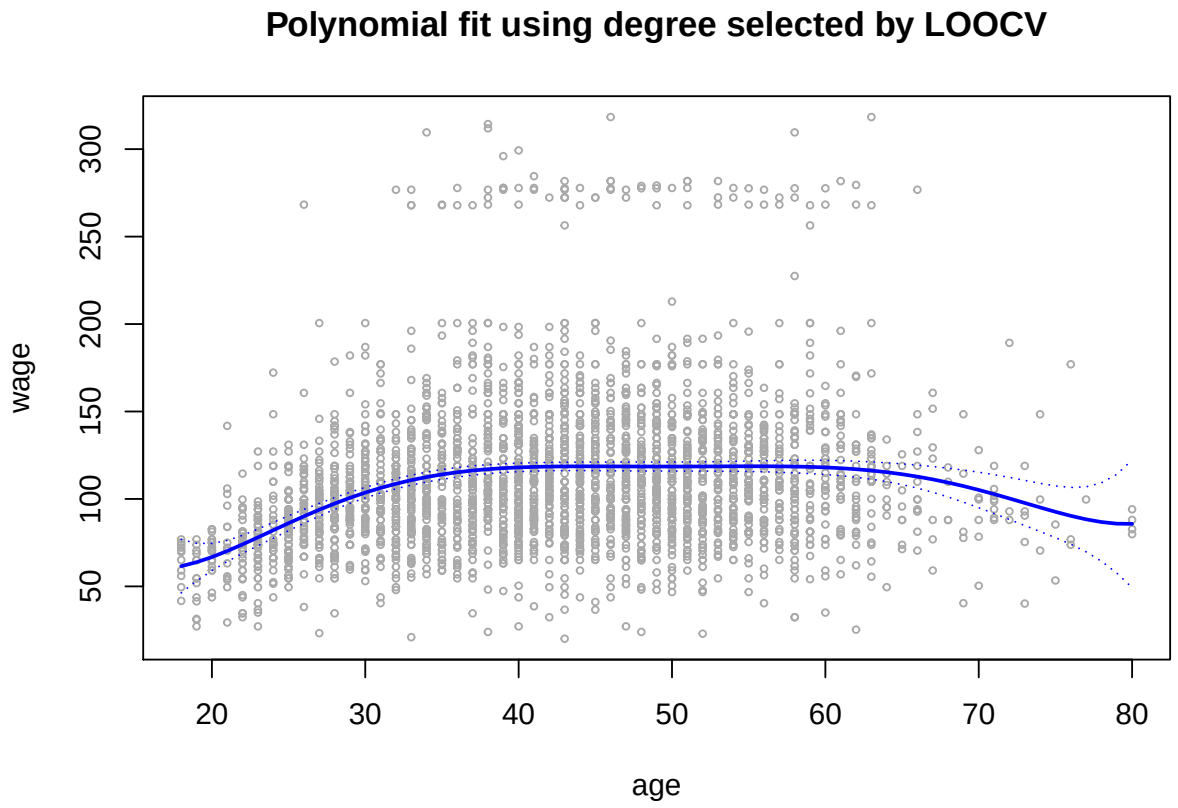
In this exercise, you will further analyze the Wage data set considered throughout this chapter.

a)

Perform polynomial regression to predict wage using age. Use cross-validation to select the optimal degree d for the polynomial. What degree was chosen, and how does this compare to the results of hypothesis testing using ANOVA? Make a plot of the resulting polynomial fit to the data.

```
rm(list = ls())
library(ISLR)
library(ggplot2)

attach(Wage)

max.poly <- 7

cvFunc <- function(age, wage, degree) {
    preds <- numeric(length(age))
```

```
    for (i in 1:length(age)) {
        # Here is where we eageclude i from the training set
        age.in <- age[-i]
        age.out <- age[i]

        # Single test point
        wage.in <- wage[-i]
        wage.out <- age[i]

        fit <- lm(wage.in ~ poly(age.in, degree = degree))
        preds[i] <- predict(fit, newdata = data.frame(age.in = age.out))
    }
    return(sum((wage - preds)^2))
}

cv.err <- data.frame(sq_err = numeric(max.poly))
for (i in 1:max.poly) {
    cv.err[i, 1] <- cvFunc(age, wage, degree = i)
}

best_degree <- which.min(cv.err$sq_err)

lmpoly.fit <- lm(wage ~ poly(age, degree = best_degree))

agelims = range(age)
age.grid = seq(from = agelims[1], to = agelims[2])

preds = predict(lmpoly.fit, newdata = data.frame(age = age.grid), se = TRUE)

se.bands = cbind(preds$fit + 2 * preds$se.fit, preds$fit - 2 * preds$se.fit)

plot(age, wage, xlim = agelims, cex = 0.5, col = "darkgrey ")

lines(age.grid, preds$fit, lwd = 2, col = " blue")
matlines(age.grid, se.bands, lwd = 1, col = " blue", lty = 3)
title("Polynomial fit using degree selected by LOOCV")
```




```
fit.1 = lm(wage ~ age, data = Wage)
fit.2 = lm(wage ~ poly(age, 2), data = Wage)
fit.3 = lm(wage ~ poly(age, 3), data = Wage)
fit.4 = lm(wage ~ poly(age, 4), data = Wage)
fit.5 = lm(wage ~ poly(age, 5), data = Wage)
fit.6 = lm(wage ~ poly(age, 6), data = Wage)
fit.7 = lm(wage ~ poly(age, 7), data = Wage)
anova(fit.1, fit.2, fit.3, fit.4, fit.5, fit.6, fit.7)
```

```
## Analysis of Variance Table
##
## Model 1: wage ~ age
## Model 2: wage ~ poly(age, 2)
## Model 3: wage ~ poly(age, 3)
## Model 4: wage ~ poly(age, 4)
## Model 5: wage ~ poly(age, 5)
## Model 6: wage ~ poly(age, 6)
## Model 7: wage ~ poly(age, 7)
##   Res.Df     RSS Df Sum of Sq        F    Pr(>F)
## 1   2998 5022216
## 2   2997 4793430  1    228786 143.6926 < 2.2e-16 ***
## 3   2996 4777674  1     15756   9.8956  0.001673 **
## 4   2995 4771604  1      6070   3.8125  0.050966 .
## 5   2994 4770322  1      1283   0.8055  0.369516
## 6   2993 4766389  1      3932   2.4697  0.116165
```

```
## 7   2992 4763834  1      2555   1.6049  0.205311
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

According to the ANOVA results, the fourth order polynomial is sufficient. Higher order terms are not justified by the F test comparing fit.5 against fit.4

Leave one out cross validation has selected 6 as the best order, but looking at the cross validation error we see that the error for the fourth order model is very close to the fifth and sixth order model. We could justifiably choose 4 from these results as the best model based on the desire to have a low error and a parsimonious model.

```
library(pander)
pander(cv.err)
```

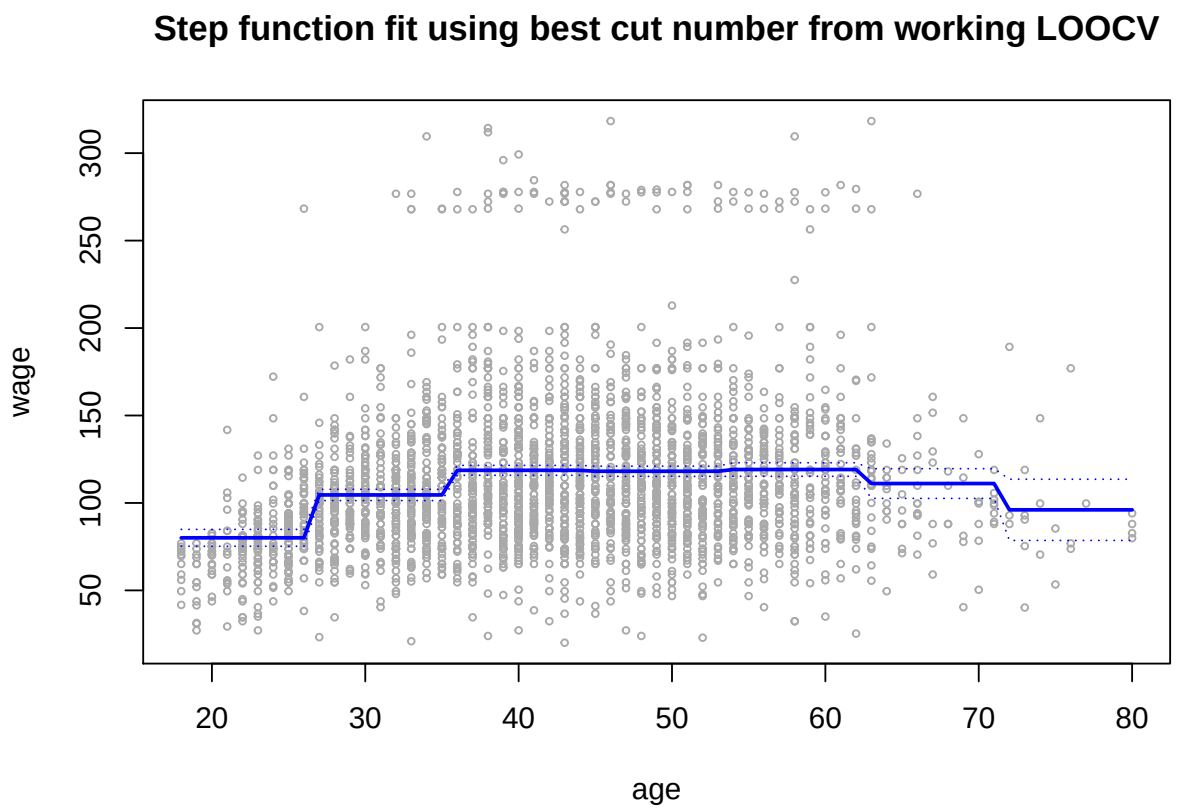
sq_err
5028705
4801588
4787879
4783788
4784636
4782357
4782436

b)

Fit a step function to predict wage using age, and perform cross-validation to choose the optimal number of cuts. Make a plot of the fit obtained.

```
max.cut <- 10
age = Wage$age
wage = Wage$wage

cvFunc <- function(age, wage, cut_val) {
    preds <- numeric(length(age))
    for (i in 1:length(age)) {
        # Here is where we eageclude i from the training set
        age.in <- age[-i]
        age.out <- age[i]

        # Single test point
        wage.in <- wage[-i]
        wage.out <- age[i]

        # fit <- lm(wage.in ~ cut(age.in, cut_val) )

        # See below - Error in model.frame.default(Terms, newdata, na.action =
        # na.action, xlev = object$xlevels) : factor cut(age.in, cut_val) has new
        # level preds[i]<- predict(fit, newdata=data.frame(age.in=age.out))
        # preds[i]<- predict(fit, newdata=data.frame(age.in=cut(age
        # ,best_cut)[age.out]))
```

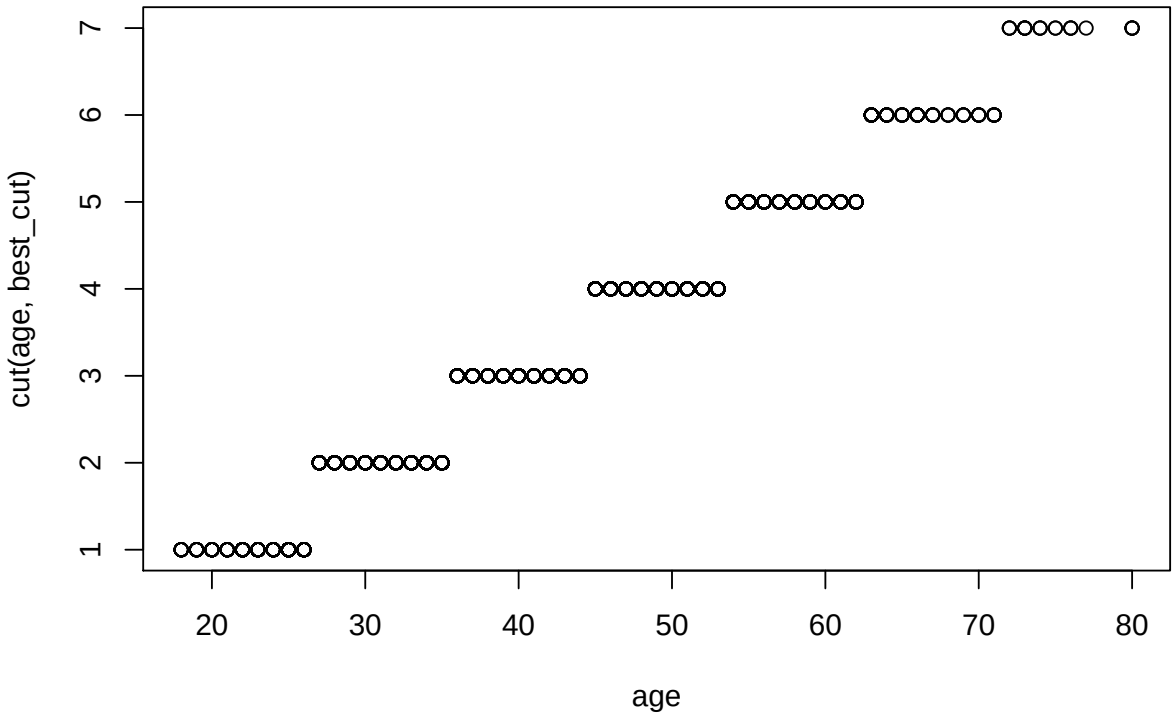
```
    }
    return(sum((wage - preds)^2))
}


cv.err <- data.frame(sq_err = numeric(max.cut))

for (i in 1:max.cut) {
    cv.err[i, 1] <- cvFunc(age, wage, cut_val = i)
}

best_cut <- 7  # LOOCV algorithm error which.min(cv.err$sq_err)

lmcut.fit <- lm(wage ~ cut(age, best_cut), Wage)


agelims = range(age)
age.grid = seq(from = agelims[1], to = agelims[2])

preds = predict(lmcut.fit, newdata = data.frame(age = age.grid), se = TRUE)

se.bands = cbind(preds$fit + 2 * preds$se.fit, preds$fit - 2 * preds$se.fit)

plot(age, wage, xlim = agelims, cex = 0.5, col = "darkgrey ")

lines(age.grid, preds$fit, lwd = 2, col = " blue")
matlines(age.grid, se.bands, lwd = 1, col = " blue", lty = 3)
title("Step function fit using best cut number from working LOOCV")
```



There was one successful run of the LOOCV for the step function regression - then I ran into the error captured in the code comments above. Much effort was expended to get it working again, but we ran out of time so the best_cut was hard coded to be the value I recall from the one run that worked.

Here's what we learned from debugging - the cut function creates a factor variable

```
plot(age, cut(age, best_cut))
```



The expression used in the plot `predict (lmcut.fit ,newdata =data.frame(age=age.grid),se=TRUE)` is able to convert the data frame to a factor and calculate the predicted values. For other cases like a single value in LOOCV or a split of the data as in a validation set we got the error that reads like there was a problem converting the data to a factor. I checked the type of the data in all cases and verified it was an int. I tried converting the input data to a factor (see code) but that did not work either.

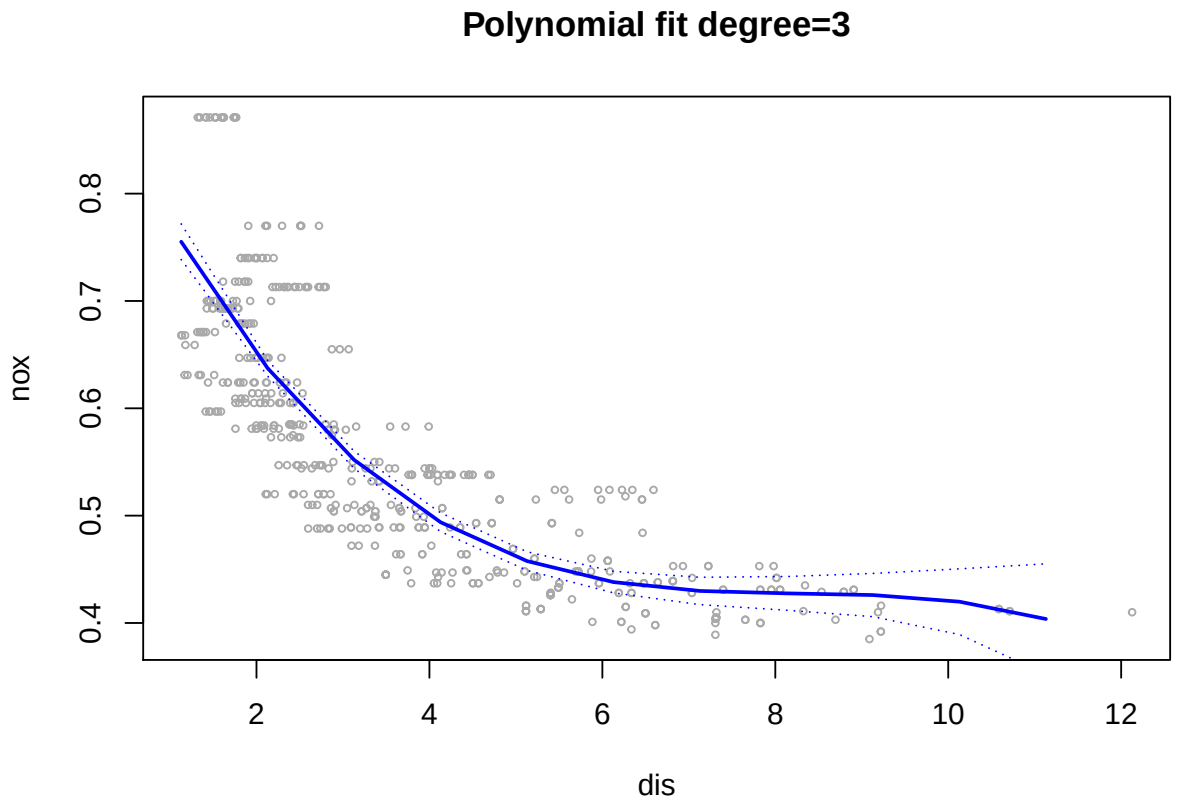
Chapter 7

Problem 9

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. 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. Report the regression output, and plot the resulting data and polynomial fits.

```
rm(list = ls())
library(MASS)
attach(Boston)
lmpoly.fit <- lm(nox ~ poly(dis, degree = 3))

dislims = range(dis)
dis.grid = seq(from = dislims[1], to = dislims[2])

preds = predict(lmpoly.fit, newdata = data.frame(dis = dis.grid), se = TRUE)
```

```
se.bands = cbind(preds$fit + 2 * preds$se.fit, preds$fit - 2 * preds$se.fit)

plot(dis, nox, xlim = dislims, cex = 0.5, col = "darkgrey ")

lines(dis.grid, preds$fit, lwd = 2, col = " blue")
matlines(dis.grid, se.bands, lwd = 1, col = " blue", lty = 3)
title("Polynomial fit degree=3")
```

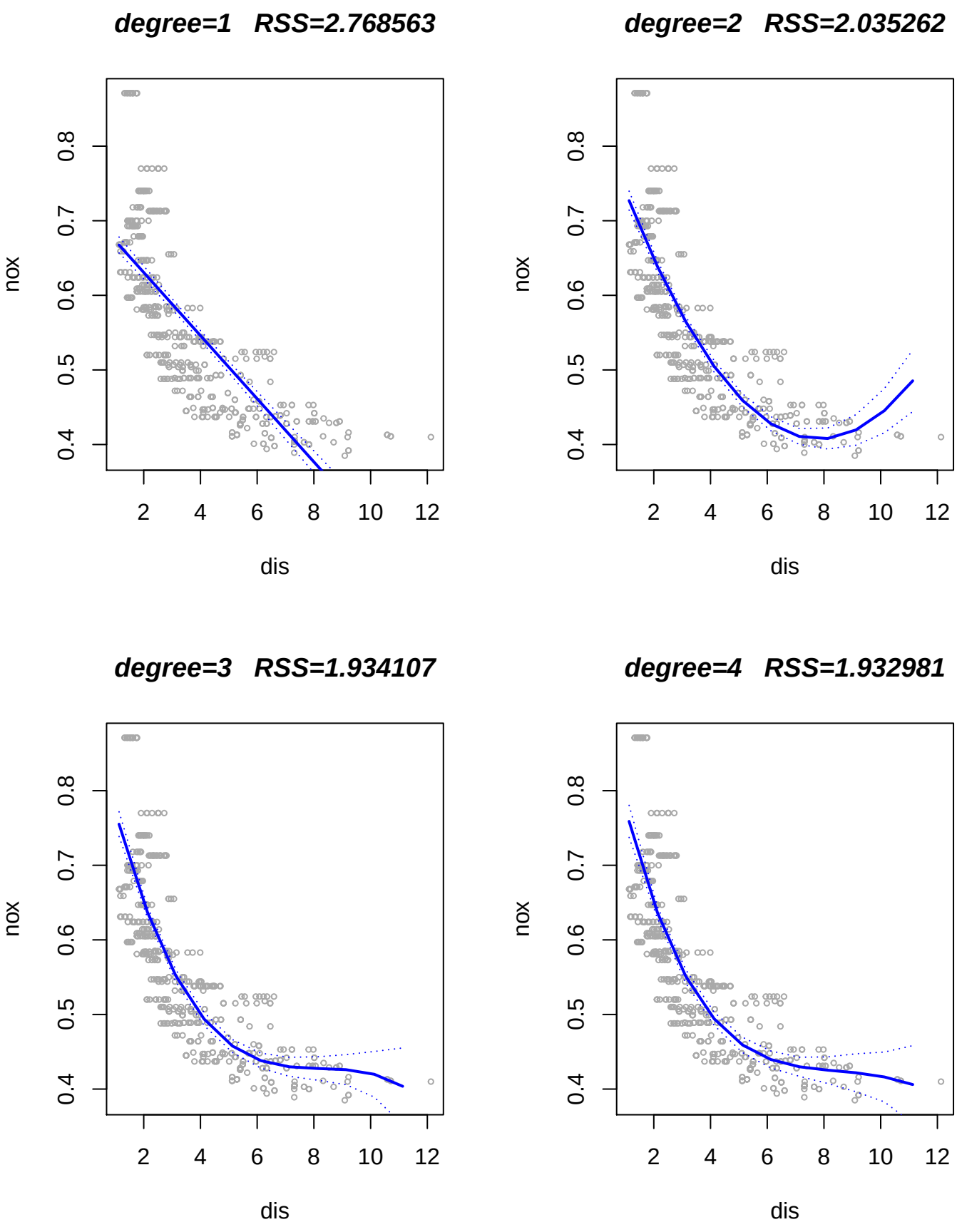


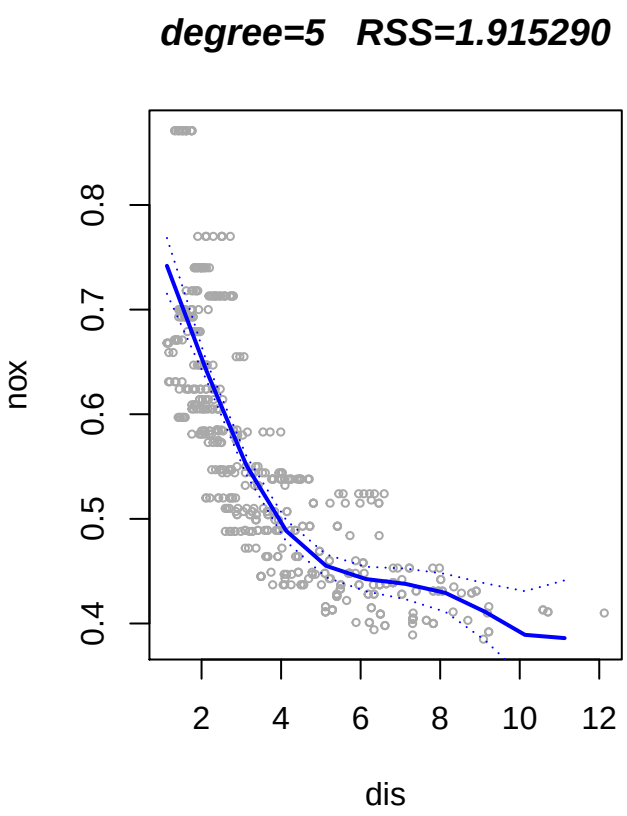
b)

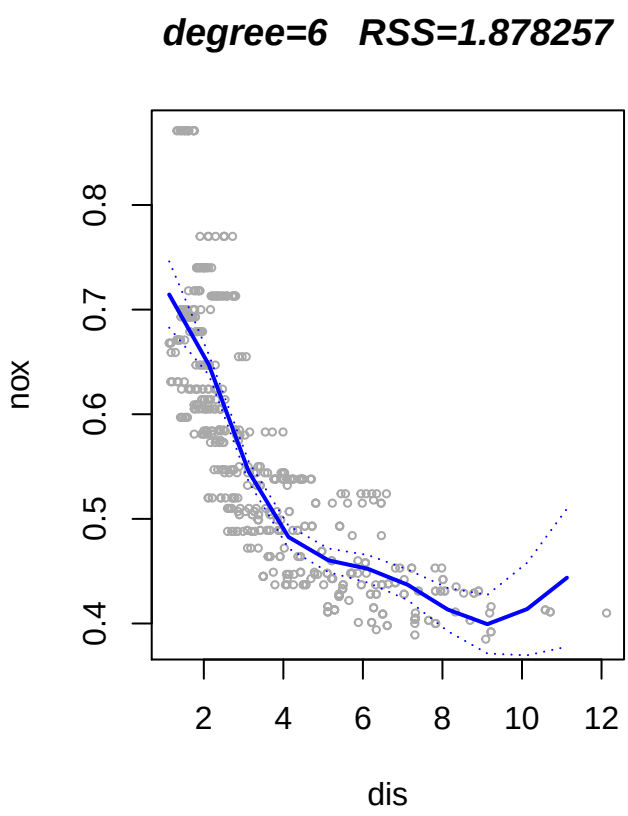
Plot the polynomial fits for a range of different polynomial degrees (say, from 1 to 10), and report the associated residual sum of squares.

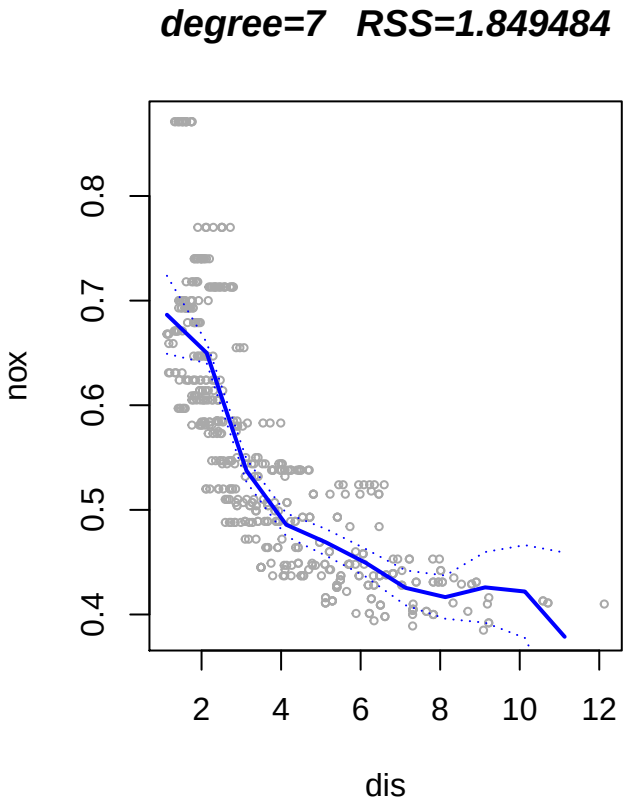
```
max.poly <- 10
nox <- Boston$nox
dis <- Boston$dis
plotFunc <- function(dis, nox, degree_val) {
    lmpoly.fit <- lm(nox ~ poly(dis, degree = degree_val))

    dislims = range(dis)
    dis.grid = seq(from = dislims[1], to = dislims[2])

    preds = predict(lmpoly.fit, newdata = data.frame(dis = dis.grid), se = TRUE)
```

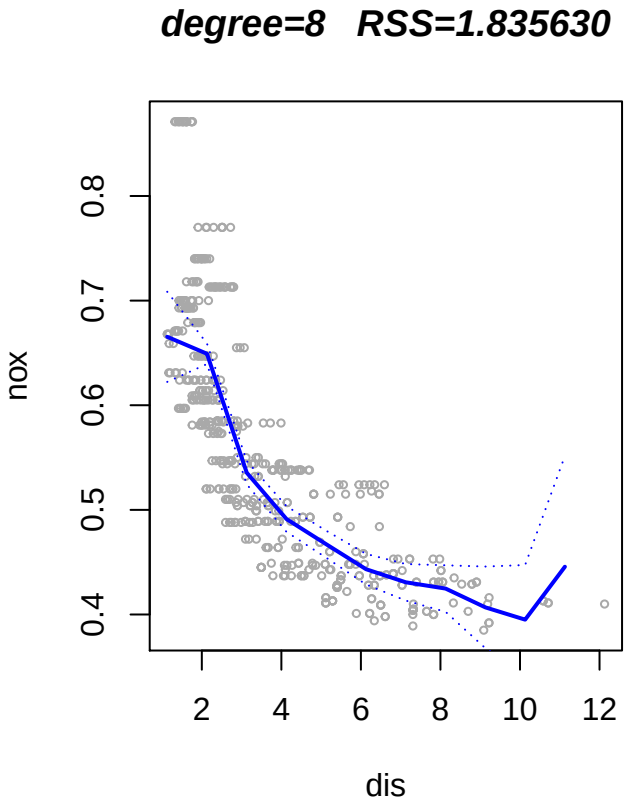
```
    se.bands = cbind(preds$fit + 2 * preds$se.fit, preds$fit - 2 * preds$se.fit)

    plot(dis, nox, xlim = dislims, cex = 0.5, col = "darkgrey ")

    lines(dis.grid, preds$fit, lwd = 2, col = " blue")
    matlines(dis.grid, se.bands, lwd = 1, col = " blue", lty = 3)
    title(c(sprintf("degree=%d   RSS=%f", degree_val, sum(lmpoly.fit$residuals^2))),
        cex = 0.5, font.main = 4)

}

for (i in 1:max.poly) {
    plotFunc(dis, nox, degree_val = i)
}
```

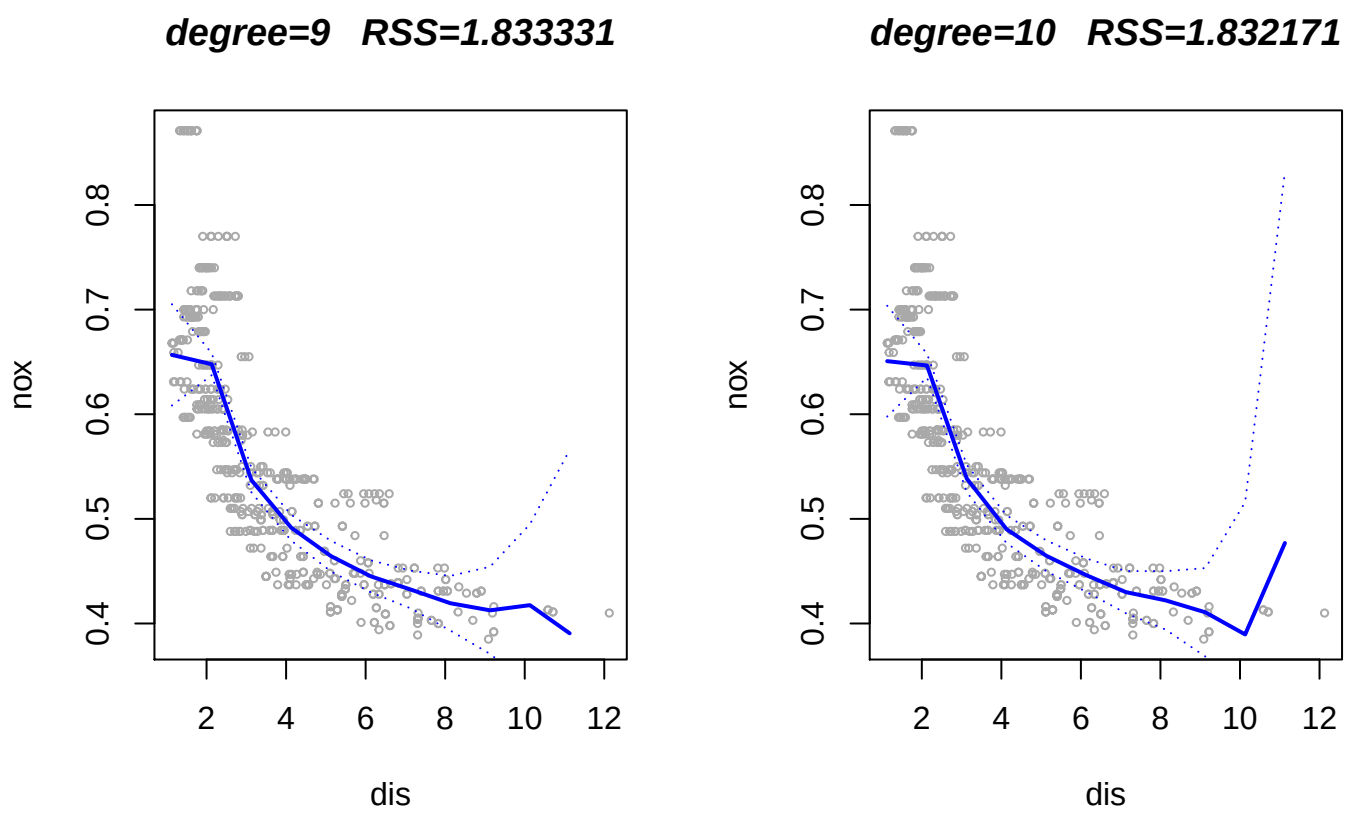

degree=1 RSS=2.768563
degree=2 RSS=2.035262
degree=3 RSS=1.934107
degree=4 RSS=1.932981
nox
dis
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12


degree=5 RSS=1.915290
nox
dis
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12


degree=6 RSS=1.878257
nox
dis
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12


degree=7 RSS=1.849484
nox
dis
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12


degree=8 RSS=1.835630
nox
dis
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12



c)

Perform cross-validation or another approach to select the optimal degree for the polynomial, and explain your results.

```
max.poly <- 10

dis = Boston$dis
nox = Boston$nox


cvFunc <- function(dis, nox, degree) {
    preds <- numeric(length(dis))
    for (i in 1:length(dis)) {
        # Here is where we exclude i from the training set
        dis.in <- dis[-i]
        dis.out <- dis[i]

        # Single test point
        nox.in <- nox[-i]
        nox.out <- dis[i]

        fit <- lm(nox.in ~ poly(dis.in, degree = degree))
        preds[i] <- predict(fit, newdata = data.frame(dis.in = dis.out))
```

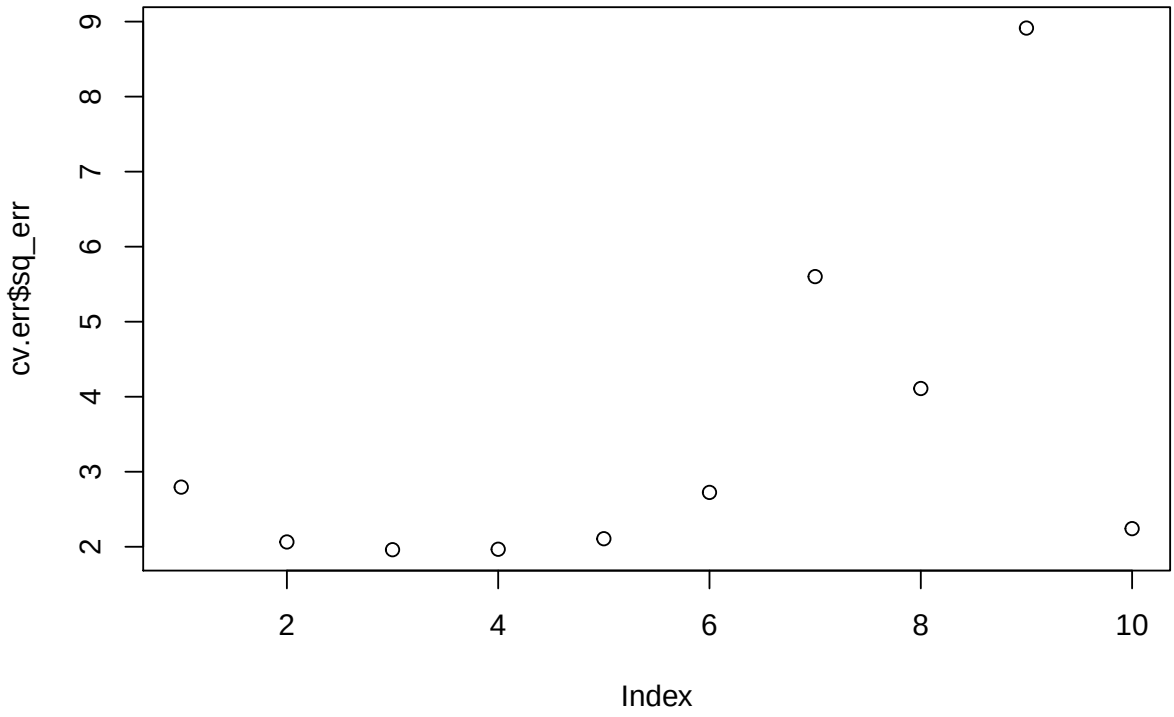
```
    }
    return(sum((nox - preds)^2))
}

cv.err <- data.frame(sq_err = numeric(max.poly))
for (i in 1:max.poly) {
    cv.err[i, 1] <- cvFunc(dis, nox, degree = i)
}

which.min(cv.err$sq_err)
```

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

```
library(pander)
pander(cv.err)
```

sq_err
2.795
2.064
1.961
1.967
2.107
2.724
5.601
4.109
8.914
2.242

```
plot(cv.err$sq_err)
```

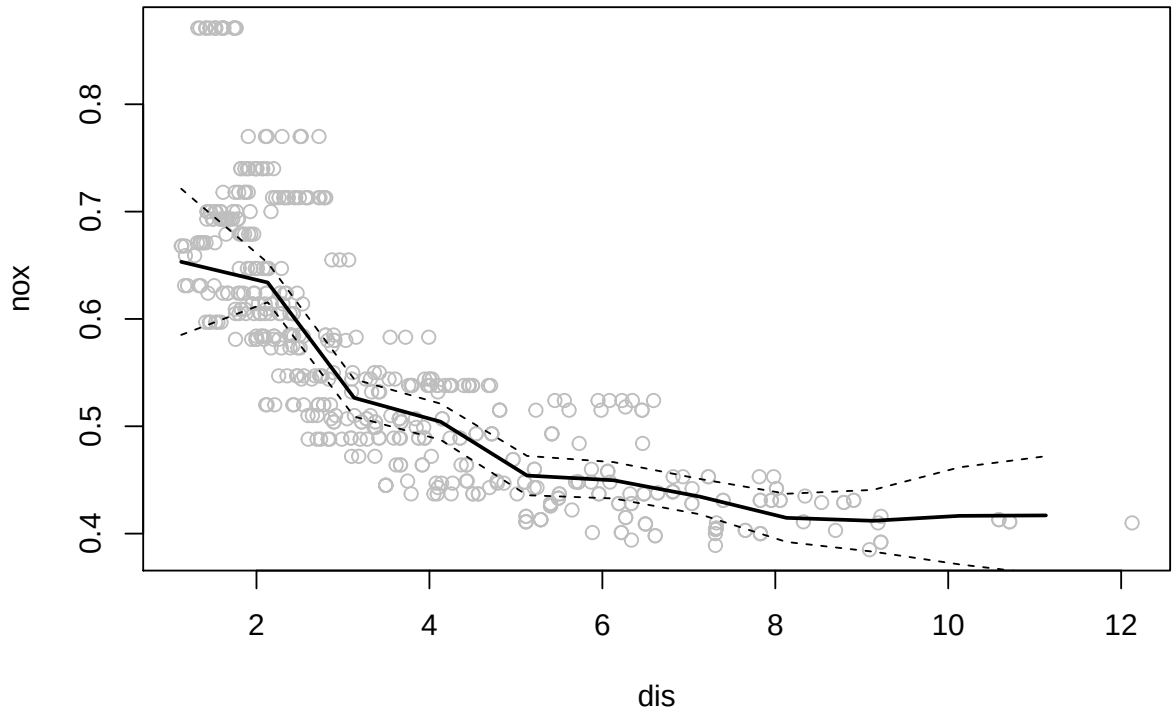


We've used leave one out cross validation above to select the best degree based on the CV_n error rate. Interestingly, there is a marked drop in the CV_n at degree of 10. The best degree based on the LOOCV algorithm is 3.

d)

Use the bs() function to fit a regression spline to predict nox using dis. Report the output for the fit using four degrees of freedom. How did you choose the knots? Plot the resulting fit.

```
library(splines)
quantileLocs <- quantile(dis, ppoints(10))
fit = lm(nox ~ bs(dis, knots = quantileLocs), data = Boston)
dislims = range(dis)
dis.grid = seq(from = dislims[1], to = dislims[2])
pred = predict(fit, newdata = list(dis = dis.grid), se = T)
plot(dis, nox, col = " gray ")
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")
```

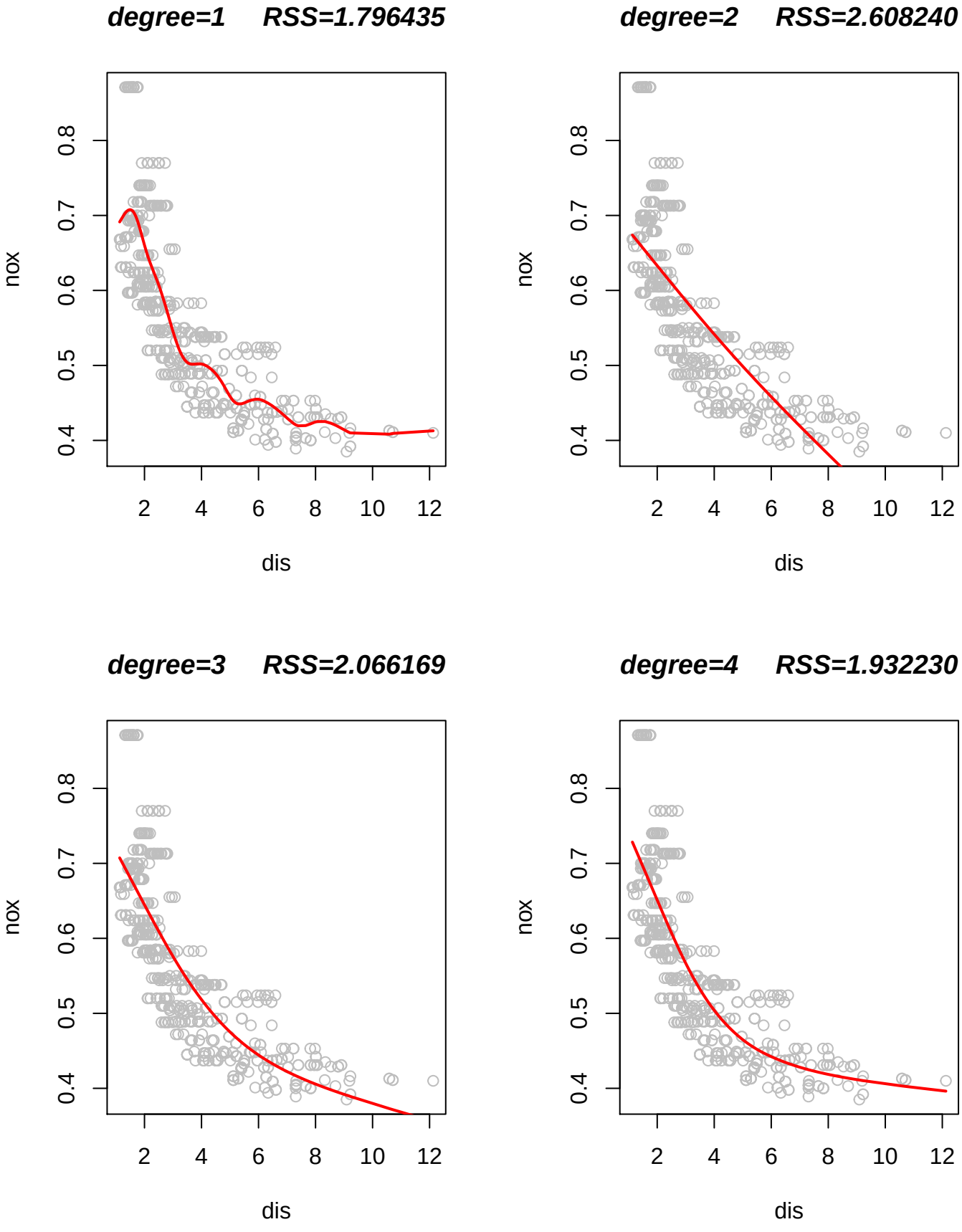


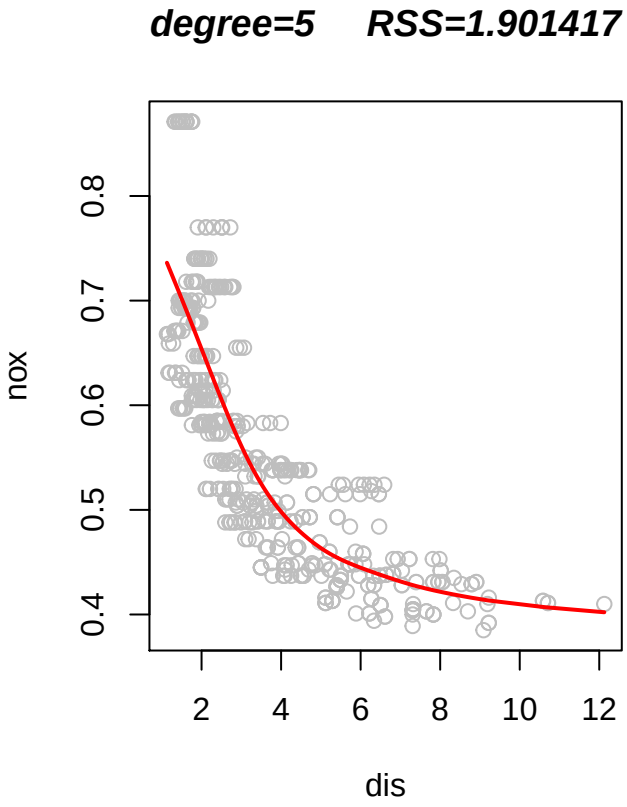
We chose the knots to be distributed according to the deciles of the data.

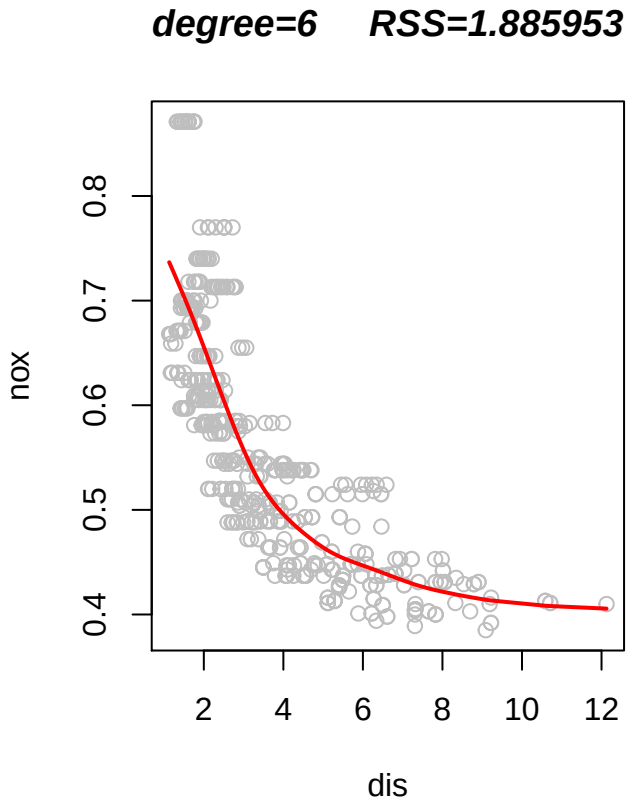
e)

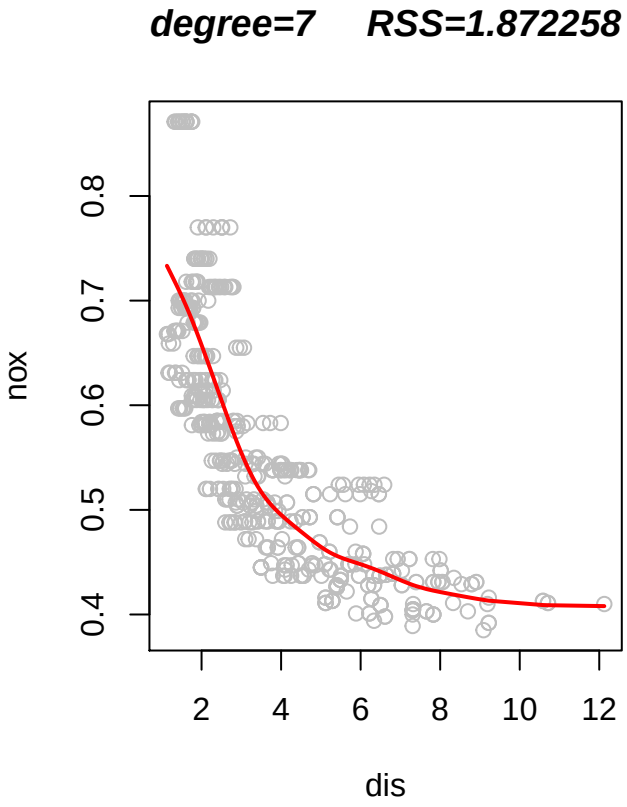
Now fit a regression spline for a range of degrees of freedom, and plot the resulting fits and report the resulting RSS. Describe the results obtained.

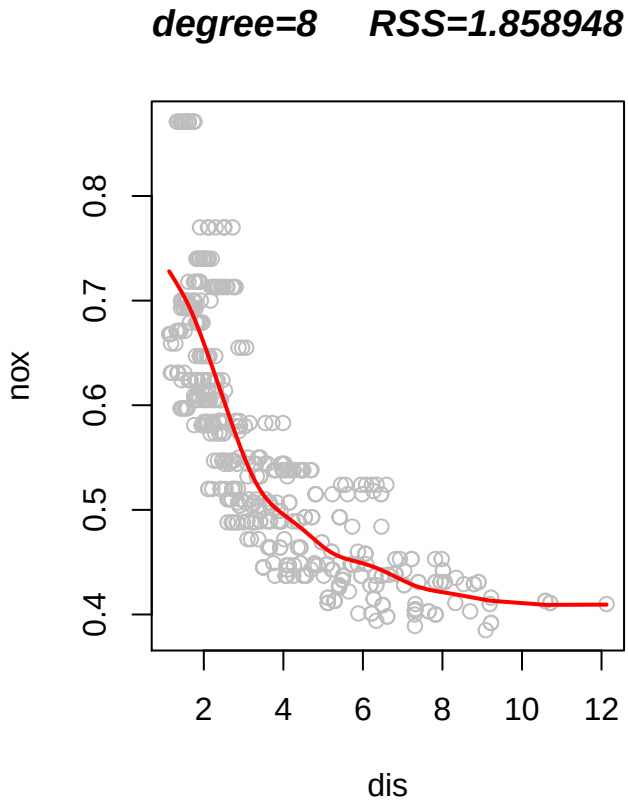
```
max.df <- 10
nox <- Boston$nox
dis <- Boston$dis
plotFunc <- function(dis, nox, degree_val) {
    fit = smooth.spline(dis, nox, df = degree_val)
    pred = predict(fit, se = T)
    plot(dis, nox, col = " gray ")
    lines(fit, col = "red ", lwd = 2)
    RSS <- sum(residuals(fit)^2)

    title(c(sprintf("degree=%d     RSS=%f", degree_val, RSS)), cex = 0.5, font.main = 4)

}

for (i in 1:max.poly) {
    plotFunc(dis, nox, degree_val = i)
}
```

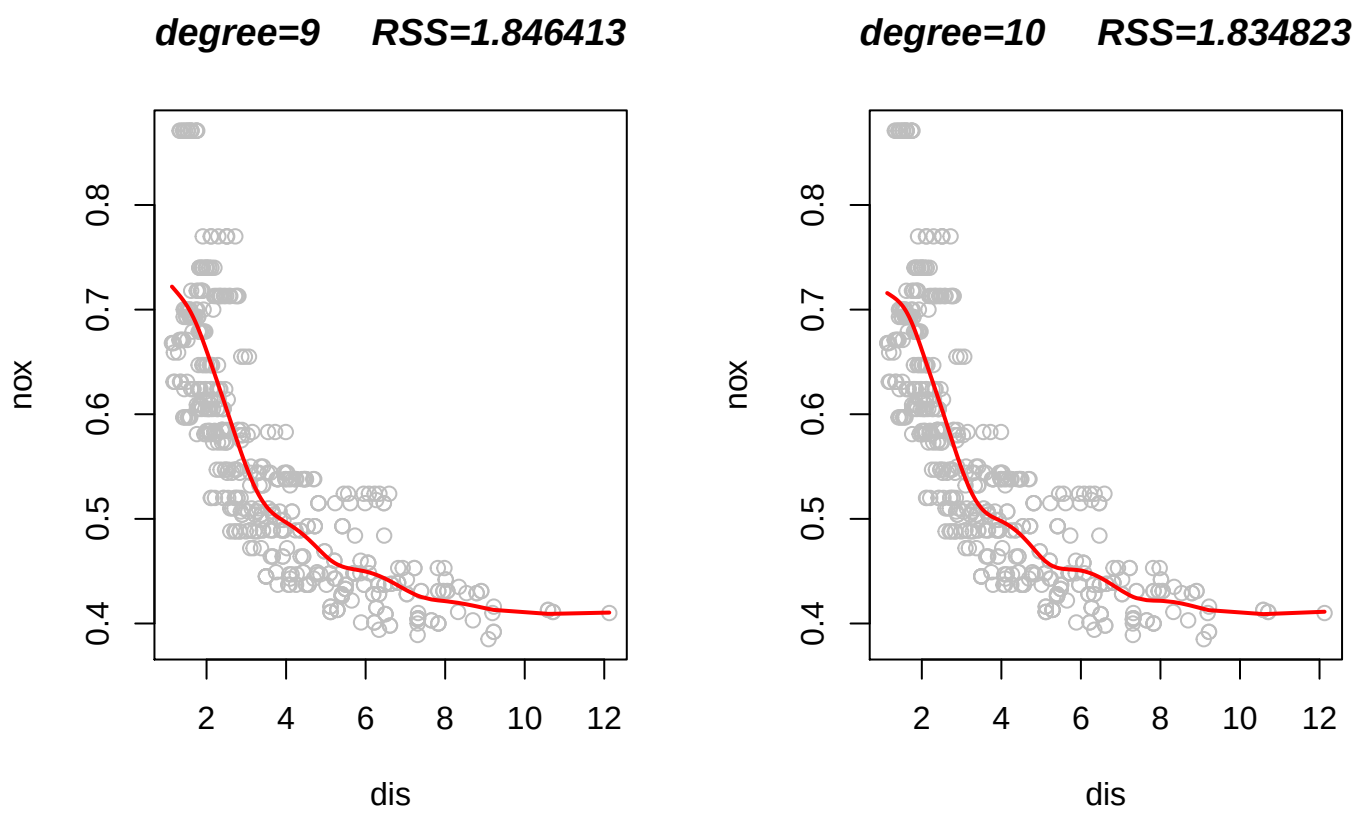

degree=1 RSS=1.796435
degree=2 RSS=2.608240
degree=3 RSS=2.066169
degree=4 RSS=1.932230
nox
dis
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12


degree=5 RSS=1.901417
nox
dis
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12


degree=6 RSS=1.885953
nox
dis
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12


degree=7 RSS=1.872258
nox
dis
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12


degree=8 RSS=1.858948
nox
dis
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12



f)

Perform cross-validation or another approach in order to select the best degrees of freedom for a regression spline on this data. Describe your results.

```
dis = Boston$dis
nox = Boston$nox

cvFunc <- function(dis, nox, degree_val) {
    preds <- numeric(length(dis))
    for (i in 1:length(dis)) {
        # Here is where we exclude i from the training set
        dis.in <- dis[-i]
        dis.out <- dis[i]

        # Single test point
        nox.in <- nox[-i]
        nox.out <- dis[i]

        fit = smooth.spline(dis.in, nox.in, df = degree_val)

        preds[i] <- predict(fit, dis.out)$y
    }
    return(sum((nox - preds)^2))
```

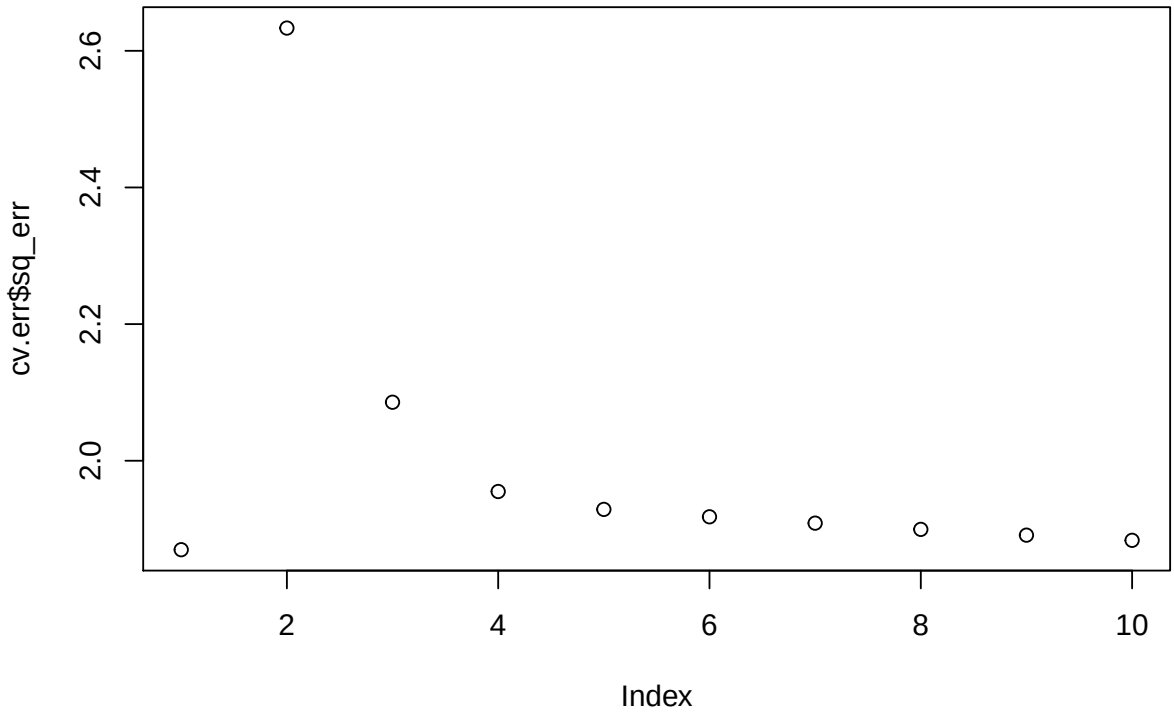
```
}

cv.err <- data.frame(sq_err = numeric(max.poly))
for (i in 1:max.df) {
    cv.err[i, 1] <- cvFunc(dis, nox, degree_val = i)
}

which.min(cv.err$sq_err)
```

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

```
library(pander)
pander(cv.err)
```

sq_err
1.87
2.633
2.086
1.955
1.929
1.918
1.909
1.9
1.891
1.884

```
plot(cv.err$sq_err)
```



We've used leave one out cross validation above to select the best degree based on the CV_n error rate. There are two candidates for best degree based on CV_n degree 1 and degree 10.

Chapter 7

Problem 10

This question relates to the College data set.

a)

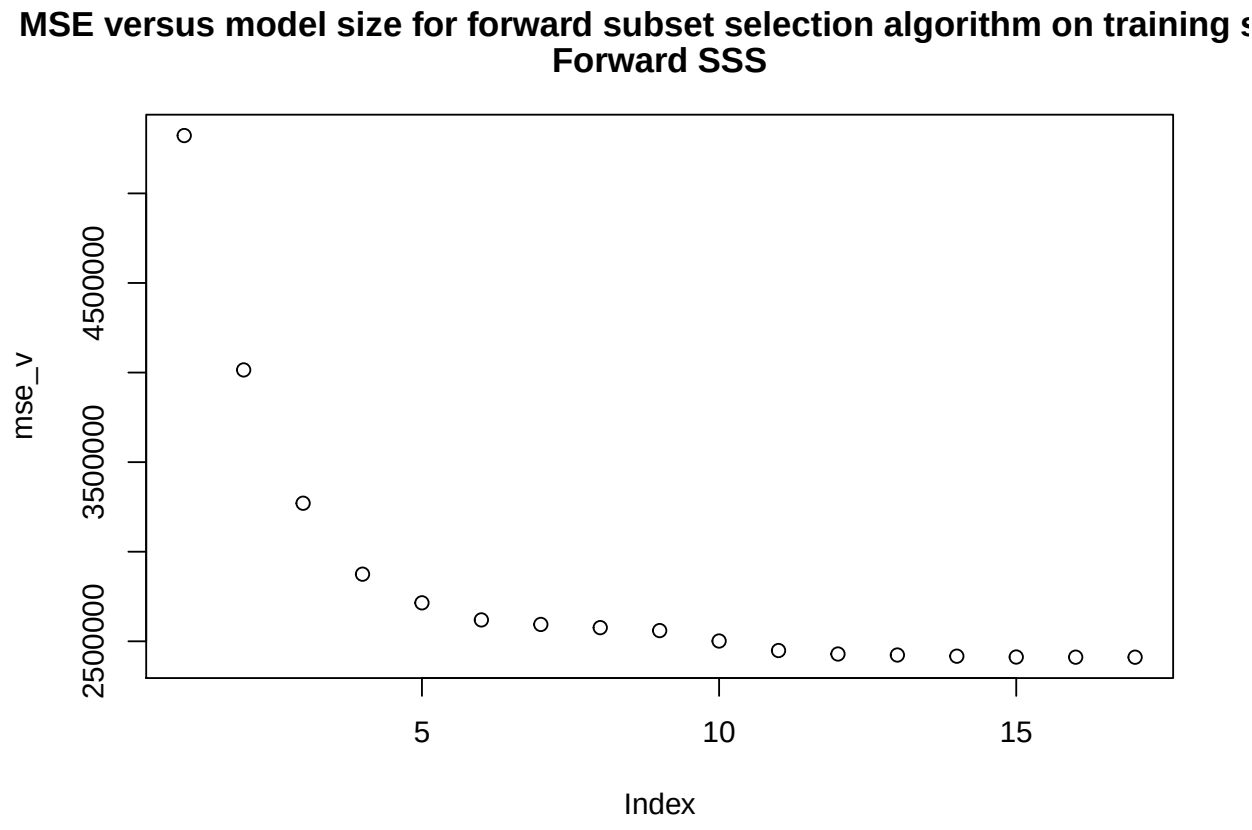
Split the data into a training set and a test set. Using out-of-state tuition as the response and the other variables as the predictors, perform forward stepwise selection on the training set in order to identify a satisfactory model that uses just a subset of the predictors.

```
rm(list = ls())
library(ISLR)
attach(College)

train = sample(nrow(College), floor(nrow(College) * 2/3))
DF <- College
DFTrain <- DF[train, ]
DFTest <- DF[-train, ]
```

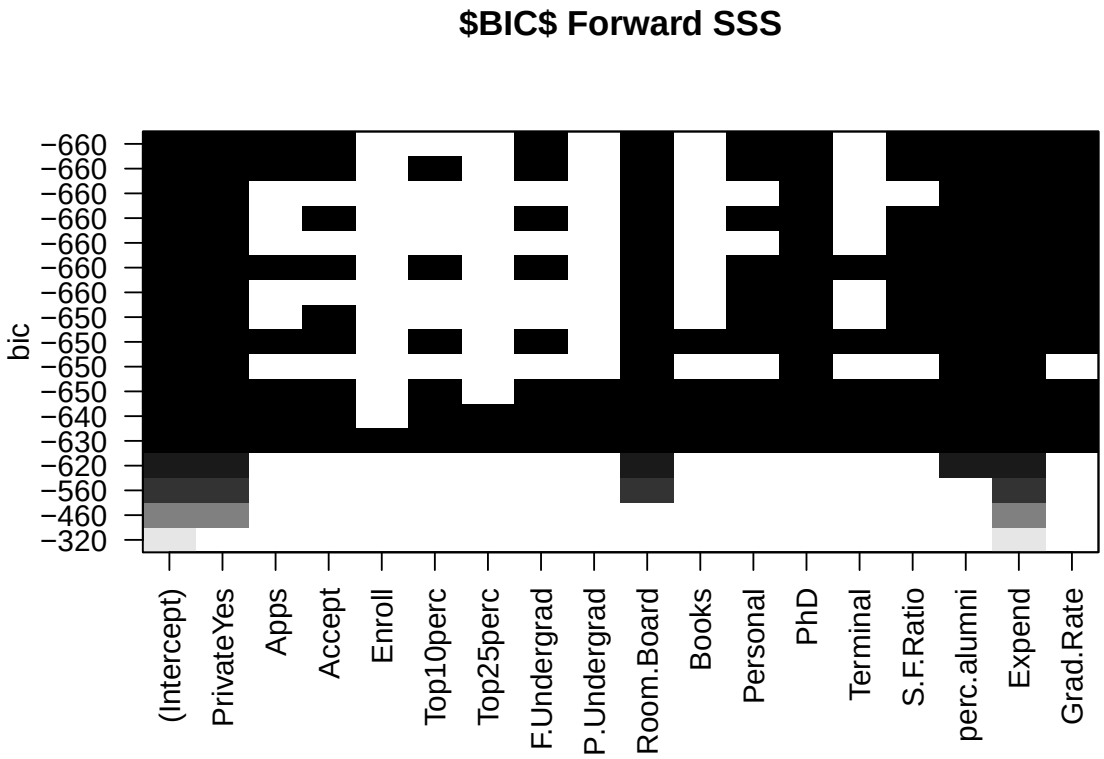
```
library(pander)
pander(names(DF))
```

Private, *Apps*, *Accept*, *Enroll*, *Top10perc*, *Top25perc*, *F.Undergrad*, *P.Undergrad*, *Outstate*, *Room.Board*, *Books*, *Personal*, *PhD*, *Terminal*, *S.F.Ratio*, *perc.alumni*, *Expend* and *Grad.Rate*

```
library(leaps)
regfit.full <- regsubsets(Outstate ~ ., data = DFTrain, method = "forward",
    nvmax = 18)

reg.summary <- summary(regfit.full)

mse_v <- reg.summary$rss/nrow(DF)

plot(mse_v)
title(c("MSE versus model size for forward subset selection algorithm on training set.",
    "Forward SSS"))
```




```
plot(regfit.full, scale = "bic")
title("$BIC$ Forward SSS")
```



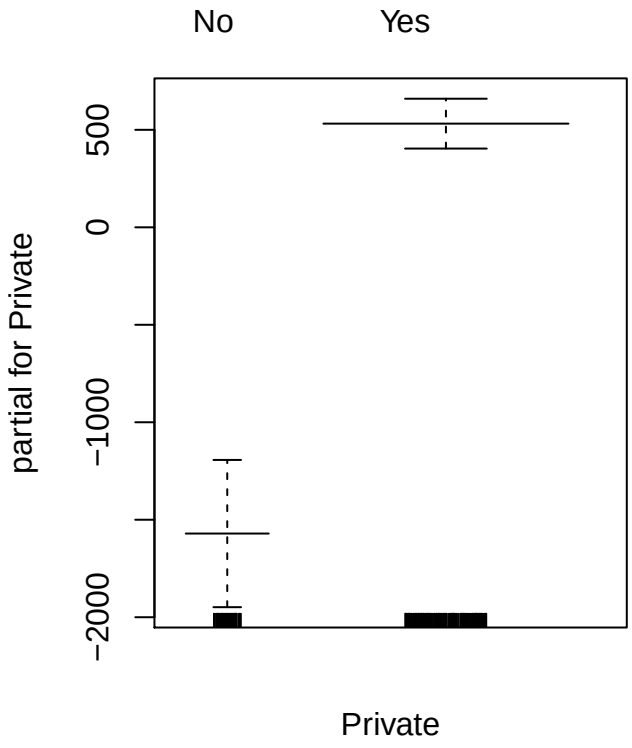

```
model_fss8 <- coef(regfit.full, 8)
pander(names(model_fss8))
```

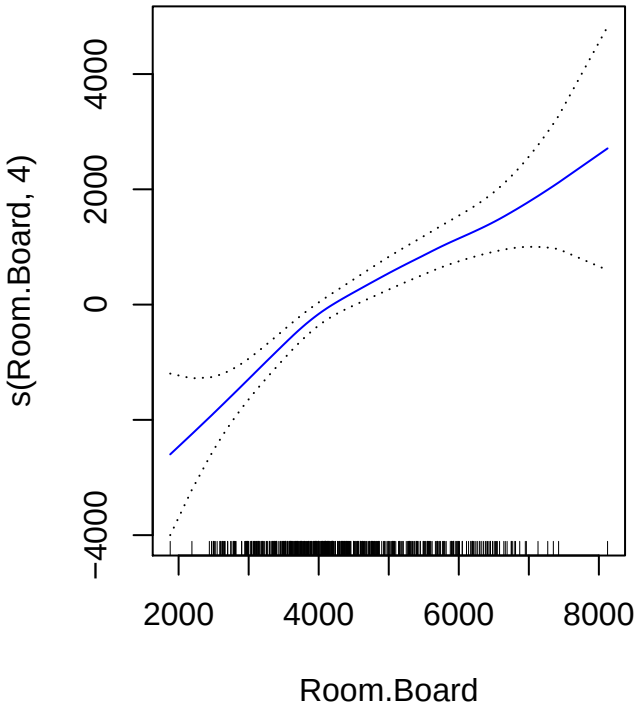
(Intercept), *PrivateYes*, *Room.Board*, *Personal*, *PhD*, *S.F.Ratio*, *perc.alumni*, *Expend* and *Grad.Rate*

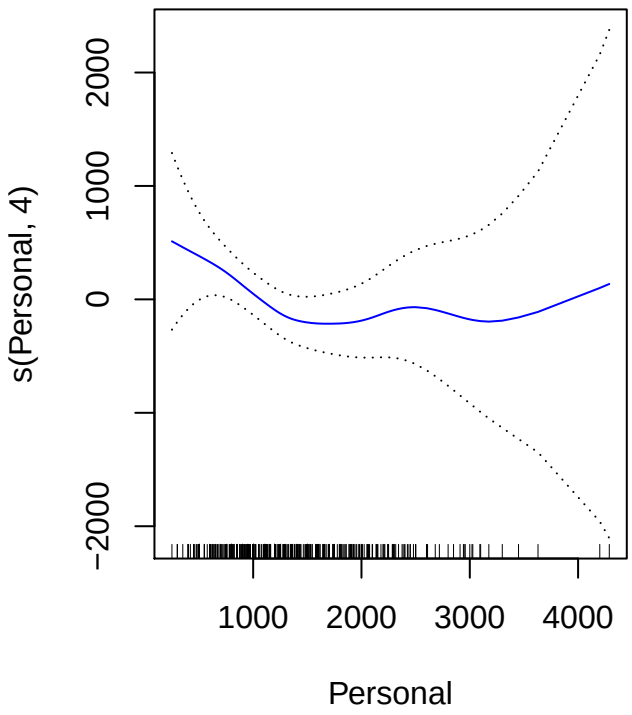
b)

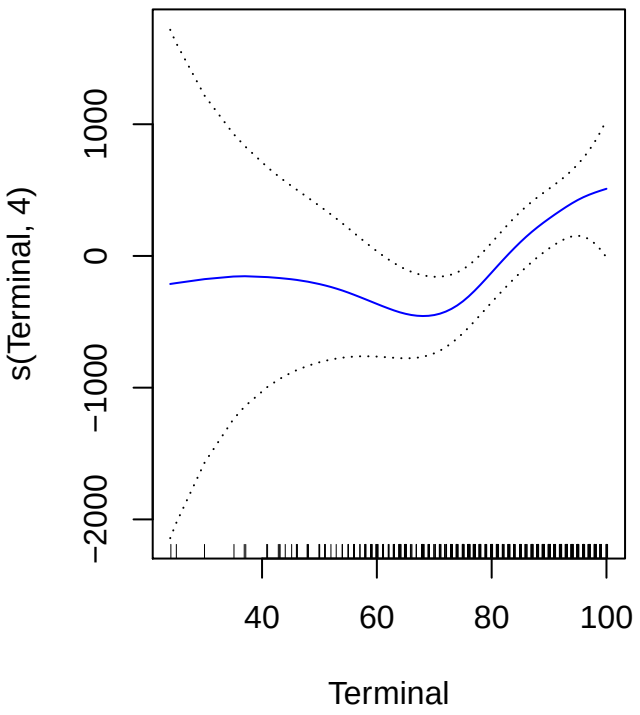
Fit a GAM on the training data, using out-of-state tuition as the response and the features selected in the previous step as the predictors. Plot the results, and explain your findings.

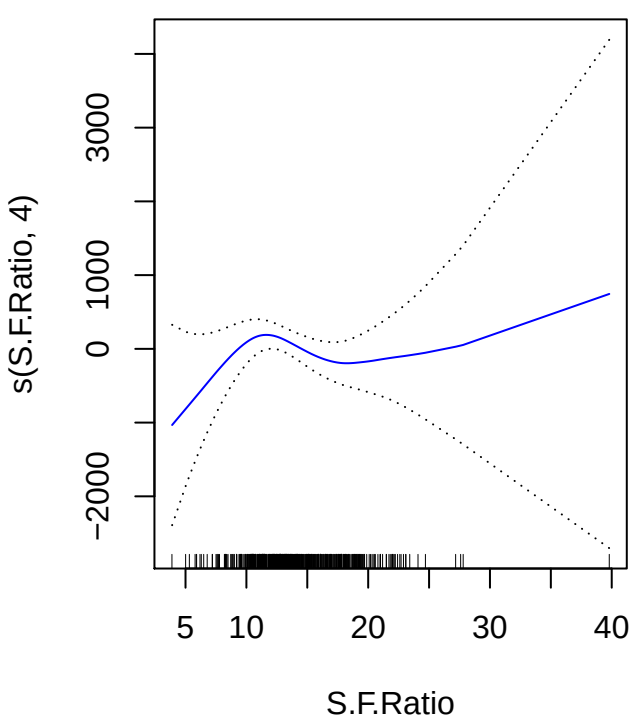
```
library(gam)

gam.fit = gam(Outstate ~ Private + s(Room.Board, 4) + s(Personal, 4) + s(Terminal,
    4) + s(S.F.Ratio, 4) + s(perc.alumni) + s(Expend, 4) + s(Grad.Rate, 4),
    data = DFTrain)

plot(gam.fit, se = TRUE, col = "blue ")
```

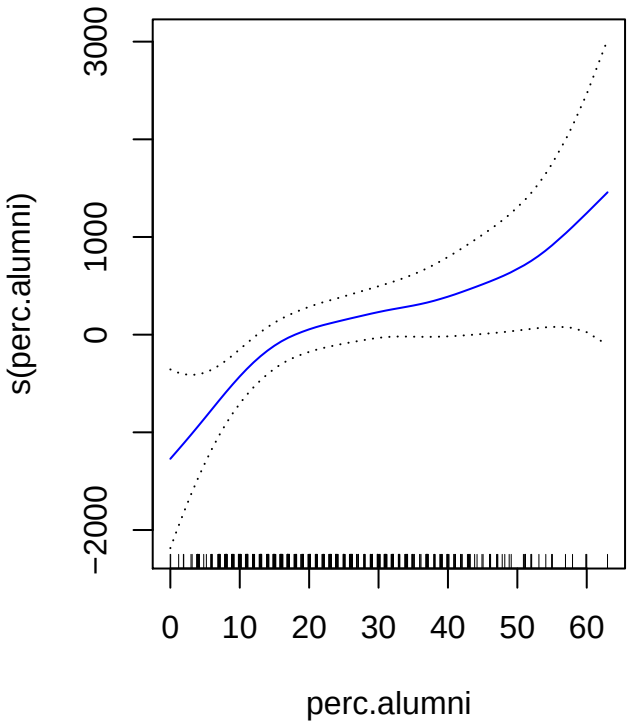

No
Yes
partial for Private
−2000
−1000
0
500
Private

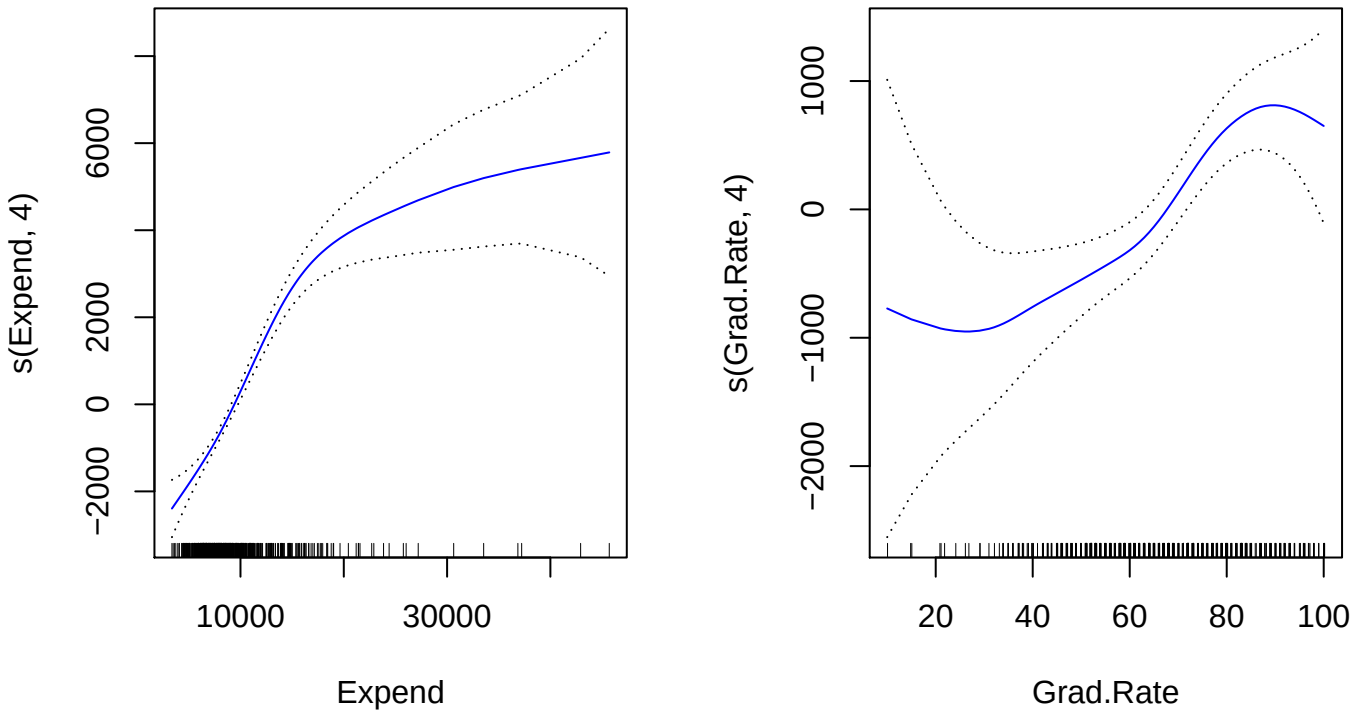

s(Room.Board, 4)
−4000
0
2000
4000
2000
4000
6000
8000
Room.Board


s(Personal, 4)
−2000
0
1000
2000
1000
2000
3000
4000
Personal


s(Terminal, 4)
−2000
−1000
0
1000
40
60
80
100
Terminal


s(S.F.Ratio, 4)
−2000
0
1000
3000
5
10
20
30
40
S.F.Ratio


s(perc.alumni)
−2000
0
1000
3000
0
10
20
30
40
50
60
perc.alumni

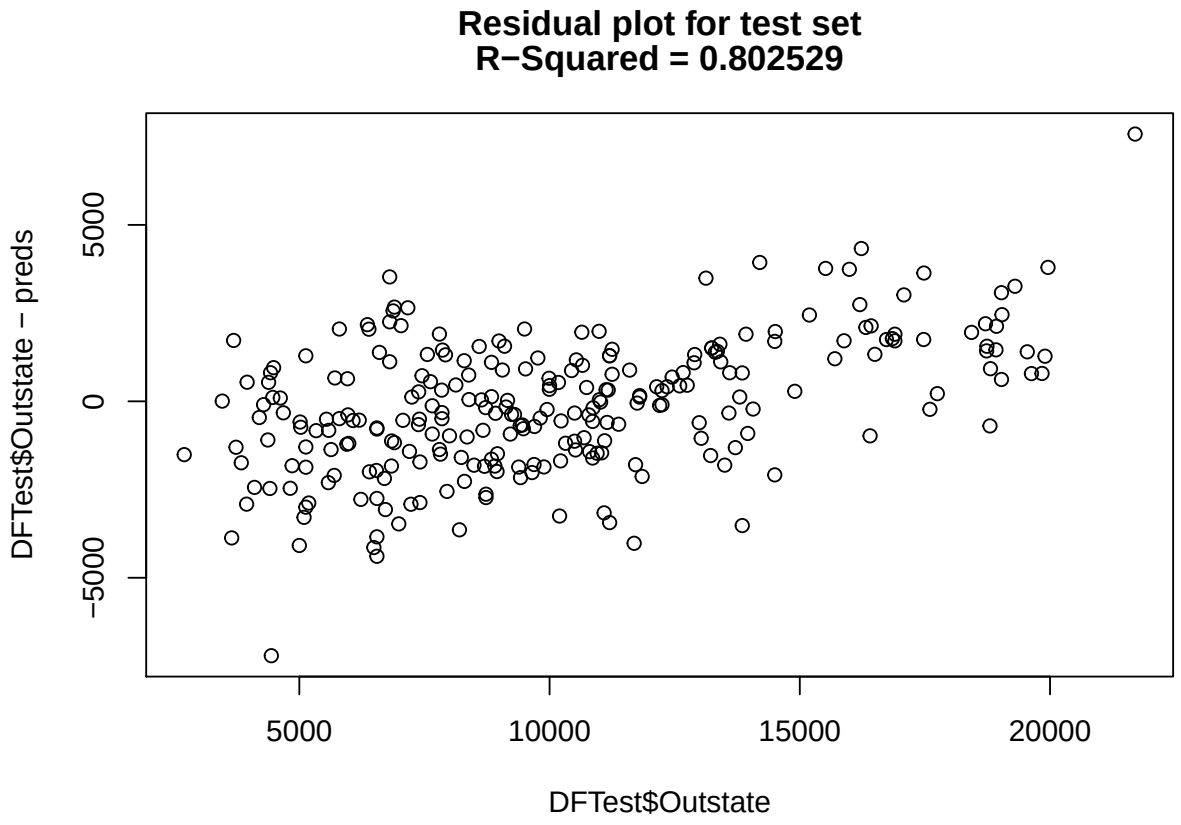


We have used the gam function to fit a univariate smoothing spline with 4 degrees of freedom to each predictor. The plots show the univariate fits. One of the predictors selected by the forward SSS algorithm is a factor and is not fit to a smoothing spline. There is strong evidence of non linear relationships in the data. The variables $Personal, S.F.Ratio, perc.alumni, expend$ show this particularly.

c)

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

```
preds = predict(gam.fit, newdata = DFTest)

RSS <- sum((preds - DFTest$Outstate)^2)
TSS <- sum((DFTest$Outstate - mean(DFTest$Outstate))^2)
RS2_Test <- 1 - (RSS/TSS)

plot(DFTest$Outstate, DFTest$Outstate - preds)
title(c("Residual plot for test set", sprintf("R-Squared = %f", RS2_Test)))
```




```
preds = predict(gam.fit, newdata = DFTrain)
RSS <- sum((preds - DFTrain$Outstate)^2)
TSS <- sum((DFTrain$Outstate - mean(DFTrain$Outstate))^2)
RS2_Train <- 1 - (RSS/TSS)
```

We see no significant trend in the residual plot, indicating that there is no unaccounted for non-linear relationships in the model. We also see that the training and test set R^2 statistic indicate a reasonable fit. As expected the test R^2 statistic is below the training R^2 value.

d)

For which variables, if any, is there evidence of a non-linear relationship with the response?

The variables $Personal, S.F.Ratio, perc.alumni, expend$ particularly show a non -linear relationship with the response.