Econ 187 HW2

Yiting Zhang

2022-04-28

Problem 6.9a

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.

```
data(College)

set.seed(12345678)
train <- sample(1:dim(College)[1], dim(College)[1]*.75, rep=FALSE)
test <- -train
c.training<- College[train, ]
c.testing= College[test, ]
```

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

```
# numeric data
c.training.scaled <- c.training %>% mutate_if(is.numeric,function(x) ifelse(is.na(x),median(x,na.rm=T),x

c.testing.scaled <- c.testing %>% mutate_if(is.numeric,function(x) ifelse(is.na(x),median(x,na.rm=T),x)

# categorical data
# (1) impute with mode
c.training.scaled <- c.training.scaled %>% mutate_if(is.character,function(x) ifelse(is.na(x),mode(x),x
c.testing.scaled <- c.testing.scaled %>% mutate_if(is.character,function(x) ifelse(is.na(x),mode(x),x))

# (2) encode data
c.training.scaled <- c.training.scaled %>% mutate_if(is.character,function(x) as.integer(factor(x)))
c.testing.scaled <- c.testing.scaled %>% mutate_if(is.character,function(x) as.integer(factor(x)))
```

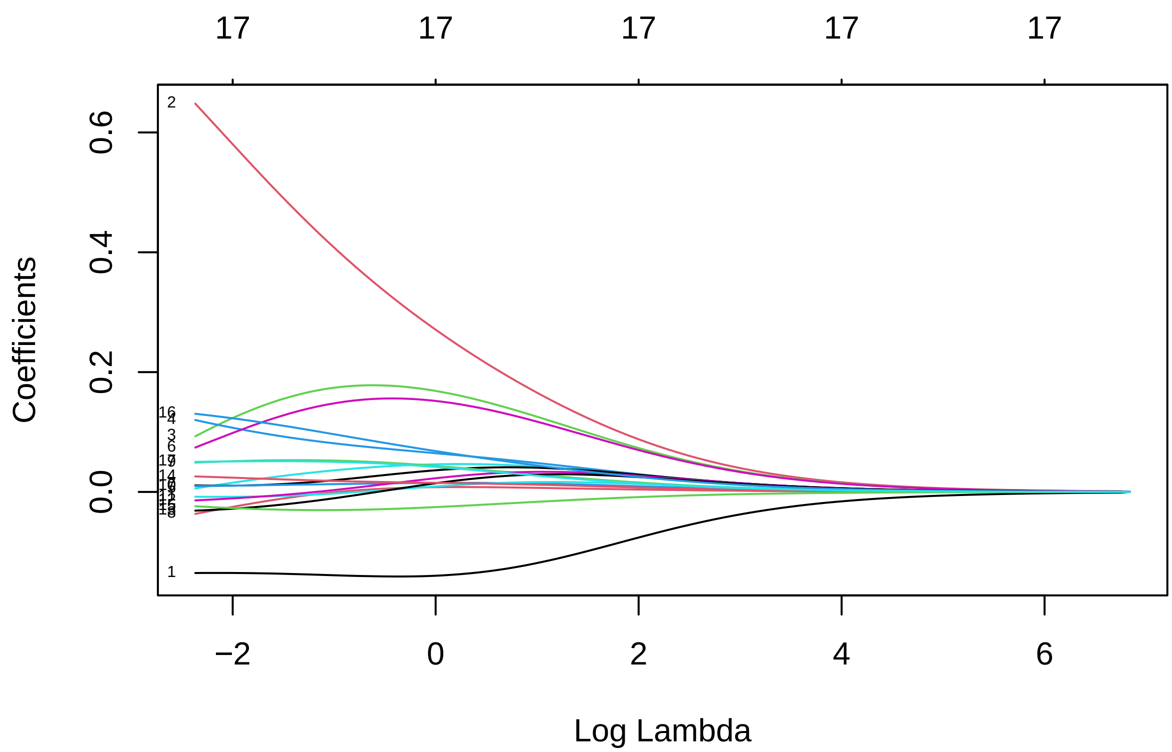
```
lm.fit = lm(Apps~., data=c.training.scaled)
lm.pred = predict(lm.fit, c.testing.scaled, type="response")
lm_MSE <- mean((lm.pred - c.testing.scaled$Apps)^2); lm_MSE
```

```
## [1] 0.04170952
```

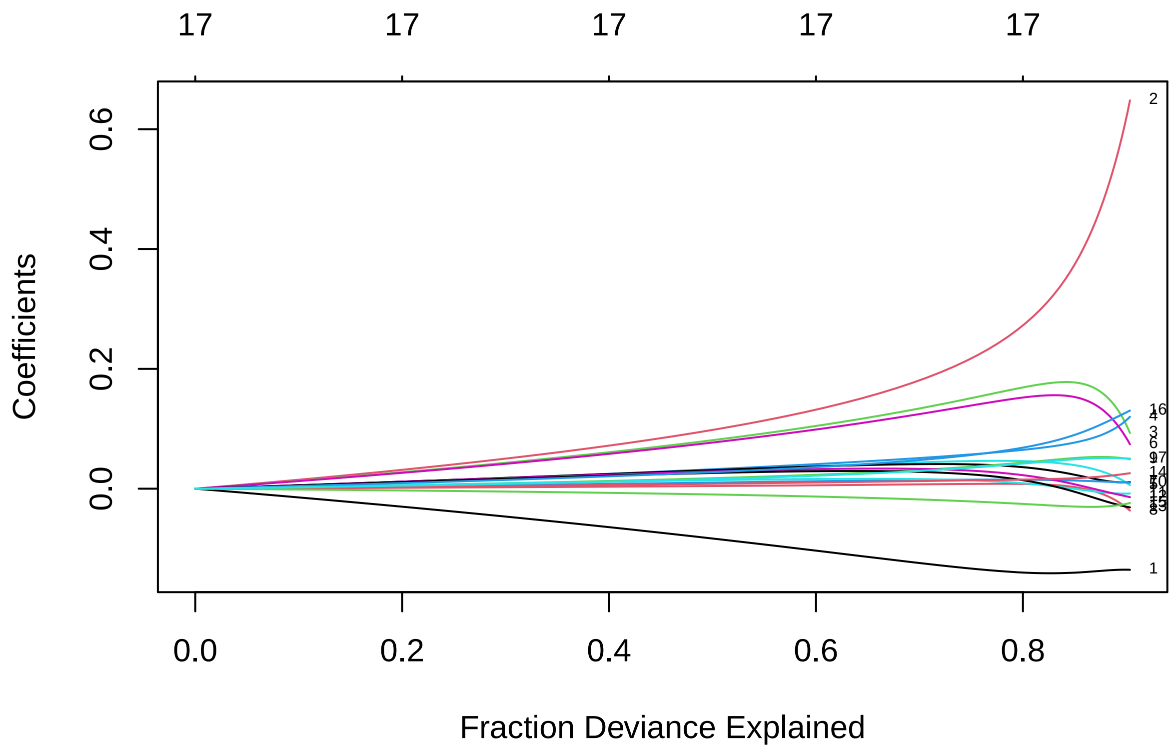
The OLS MSE is 0.0417.

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

```
model_ridge = cv.glmnet(x = data.matrix(c.training.scaled[,-which(names(c.training) %in% c("Apps"))]),
y=c.training.scaled$Apps, alpha = 0)

plot(model_ridge$glmnet.fit, "lambda", label=TRUE)
```



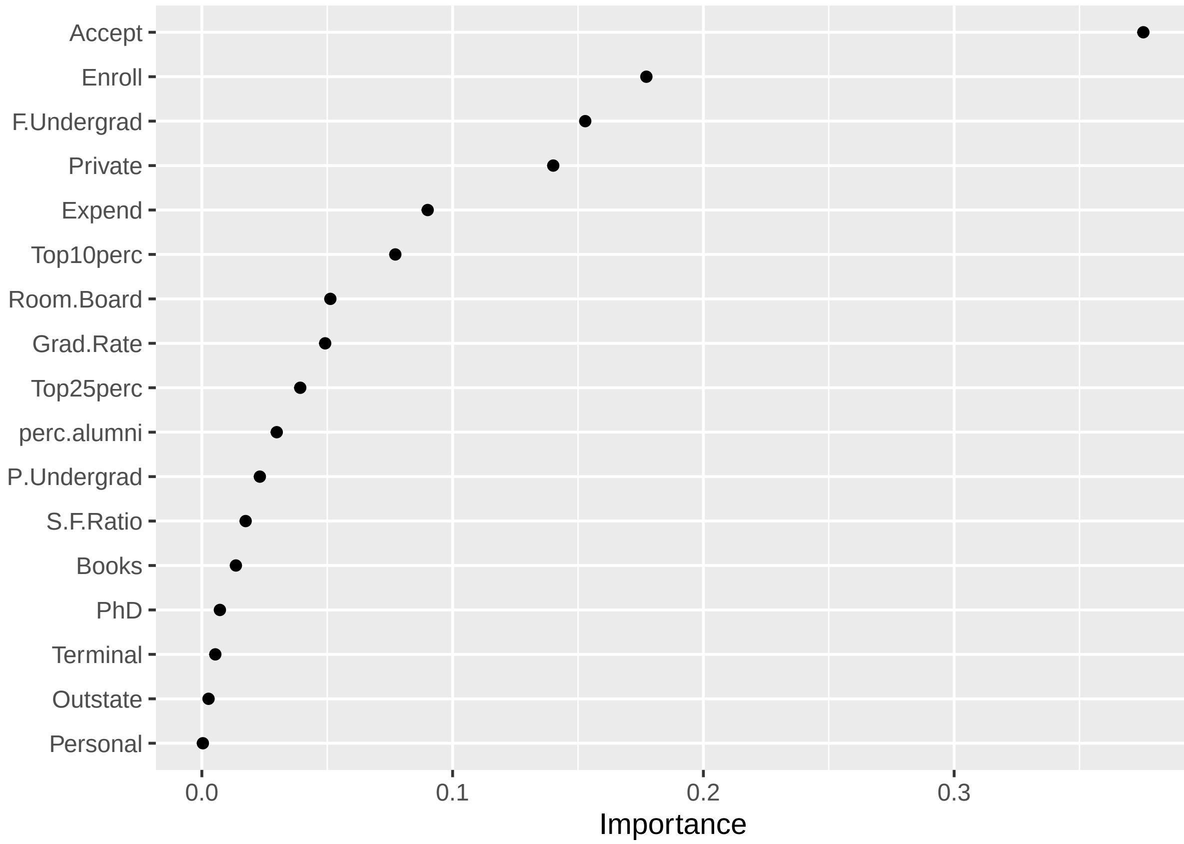

```
plot(model_ridge$glmnet.fit,xvar="dev",label=TRUE)
```




```
library(vip)
```

```
##
## Attaching package: 'vip'

## The following object is masked from 'package:utils':
##
##     vi
```

```
vip(model_ridge, num_features = 30, geom = "point")
```



This shows that the Private variable is the most important one by far, followed by whether the college is top 10 percent or not. This makes sense because the top 10 colleges are all private and private colleges tend to have a better rank.

```
# cross validation to train our ridge model to find the best lambda

train_control <- trainControl(method = "repeatedcv",
                              number = 5,
                              repeats = 1,
                              search = "random",
                              verboseIter = FALSE)

ridge_model     <- train(Apps ~ .,
                       data = c.training.scaled,
                       metrics = 'RMSE',
                       method = "ridge",
                       tuneLength = 25,
                       trControl = train_control)

# Predict using the testing data
ridge_pred = predict(ridge_model, newdata = c.testing.scaled)

# Evaluate performance
postResample(pred = ridge_pred, obs = c.testing.scaled[,'Apps'])
```

```
##      RMSE  Rsquared       MAE
```

```
## 0.2042257 0.9595370 0.1371349
```

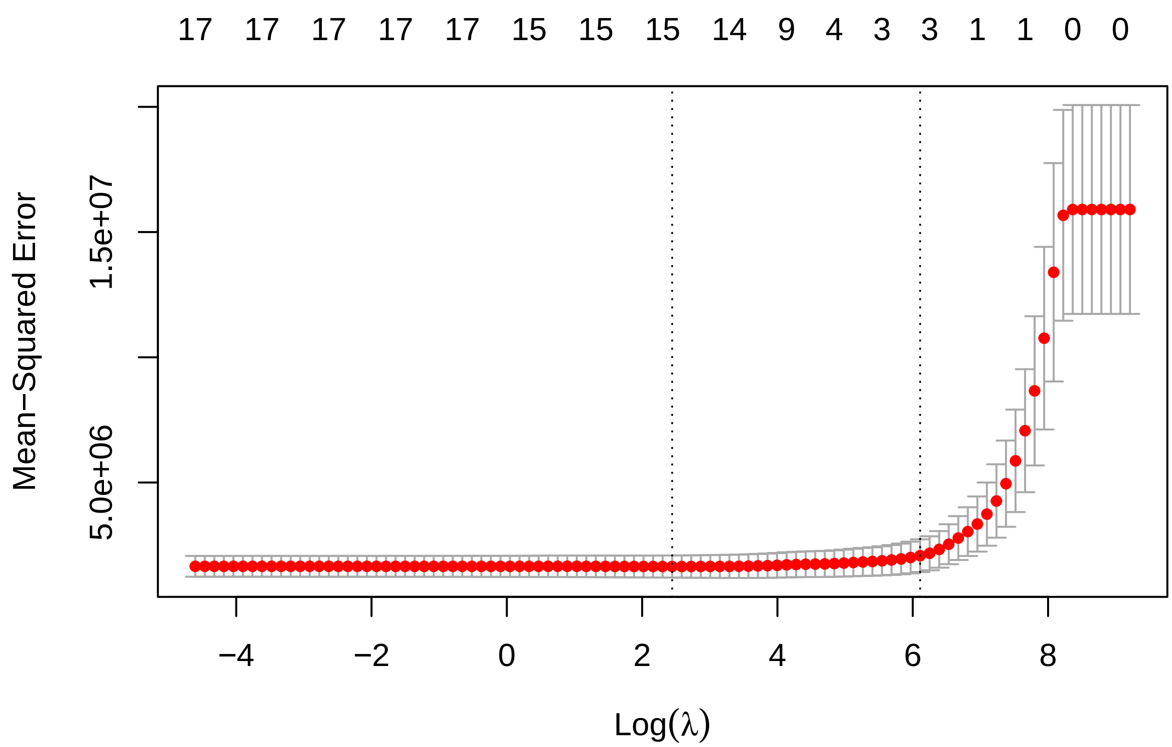
```
ridge_MSE <- mean((ridge_pred - c.testing.scaled[, "Apps"])^2); ridge_MSE
```

```
## [1] 0.04170816
```

The Ridge MSE is 0.04170833 The ridge model shows the best performance to be 0.2042257 (RMSE), with an R-squared of 0.959537

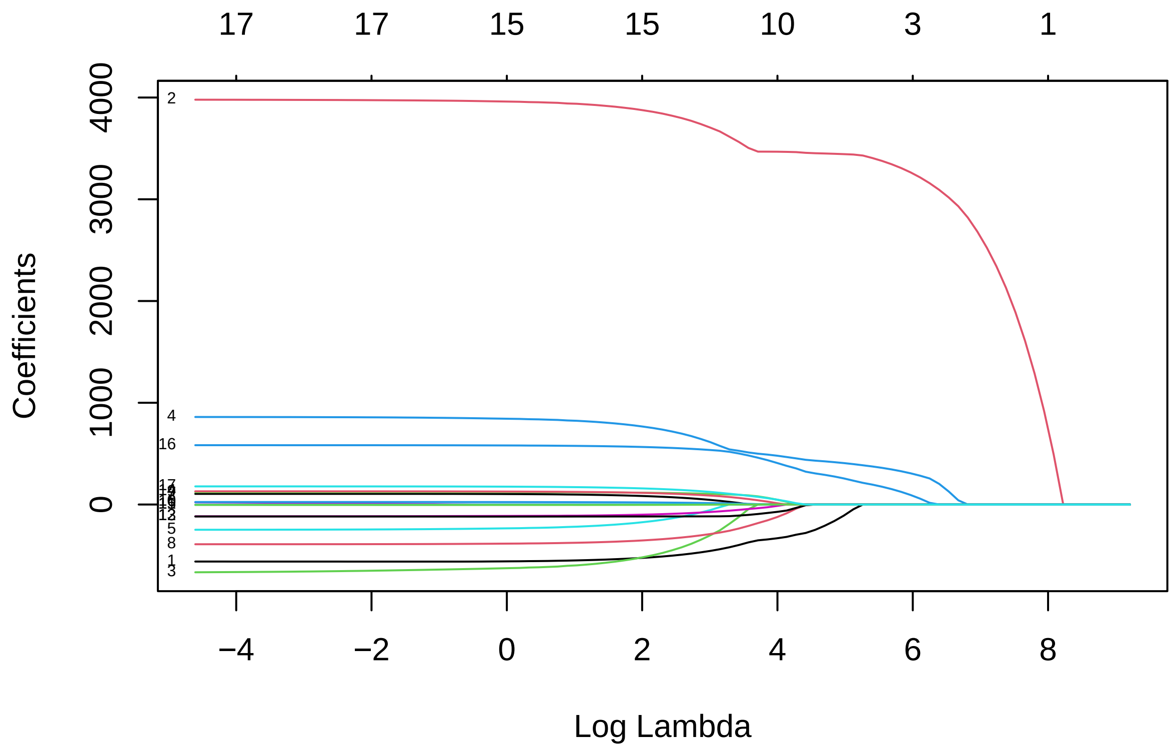
(d) Fit a lasso model on the training set, with lambda chosen by cross validation. Report the test error obtained, along with the number of non-zero coefficient estimates.

```
lambda_try <- 10^seq(-2, 4, length.out = 99)
cv_lasso = cv.glmnet(x = data.matrix(c.training.scaled[,-which(names(c.training) %in% c("Apps"))]),
y=c.training$Apps, alpha = 1, lambda=lambda_try,standardize = TRUE, nfolds = 10)

#choose best lambda
# Plot cross-validation results
plot(cv_lasso)
```



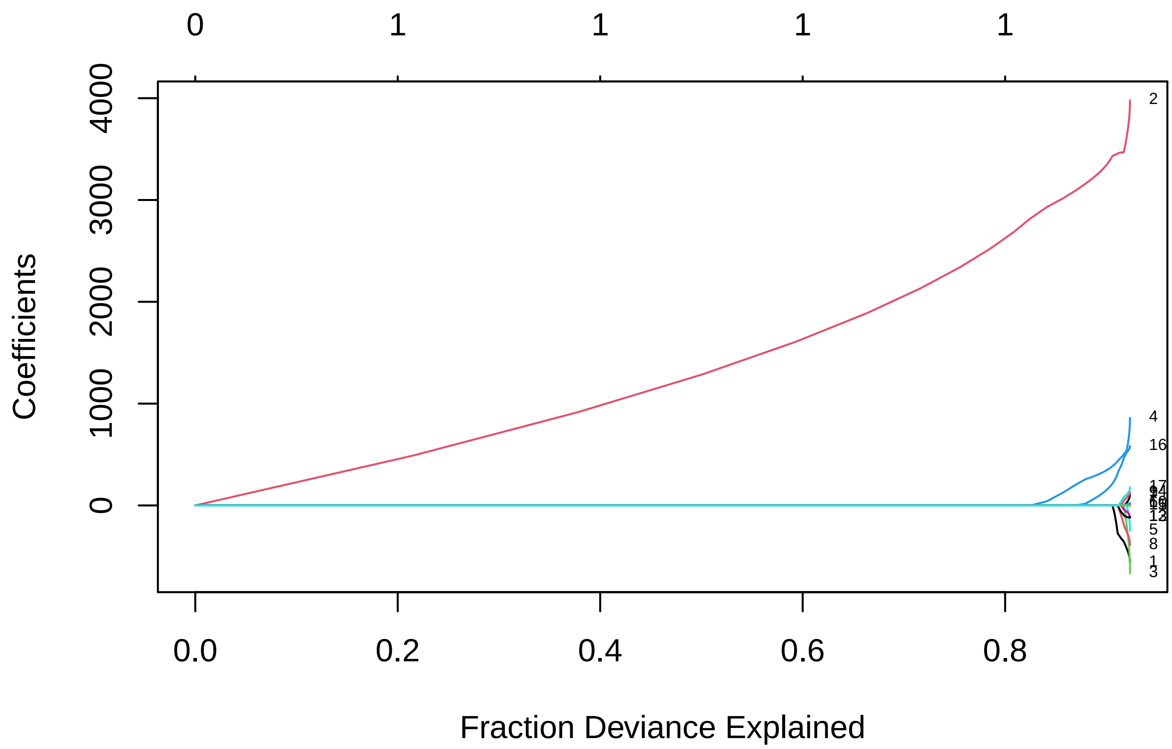

```
# Best cross-validated lambda
lambda_cv <- cv_lasso$lambda.min

# Fit final model
```

```r
model_lasso <- glmnet(x = data.matrix(c.training.scaled[,-which(names(c.training) %in% c("Apps"))]), y=

plot(cv_lasso$glmnet.fit, "lambda", label=TRUE)
```

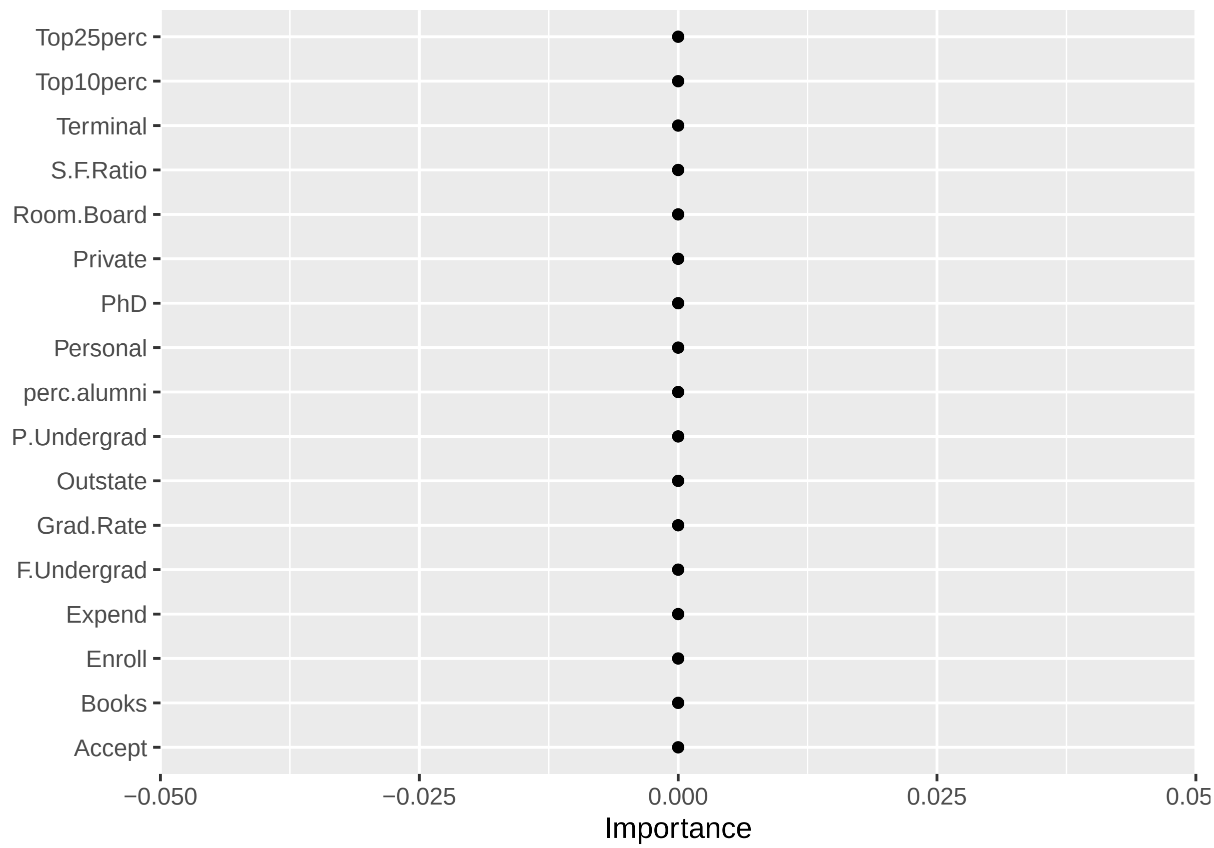



```r
plot(cv_lasso$glmnet.fit,xvar="dev",label=TRUE)
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```




```
vip(model_lasso, num_features = 30, geom = "point")
```



Lasso also shows that Accept is the most important variable.

```
train_control <- trainControl(method = "repeatedcv",
                              number = 5,
                              repeats = 5,
                              search = "random",
                              verboseIter = FALSE)

lasso_model     <- train(Apps ~ .,
                       data = c.training.scaled,
                       metrics = 'RMSE',
                       method = "glmnet",
                       tuneGrid = expand.grid(alpha = 1,
                                              lambda = 1),
                       tuneLength = 25,
                       trControl = train_control)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
# Predict using the testing data
lasso_pred = predict(lasso_model, newdata = c.testing.scaled)

# Evaluate performance
postResample(pred = na.omit(lasso_pred), obs = c.testing.scaled[,'Apps'])
```

```
##      RMSE  Rsquared       MAE
## 0.9974326        NA 0.7012511
```

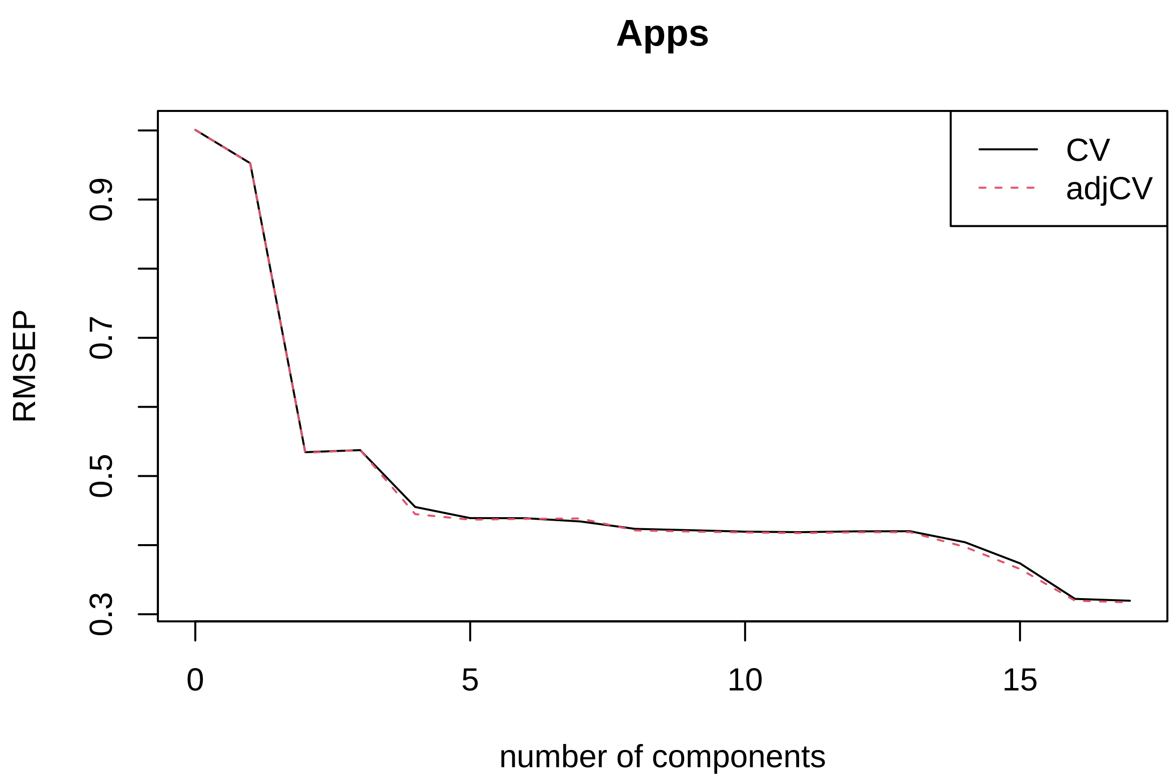
```
lasso_MSE <- mean((lasso_pred - c.testing.scaled[, "Apps"])^2); lasso_MSE
```

```
## [1] 0.9948718
```

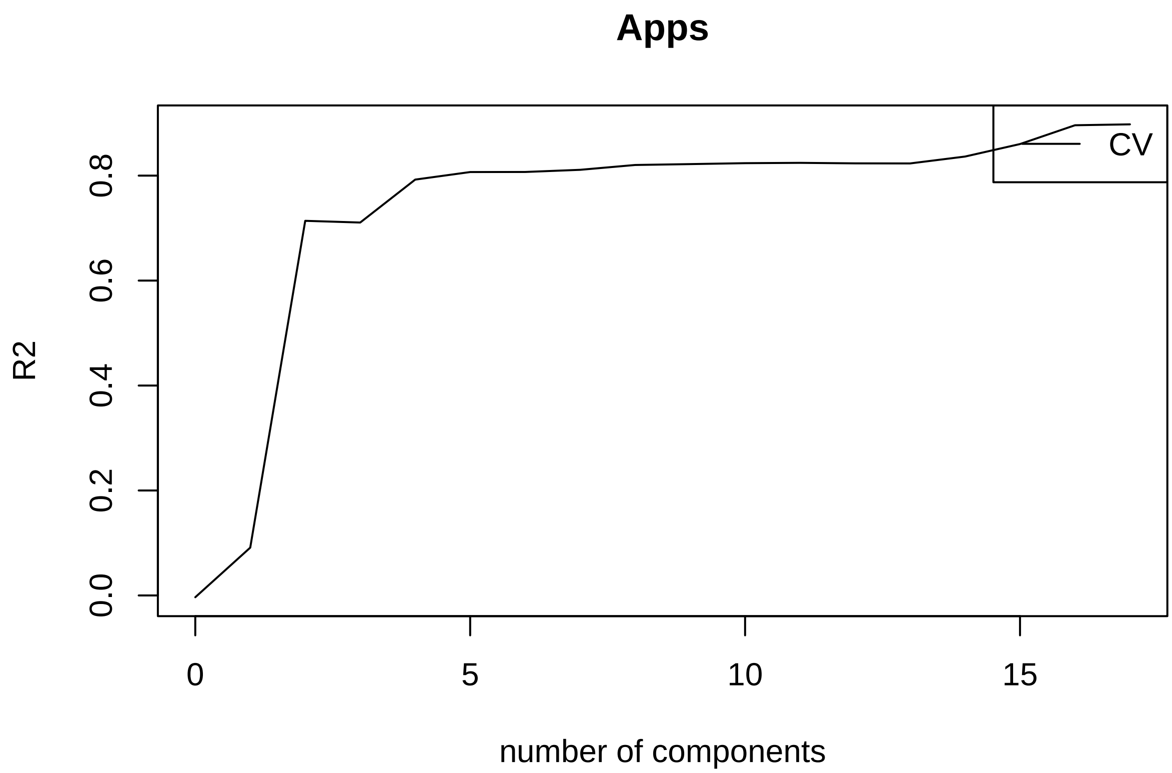
The lasso model shows the best performance as 0.9974326. The Lasso MSE is 0.9948718.

(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.fit <- pcr(Apps ~ .,data = c.training.scaled,validation = "CV")

validationplot(pcr.fit, val.type = "RMSEP",
               legendpos='topright')
```




```
validationplot(pcr.fit, val.type = "R2",
               legendpos='topright')
```



From the cross validation,the lowest cross-validation error is in the range of 3 to 10 components. Therefore, we will test ncomps in [21,40] to see which principal component regression model performs best on our test set.

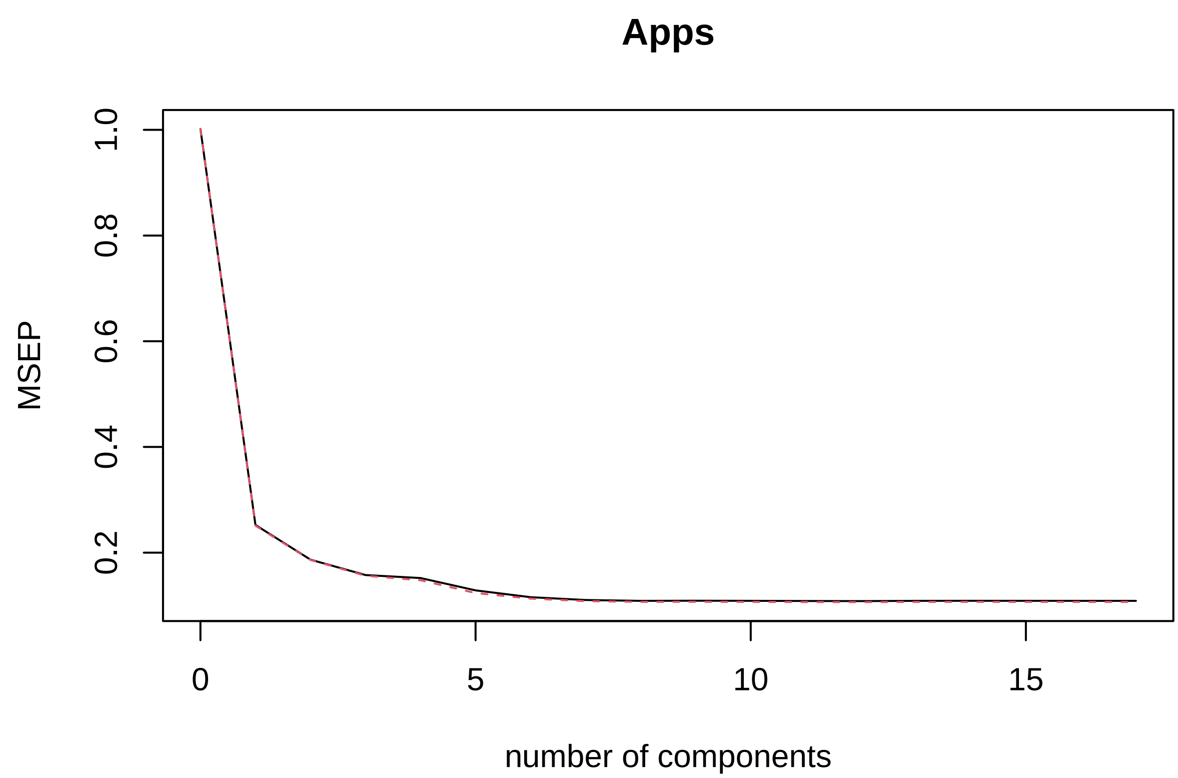
```
pcr.pred<- predict(pcr.fit, c.testing.scaled, ncomp = 7)
pred_pcr <- data.frame(pcr.pred)
pcr_MSE <- sqrt(mean(c.testing.scaled[, "Apps"] - pred_pcr$Apps.7.comps)^2); pcr_MSE
```

```
## [1] 0.001704798
```

PCR MSE is 0.001704798.

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

```
pls.fit <- plsr(Apps ~ ., data = c.training.scaled, scale = TRUE, validation = "CV")
validationplot(pls.fit, val.type = "MSEP")
```



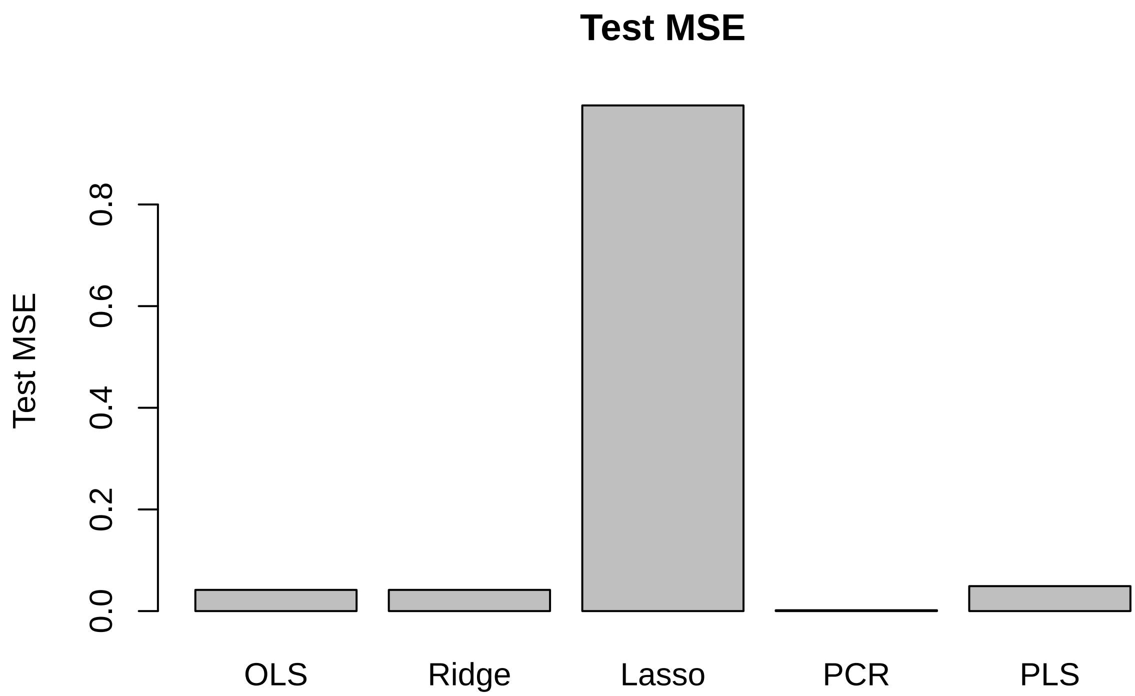

```
pls.pred <- predict(pls.fit, c.testing.scaled, ncomp = 5)
pred_pls <- data.frame(pls.pred)
pls_MSE <- mean((c.testing.scaled[, "Apps"] - pred_pls$Apps.5.comps)^2); pls_MSE
```

```
## [1] 0.04910368
```

PLS MSE is 0.04910368.

(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?

```
barplot(c(lm_MSE, ridge_MSE, lasso_MSE, pcr_MSE, pls_MSE),
        names.arg=c("OLS","Ridge", "Lasso", "PCR", "PLS"), main = "Test MSE",
        ylab = "Test MSE")
```



OLS, Ridge,and Pls model are pretty similar in their MSE. Lasso has a significantly high MSE. PCR stands out the most with the least MSE, so PCR could be a good model.

Problem 6.11a

11. We will now try to predict per capita crime rate in the Boston data set.

(a) Try out some of the regression methods explored in this chapter, such as best subset selection, the lasso, ridge regression, and PCR. Present and discuss results for the approaches that you consider.

```
data(Boston)

set.seed(123)
train <- sample(1:dim(Boston)[1], dim(Boston)[1]*.75, rep=FALSE)
test <- -train
b.training<- Boston[train, ]
b.testing= Boston[test, ]
```

```
sum(is.na(Boston$crim))
```

```
## [1] 0
```

```r
library(leaps)
# best subset selection by identifying the best model that contains a given number of predictors
regfit.full <- regsubsets(crim ~ ., Boston)
summary(regfit.full)
```

```
## Subset selection object
## Call: regsubsets.formula(crim ~ ., Boston)
## 13 Variables  (and intercept)
##         Forced in Forced out
## zn          FALSE      FALSE
## indus       FALSE      FALSE
## chas        FALSE      FALSE
## nox         FALSE      FALSE
## rm          FALSE      FALSE
## age         FALSE      FALSE
## dis         FALSE      FALSE
## rad         FALSE      FALSE
## tax         FALSE      FALSE
## ptratio     FALSE      FALSE
## black       FALSE      FALSE
## lstat       FALSE      FALSE
## medv        FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          zn  indus chas nox rm  age dis rad tax ptratio black lstat medv
## 1  ( 1 ) " " " "   " "  " " " " " " " " "*" " " " "     " "   " "   " " 
## 2  ( 1 ) " " " "   " "  " " " " " " " " "*" " " " "     " "   "*"   " " 
## 3  ( 1 ) " " " "   " "  " " " " " " " " "*" " " " "     "*"   "*"   " " 
## 4  ( 1 ) "*" " "   " "  " " " " " " "*" "*" " " " "     " "   " "   "*" 
## 5  ( 1 ) "*" " "   " "  " " " " " " "*" "*" " " " "     "*"   " "   "*" 
## 6  ( 1 ) "*" " "   " "  "*" " " " " "*" "*" " " " "     "*"   " "   "*" 
## 7  ( 1 ) "*" " "   " "  "*" " " " " "*" "*" " " "*"     "*"   " "   "*" 
## 8  ( 1 ) "*" " "   " "  "*" " " " " "*" "*" " " "*"     "*"   "*"   "*" 
```

```r
regfit.full <- regsubsets(crim ~ ., data = Boston,
    nvmax = 13)
reg.summary <- summary(regfit.full)
names(reg.summary)
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

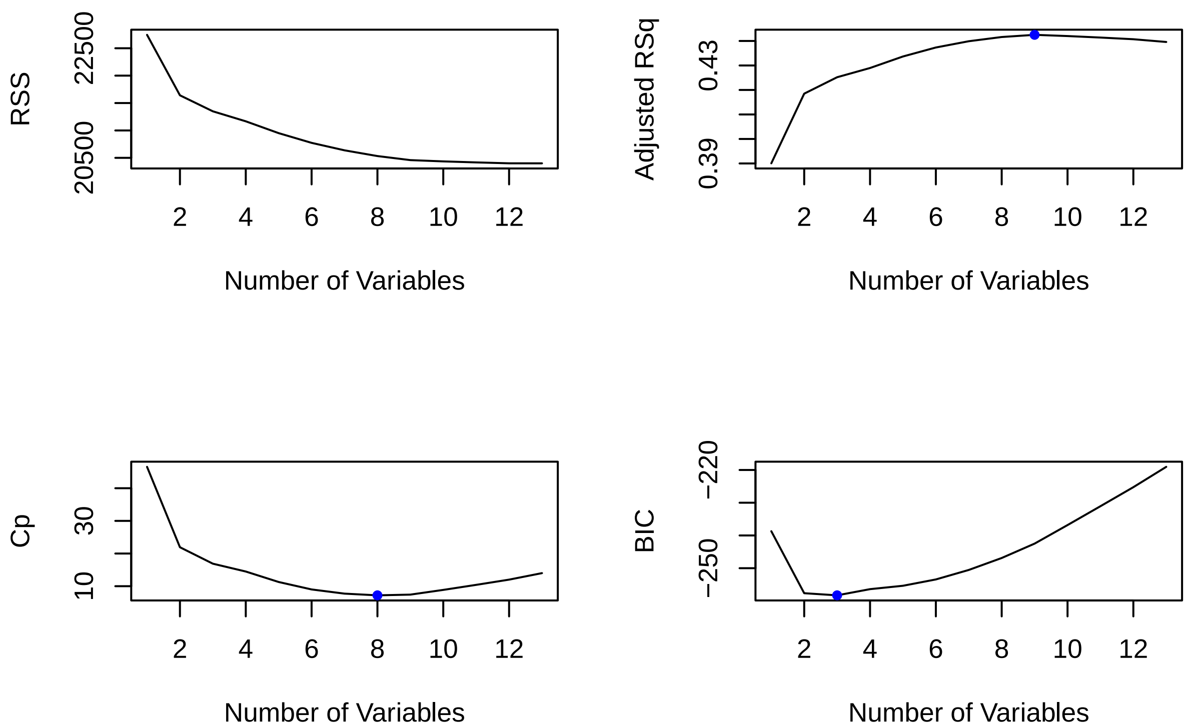
```r
reg.summary$rsq
```

```
##  [1] 0.3912567 0.4207965 0.4286123 0.4334892 0.4392738 0.4440173 0.4476594
##  [8] 0.4504606 0.4524408 0.4530572 0.4535605 0.4540031 0.4540104
```

As we can see from the R^2, it increases from 39% when its only one variable to 45% when its all variables.

```r
#Decide which model to select
par(mfrow=c(2,2))
plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")
```

```r
plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="l")
p = which.max(reg.summary$adjr2)
points(p,reg.summary$adjr2[p], col="blue", pch=20)

plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",type='l')
p = which.min(reg.summary$cp)
points(p,reg.summary$cp[p],col="blue", pch=20)

plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type='l')
p = which.min(reg.summary$bic)
points(p,reg.summary$bic[p],col="blue", pch=20)
```



Number of variables = 3, 8,9 could be used when fitting the models.

Linear Model

```r
#Linear Regression
#With 3 variables
lm.fit = lm(crim~rad+dis+black, data=b.training)
lm.pred = predict(lm.fit, b.testing, type="response")
lm_MSE <- mean((lm.pred - b.testing$crim)^2); lm_MSE
```

```
## [1] 21.04157
```

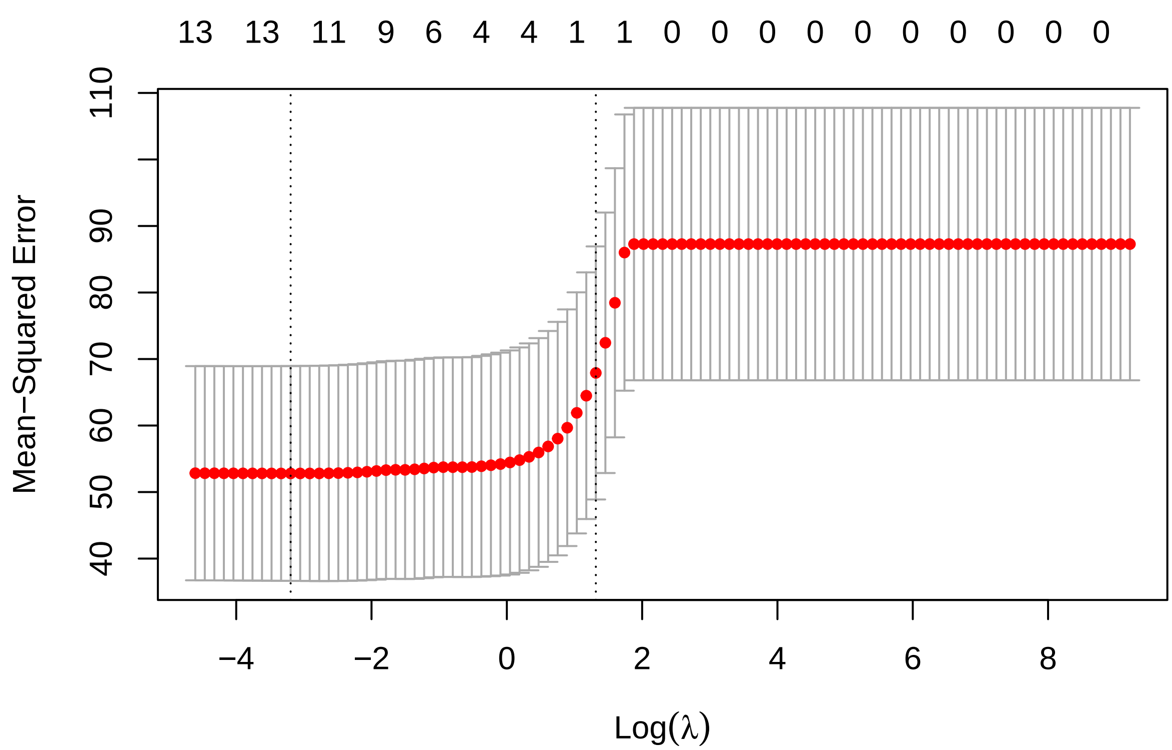
```r
#With 9 variables
lm.fit = lm(crim~rad+dis+black+lstat+medv+zn+nox+ptratio, data=b.training)
lm.pred = predict(lm.fit, b.testing, type="response")
lm_MSE <- mean((lm.pred - b.testing$crim)^2); lm_MSE
```

```
## [1] 19.14954
```

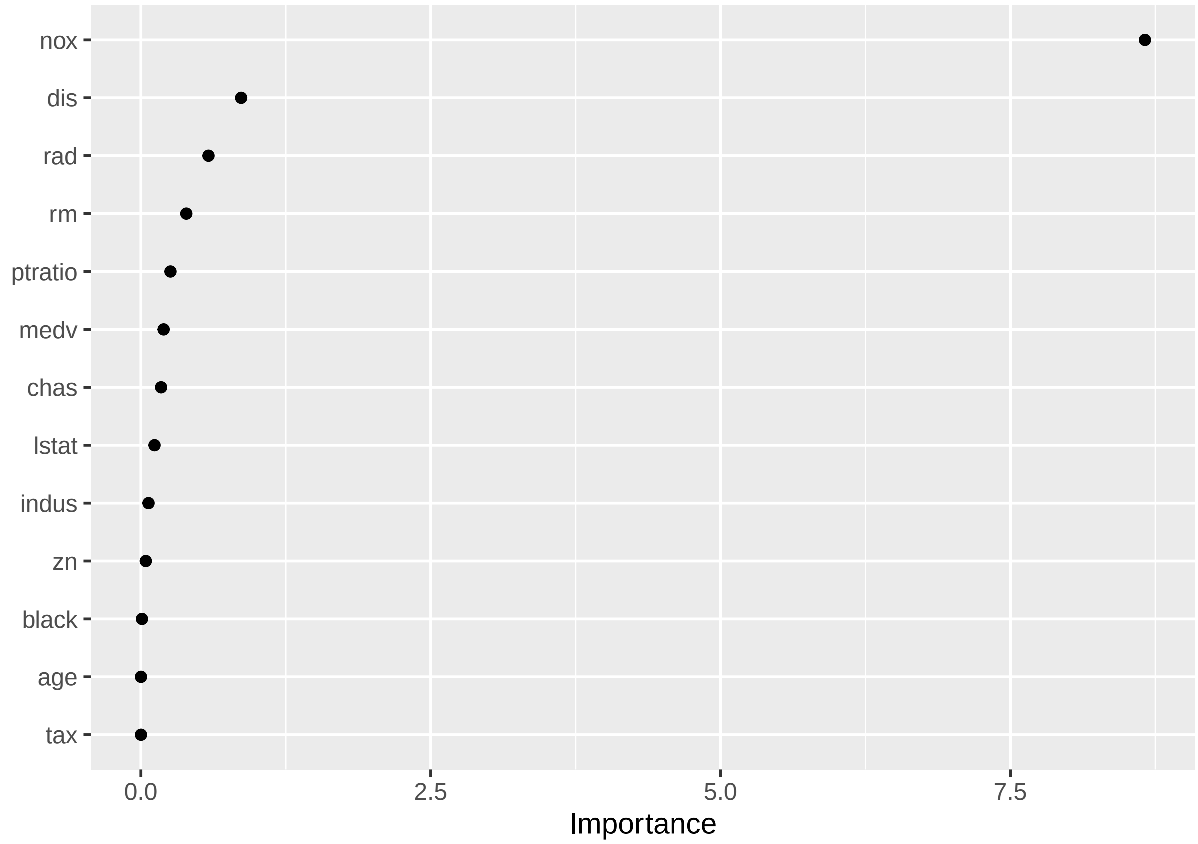
With 9 variables, the MSE only improves a little. So for the complexity of the model, I would just use three variables. MSE = 21.04

LASSO

```r
#LASSO

lambda_try <- 10^seq(-2, 4, length.out = 99)
cv_lasso = cv.glmnet(x = data.matrix(b.training[,-which(names(b.training) %in% c("crim"))]),
y=b.training$crim, alpha = 1, lambda=lambda_try,standardize = TRUE, nfolds = 10)

#choose best lambda
# Plot cross-validation results
plot(cv_lasso)
```



```r
# Best cross-validated lambda
lambda_cv <- cv_lasso$lambda.min

# Fit final model
model_lasso <- glmnet(x = data.matrix(b.training[,-which(names(b.training) %in% c("crim"))]), y=b.traini

vip(model_lasso, num_features = 30, geom = "point")
```




```r
train_control <- trainControl(method = "repeatedcv",
                              number = 5,
                              repeats = 5,
                              search = "random",
                              verboseIter = FALSE)

lasso_model     <- train(crim ~ rad+dis+black,
                       data = b.training,
                       metrics = 'RMSE',
                       method = "glmnet",
                       tuneGrid = expand.grid(alpha = 1,
                                              lambda = 1),
                       tuneLength = 25,
                       trControl = train_control)

# Predict using the testing data
lasso_pred = predict(lasso_model, newdata = b.testing)
```

```r
# Evaluate performance
postResample(pred = na.omit(lasso_pred), obs = b.testing[,'crim'])
```

```
##      RMSE  Rsquared       MAE
## 4.2008359 0.5321872 2.3963518
```

```r
lasso_MSE <- mean((lasso_pred - b.testing[, "crim"])^2); lasso_MSE
```

```
## [1] 17.64702
```

LASSO MSE is 17.64. better than OLS model.

Ridge

```r
#Ridge

model_ridge = cv.glmnet(x = data.matrix(b.training[,-which(names(b.training) %in% c("crim"))]),
y=b.training$crim, alpha = 0)
```

```r
# cross validation to train our ridge model to find the best lambda

train_control <- trainControl(method = "repeatedcv",
                              number = 5,
                              repeats = 1,
                              search = "random",
                              verboseIter = FALSE)

ridge_model     <- train(crim ~ rad+dis+black,
                       data = b.training,
                       metrics = 'RMSE',
                       method = "ridge",
                       tuneLength = 25,
                       trControl = train_control)

# Predict using the testing data
ridge_pred = predict(ridge_model, newdata = b.testing)

# Evaluate performance
postResample(pred = ridge_pred, obs = b.testing[,'crim'])
```

```
##      RMSE  Rsquared       MAE
## 4.5871199 0.5436074 2.7559918
```

```r
ridge_MSE <- mean((ridge_pred - b.testing[, "crim"])^2); ridge_MSE
```

```
## [1] 21.04167
```

Ridge yields a MSE of 21.0417, which is worse than LASSO.

(b) Propose a model (or set of models) that seem to perform well on this data set, and justify your answer. Make sure that you are evaluating model performance using validation set error, cross validation,or some other reasonable alternative, as opposed to using training error.

Using cross validation and compare the MSE results of the linear, LASSO, an Ridge model, I conclude that LASSO is the best model based on the lowest MSE as well as RMSE.

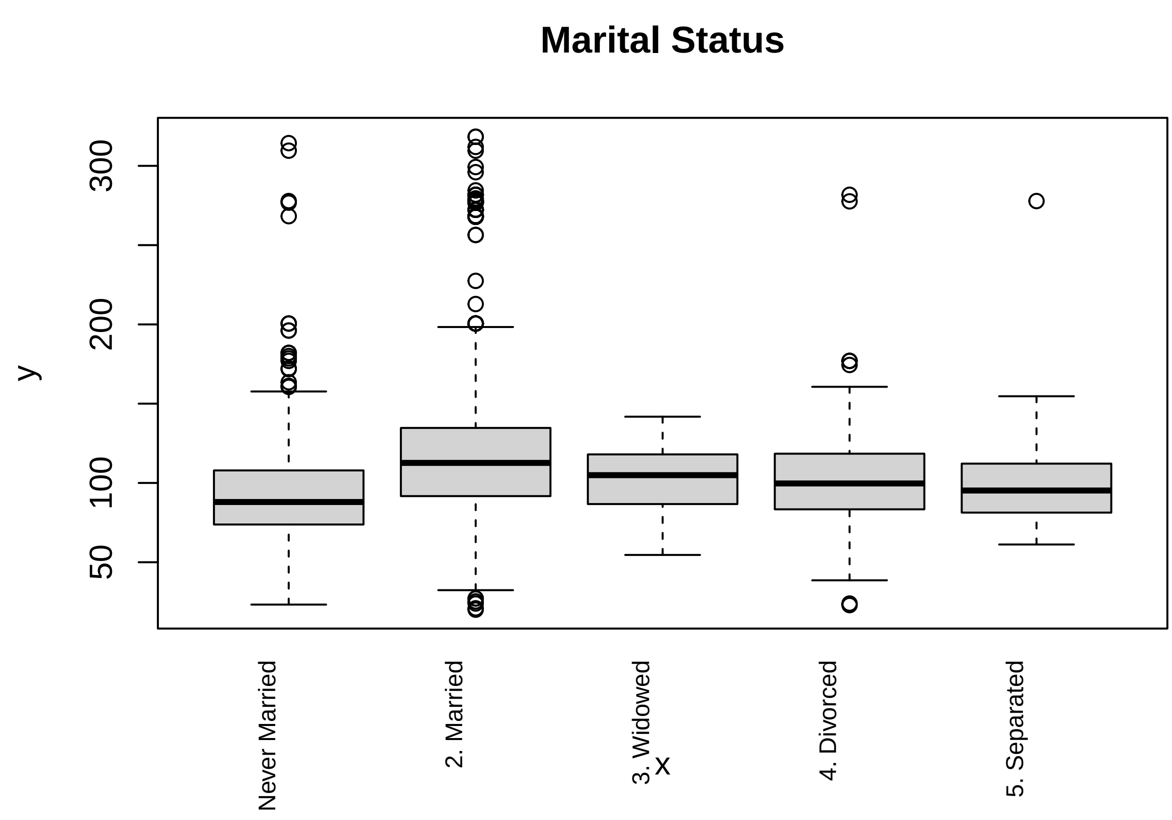
(c) Does your chosen model involve all of the features in the data set? Why or why not?

No, it doesn't. I choose the best subset of the variables which contains 3 variables from the data Boston.

Problem 7.7a

The Wage data set contains a number of other features not explored in this chapter, such as marital status (maritl), job class (jobclass), and others. Explore the relationships between some of these other predictors and wage, and use non-linear fitting techniques in order to fit flexible models to the data. Create plots of the results obtained, and write a summary of your findings.

```
data(Wage)

plot(Wage$maritl,Wage$wage, main="Marital Status", xaxt="n")
text(1:5, par("usr")[3] - 20, labels = levels(Wage$maritl),  cex = 0.75, srt = 90, pos = 2, xpd = TRUE)
```



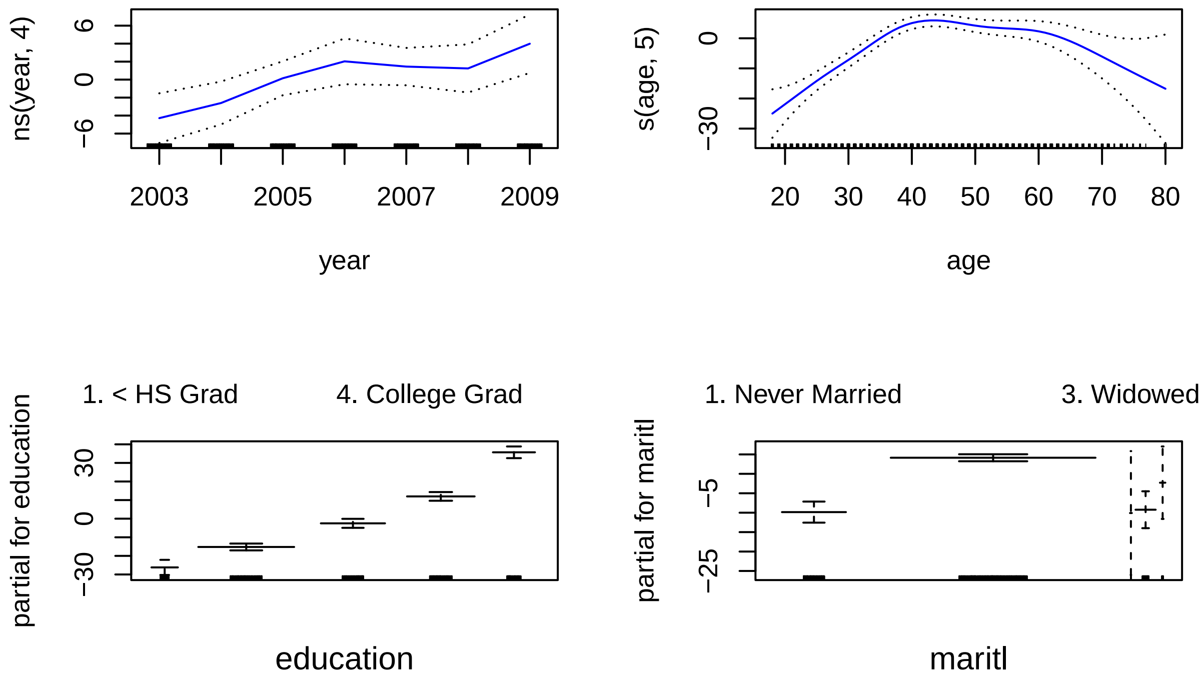
```r
plot(Wage$jobclass, Wage$wage, main="Job Class")
```

People that are married and work in the information job class tend to have a higher wage.

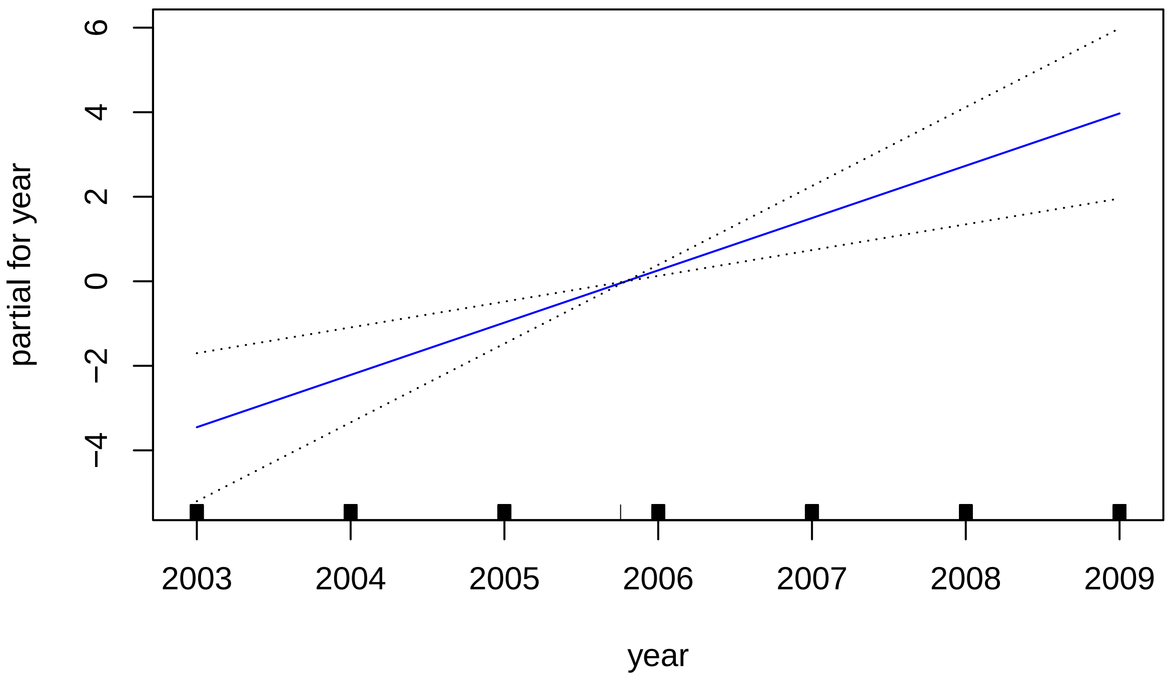
```r
gam1 <- gam(wage ~ ns(year, 4) + s(age, 5) + education, data = Wage)
gam2 <- gam(wage ~ ns(year, 4) + s(age, 5) + education + jobclass, data = Wage)
gam3 <- gam(wage ~ ns(year, 4) + s(age, 5) + education + maritl, data = Wage)
gam4 <- gam(wage ~ ns(year, 4) + s(age, 5) + education + jobclass + maritl, data = Wage)
anova(gam1, gam2, gam3, gam4)
```

```
## Analysis of Deviance Table
##
## Model 1: wage ~ ns(year, 4) + s(age, 5) + education
## Model 2: wage ~ ns(year, 4) + s(age, 5) + education + jobclass
## Model 3: wage ~ ns(year, 4) + s(age, 5) + education + maritl
## Model 4: wage ~ ns(year, 4) + s(age, 5) + education + jobclass + maritl
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1      2986    3690853
## 2      2985    3678665  1    12188 0.0014496 **
## 3      2982    3596496  3    82169 9.531e-15 ***
## 4      2981    3582643  1    13852 0.0006863 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

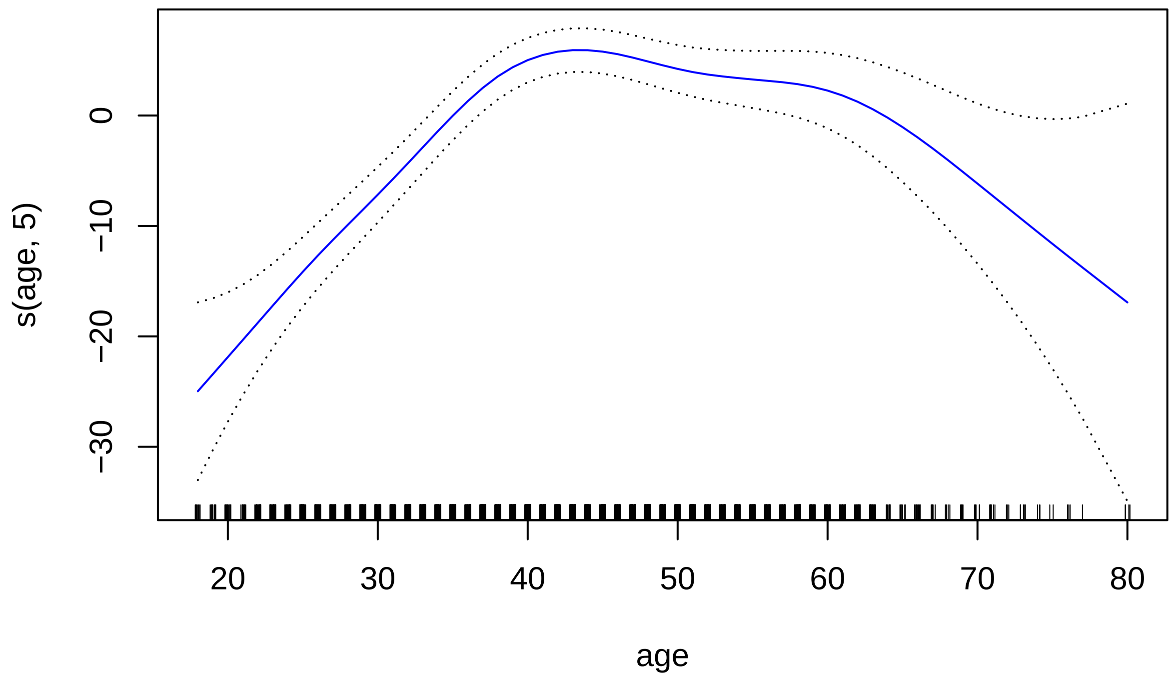
```r
par(mfrow = c(2, 2))
plot.Gam(gam3, se = TRUE, col = "blue")
```



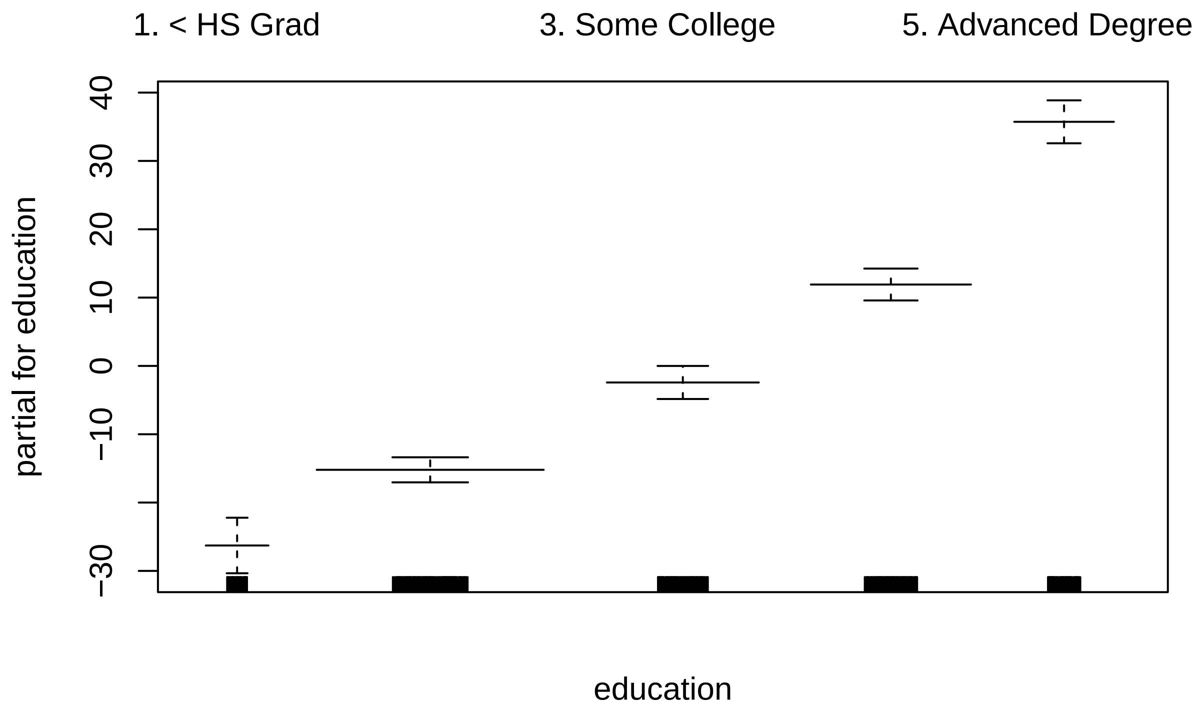

```r
gam <- gam(wage ~ year + s(age, 5) + education + maritl, data = Wage)

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

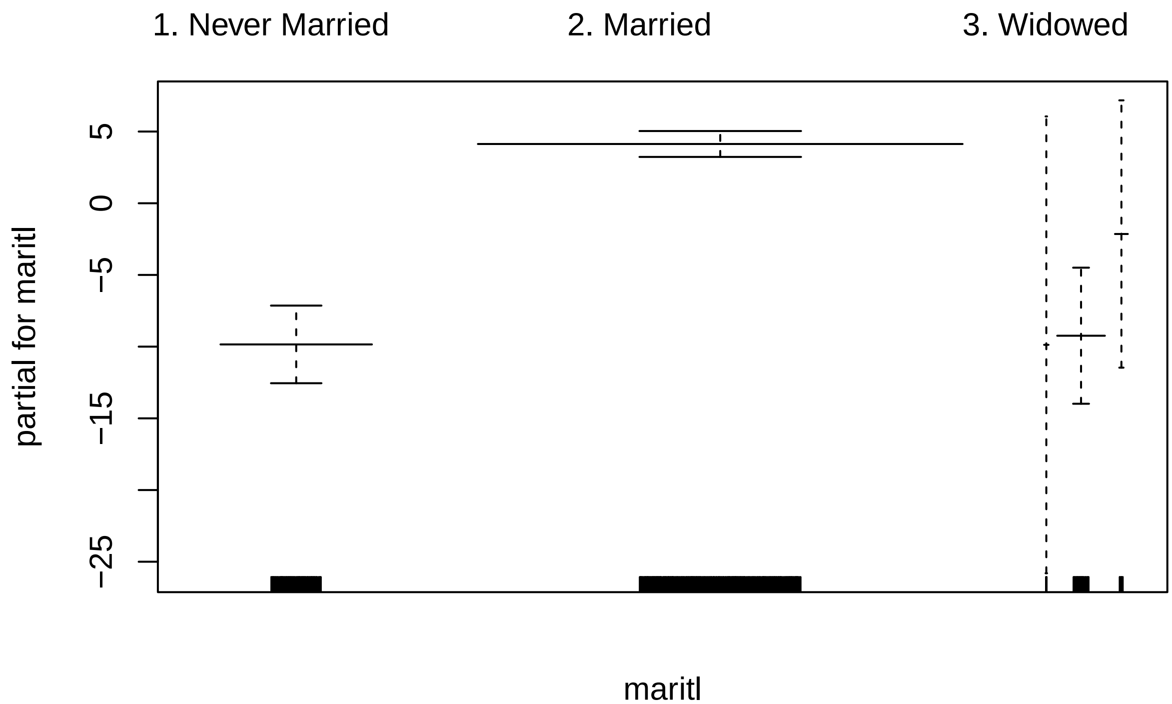


6
4
2
0
−2
−4
partial for year
2003
2004
2005
2006
2007
2008
2009
year



0
−10
−20
−30
s(age, 5)
20
30
40
50
60
70
80
age


1. < HS Grad
3. Some College
5. Advanced Degree
partial for education
40
30
20
10
0
−10
−30
education



Year looks linear from gam3, so we can probably use linear variable for year. We can see that wage increases with years and education, and generally with age until 50.

Problem 7.9a

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.

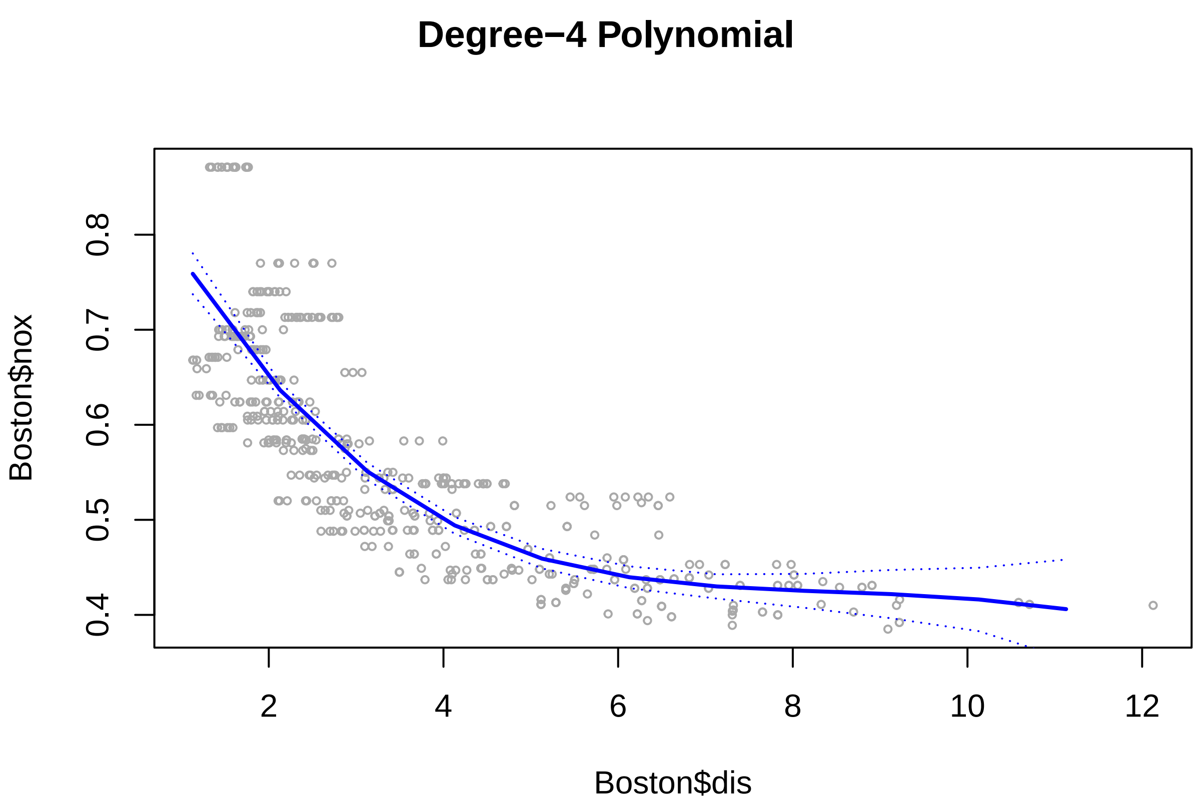
(a) Use the poly() function to fit a cubic polynomial regression to predict nox using dis. Report the regression output, and plot the resulting data and polynomial fits.

```
fit <- lm(nox ~ poly(dis, 4), data = Boston)
coef(summary(fit))
```

```
##                  Estimate  Std. Error     t value      Pr(>|t|)
## (Intercept)    0.55469506 0.002761339 200.8790240  0.000000e+00
## poly(dis, 4)1 -2.00309590 0.062114782 -32.2482963 2.540459e-124
## poly(dis, 4)2  0.85632995 0.062114782  13.7862506  6.924872e-37
## poly(dis, 4)3 -0.31804899 0.062114782  -5.1203430  4.356581e-07
## poly(dis, 4)4  0.03354668 0.062114782   0.5400757  5.893848e-01
```

```r
#We now create a grid of values for dis at which we want predictions, and then call the generic predict

dislims <- range(Boston$dis)
dis.grid <- seq(from = dislims[1], to = dislims[2])
preds <- predict(fit, newdata = list(dis = dis.grid),
    se = TRUE)
se.bands <- cbind(preds$fit + 2 * preds$se.fit,
    preds$fit - 2 * preds$se.fit)
```

```r
par( mar = c(4.5, 4.5, 1, 1),
    oma = c(0, 0, 4, 0))
plot(Boston$dis, Boston$nox, xlim = dislims, cex = .5, col = "darkgrey")
title("Degree-4 Polynomial", outer = T)
lines(dis.grid, preds$fit, lwd = 2, col = "blue")
matlines(dis.grid, se.bands, lwd = 1, col = "blue", lty = 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.

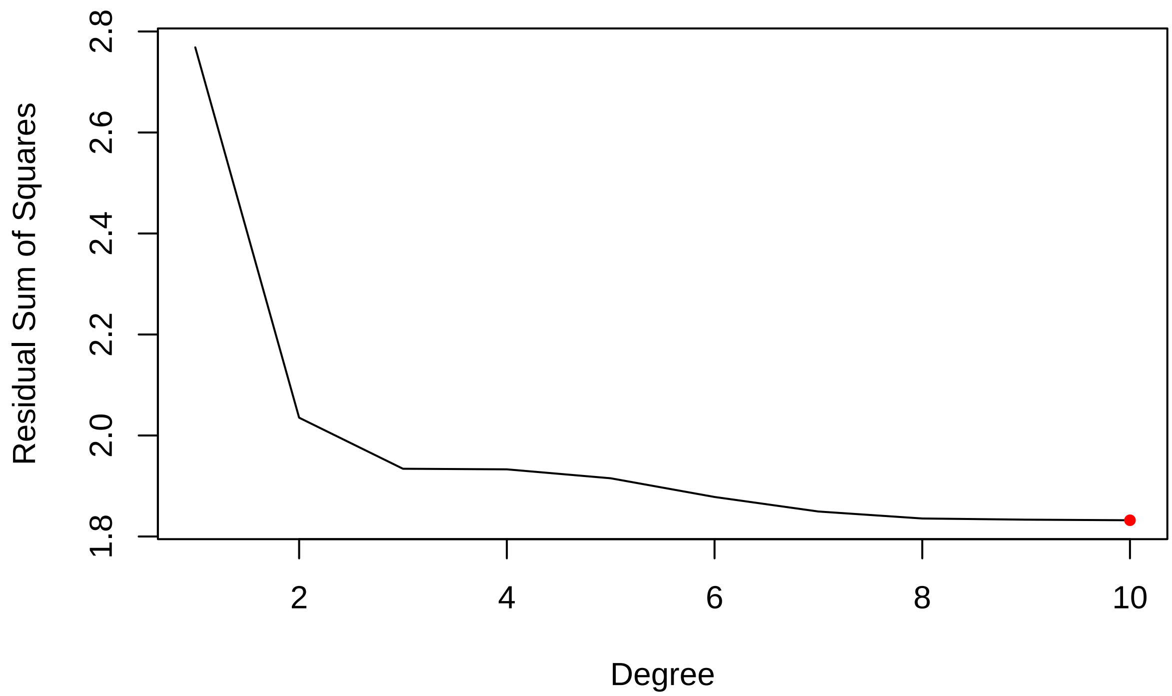
```r
#create a for loop
residss <- rep(NA, 10)
for (i in 1:10) {
  fit <- lm(nox ~ poly(dis, i), data = Boston)
  residss[i] <- sum(fit$residuals ^ 2)
```

```
  print(anova(lm(nox ~ poly(dis, i), data = Boston)))
}
```

```
## Analysis of Variance Table
##
## Response: nox
##                 Df Sum Sq Mean Sq F value    Pr(>F)
## poly(dis, i)   1 4.0124  4.0124  730.43 < 2.2e-16 ***
## Residuals    504 2.7686  0.0055
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
##
## Response: nox
##                 Df Sum Sq Mean Sq F value    Pr(>F)
## poly(dis, i)   2 4.7457 2.37285  586.43 < 2.2e-16 ***
## Residuals    503 2.0353 0.00405
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
##
## Response: nox
##                 Df Sum Sq Mean Sq F value    Pr(>F)
## poly(dis, i)   3 4.8468 1.61562  419.34 < 2.2e-16 ***
## Residuals    502 1.9341 0.00385
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
##
## Response: nox
##                 Df Sum Sq Mean Sq F value    Pr(>F)
## poly(dis, i)   4  4.848 1.21199  314.13 < 2.2e-16 ***
## Residuals    501  1.933 0.00386
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
##
## Response: nox
##                 Df Sum Sq Mean Sq F value    Pr(>F)
## poly(dis, i)   5 4.8657 0.97313  254.04 < 2.2e-16 ***
## Residuals    500 1.9153 0.00383
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
##
## Response: nox
##                 Df Sum Sq Mean Sq F value    Pr(>F)
## poly(dis, i)   6 4.9027 0.81712  217.08 < 2.2e-16 ***
## Residuals    499 1.8783 0.00376
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
```

```
##
## Response: nox
##                Df Sum Sq Mean Sq F value    Pr(>F)
## poly(dis, i)   7 4.9315 0.70450   189.7 < 2.2e-16 ***
## Residuals    498 1.8495 0.00371
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
##
## Response: nox
##                Df Sum Sq Mean Sq F value    Pr(>F)
## poly(dis, i)   8 4.9453 0.61817  167.37 < 2.2e-16 ***
## Residuals    497 1.8356 0.00369
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
##
## Response: nox
##                Df Sum Sq Mean Sq F value    Pr(>F)
## poly(dis, i)   9 4.9476 0.54974  148.73 < 2.2e-16 ***
## Residuals    496 1.8333 0.00370
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
##
## Response: nox
##                Df Sum Sq Mean Sq F value    Pr(>F)
## poly(dis, i)  10 4.9488 0.49488   133.7 < 2.2e-16 ***
## Residuals    495 1.8322 0.00370
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
plot(1:10, residss, type = 'l', xlab = "Degree", ylab = "Residual Sum of Squares")
points(which.min(residss),residss[which.min(residss)], col="red", pch=20)
```




```
fit2 <- lm(nox ~ poly(dis, 2), data = Boston)
fit3 <- lm(nox ~ poly(dis, 3), data = Boston)
fit4 <- lm(nox ~ poly(dis, 4), data = Boston)
fit5 <- lm(nox ~ poly(dis, 5), data = Boston)
fit6 <- lm(nox ~ poly(dis, 6), data = Boston)
anova(fit2,fit3,fit4,fit5,fit6)
```

```
## Analysis of Variance Table
##
## Model 1: nox ~ poly(dis, 2)
## Model 2: nox ~ poly(dis, 3)
## Model 3: nox ~ poly(dis, 4)
## Model 4: nox ~ poly(dis, 5)
## Model 5: nox ~ poly(dis, 6)
##   Res.Df    RSS Df Sum of Sq       F   Pr(>F)
## 1    503 2.0353
## 2    502 1.9341  1  0.101155 26.8741 3.16e-07 ***
## 3    501 1.9330  1  0.001125  0.2990  0.58477
## 4    500 1.9153  1  0.017691  4.7001  0.03063 *
## 5    499 1.8783  1  0.037033  9.8385  0.00181 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

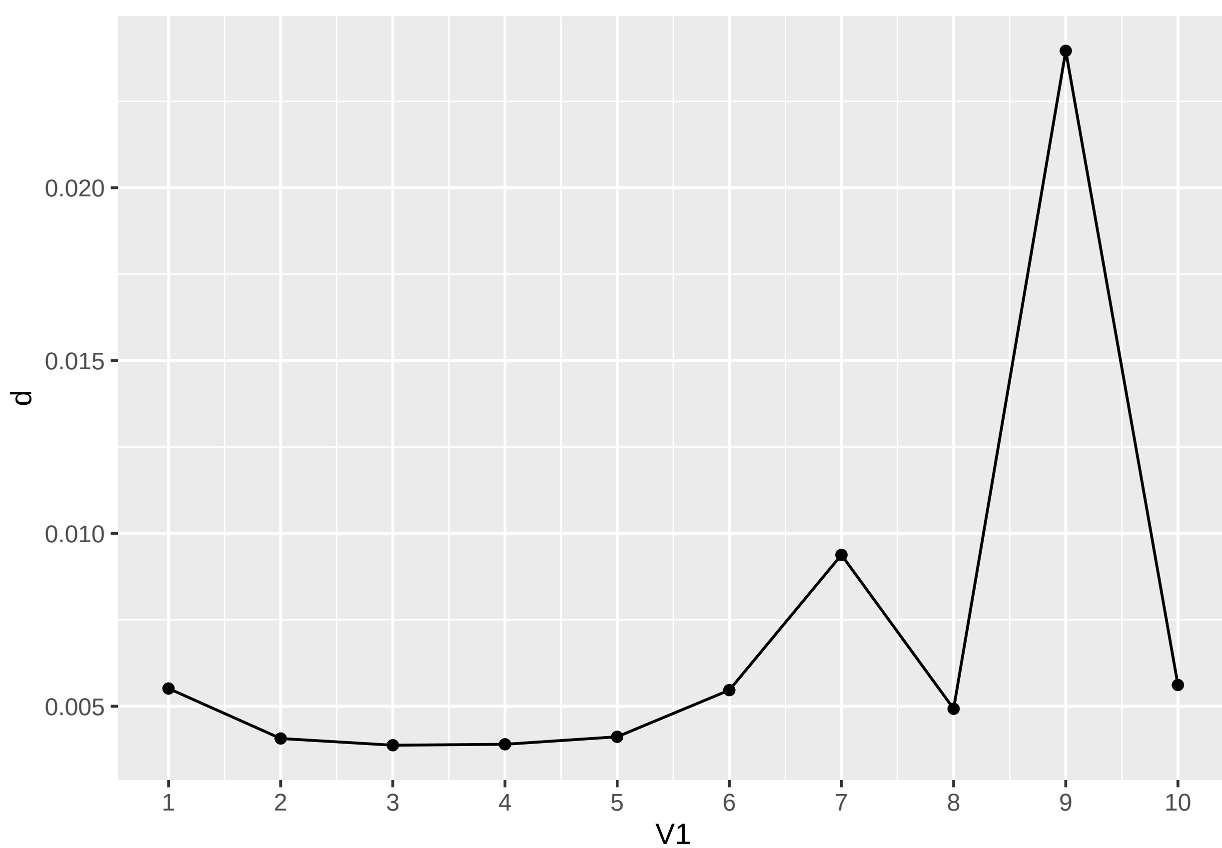
Cubic polynomial is the best fit.

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

```
library(boot)
```

```
##
## Attaching package: 'boot'

## The following object is masked from 'package:lattice':
##
##     melanoma

## The following object is masked from 'package:psych':
##
##     logit
```

```
#corss validation
d <- rep(NA,10)
for (i in 1:10) {
    glm.fit = glm(nox ~ poly(dis, i), data = Boston)
    d[i] = cv.glm(Boston, glm.fit, K = 10)$delta[2]
}

cv.plot <- data.table(seq(1:10),d,keep.rownames = TRUE)
ggplot(cv.plot, aes(V1,d)) + geom_point() + geom_line() +
  scale_x_continuous(breaks = c(0:10))
```



```
which.min(d)
```

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

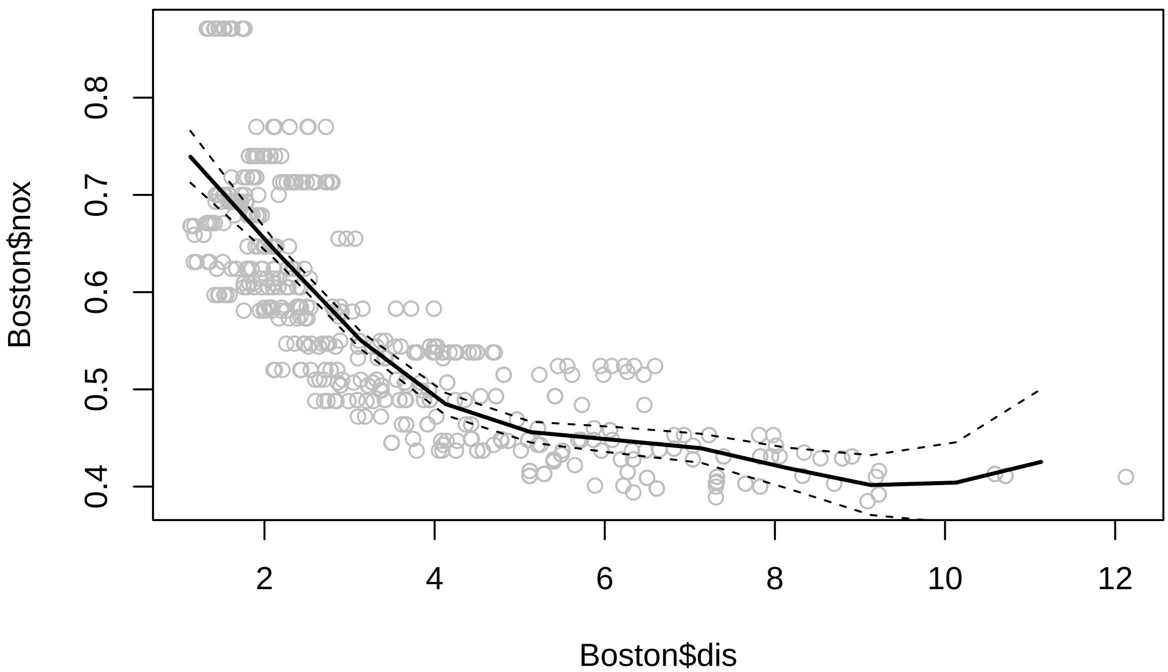
4 degrees of polynomial appears to be the best selection.

(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.

```
range(Boston$dis)
```

```
## [1]  1.1296 12.1265
```

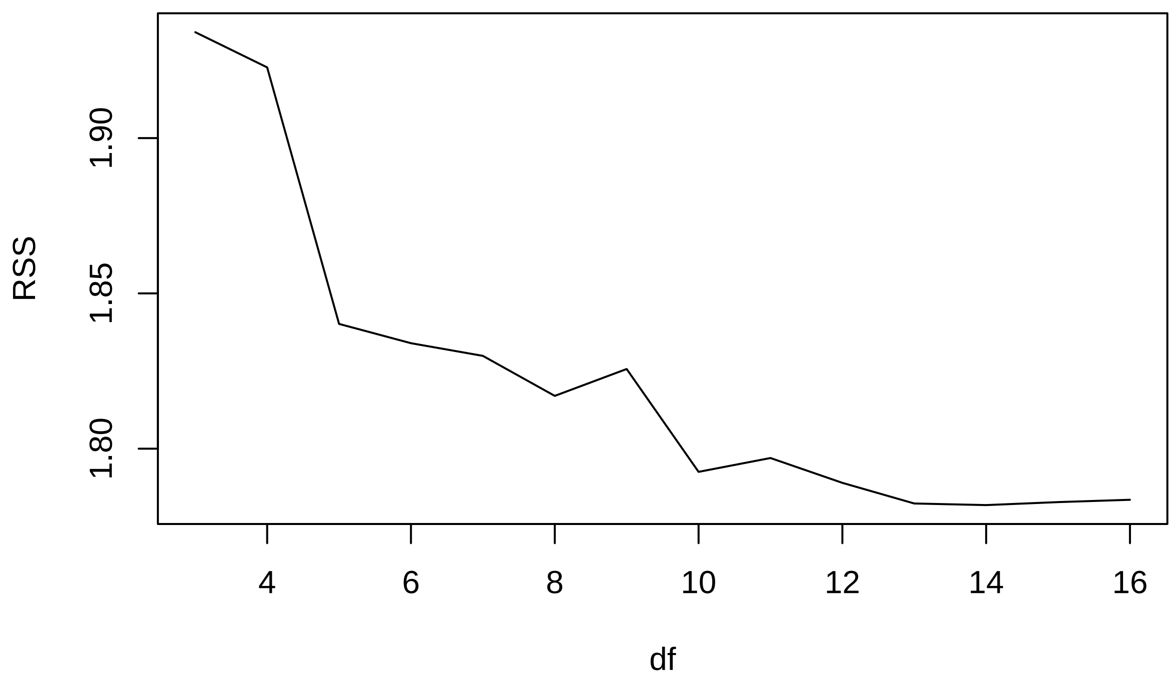
```
spline.fit <- lm(nox ~ bs(dis, df=4, knots=c(4,7,10)),data=Boston)
spline.pred <- predict(spline.fit, newdata = list(dis = dis.grid), se = T)
plot(Boston$dis, Boston$nox, col = "gray")
lines(dis.grid, spline.pred$fit, lwd = 2)
lines(dis.grid, spline.pred$fit + 2 * spline.pred$se, lty = "dashed")
lines(dis.grid, spline.pred$fit - 2 * spline.pred$se, lty = "dashed")
```



The range of dis is approximately 1 to 13, so we set the knots at the points where we can split the data equally.

(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.

```r
cv.range <- 3:16
res<- c()
for (i in cv.range) {
  fit <- lm(nox ~ bs(dis, df = i), data = Boston)
  res <- c(res, sum(fit$residuals ^ 2))
}
plot(cv.range, res, type = 'l', xlab = 'df', ylab = 'RSS')
```



The plot shows that 10 degrees of freedom is the optimal choice.

(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.

```r
library(boot)
#corss validation
cv.range <- 3:16
mse <- rep(NA,10)
res<- c()
for (i in cv.range) {
    glm.fit = glm(nox ~ bs(dis,df= i), data = Boston)
    mse = cv.glm(Boston, glm.fit, K = 10)$delta[1]
    res <- c(res, mse)
}
```

```
## Warning in bs(dis, degree = 3L, knots = numeric(0), Boundary.knots = c(1.137, :
```

```
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = numeric(0), Boundary.knots = c(1.137, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = numeric(0), Boundary.knots = c(1.1296, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = numeric(0), Boundary.knots = c(1.1296, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`50%` = 3.1121), Boundary.knots =
## c(1.137, : some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`50%` = 3.1121), Boundary.knots =
## c(1.137, : some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`50%` = 3.1523), Boundary.knots =
## c(1.1296, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = c(`50%` = 3.1523), Boundary.knots =
## c(1.1296, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = c(`33.33333%` = 2.3817, `66.66667%`
## = 4.25363333333333: some 'x' values beyond boundary knots may cause ill-
## conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`33.33333%` = 2.3817, `66.66667%`
## = 4.25363333333333: some 'x' values beyond boundary knots may cause ill-
## conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`33.33333%` = 2.41703333333333, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`33.33333%` = 2.41703333333333, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`25%` = 2.1223, `50%` = 3.2721, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`25%` = 2.1223, `50%` = 3.2721, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`20%` = 1.9312, `40%` = 2.61486, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`20%` = 1.9312, `40%` = 2.61486, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases
```

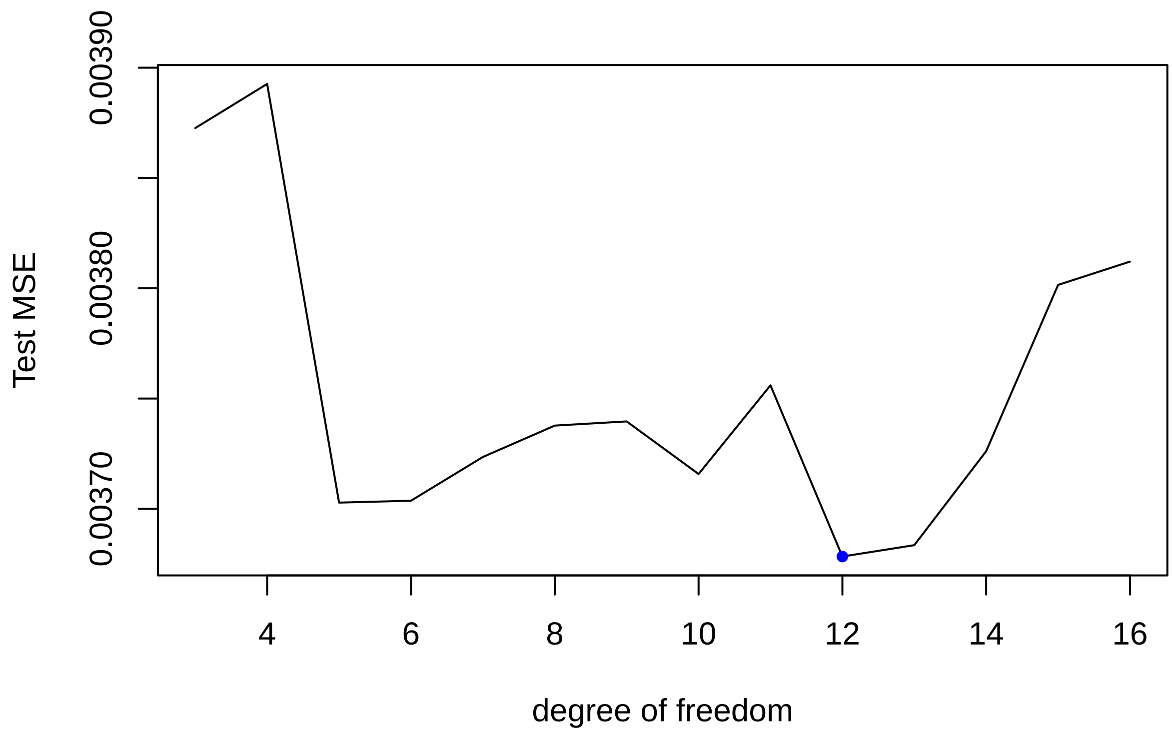
```
## Warning in bs(dis, degree = 3L, knots = c(‘20%‘ = 1.9769, ‘40%‘ = 2.72062, :
## some ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘20%‘ = 1.9769, ‘40%‘ = 2.72062, :
## some ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘16.66667%‘ = 1.83066666666667, : some
## ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘16.66667%‘ = 1.83066666666667, : some
## ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘16.66667%‘ = 1.86223333333333, : some
## ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘16.66667%‘ = 1.86223333333333, : some
## ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘14.28571%‘ = 1.7912, ‘28.57143%‘ =
## 2.2004, : some ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘14.28571%‘ = 1.7912, ‘28.57143%‘ =
## 2.2004, : some ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘14.28571%‘ = 1.79078571428571, : some
## ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘14.28571%‘ = 1.79078571428571, : some
## ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘12.5%‘ = 1.7275, ‘25%‘ = 2.08285, :
## some ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘12.5%‘ = 1.7275, ‘25%‘ = 2.08285, :
## some ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘12.5%‘ = 1.748425, ‘25%‘ = 2.1038, :
## some ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘12.5%‘ = 1.748425, ‘25%‘ = 2.1038, :
## some ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘11.11111%‘ = 1.73271111111111, : some
## ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘11.11111%‘ = 1.73271111111111, : some
## ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘10%‘ = 1.6634, ‘20%‘ = 1.9784, : some
## ’x’ values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(‘10%‘ = 1.6634, ‘20%‘ = 1.9784, : some
## ’x’ values beyond boundary knots may cause ill-conditioned bases
```

```
## Warning in bs(dis, degree = 3L, knots = c(`10%` = 1.6311, `20%` = 1.9512, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`10%` = 1.6311, `20%` = 1.9512, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`9.090909%` = 1.61450909090909, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`9.090909%` = 1.61450909090909, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`9.090909%` = 1.61097272727273, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`9.090909%` = 1.61097272727273, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`8.333333%` = 1.58948333333333, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`8.333333%` = 1.58948333333333, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`8.333333%` = 1.61698333333333, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`8.333333%` = 1.61698333333333, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`7.692308%` = 1.57836153846154, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`7.692308%` = 1.57836153846154, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`7.692308%` = 1.5804, `15.38462%` =
## 1.8195, : some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`7.692308%` = 1.5804, `15.38462%` =
## 1.8195, : some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`7.142857%` = 1.54201428571429, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`7.142857%` = 1.54201428571429, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`7.142857%` = 1.54498571428571, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`7.142857%` = 1.54498571428571, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases
```

```r
which.min(res)
```

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

```r
plot(cv.range, res, type = 'l', xlab = 'degree of freedom', ylab = 'Test MSE')
points(which.min(res)+2, res[which.min(res)], col = 'blue', pch = 20)
```



It shows that 10 degrees of freedom is the optimal choice.

Problem 7.10a

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.

```r
library(leaps)
data(College)

set.seed(12345678)
train <- sample(1:dim(College)[1], dim(College)[1]*.75, rep=FALSE)
test <- -train
```

```r
c.training<- College[train, ]
c.testing= College[test, ]
```

```r
f.atrix <- model.matrix(Outstate ~ ., data=c.training)
forward.step.fit <- regsubsets(Outstate ~ ., data = College, subset = f.atrix, method = 'forward')
summary(forward.step.fit)
```

```
## Subset selection object
## Call: regsubsets.formula(Outstate ~ ., data = College, subset = f.atrix,
##     method = "forward")
## 17 Variables  (and intercept)
##             Forced in Forced out
## PrivateYes      FALSE      FALSE
## Apps            FALSE      FALSE
## Accept          FALSE      FALSE
## Enroll          FALSE      FALSE
## Top10perc       FALSE      FALSE
## Top25perc       FALSE      FALSE
## F.Undergrad     FALSE      FALSE
## P.Undergrad     FALSE      FALSE
## Room.Board      FALSE      FALSE
## Books           FALSE      FALSE
## Personal        FALSE      FALSE
## PhD             FALSE      FALSE
## Terminal        FALSE      FALSE
## S.F.Ratio       FALSE      FALSE
## perc.alumni     FALSE      FALSE
## Expend          FALSE      FALSE
## Grad.Rate       FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: forward
##          PrivateYes Apps Accept Enroll Top10perc Top25perc F.Undergrad
## 1  ( 1 ) " "        " "  " "    " "    " "       " "       " "
## 2  ( 1 ) " "        " "  " "    " "    " "       " "       " "
## 3  ( 1 ) " "        " "  " "    " "    " "       " "       " "
## 4  ( 1 ) " "        " "  " "    " "    " "       " "       " "
## 5  ( 1 ) " "        " "  " "    " "    " "       " "       "*"
## 6  ( 1 ) " "        " "  "*"    " "    " "       " "       "*"
## 7  ( 1 ) " "        " "  "*"    " "    " "       "*"       "*"
## 8  ( 1 ) " "        " "  "*"    " "    " "       "*"       "*"
##          P.Undergrad Room.Board Books Personal PhD Terminal S.F.Ratio
## 1  ( 1 ) " "         " "        " "   " "      " " " "      " "
## 2  ( 1 ) " "         "*"        " "   " "      " " " "      " "
## 3  ( 1 ) " "         "*"        " "   " "      " " " "      " "
## 4  ( 1 ) " "         "*"        " "   " "      " " " "      " "
## 5  ( 1 ) " "         "*"        " "   " "      " " " "      " "
## 6  ( 1 ) " "         "*"        " "   " "      " " " "      " "
## 7  ( 1 ) " "         "*"        " "   " "      " " " "      " "
## 8  ( 1 ) " "         "*"        " "   "*"      " " " "      " "
##          perc.alumni Expend Grad.Rate
## 1  ( 1 ) " "         "*"    " "
## 2  ( 1 ) " "         "*"    " "
## 3  ( 1 ) "*"         "*"    " "
```

```
## 4  ( 1 ) "*"          "*"     "*"
## 5  ( 1 ) "*"          "*"     "*"
## 6  ( 1 ) "*"          "*"     "*"
## 7  ( 1 ) "*"          "*"     "*"
## 8  ( 1 ) "*"          "*"     "*"
```

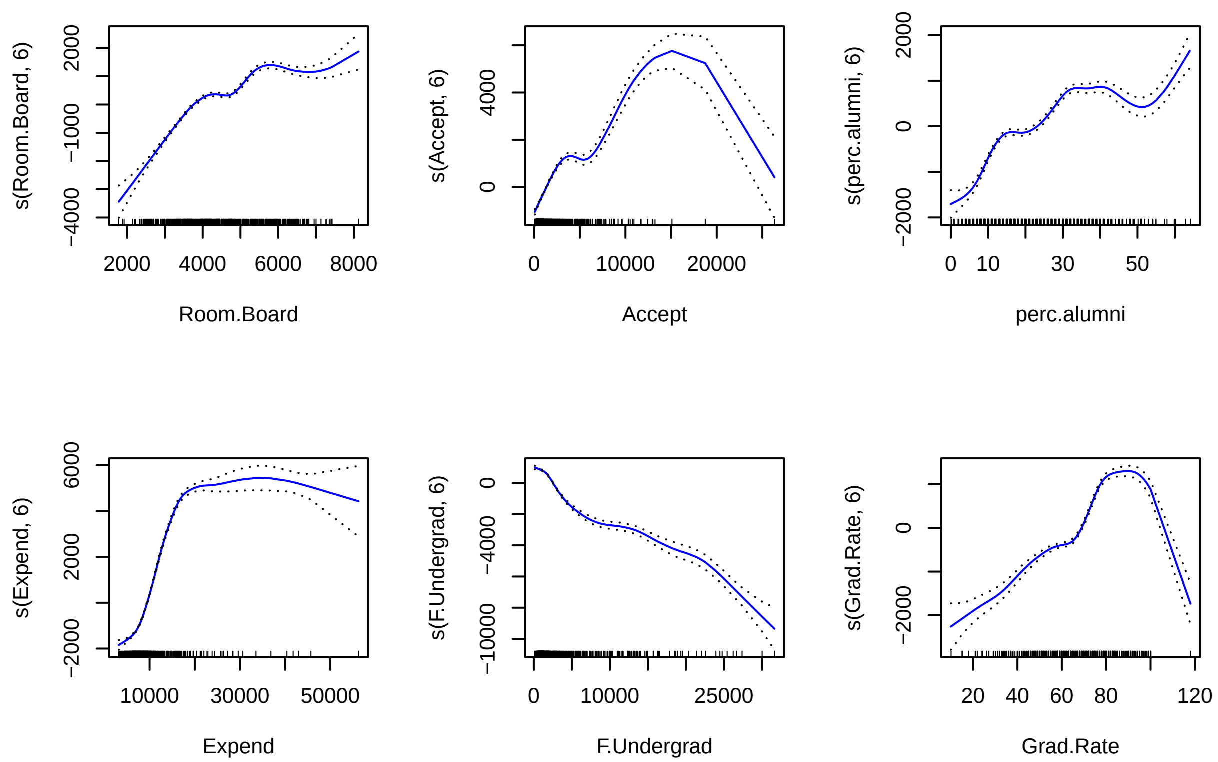
Accept, F.Undergrad, Room.Board, perc.alumni, Expend, Grad.Rate are statistically significant.

```r
coef(forward.step.fit, id = 6)
```

```
##   (Intercept)        Accept    F.Undergrad     Room.Board    perc.alumni
## -1519.0812446     0.4349844     -0.3205914      1.2292685     75.2468284
##        Expend     Grad.Rate
##     0.2752600    44.1710397
```

(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.

```r
gam.c1 <- gam(Outstate ~ s(Room.Board, 6) + s(Accept, 6) + s(perc.alumni, 6) + s(Expend, 6) + s(F.Underg
par(mfrow = c(2,3))
plot(gam.c1, se = TRUE, col = 'blue')
```



The fit curves shows that all the variables are have pretty good fit.

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

```r
gam.pred <- predict(gam.c1, c.testing)
RSS <- sum((c.testing$Outstate - gam.pred)^2) # based on equation (3.16)
TSS <- sum((c.testing$Outstate - mean(c.testing$Outstate)) ^ 2)
1 - (RSS / TSS)  # R-squared
```

```
## [1] 0.7940107
```

```r
gam_MSE <- mean((gam.pred - c.testing$Outstate)^2); gam_MSE
```

```
## [1] 3629447
```

It yields an MSE of 3622744. The R-squared is 0.79, which is not too bad.

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

```r
summary(gam.c1)
```

```
##
## Call: gam(formula = Outstate ~ s(Room.Board, 6) + s(Accept, 6) + s(perc.alumni,
##     6) + s(Expend, 6) + s(F.Undergrad, 6) + s(Grad.Rate, 6),
##     data = College, subset = f.atrix)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -7779.44  -834.79    35.89   894.31  6044.19
##
## (Dispersion Parameter for gaussian family taken to be 2443131)
##
##     Null Deviance: 111942903672 on 6991 degrees of freedom
## Residual Deviance: 16991978547 on 6955 degrees of freedom
## AIC: 122725.2
## 3316 observations deleted due to missingness
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##                      Df     Sum Sq    Mean Sq  F value    Pr(>F)
## s(Room.Board, 6)      1 3.5205e+10 3.5205e+10 14409.83 < 2.2e-16 ***
## s(Accept, 6)          1 3.6365e+08 3.6365e+08   148.84 < 2.2e-16 ***
## s(perc.alumni, 6)     1 1.4306e+10 1.4306e+10  5855.51 < 2.2e-16 ***
## s(Expend, 6)          1 1.5021e+10 1.5021e+10  6148.35 < 2.2e-16 ***
## s(F.Undergrad, 6)     1 4.4552e+09 4.4552e+09  1823.55 < 2.2e-16 ***
## s(Grad.Rate, 6)       1 1.9477e+09 1.9477e+09   797.22 < 2.2e-16 ***
## Residuals          6955 1.6992e+10 2.4431e+06
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                    Npar Df Npar F     Pr(F)
## (Intercept)
## s(Room.Board, 6)         5  87.02 < 2.2e-16 ***
## s(Accept, 6)             5  81.50 < 2.2e-16 ***
```

```
## s(perc.alumni, 6)       5  81.91 < 2.2e-16 ***
## s(Expend, 6)            5 410.73 < 2.2e-16 ***
## s(F.Undergrad, 6)       5  89.96 < 2.2e-16 ***
## s(Grad.Rate, 6)         5 126.13 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There is a non-linear relationship for all the variables in gam.c1.