DS502- HW4

Mahdi Alouane and Rahul Pande

1. (10 points) Section 6.8, page 259, question 2

(a) iii. holds true. Lasso puts a budget constraint on the parameters which decreases the model variance and it reduces overfitting. However, when we put a constraint, the model bias increases. From the bias-variance trade-off concept we can say that the lasso regression will give better prediction when its increase in bias is less that its decrease in variance.

(b) iii. holds true. Like Lasso above, Ridge also puts a budget constraint on the parameters which decreases the model variance and it reduces overfitting. However, when we put a constraint, the model bias increases. Similarly as above, from the bias-variance trade-off concept we can say that the Ridge regression will give better prediction when its increase in bias is less that its decrease in variance.

(c) ii. holds true. Since non-linear methods are more flexible, they have higher variance than least squares regression but lower bias. Again, from the bias variance trade-off, if the increase on model variance is less than the decrease in bias, then non-linear model will have better prediction accuracy.

2. (20 points) Section 6.8, page264, question 11

(a)

```
library(MASS)
library(leaps)
library(glmnet)
```

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

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

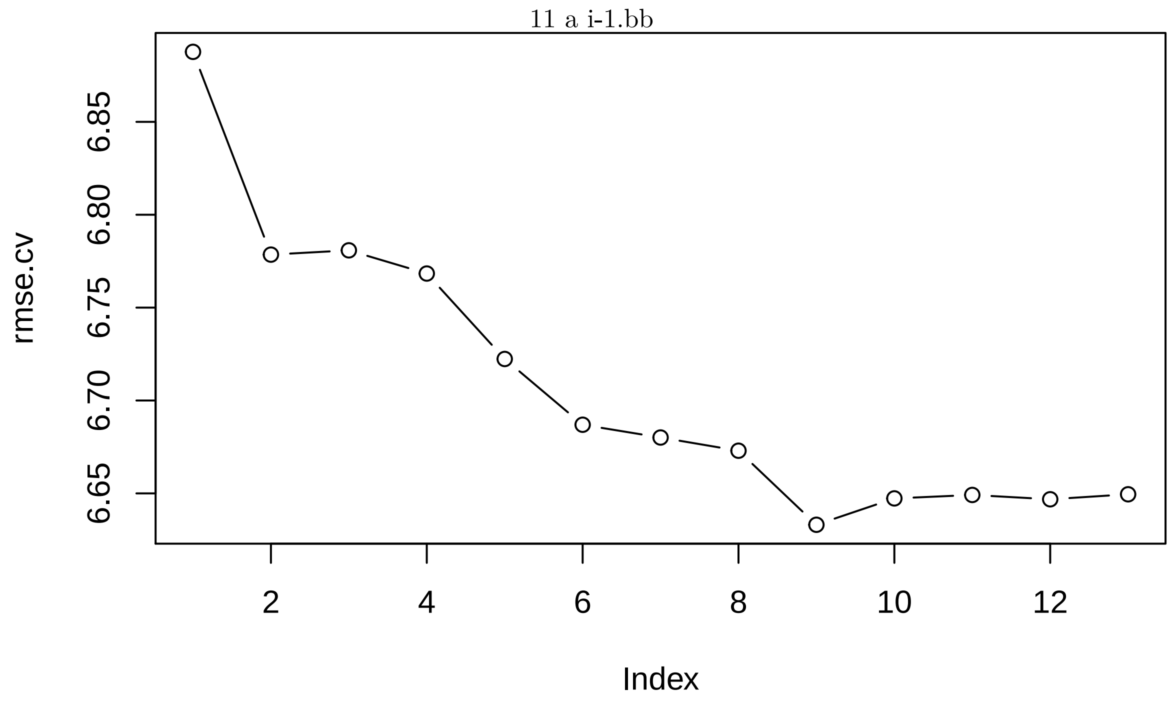
```
## Loaded glmnet 2.0-16
```

```
attach(Boston)

colSums(sapply(Boston, is.na))
```

```
##    crim      zn   indus    chas     nox      rm     age     dis     rad
##       0       0       0       0       0       0       0       0       0
##     tax ptratio   black   lstat    medv
##       0       0       0       0       0
```

```
# No NAs in the dataset

# k-fold cross validation
k = 10

n = dim(Boston)[1]
p = dim(Boston)[2]-1

set.seed(123)
folds = sample(rep(1:k, length = nrow(Boston)), replace = T)
```

```r
form.subset = as.formula("crim ~ .")

cv.errors = matrix(NA, k, p)

for (i in 1:k) {
  subset.fit <- regsubsets(form.subset, Boston[folds!=i, ], nvmax = p)
  for (n.subset in 1:p) {
    m.mat <- model.matrix(form.subset, Boston[folds==i, ])
    best.coef <- coef(subset.fit, id=n.subset)
    pred <- m.mat[, names(best.coef)] %*% best.coef
    # mean squared error
    error = mean((Boston[folds==i, ]$crim - pred)^2)
    cv.errors[i,n.subset] = error
  }

}
# root mean squared values
rmse.cv = sqrt(apply(cv.errors, 2, mean))
plot(rmse.cv, type = "b")
```



```r
which(rmse.cv == min(rmse.cv))
```

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

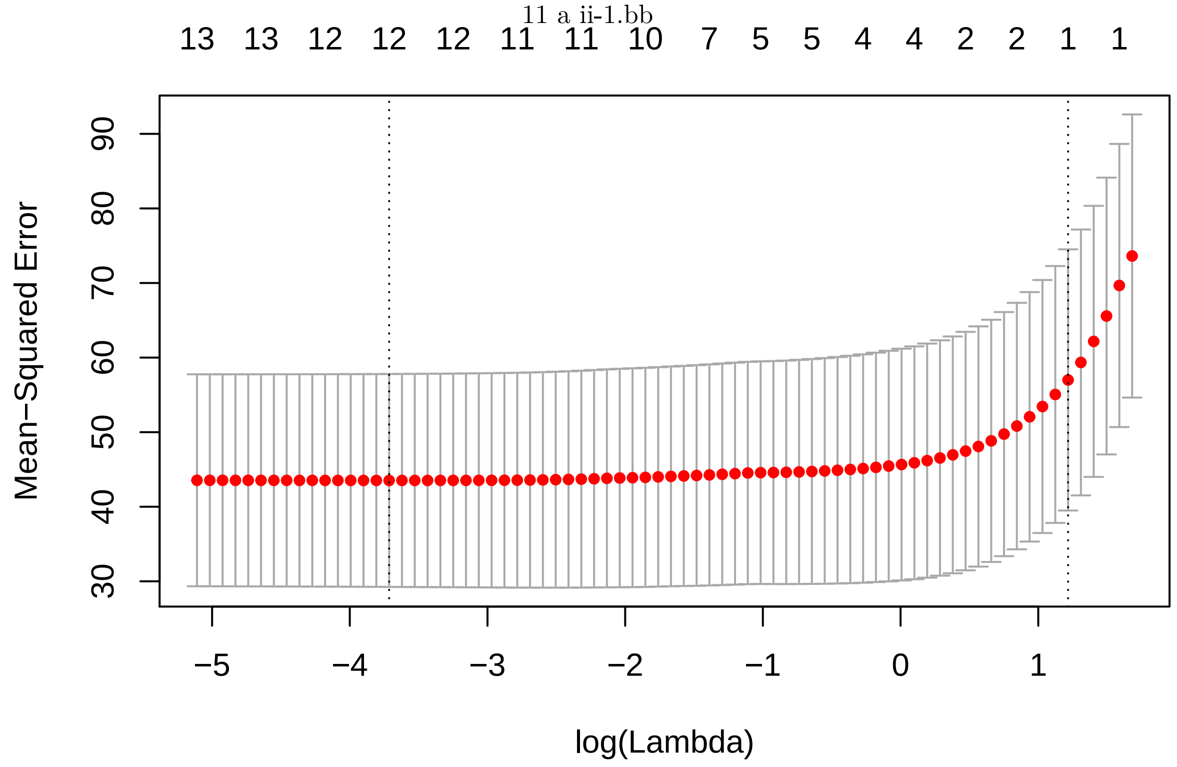
```r
rmse.cv[which(rmse.cv == min(rmse.cv))]
```

```
## [1] 6.633116
```

```r
# Lasso regression

X = model.matrix(crim ~ ., data = Boston)[, -1]
y = Boston$crim
cv.lasso = cv.glmnet(X, y, type.measure = "mse", nfolds = 10)
```

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

11 a ii-1.bb



```r
coef(cv.lasso)[,1]
```

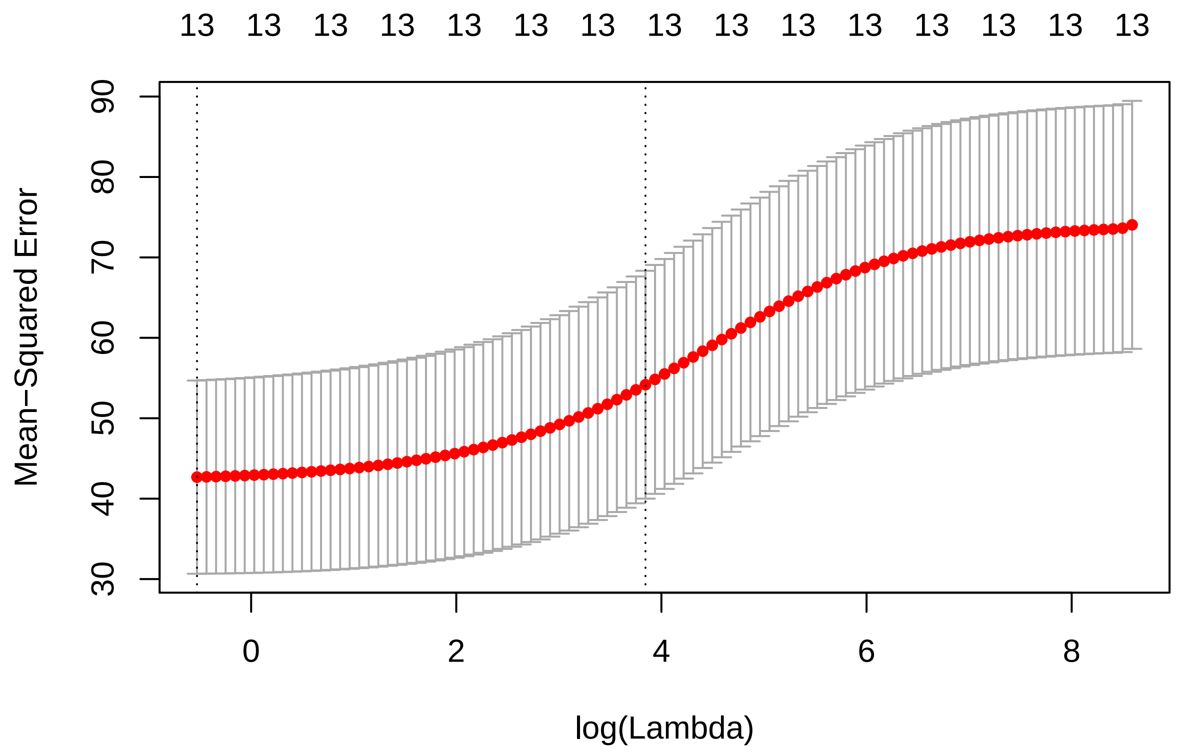
```
## (Intercept)          zn       indus        chas         nox          rm 
##   1.4186415   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000 
##         age         dis         rad         tax     ptratio       black 
##   0.0000000   0.0000000   0.2298449   0.0000000   0.0000000   0.0000000 
##       lstat        medv 
##   0.0000000   0.0000000
```

```r
# One standard error lamda to avoid overfitting
chosen.lambda = cv.lasso$lambda.1se

# root mean square error for the chosen lamdba
sqrt(cv.lasso$cvm[cv.lasso$lambda == chosen.lambda])
```

```
## [1] 7.549995
```

```r
cv.ridge = cv.glmnet(X, y, type.measure = "mse", alpha = 0, nfolds = 10)
plot(cv.ridge)
```

11 a iii-1.bb



```
coef(cv.ridge)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  0.757566248
## zn          -0.002519179
## indus        0.036530724
## chas        -0.259774554
## nox          2.423435801
## rm          -0.169106780
## age          0.007845252
## dis         -0.125063596
## rad          0.067739853
## tax          0.002972047
## ptratio      0.093676787
## black       -0.003748456
## lstat        0.049092196
## medv        -0.032058198
```

```
# One standard error lamda to avoid overfitting
chosen.lambda = cv.ridge$lambda.1se

# root mean square error for the chosen lambda
sqrt(cv.ridge$cvm[cv.ridge$lambda == chosen.lambda])
```

```
## [1] 7.359946
```

```
library(pls)
```

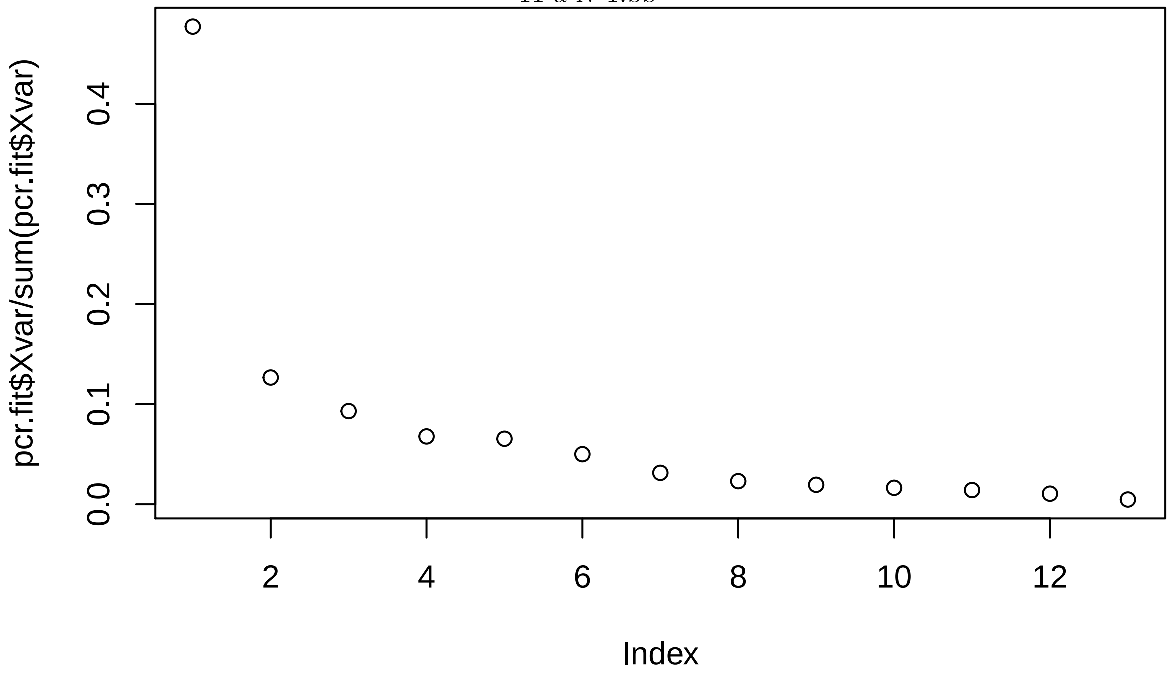
```
##
## Attaching package: 'pls'
## The following object is masked from 'package:stats':
##
##     loadings
```

```
pcr.fit = pcr(crim ~ ., data = Boston, scale = TRUE, validation = "CV", segments= 10)
summary(pcr.fit)
```

```
## Data:    X dimension: 506 13
##  Y dimension: 506 1
## Fit method: svdpc
## Number of components considered: 13
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            8.61    7.189    7.185    6.762    6.770    6.769    6.784
## adjCV         8.61    7.187    7.183    6.757    6.764    6.764    6.778
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       6.789    6.653    6.696     6.680     6.673     6.626     6.563
## adjCV    6.782    6.645    6.686     6.671     6.663     6.615     6.551
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       47.70    60.36    69.67    76.45    82.99    88.00    91.14
## crim    30.69    30.87    39.27    39.61    39.61    39.86    40.14
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X       93.45    95.40     97.04     98.46     99.52     100.0
## crim    42.47    42.55     42.78     43.04     44.13      45.4
```

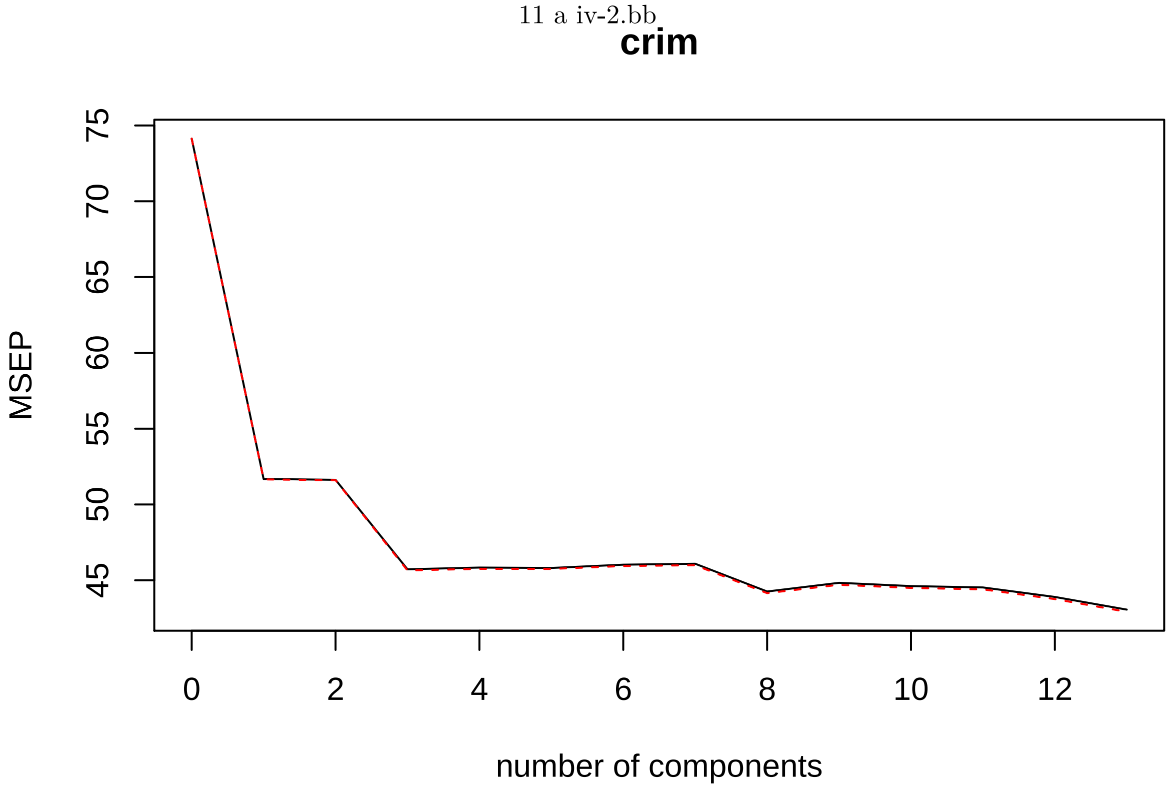
```
# variance explanation
plot(pcr.fit$Xvar/sum(pcr.fit$Xvar))
```

11 a iv-1.bb



```
# validation plot
validationplot(pcr.fit,val.type='MSEP')
```

11 a iv-2.bb



```
ncomp = 4

set.seed(123)
folds = sample(rep(1:k, length = nrow(Boston)), replace = T)

cv.errors = rep(0, k)

for (i in 1:k) {
  pcr.fit <- pcr(crim ~ ., data = Boston[folds !=i, ], scale = TRUE)
  pred = predict( pcr.fit, Boston[folds==i, ], ncomp=ncomp)
  error = mean((Boston[folds==i, ]$crim - pred)^2)
  cv.errors[i] = error
}
sqrt(mean(cv.errors))
```

```
## [1] 6.875975
```

(b)

```
results <- rbind(
  c("Best Subset", 6.633116, 9),
  c("Lasso Regression", 7.549995, 1),
  c("Ridge Regression", 7.359946, 13),
  c("PCR", 6.875975, 4)
)
colnames(results) <- c("Method", "MSE", "# predictors")
```

```r
knitr::kable(results)
```

Method	MSE	# predictors
Best Subset	6.633116	9
Lasso Regression	7.549995	1
Ridge Regression	7.359946	13
PCR	6.875975	4

From the above table, we chose PCR model with 4 predictors as it has the mse very close to the lowest mse and is simpler model than the best subset model, since it has 4 predictors against the 9 predictors and thus has lower chances of overfitting the data. Second choice would be Best Subset model with 9 predictors as it has the lowest cross validation mse and has less number of predictors than Ridge Regression. Lasso Regression here seems to be underfit in this case.

(c) No. PCR has only 4 predictors since from the graph, after 4 predictors, adding another predictor does not increase the explained variance a lot. Hence we have taken only 4 predictors. We can also have criterion like taking n components which explain at least x% of the variance.

3. (10 points) Section 7.9, Page 298, question 3

We have $b_1(X) = X, b_2(X) = (X-1)^2 * I(X \geq 1)$

For $X \geq 1, I(X \geq 1) = 1$, and $X < 1, I(X \geq 1) = 0$

Substituting $\hat{\beta}_0 = 1, \hat{\beta}_1 = 1, \hat{\beta}_2 = 2$ in

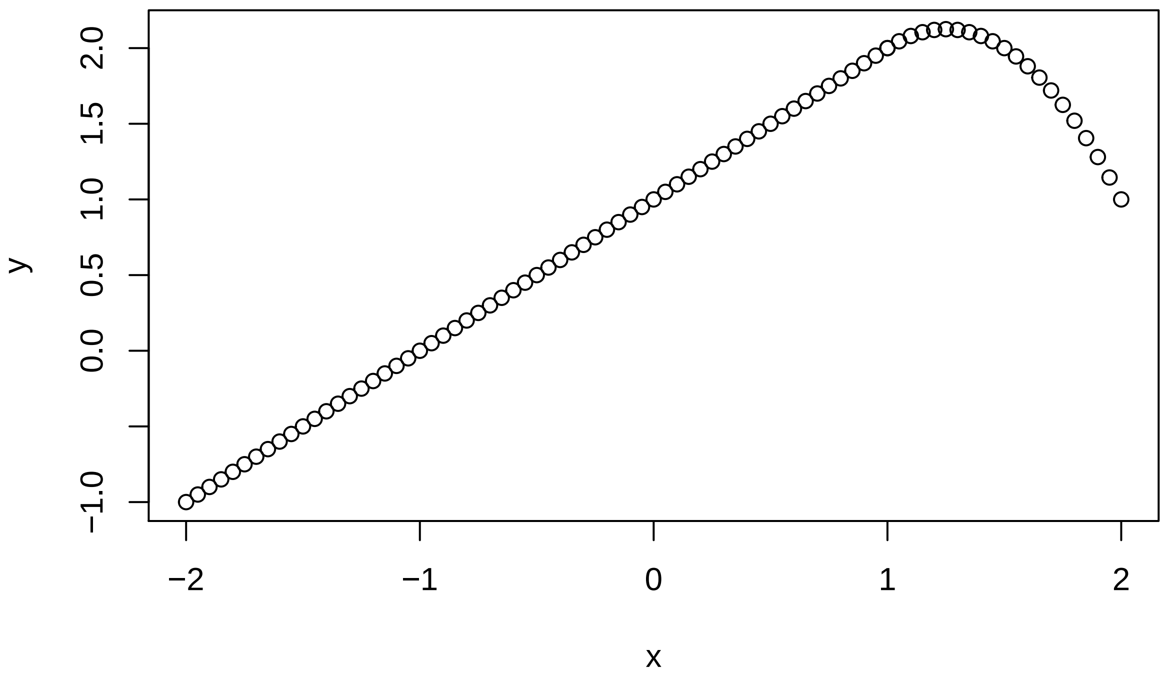
$Y = \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \epsilon$

We get,

$\hat{Y} = 1 + b_1(X) - 2b_2(X)$

For $X \geq 1, \hat{Y} = 1 + X - 2(X-1)^2 = -1 + 5X - 2X^2$, and for $X < 1, \hat{Y} = 1 + X$

```r
x_lower = seq(-2, 1, by = 0.05)
x_upper = seq(1, 2, by = 0.05)

y_lower = 1 + x_lower
y_upper = -1 + 5 * x_upper - 2 * (x_upper ^ 2)

x <- c(x_lower, x_upper)
y <- c(y_lower, y_upper)

plot(x,y)
```



2-1.bb

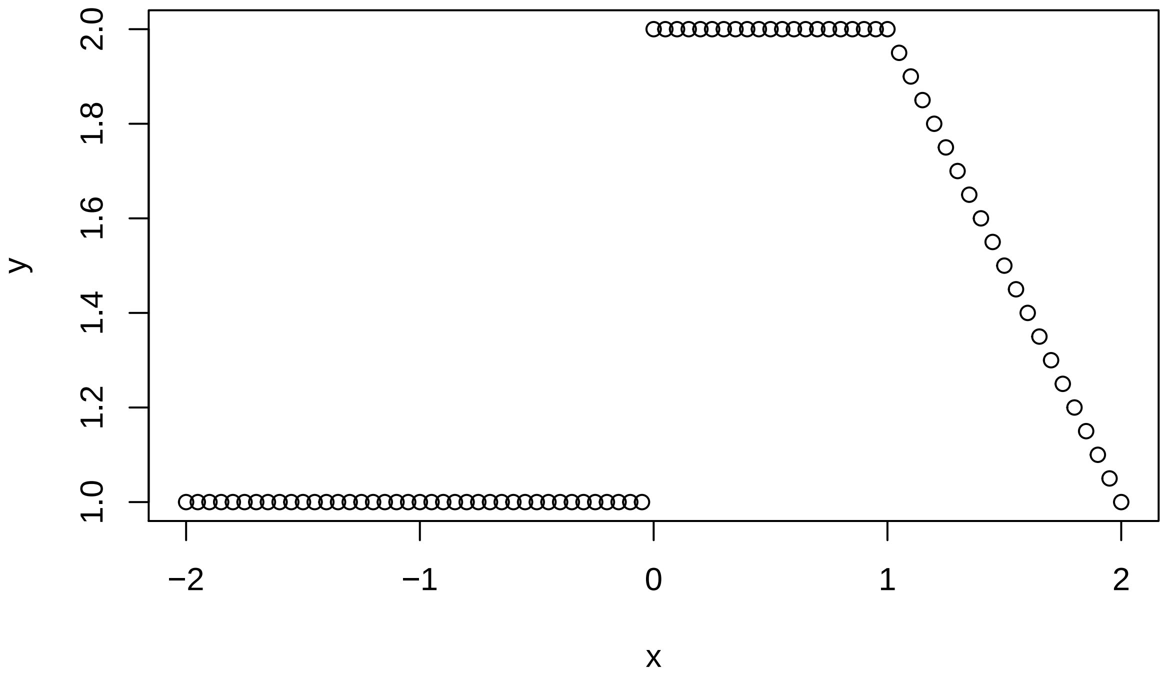
$Y = 1 + X, for X < 1$ $Now, for X = 0, Y = 1$

Therefore y-intercept is 1. Slope is 1 when $X < 1$ and by taking derivative for Y where $X \geq 1$, we get slope as $5 - 4X$

4. (10 points) Section 7.9, Page 298, question 4

Similary from above, we can split the function into multiple domains. Since there are a lot of cuts in this, we use the `I` function in R to enforce the conditions on x. It is as below.

```
x = seq(-2, 2, 0.05)
y = 1 + 1 * I(x <= 2 & x >= 0) - (x-1) * I(x <= 2 & x >= 1)  + 3 * (x-3) * I (x <= 4 & x >= 3) + I(x <=

plot(x, y)
```



4-1.bb

```
# y-intercept
y[which(x==0)]
```

```
## [1] 2
```

The y-intercept is 2 (`y[which(x==0)]`). Slope is -1 for $1 \leq X \leq 2$ and 0 for $-2 \leq X < 0$ and $0 < X \leq 1$. The function is discontinuos at `x=0`

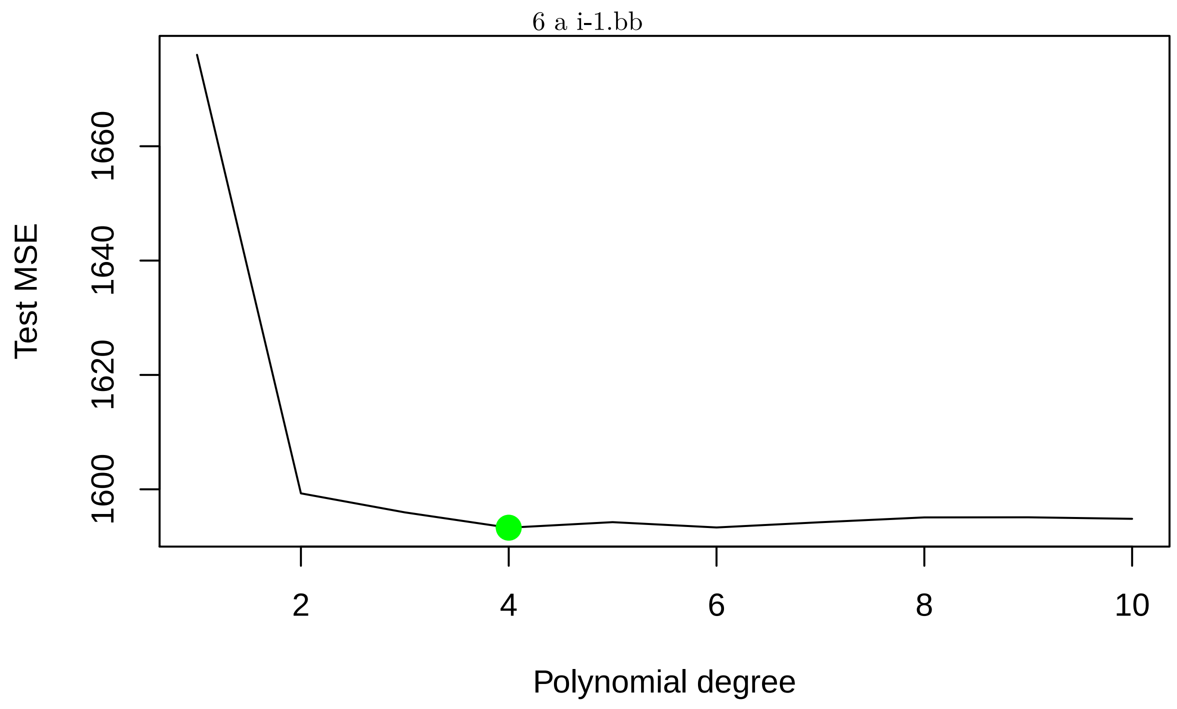
5. (10 points) Section 7.9, Page 299, question 6

a.

First, we start by performing a 10-fold cross validation as shown below:

```
# Import the required libraries
library(ISLR)
library(boot)

# Fix the random seed
set.seed(4)

# Initialize the error/degree vector
errors = rep(NA, 10)

# For each degree from 1 to 10
for (d in 1:10) {
    # Fit a polynomial model of degree d
    fit = glm(wage ~ poly(age, d), data = Wage)
    # Estimate of the test MSE with 10-fold cross validation
    errors[d] = cv.glm(Wage, fit, K = 10)$delta[1]
}
# Plot MSE vs. Degree
plot(1:10, errors, xlab = "Polynomial degree", ylab = "Test MSE", type = "l")
```

```r
# Highlight the lowest test MSE value in the plot
points(which.min(errors), errors[which.min(errors)], col = "green", cex = 2.5, pch = 20)
```



We can see that the polynomial model with `d = 4` outperforms the other models on the testing set which corresponds to the optimal model. In the section below, we use ANOVA for testing the null hypothesis that this model is complex enough to explain the data vs. the alternative hypothesis that a more complex model is required.

```r
# We fit different polynomial models going from degree 1 to 10
fit1 = lm(wage ~ age, data = Wage)
fit2 = lm(wage ~ poly(age, 2), data = Wage)
fit3 = lm(wage ~ poly(age, 3), data = Wage)
fit4 = lm(wage ~ poly(age, 4), data = Wage)
fit5 = lm(wage ~ poly(age, 5), data = Wage)
fit6 = lm(wage ~ poly(age, 6), data = Wage)
fit7 = lm(wage ~ poly(age, 7), data = Wage)
fit8 = lm(wage ~ poly(age, 8), data = Wage)
fit9 = lm(wage ~ poly(age, 9), data = Wage)
fit10 = lm(wage ~ poly(age, 10), data = Wage)

# We use the null hypothesis test ANOVA
anova(fit1, fit2, fit3, fit4, fit5, fit6, fit7, fit8, fit9, fit10)
```

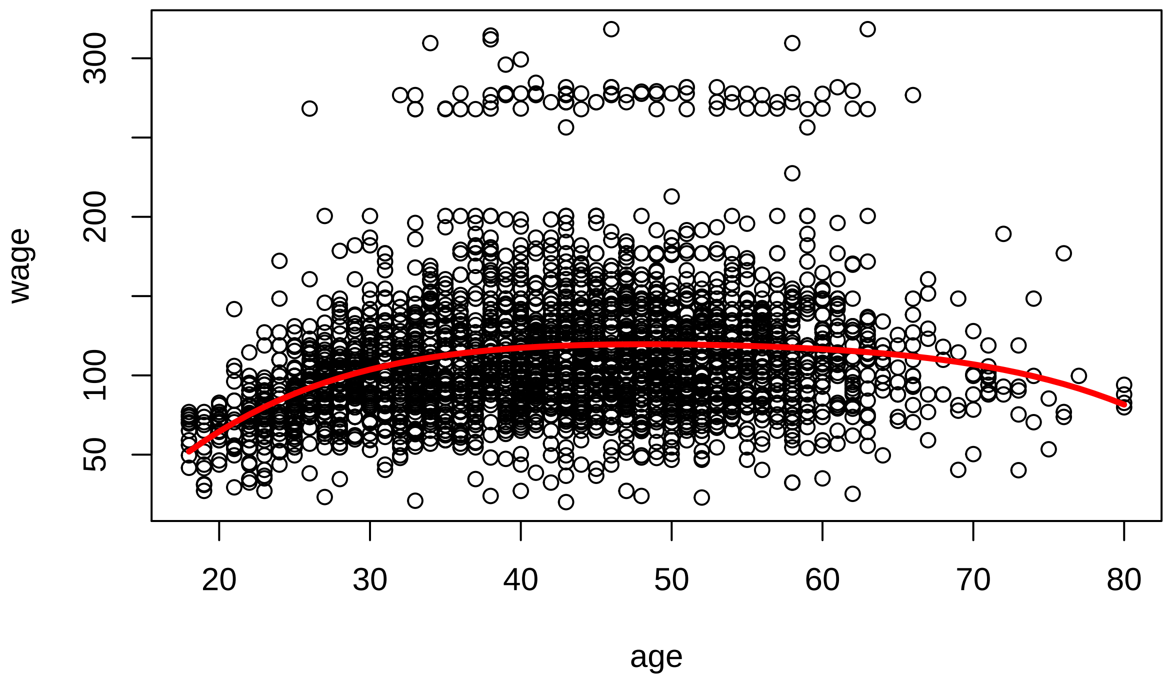
```
## Analysis of Variance Table
##
## Model  1: wage ~ age
## Model  2: wage ~ poly(age, 2)
## Model  3: wage ~ poly(age, 3)
## Model  4: wage ~ poly(age, 4)
## Model  5: wage ~ poly(age, 5)
## Model  6: wage ~ poly(age, 6)
## Model  7: wage ~ poly(age, 7)
## Model  8: wage ~ poly(age, 8)
## Model  9: wage ~ poly(age, 9)
## Model 10: wage ~ poly(age, 10)
##    Res.Df     RSS Df Sum of Sq        F    Pr(>F)
## 1    2998 5022216
## 2    2997 4793430  1    228786 143.7638 < 2.2e-16 ***
## 3    2996 4777674  1     15756   9.9005  0.001669 **
## 4    2995 4771604  1      6070   3.8143  0.050909 .
## 5    2994 4770322  1      1283   0.8059  0.369398
## 6    2993 4766389  1      3932   2.4709  0.116074
## 7    2992 4763834  1      2555   1.6057  0.205199
## 8    2991 4763707  1       127   0.0796  0.777865
## 9    2990 4756703  1      7004   4.4014  0.035994 *
## 10   2989 4756701  1         3   0.0017  0.967529
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can observe that a polynomial fit with degree 4 provides a statistically significant fit to the data (degree 3 too), however, the other models are not justified since they provide higher p-values. We can say the the ANOVA test proves the results found in the plot above.

In this section, we fit the data with the previous model:

```r
# Plot the Wage vs. Age data
plot(wage ~ age, data = Wage)

# Calculate the age grid (lower and upper limits)
limits = range(Wage$age)
ageGrid = seq(from = limits[1], to = limits[2])

# Fit the data with the quartic polynomial model
fit = lm(wage ~ poly(age, 4), data = Wage)

# Calculate the predictions for this model
predictions = predict(fit, newdata = list(age = ageGrid))

# Display the model on the plot
lines(ageGrid, predictions, col = "red", lwd = 3)
```

6 a iii-1.bb

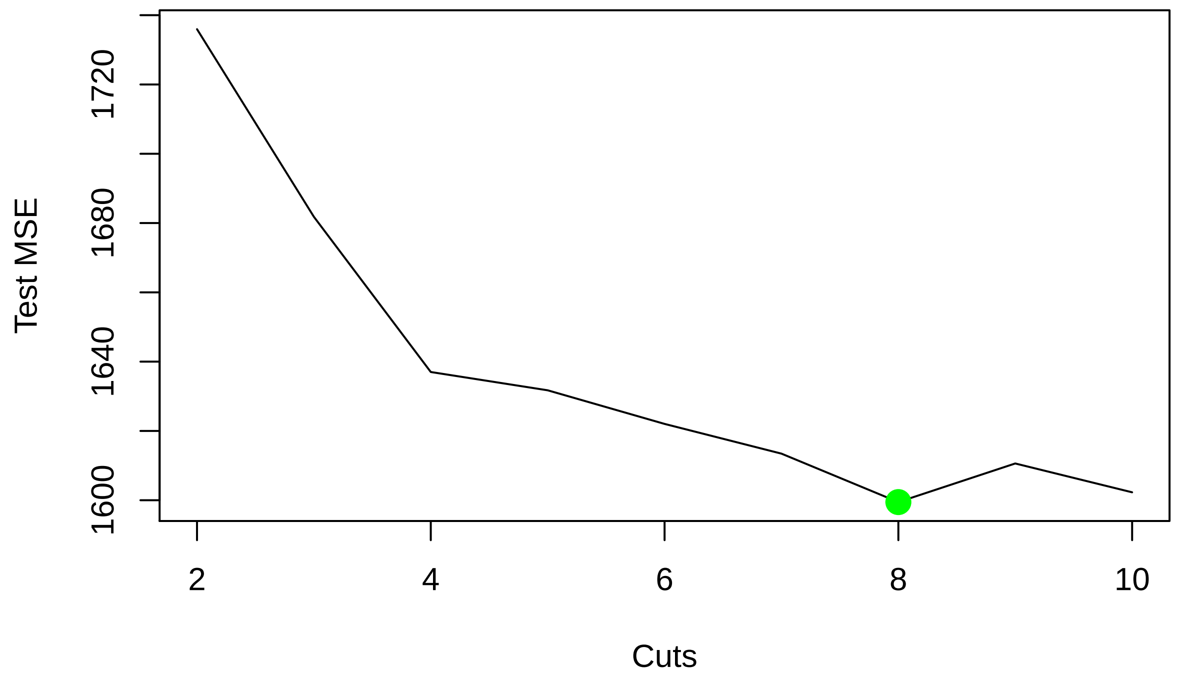


b.

We repeat the same process for the step function using 10-fold cross validation.

```r
errors <- rep(NA, 10)
for (intervals in 2:10) {
    Wage$age.cut = cut(Wage$age, intervals)
    fit = glm(wage ~ age.cut, data = Wage)
    errors[intervals] = cv.glm(Wage, fit, K = 10)$delta[1]
}
plot(2:10, errors[-1], xlab = "Cuts", ylab = "Test MSE", type = "l")

points(which.min(errors), errors[which.min(errors)], col = "green", cex = 2.5, pch = 20)
```

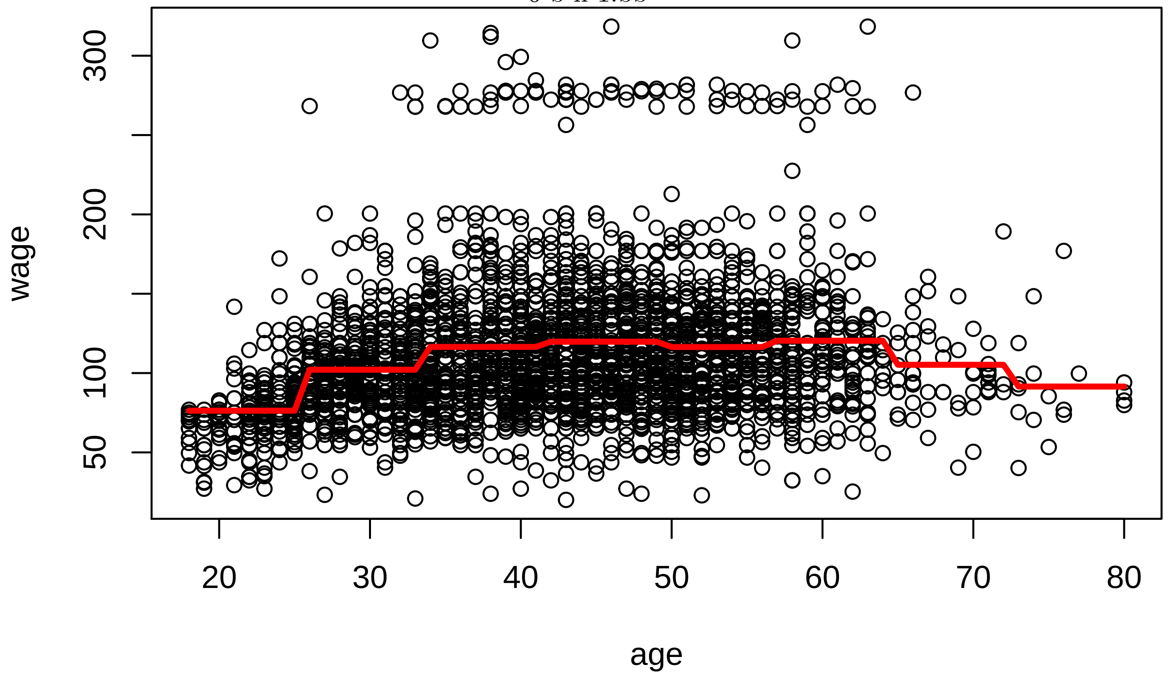
6 b i-1.bb



We can observe that the optimal number of `cuts = 8` since it corresponds to the lowest error on the testing set. In the section below, we plot the data and the 8-cuts step model fit.

```
plot(wage ~ age, data = Wage)
agelims = range(Wage$age)
age.grid = seq(from = agelims[1], to = agelims[2])
fit = glm(wage ~ cut(age, 8), data = Wage)
preds = predict(fit, data.frame(age = age.grid))
lines(age.grid, preds, col = "red", lwd = 3)
```

6 b ii-1.bb



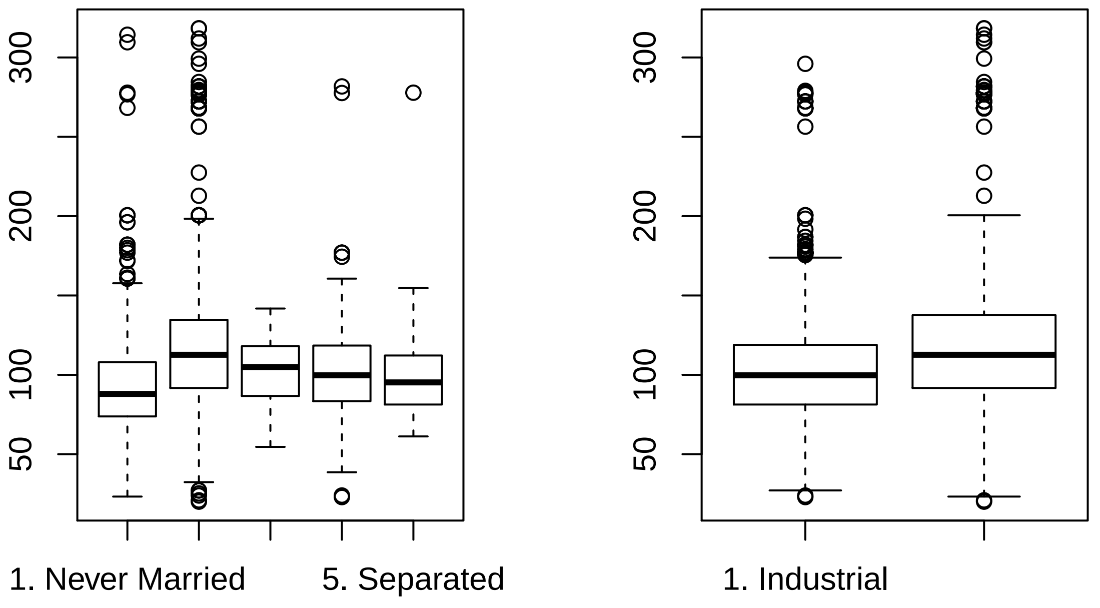
6. (20 points) Section 7.9, Page 299, question 7

Let's first begin by exploring the relationship between `wage` from a side and `maritl` and `jobclass` from the other side.

```r
# Fix the random seed
set.seed(1)

 # Display the summary of the features maritl and jobclass (index of values)
summary(Wage[, c("maritl", "jobclass")] )
```

```
##                maritl                jobclass
##  1. Never Married: 648   1. Industrial :1544
##  2. Married      :2074   2. Information:1456
##  3. Widowed      :  19
##  4. Divorced     : 204
##  5. Separated    :  55
```

```r
# Display the wage vs. maril and wage vs. jobclass side by side
par(mfrow = c(1, 2))
plot(Wage$maritl, Wage$wage)
plot(Wage$jobclass, Wage$wage)
```



7 i-1.bb

From the plots above, we can see that a `married` marital status corresponds to higher wages (followed by widowed) and job class corresponding to `informational` presents higher wages too on an average.

In this section, let's experiment different models using natural spline functions of the variables `martil`, `jobclass`, `year`, `education` and `age`.

```r
library(gam)
```

```
## Loading required package: splines
```

```
## Loaded gam 1.16
```

```r
fit0 = gam(wage ~ lo(age,span=200,degree=1), data = Wage)
fit1 = gam(wage ~ lo(age,span=200,degree=1) + year, data = Wage)
fit2 = gam(wage ~ s(year,4)+lo(age,span=200,degree=1), data = Wage)
```

```r
fit4 = gam(wage ~ s(year,4)+lo(age,span=200,degree=1) + education, data = Wage)
fit5 = gam(wage ~ s(year,4)+lo(age,span=200,degree=1) + education + jobclass, data = Wage)
fit6 = gam(wage ~ s(year,4)+lo(age,span=200,degree=1) + education + maritl, data = Wage)
fit7 = gam(wage ~ s(year,4)+lo(age,span=200,degree=1) + education + jobclass + maritl, data = Wage)
anova(fit0, fit1, fit2, fit4, fit5, fit6, fit7)
```

```
## Analysis of Deviance Table
##
## Model 1: wage ~ lo(age, span = 200, degree = 1)
## Model 2: wage ~ lo(age, span = 200, degree = 1) + year
## Model 3: wage ~ s(year, 4) + lo(age, span = 200, degree = 1)
## Model 4: wage ~ s(year, 4) + lo(age, span = 200, degree = 1) + education
## Model 5: wage ~ s(year, 4) + lo(age, span = 200, degree = 1) + education +
##     jobclass
## Model 6: wage ~ s(year, 4) + lo(age, span = 200, degree = 1) + education +
##     maritl
## Model 7: wage ~ s(year, 4) + lo(age, span = 200, degree = 1) + education +
##     jobclass + maritl
##   Resid. Df Resid. Dev     Df Deviance  Pr(>Chi)
## 1      2998    5022190
## 2      2997    5004580 1.0000    17610 0.0001540 ***
## 3      2994    4999892 2.9996     4689 0.2822804
## 4      2990    3849469 4.0000  1150423 < 2.2e-16 ***
## 5      2989    3835853 1.0000    13616 0.0008753 ***
## 6      2986    3685252 3.0000   150601 < 2.2e-16 ***
## 7      2985    3670120 1.0000    15132 0.0004512 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can conclude that models 4 and 6 are statistically significant with respectively p-values equal to `< 2.2e-16` and `< 2.2e-16`. Hence, we can say that the model gets a statistically significant improvement by including the `year spline`, `age local regression`, `education` and `maritl`. However, the model is less significant for the cases where we introduced `jobclass`.

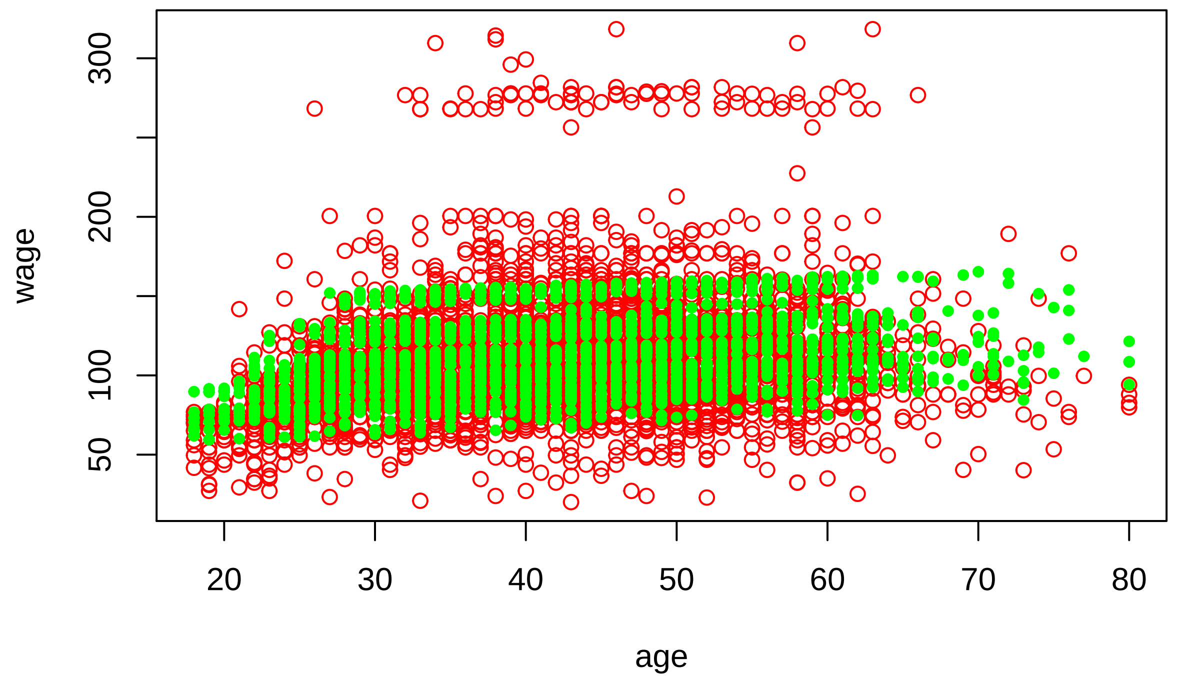
In this section, we are going to plot our 7th model as shown below:

```r
attach(Wage)
```

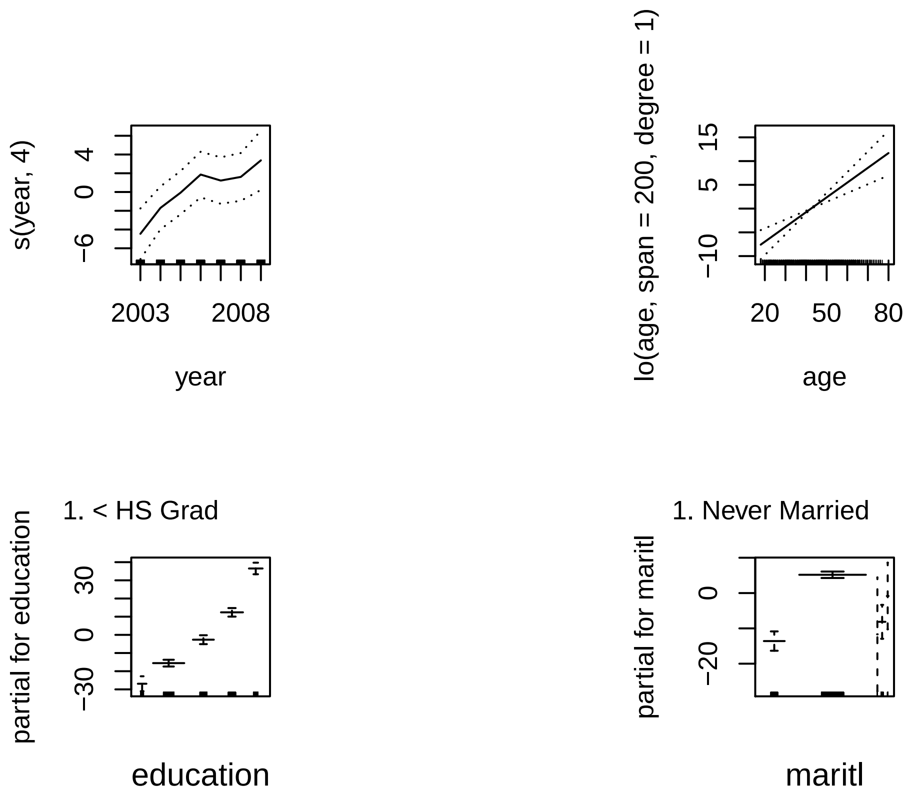
```
## The following object is masked from Boston:
##
##     age
```

```r
# Calculate the prediction with our model
myPreds = predict(fit6,se=TRUE)

# Plot the data
plot(age,wage, col='red')

# Plot the predictions
points(age,myPreds$fit,col='green',pch=20)
```

7 iii-1.bb



```r
# Fancy GAM plot
par(mfrow = c(2, 2), pty = "s")
plot(fit6,se=TRUE)
```

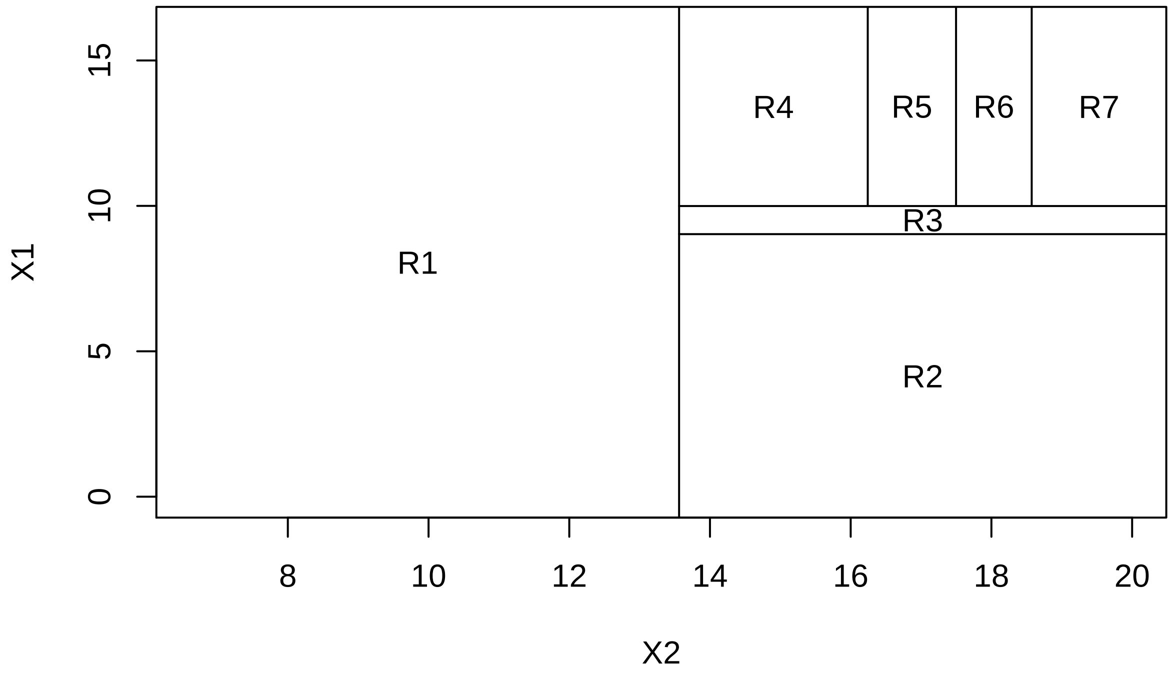


7 iii-2.bb

We can observe our data is not fitted with a linear model (plane) but instead is fitted with multiple lines each corresponding to a value of the categorical variables `education` and `mirtl`. We can see that our model is better fitted by combining multiple models such as `spline`, `local regression` and `linear regression` with `continous` and `categorical` variables.

7. (10 points) Section 8.4, Page 332, question 1

```r
library(tree)
syn_data <- data.frame(mvrnorm(n= 100, mu = c(10,15), matrix(c(10,3,3,5),2,2)))
color <- sample(c("red", "blue"), nrow(syn_data), replace = T)
syn_data$class <- ifelse(color == "red", 1, 0)
tree.fit <- tree(class ~ ., data = syn_data, control = tree.control(nobs = nrow(syn_data), mindev = 0.0

partition_plot(tree.fit)
```



1-1.bb

8. (10 points) Section 8.4, Page 333-334, question 8

a.

```r
library(ISLR)
set.seed(1)

# Splitting the data according to a 50/50 ratio

split = sample(1:nrow(Carseats), nrow(Carseats) / 2)
Carseats.train = Carseats[split, ]
Carseats.test = Carseats[-split, ]
```

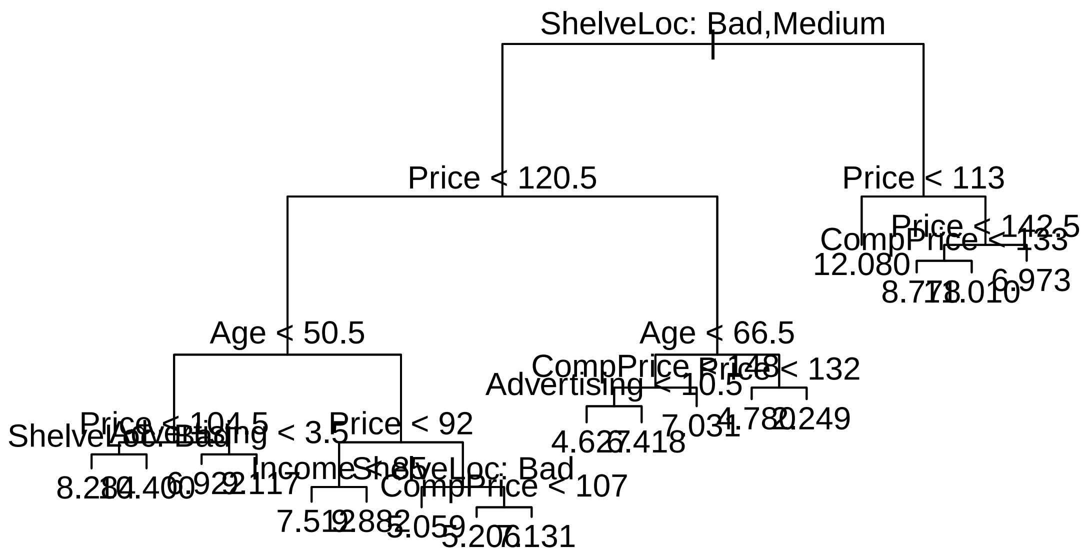
b.

```r
# Import the library
library(tree)

tree.carseats =  tree(Sales ~ ., data = Carseats.train)
summary(tree.carseats)
```

```
##
## Regression tree:
```

```
## tree(formula = Sales ~ ., data = Carseats.train)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Age"         "Advertising" "Income"
## [6] "CompPrice"
## Number of terminal nodes:  18
## Residual mean deviance:  2.36 = 429.5 / 182
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -4.2570 -1.0360  0.1024  0.0000  0.9301  3.9130
```

```
plot(tree.carseats)
text(tree.carseats, pretty = 0)
```



8 b-1.bb

We can see that the most important variable in the tree is `ShelveLoc` and then `Price` which means that knowing the shelve location would be the information that helps us the most to predict the `sales` of the carseats.

```
pred <- predict(tree.carseats, newdata = Carseats.test)
mean((pred - Carseats.test$Sales)^2)
```

```
## [1] 4.148897
```

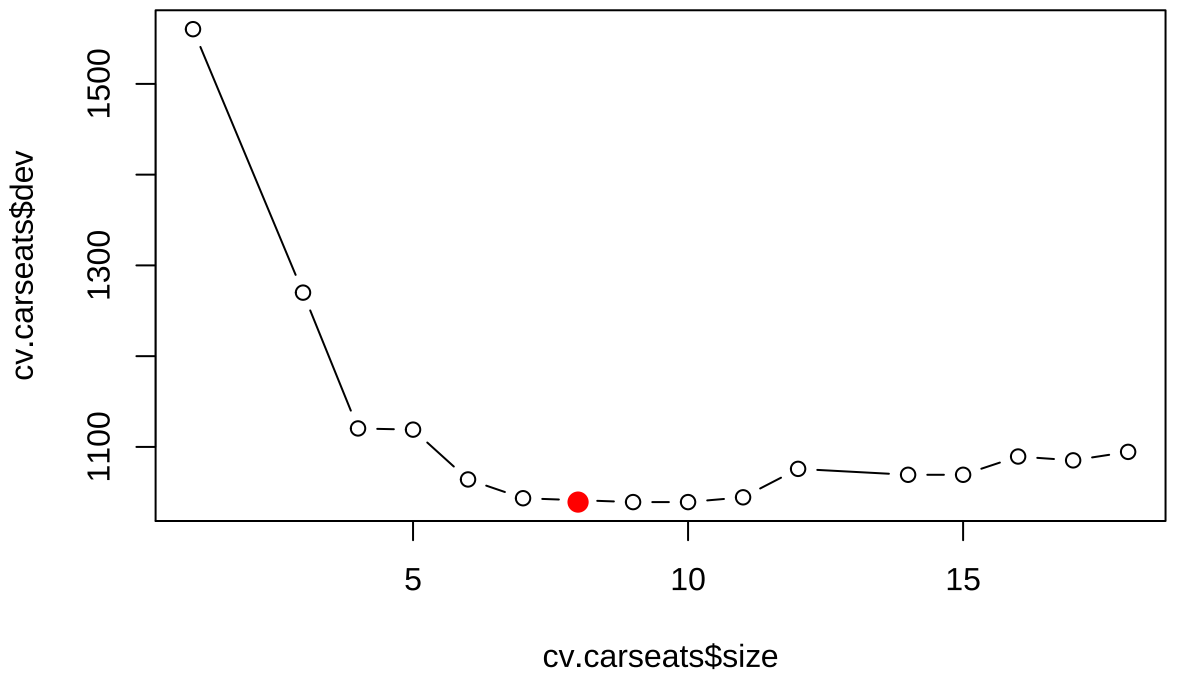
In this case, we obtain about `4.15` as test MSE.

c.

In this section, we use cross-validation to know which size is optimal for the tree complexity.

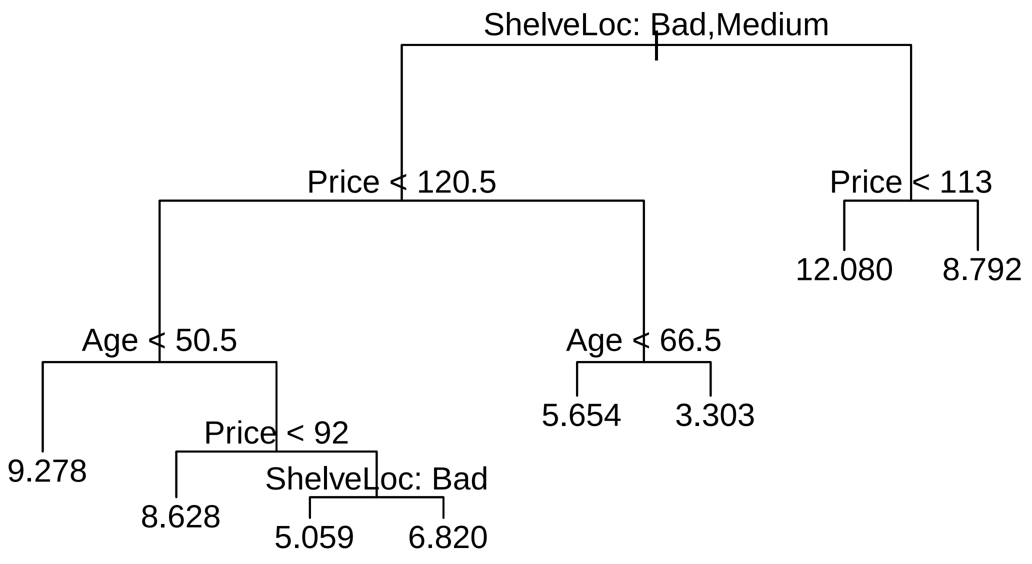
```
cv.carseats <- cv.tree(tree.carseats)
plot(cv.carseats$size, cv.carseats$dev, type = "b")
tree.min <- which.min(cv.carseats$dev)
points(tree.min, cv.carseats$dev[tree.min], col = "red", cex = 2, pch = 20)
```

8 c i-1.bb



We can observe that a size of `8` corresponds to the minimal deviance. Hence, we draw the tree with `size = 8` and we compute the test MSE for this tree.

```r
prune.carseats = prune.tree(tree.carseats, best = 8)
plot(prune.carseats)
text(prune.carseats, pretty = 0)
```



8 c ii-1.bb

```r
pred = predict(prune.carseats, newdata = Carseats.test)
mean((pred - Carseats.test$Sales)^2)
```

```
## [1] 5.09085
```

We can see that pruning the tree is not helpful in this case since the test MSE increased after pruning from `4.15` to about `5.09`. However, the tree seems way more human readable than the first one and presents the most important variables in the same order.

d.

```r
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.
```

```r
bag.carseats <- randomForest(Sales ~ ., data = Carseats.train, mtry = 10, ntree = 500, importance = TRUI
pred.bag <- predict(bag.carseats, newdata = Carseats.test)
mean((pred.bag - Carseats.test$Sales)^2)
```

```
## [1] 2.633915
```

As we can see, using bagging decreased the test MSE to about `2.63` which means it outperforms the previous two models.

```r
importance(bag.carseats)
```

```
##                %IncMSE IncNodePurity
## CompPrice   16.9874366    126.852848
## Income       3.8985402     78.314126
## Advertising 16.5698586    123.702901
## Population   0.6487058     62.328851
## Price       55.3976775    514.654890
## ShelveLoc   42.7849818    319.133777
## Age         20.5135255    185.582077
## Education    3.4615211     42.253410
## Urban       -2.5125087      8.700009
## US           7.3586645     18.180651
```

We can observe also that the most important variables are `ShelveLoc`, `Price` and `Age` which were also the most important variables for our previous models.

e.

```r
mse.vec <- NA
for (a in 1:10){
  rf.carseats <-  randomForest(Sales ~ . , data=Carseats.train,
                               mtry=a, ntree=500, importance=TRUE)
  rf.pred <-  predict(rf.carseats, Carseats.test)
  mse.vec[a] <- mean((Carseats.test$Sales - rf.pred)^2)
}

# Number of variables used in the best model
which.min(mse.vec)
```

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

```r
# Test MSE corresponding to the best model
mse.vec[which.min(mse.vec)]
```

```
## [1] 2.56175
```

After applying random forest, we observe that the best model having to the lowest test MSE `2.56` corresponds to a tree using 10 predictors.

```r
# Most important variables corresponding to the best model
rf.carseats <-  randomForest(Sales ~ . , data = Carseats,
                             mtry=9, ntree=500, importance=TRUE)
importance(rf.carseats)
```

```
##                %IncMSE IncNodePurity
## CompPrice   37.4090331     328.33305
## Income      11.6101066     167.93255
## Advertising 29.6279922     235.62302
## Population  -0.5069106     104.08331
## Price       81.7594068     899.59222
## ShelveLoc   86.8711457     972.32922
## Age         27.2728885     286.99161
## Education    2.8298995      84.28843
## Urban       -1.8371710      12.60837
## US           5.2177662      16.76991
```

Our best model shows also that the most important predictors in order are `ShelveLoc`, `Price`, `CompPrice`, `Advertising` and `Age`.

From the experiments above, we can conclude that the number of predictors `m` helps reduce the test MSE and hence, helps predicting our data with higher accuracy which explains why pruning the same tree in `b` generated a higher test MSE. However, there is a trade-off between the model's `accuracy` and `complexity` which makes the model more accurate but less readable and more complex.