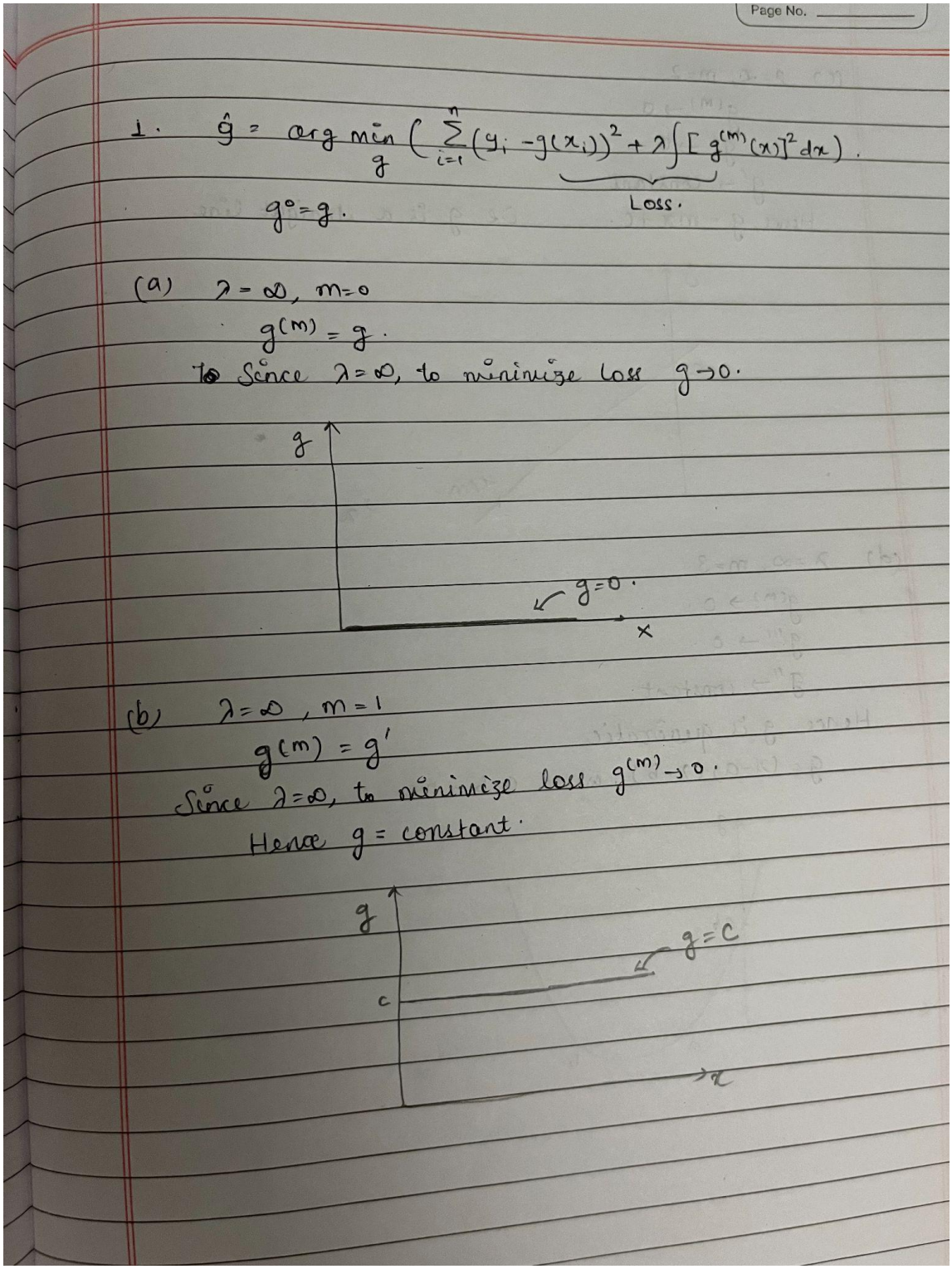
HW 4 submission

1.



1. $\hat{g} = \underset{g}{\arg\min} \left(\sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int [g^{(m)}(x)]^2 dx \right)$.

Loss.

$g^0 = g$.

(a) $\lambda = \infty$, $m = 0$

$g^{(m)} = g$.

Since $\lambda = \infty$, to minimize loss $g \to 0$.

(b) $\lambda = \infty$, $m = 1$

$g^{(m)} = g'$

Since $\lambda = \infty$, to minimize loss $g^{(m)} \to 0$.

Hence g = constant.

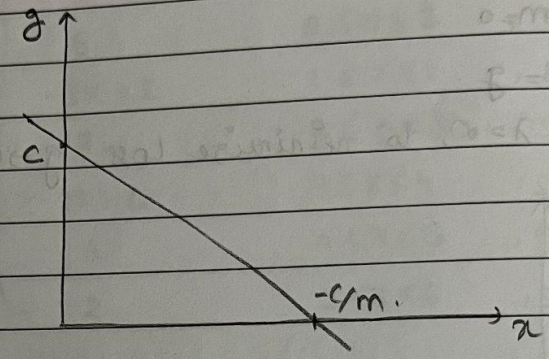
(c) $\lambda = \infty$, $m = 2$

$g^{(m)} \to 0$.

$g'' \to 0$.

$g' \to$ constant.

Hence, $g = mx + c$. i.e g is a straight line.



(d) $\lambda = \infty$, $m = 3$.

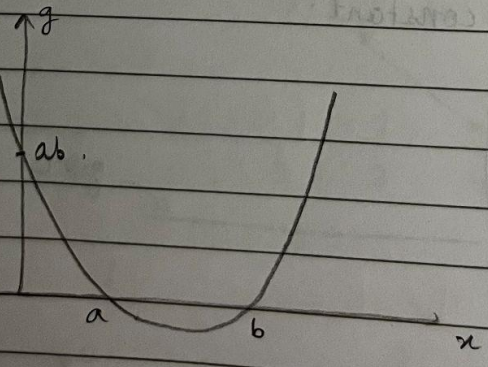
$g^{(m)} \to 0$.

$g''' \to 0$.

$g'' \to$ constant.

Hence, g is quadratic

$g = (x-a)(x-b)$, where $a, b \to$ +ve.

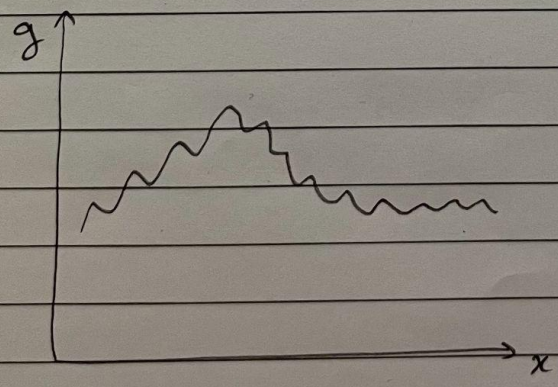


(e) $\lambda = 0$, $m = 3$

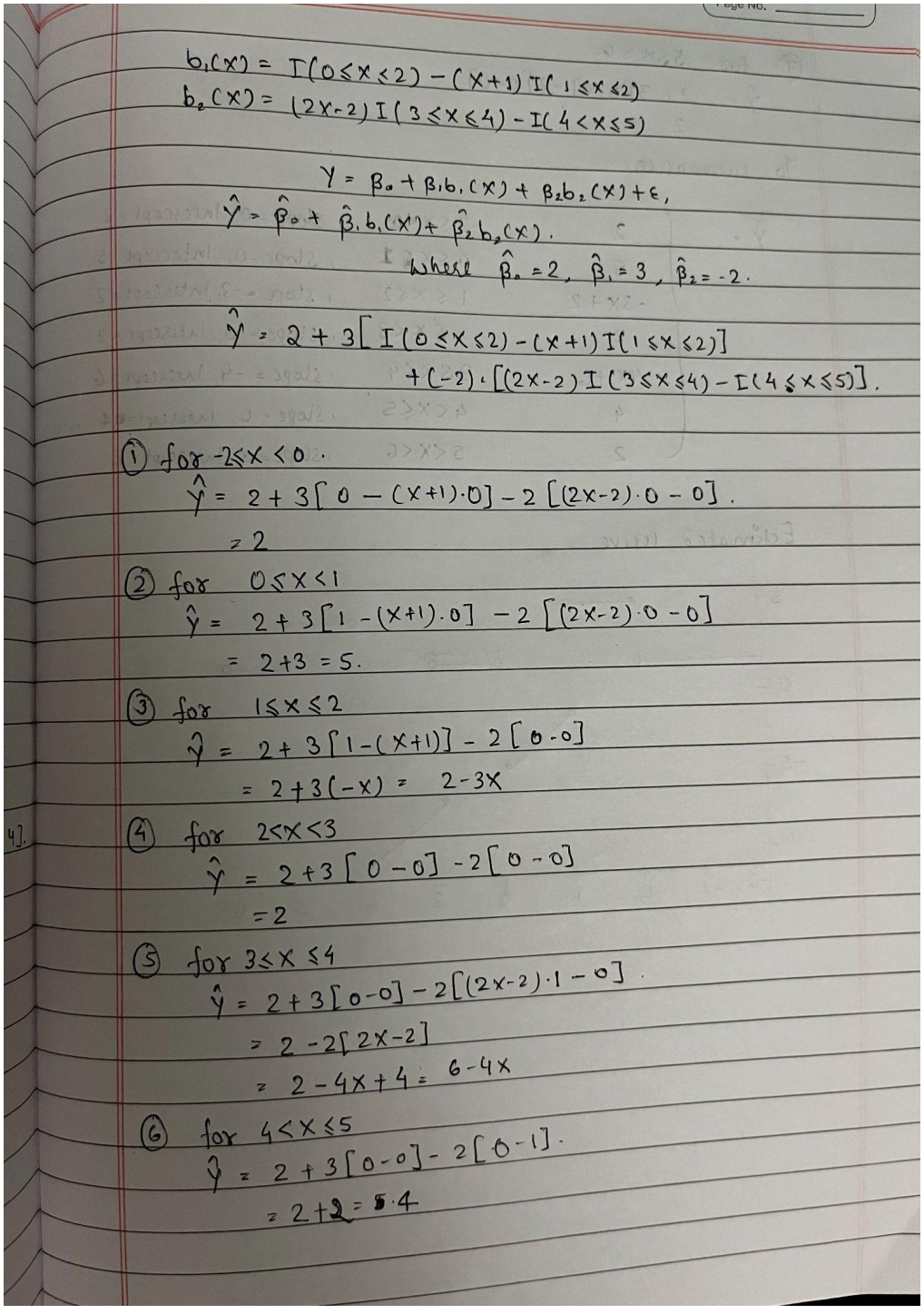
$$\hat{g} = \underset{g}{\operatorname{argmin}} \left(\sum_{i=1}^{n} (y_i - g(x_i))^2 + 0 \right).$$

When $\lambda = 0$, the g will be very jumpy as penalty term has no effect.

It will be a function that exactly interpolates all the observations.



2.



$b_1(x) = I[0 \le x \le 2) - (x+1) I(1 \le x \le 2)$

$b_2(x) = (2x-2) I(3 \le x \le 4) - I(4 < x \le 5)$

$$Y = \beta_0 + \beta_1 b_1(x) + \beta_2 b_2(x) + \varepsilon,$$

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 b_1(x) + \hat{\beta}_2 b_2(x).$$

where $\hat{\beta}_0 = 2$, $\hat{\beta}_1 = 3$, $\hat{\beta}_2 = -2$.

$$\hat{y} = 2 + 3[I(0 \le x \le 2) - (x+1) I(1 \le x \le 2)] + (-2) \cdot [(2x-2) I(3 \le x \le 4) - I(4 \le x \le 5)].$$

① for $-2 \le x < 0$.

$\hat{y} = 2 + 3[0 - (x+1) \cdot 0] - 2[(2x-2) \cdot 0 - 0]$.

$= 2$

② for $0 \le x < 1$

$\hat{y} = 2 + 3[1 - (x+1) \cdot 0] - 2[(2x-2) \cdot 0 - 0]$

$= 2 + 3 = 5$.

③ for $1 \le x \le 2$

$\hat{y} = 2 + 3[1 - (x+1)] - 2[0 - 0]$

$= 2 + 3(-x) = 2 - 3x$

④ for $2 < x < 3$

$\hat{y} = 2 + 3[0 - 0] - 2[0 - 0]$

$= 2$

⑤ for $3 \le x \le 4$

$\hat{y} = 2 + 3[0 - 0] - 2[(2x-2) \cdot 1 - 0]$.

$= 2 - 2[2x - 2]$

$= 2 - 4x + 4 = 6 - 4x$

⑥ for $4 \le x \le 5$

$\hat{y} = 2 + 3[0 - 0] - 2[0 - 1]$.

$= 2 + 2 = 5.4$

FABER-CASTE
Date
Page No.

(7) for $5 \leq x \leq 6$.

$\hat{y} = 2 + 3[0-0] - 2[0-0]$

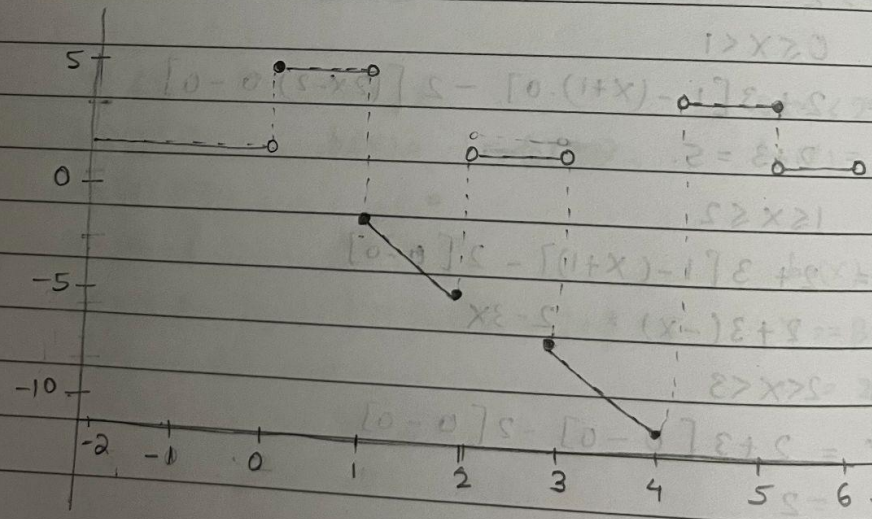
$= 2$

To summarise:

$$\hat{y} = \begin{cases} 2 & -2 \leq x < 0 \\ 5 & 0 \leq x \leq 1 \\ -3x + 2 & 1 \leq x \leq 2 \\ 2 & 2 \leq x < 3 \\ -4x + 6 & 3 \leq x \leq 4 \\ 4 & 4 < x \leq 5 \\ 2 & 5 < x < 6 \end{cases}$$

, slope = 0, Intercept = 2
, slope = 0, Intercept = 5
, slope = -3, Intercept = 2
, slope = 0, Intercept = 2
, slope = -4, Intercept = 6
, slope = 0, Intercept = 4
, slope = 0, Intercept = 2.

Estimated Curve:



3.

$$f(X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 (X - \psi)^3_+$$

In order to prove that above function is a cubic spline, we need to show that:

1. It is continuous at ψ
2. It is piece-wise cubic at ψ
3. It's first derivative is continuous at ψ
4. It's second derivative is continuous at ψ
5. It's third derivative is discontinuos at ψ(not necessary)

All the above conditions are checked below:

① The function is defined at ψ.

Let see if $f(\psi^+) = f(\psi^-)$.

$$\lim_{x \to \psi^-} f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$

$$\lim_{x \to \psi^+} f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-\psi)^3.$$

$$\lim_{x \to \psi^-} f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$
$$= \beta_0 + \beta_1 \psi + \beta_2 \psi^2 + \beta_3 \psi^3.$$

$$\lim_{x \to \psi^+} f(x) = \beta_0 + \beta_1 \psi + \beta_2 \psi^2 + \beta_3 \psi^3 + \beta_4 (\psi - \psi)^3.$$
$$= \beta_0 + \beta_1 \psi + \beta_2 \psi^2 + \beta_3 \psi^3 + 0.$$

Hence $\lim_{x \to \psi^-} f(x) = \lim_{x \to \psi^+} f(x)$.

② $f'(x) = \beta_1 + 2\beta_2 x + 3\beta_3 x^2 + 3\beta_4 (x-\psi)^2$ if $x > \psi$

$f'(x) = \beta_1 + 2\beta_2 x + 3\beta_3 x^2$ if otherwise i.e $x \leq \psi$

$$\lim_{x \to \psi^+} f'(x) = \beta_1 + 2\beta_2 \psi + 3\beta_3 \psi^2 + 3(\beta_4 (\psi - \psi)^2)$$
$$= \beta_1 + 2\beta_2 \psi + 3\beta_3 \psi^2 + 0.$$

$$\lim_{x \to \psi^-} f'(x) = \beta_1 + 2\beta_2 \psi + 3\beta_3 \psi^2 + 0.$$

Hence.

$$\lim_{x \to \psi^+} f'(x) = \lim_{x \to \psi^-} f'(x)$$

④ $f''(x) = 2\beta_2 + 6\beta_3 x + 6\beta_4 (x - \psi)$ if $x > \psi$

$f''(x) = 2\beta_2 + 6\beta_3 x + 0$ if $x \leq \psi$.

$$\lim_{x \to \psi^+} f''(x) = 2\beta_2 + 6\beta_3 \psi + 6\beta_4 (\psi - \psi)$$

$$= 2\beta_2 + 6\beta_3 \psi + 0.$$

$$\lim_{x \to \psi^-} f''(x) = 2\beta_2 + 6\beta_3 \psi$$

Hence

$$\lim_{x \to \psi^+} f''(x) = \lim_{x \to \psi^-} f''(x).$$

⑤ $f'''(x) = 6\beta_2 + 6\beta_4$. $x > \psi$.

$f'''(x) = 6\beta_3$ $x \leq \psi$.

$$\lim_{x \to \psi^+} f'''(x) = 6\beta_3 + 6\beta_4.$$

$$\lim_{x \to \psi^-} f'''(x) = 6\beta_3.$$

Hence, $\lim_{x \to \psi^+} f'''(x) \neq \lim_{x \to \psi^-} f'''(x)$.

(2) $\lim_{x \to \psi^+} f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-\psi)^3$.

$= \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-\psi)(x^2+\psi^2-2x\psi)$.

$= \beta_0 + \beta_1 x (\beta_0 - \beta_1 \psi^3) + (\beta_1 + 3\beta_4 \psi^2) x^2 + (\beta_2 - 3\beta_4 \psi) x^2 + (\beta_3 + \beta_4) x^3$

$\lim_{x \to \psi^-} f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$.

Hence both $\lim_{x \to \psi^+} f(x)$ & $\lim_{x \to \psi^-} f(x)$ are piece-wise cubic polynomial

We can see that all the above mentioned conditions are met and hence it is proved that the mentioned function is always a cubic spline with knot at ψ.

4.

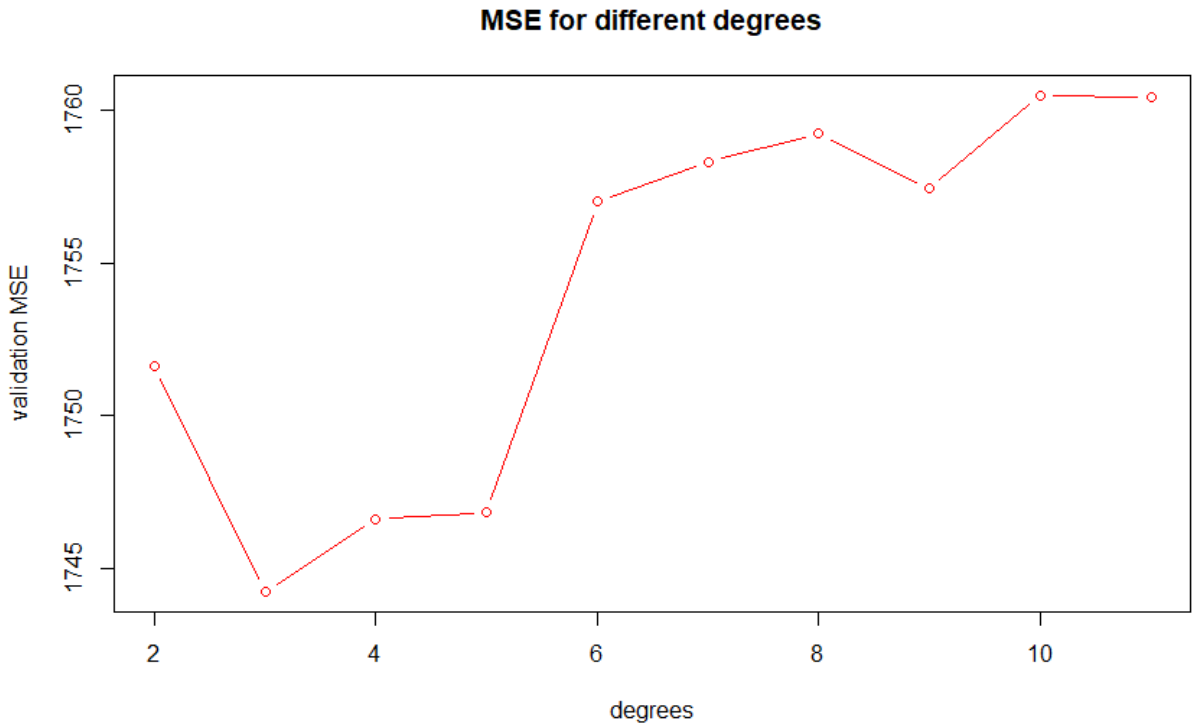
```
library(ISLR2)
attach(Wage)

par(mfrow = c(1,1))

get_mse = function(estimate, truth) {
  mean(mean((estimate - truth) ^ 2))
}

set.seed(2037)
df <- data.frame(Wage)
train=sample(1:nrow(df),2*nrow(df)/3)
X <- data.frame(df[, 'age'])
colnames(X) <- 'age'
Y <- data.frame(df['wage'])
colnames(Y) <- 'wage'
df_train <- df[train,c('age','wage')]
df_test <- df[-train,c('age','wage')]
```

A. Polynomial

```
poly.MSE.arr <- c()
n.degree <- c(2,3,4,5,6,7,8,9,10,11)
for(i in n.degree){
  poly.fit <- lm(wage ~ poly(age , i),data=df_train[valid,])
  yhat <- predict(poly.fit,newdata=df_train[-valid,],se=TRUE)
  MSE<-get_mse(yhat$fit,df_train[-valid,]$wage)
  poly.MSE.arr<-append(poly.MSE.arr,MSE)
}
print('the degree with least MSE is: ')
print(n.degree[which.min(poly.MSE.arr)])

plot(n.degree,poly.MSE.arr,type='b',col='red',xlab='degrees',ylab='validation MSE')
title('MSE for different degrees')

#fitting with the best degree of polynomial
poly.fit <- lm(wage ~ poly(age , 3),data=df_train)
yhat <- predict(poly.fit,newdata=df_test,se=TRUE)
poly.MSE<-get_mse(yhat$fit,df_test$wage)


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

#plot the polynomial function
poly.predict <- predict(poly.fit,newdata=list(age=age.grid),se=TRUE)
se.bands <- cbind (poly.predict$fit + 2 * poly.predict$se.fit ,
                   poly.predict$fit - 2 * poly.predict$se.fit)

plot (df_test$age , df_test$wage , xlim = agelims , cex = .5, col = " darkgrey ",xlab='age',ylab='wage')
title (" Degree -3 Polynomial ")
lines (age.grid, poly.predict$fit , lwd = 2, col = " blue ")
matlines (age.grid , se.bands, lwd = 1, col = " blue ", lty = 3)
```

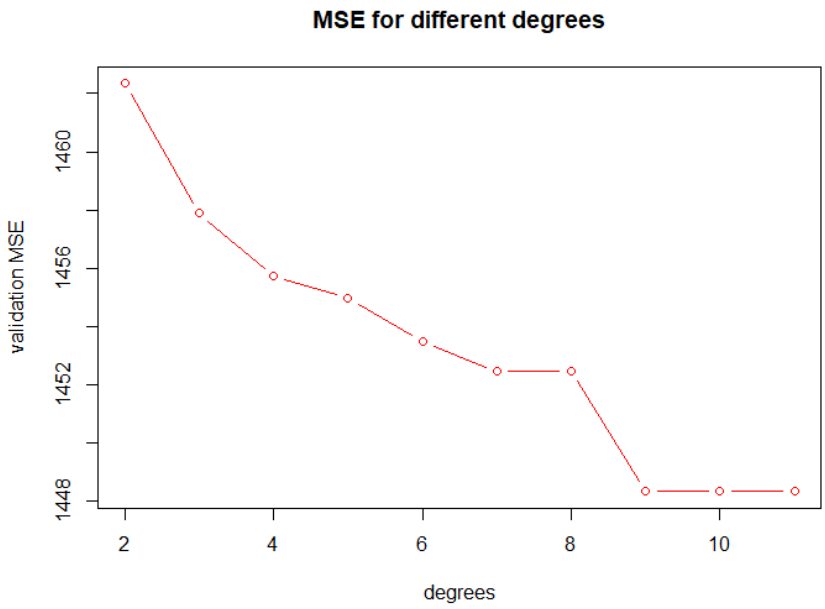
Validation set approach was used to find the degree of polynomial with least MSE. Three datasets were created-train,validation and test. The degree-3 was found to be the polynomial with least MSE.
Model was re-trained on the full training set with degree - 3. The resulting MSE on test data was **1457.88.**

The Validation MSEs for different degrees was plotted.

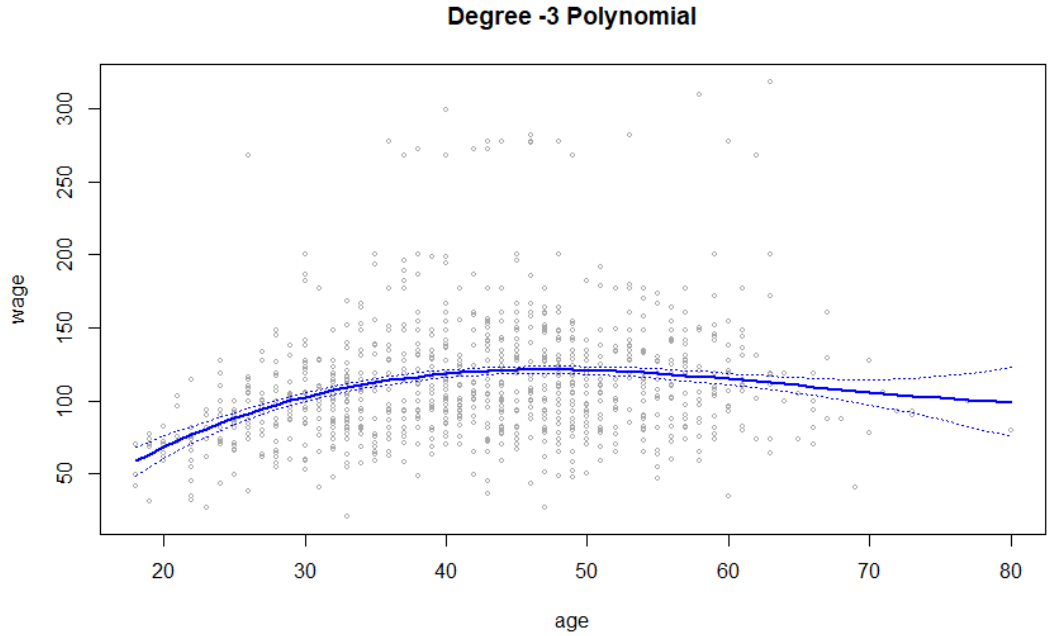


Note- Using validation set approach, the flexibility vs MSE curve acts opposed to our general expectation. With increase in flexibility as number of degrees increases, expected MSE increases but ideally it should decrease. This may be due to less number of observations used in the training and validation. When validation set approach was used on full training and test data, the flexibility curve is generated as expected. But for purpose of tuning parameters, a different validation set is created from training data and the curve was plotted as above.

Below is plot of MSEs vs number of degrees when validation set approach is used on full training and test data. In this case there is a steep decrease in MSE at degree=3 but it keeps on decreasing with further increase in number of degrees.



The polynomial fitted on the entire training set was plotted.

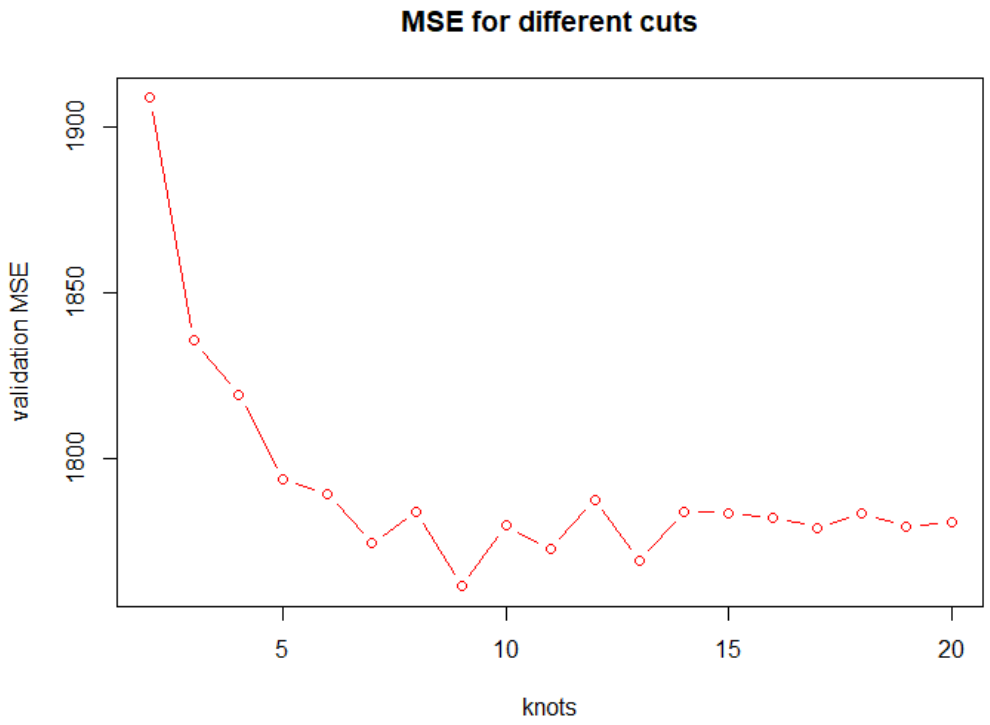


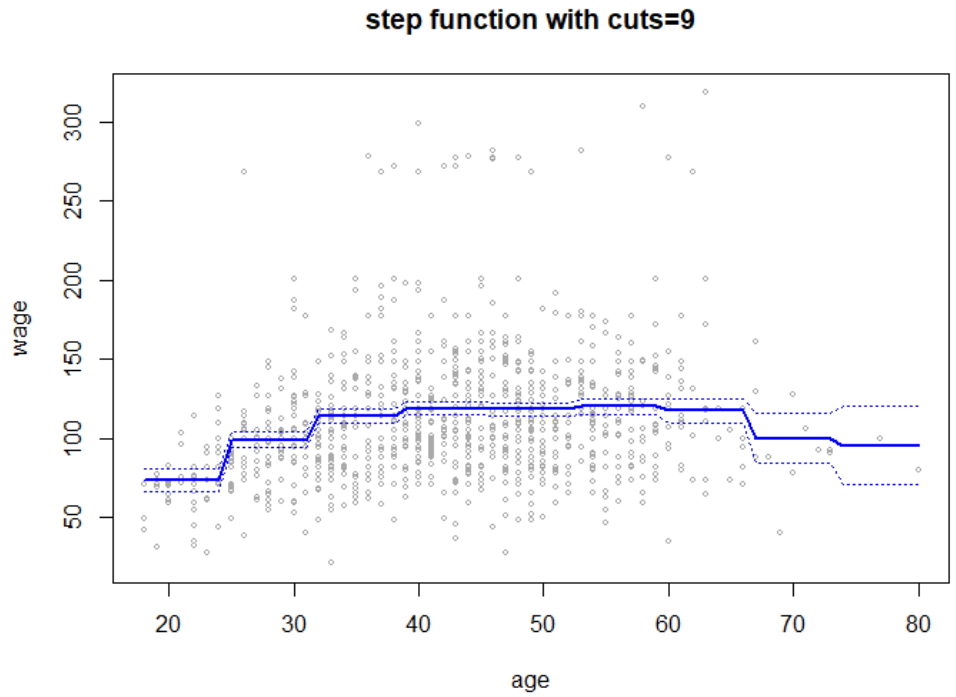
The polynomial regression model with degree=3 fits well except at the boundaries

B.

```
n.cuts <- seq(2,20,1)
step.MSE.arr<- c()
for(i in n.cuts){
  step.fit <- lm(wage ~cut (age , i),data=df_train[valid,])
  yhat <- predict(step.fit,newdata=df_train[-valid,],se=TRUE)
  MSE<-get_mse(yhat$fit,df_train[-valid,]$wage)
  step.MSE.arr<-append(step.MSE.arr,MSE)
}
print('the number of cuts with least MSE is: ')
print(n.knots[which.min(step.MSE.arr)])

plot(n.knots,step.MSE.arr,type='b',col='red',xlab='knots',ylab='validation MSE')
title('MSE for different cuts')

#fitting step function with best cut value
step.fit <- lm(wage ~ cut (age , 4), data = df_train)
yhat <- predict(step.fit,newdata=df_test,se=TRUE)
step.MSE<-get_mse(yhat$fit,df_test$wage)
summary(step.fit)


#plot the step function
step.predict <- predict(step.fit,newdata=list(age=age.grid),se=TRUE)
se.bands <- cbind (step.predict$fit + 2 * step.predict$se.fit ,
                   step.predict$fit - 2 * step.predict$se.fit)

plot (df_test$age , df_test$wage , xlim = agelims , cex = .5, col = " darkgrey ",xlab='age',ylab='wage')
title (" step function with cuts=9 ")
lines (age.grid, step.predict$fit , lwd = 2, col = " blue ")
matlines (age.grid , se.bands, lwd = 1, col = " blue ", lty = 3)
```

Best number of cuts was derived through validation approach. The best cut was found to be 9.Model was re-trained on the full training set with cut- 9. The resulting MSE on test data was **1471.978.**





Clearly, the step function doesn't fit the data as good as polynomial regression and the test error is also higher than the polynomial regression model's test error.

C.

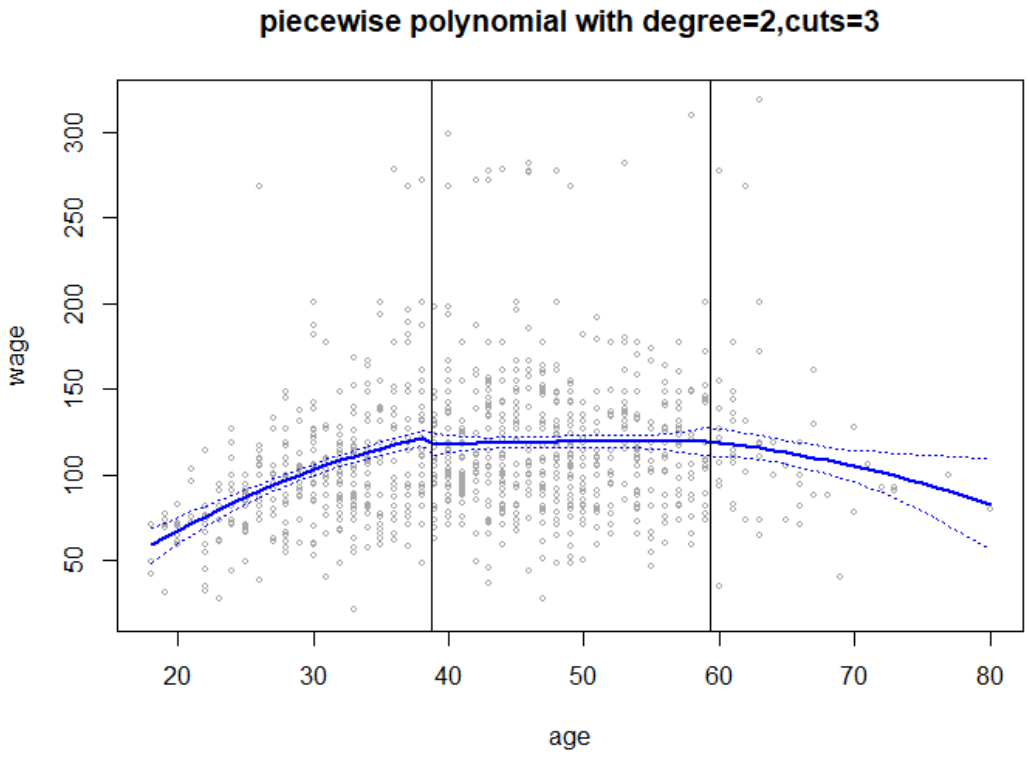
```
piece.MSE.arr<- c()
piece.var <- c()
for (i in n.degree){
  for(j in n.cuts){
    piece.fit <- lm(wage ~ cut(age,j):poly(age , i),data=df_train[valid,])
    yhat <- predict(piece.fit , newdata = df_train[-valid,])
    MSE<-get_mse(yhat,df_train[-valid,]$wage)
    piece.MSE.arr<-append(piece.MSE.arr,MSE)
    piece.var <- append(piece.var,paste(i,j))

  }
}
print('the best combination of degrees and cuts is: ')
print(piece.var[which.min(piece.MSE.arr)])


#fitting piecewise polynomial function with best degree and cut value
piece.fit <- lm(wage ~ cut(age,3):poly(age , 2), data = df_train)
yhat <- predict(piece.fit,newdata=df_test,se=TRUE)
piece.MSE<-get_mse(yhat$fit,df_test$wage)
summary(piece.fit)


#plot the step function
step.predict <- predict(piece.fit,newdata=list(age=age.grid),se=TRUE)
se.bands <- cbind (step.predict$fit + 2 * step.predict$se.fit ,
                   step.predict$fit - 2 * step.predict$se.fit)

plot (df_test$age , df_test$wage , xlim = agelims , cex = .5, col = " darkgrey ",xlab='age',ylab='wage')
title (" piecewise polynomial with degree=2,cuts=3 ")
lines (age.grid, step.predict$fit , lwd = 2, col = " blue ")
matlines (age.grid , se.bands, lwd = 1, col = " blue ", lty = 3)
```

Best number of degrees and cuts were derived through validation approach. The best cut was found to be 3 and degree to be 2. Model was re-trained on the full training set with cut- 3 and degree-2. This function has a total of 12 degrees of freedom. The resulting MSE on test data was **1450.633.**

```
> piece.fit <- lm(wage ~ cut(age,3):poly(age , 2), data = df_train)
> summary(piece.fit)

Call:
lm(formula = wage ~ cut(age, 3):poly(age, 2), data = df_train)

Residuals:
   Min     1Q Median     3Q    Max
-98.43 -25.06  -5.29  15.31 199.41

Coefficients:
                                    Estimate Std. Error t value Pr(>|t|)
(Intercept)                          117.451      5.345  21.972  < 2e-16 ***
cut(age, 3)(17.9,38.7]:poly(age, 2)1  474.442    287.284   1.651  0.09880 .
cut(age, 3)(38.7,59.3]:poly(age, 2)1   80.128    220.302   0.364  0.71611
cut(age, 3)(59.3,80.1]:poly(age, 2)1  212.041    266.274   0.796  0.42594
cut(age, 3)(17.9,38.7]:poly(age, 2)2 -539.437    167.761  -3.216  0.00132 **
cut(age, 3)(38.7,59.3]:poly(age, 2)2  -54.638    282.692  -0.193  0.84676
cut(age, 3)(59.3,80.1]:poly(age, 2)2 -313.755    165.475  -1.896  0.05809 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 40.77 on 1993 degrees of freedom
Multiple R-squared:  0.08622,   Adjusted R-squared:  0.08347
F-statistic: 31.34 on 6 and 1993 DF,  p-value: < 2.2e-16
```

```
#plot the step function
step.predict <- predict(piece.fit,newdata=list(age=age.grid),se=TRUE)
se.bands <- cbind (step.predict$fit + 2 * step.predict$se.fit ,
                   step.predict$fit - 2 * step.predict$se.fit)

plot (df_test$age , df_test$wage , xlim = agelims , cex = .5, col = " darkgrey ",xlab='age',ylab='wage')
title (" piecewise polynomial with degree=2,cuts=3 ")
lines (age.grid, step.predict$fit , lwd = 2, col = " blue ")
matlines (age.grid , se.bands, lwd = 1, col = " blue ", lty = 3)
abline(v=38.7)
abline(v=59.3)
```

A piece-wise polynomial function with degree=2 and cut=3 was plotted.



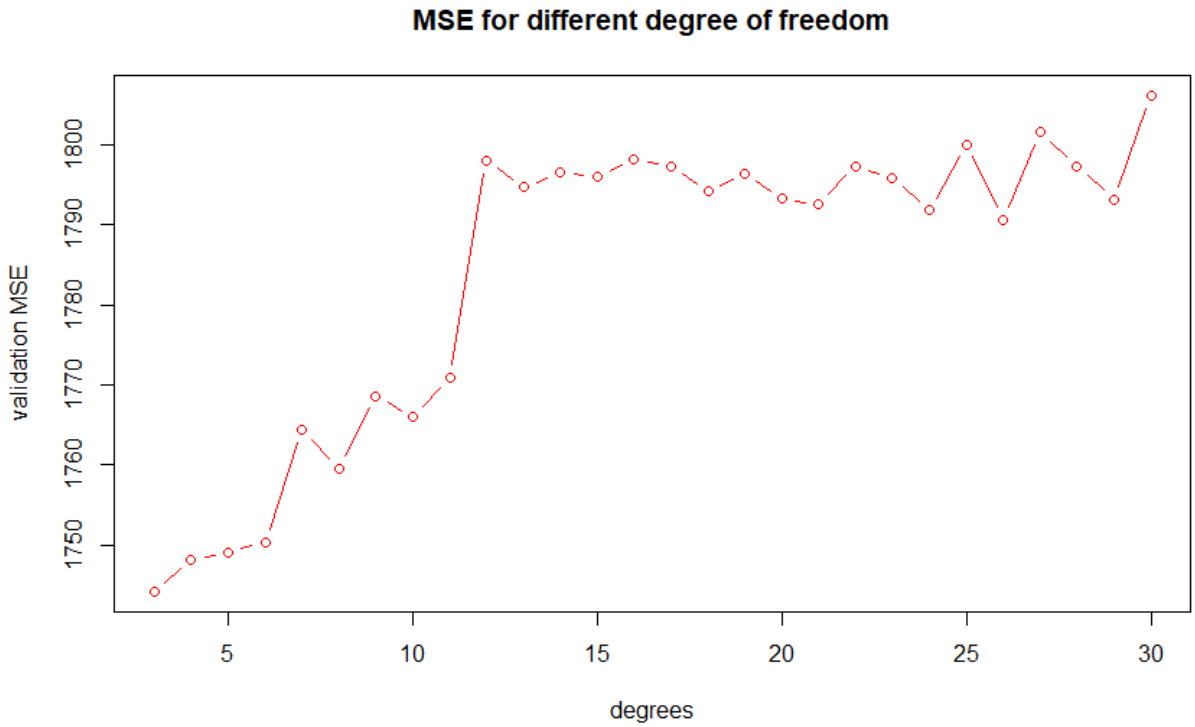
D.

```
library(splines)
spline.MSE.arr <- c()
n.dof <- seq(3,30,1)
for(i in n.dof){
  spline.fit <- lm(wage ~ bs(age , df=i,degree=3), data = df_train[valid,])
  yhat <- predict(spline.fit,newdata=df_train[-valid,],se=TRUE)
  MSE<-get_mse(yhat$fit,df_train[-valid,]$wage)
  spline.MSE.arr<-append(spline.MSE.arr,MSE)
}
print('the dof with least MSE is: ')
print(n.dof[which.min(spline.MSE.arr)])

plot(n.dof,spline.MSE.arr,type='b',col='red',xlab='degrees',ylab='validation MSE')
title('MSE for different degree of freedom')


spline.fit <- lm(wage ~ bs(age , df=4,degree=3), data = df_train)
yhat <- predict(spline.fit,newdata=df_test,se=TRUE)
spline.MSE<-get_mse(yhat$fit,df_test$wage)
summary(spline.fit)


splines.predict <- predict (spline.fit , newdata = list (age = age.grid), se = T)
#spline_MSE <- get_mse(splines.predict$fit,age.grid)
se.bands <- cbind (splines.predict$fit + 2 * splines.predict$se.fit ,
                   splines.predict$fit - 2 * splines.predict$se.fit)
plot (df_test$age , df_test$wage , xlim = agelims , cex = .5, col = " darkgrey ",xlab='age',ylab='wage')
title (" cubic spline with dof=5 ")
lines (age.grid, splines.predict$fit , lwd = 2, col = " red ")
matlines (age.grid , se.bands, lwd = 1, col = " red ", lty = 3)
```

Best degree of freedom was derived through the validation set approach. The best dof found to be 3 but this will essentially become a cubic polynomial with 0 knots. Hence, we can increase the dof and get better results while

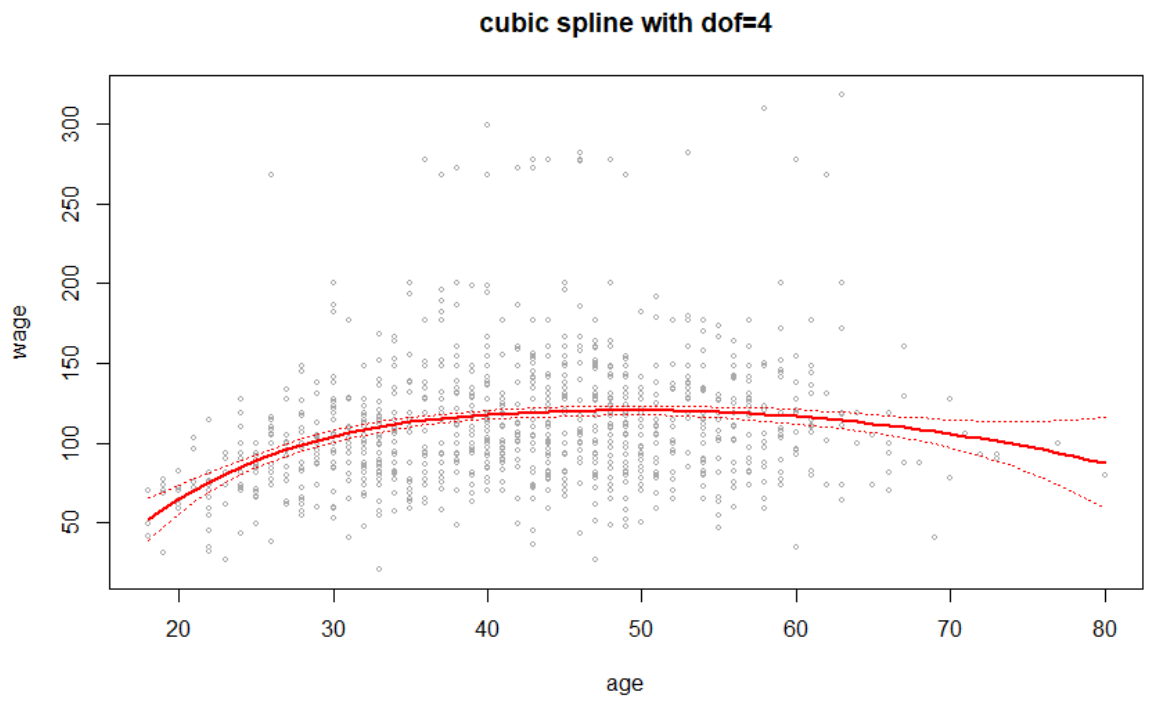
dof being still less than dof of a cubic polynomial which has 4 degrees of freedom. With the same degree of freedom, we can get a cubic spline with 1 knot. Model was re-trained on the full training set with df=4. The resulting MSE on test data was **1456.842.**

The Validation MSEs for different degree of freedoms was plotted.



Note-With increase in degrees,the MSE increases but ideally it should decrease. This could be due to the data in the validation set and since less observations are used for training.

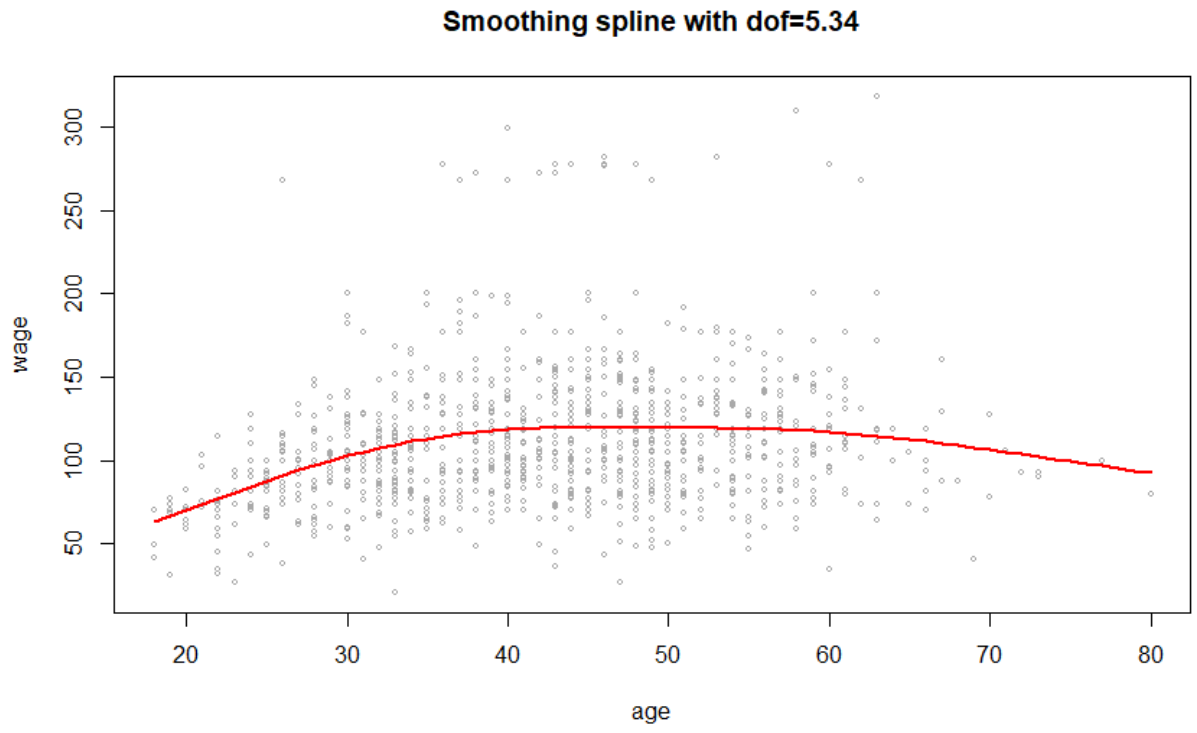
A cubic spline function with dof=4 was plotted for different values of age.



E.

```
smooth.fit <- smooth.spline(df_train$age ,df_train$wage ,cv=TRUE)
smooth.fit$df
smooth.fit <- smooth.spline(df_train$age,df_train$wage, df=5.34 )


smooth.predict1 <- predict (smooth.fit , df_test$age)

#y_actual <- df_test['wage'][df_test['age']== xx]
smooth_MSE1 <- get_mse(smooth.predict1$y,df_test$wage)

smooth.predict <- predict(smooth.fit , age.grid)
plot (df_test$age , df_test$wage , xlim = agelims , cex = .5, col = " darkgrey ",xlab='age',ylab='wage')
title (" Smoothing spline with dof=5.34 ")
lines (age.grid, smooth.predict$y , lwd = 2, col = " red ")
```

Best degree of freedom was derived through cross validation approach. The best dof found to be 5.34. Model was re-trained on the full training set with df=5.34. The resulting MSE on test data was **1453.608.**
This gives better results than all other functions except the piece-wise polynomial which had much higher degree of freedom than smoothing spline.
The smoothing spline with degree of freedom = 5.34 was plotted.



Overall piece-wise polynomial yields the best result but it is complex and uses more degree of freedoms. On the other hand, splines and smoothing splines perform better with added flexibility but with less degree of freedom. Let's see the results if we increase the degree of freedom for cubic splines and smoothing splines but keeping it less than the piece-wise polynomial.
With df=10, test MSE in cubic spline was **1447.744.**
With df=10, test MSE in smoothing spline was **1499.957**

Hence, it was observed that spline and smoothing spline yields better results with less dof and added flexibility.

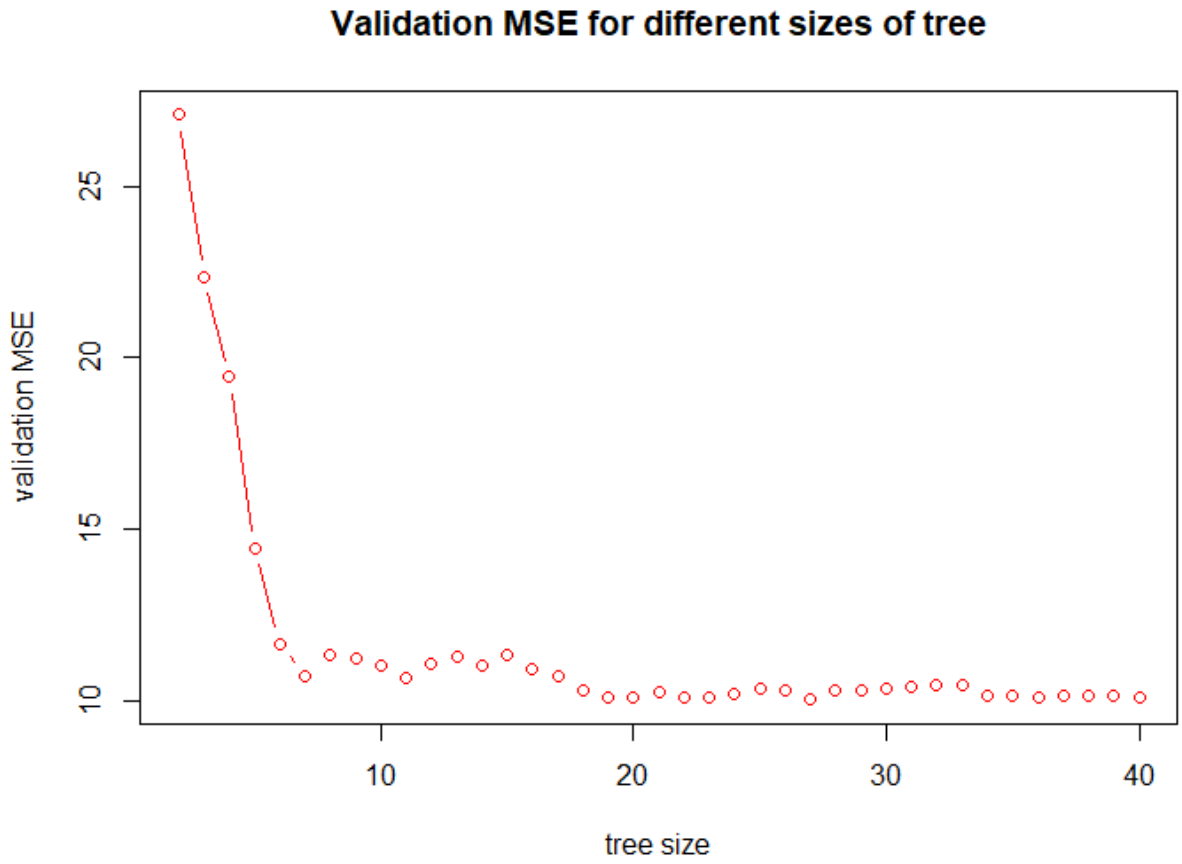
5.

A.

```
library(ISLR2)

par(mfrow=c(1,1))
set.seed(291)
valid=sample(1:length(train),length(train)/2)

df_Auto <- data.frame(Auto)
head(df_Auto)
library (tree)
df_Auto$year <- as.factor(df_Auto$year)
df_Auto$origin <- as.factor(df_Auto$origin)
set.seed(100)
train=sample(1:nrow(df_Auto),2*nrow(df_Auto)/3)
df_train <- df_Auto[train,][names(df_Auto)!='name']
df_test <- df_Auto[-train,][names(df_Auto)!='name']


set.seed(100)
tree.auto <- tree (mpg ~ .  , df_train,control = tree.control(nobs = length(train), mindev = 0))
summary(tree.auto)
plot (tree.auto)
text (tree.auto , pretty = 0)
cv.auto <- cv.tree(tree.auto)
plot(cv.auto$size , cv.auto$dev, type = "b")

#what is the difference between deviance and MSE
tree.size.par<- seq(2,40,1)
tree.MSE <- c()
for (i in tree.size.par){
  print(i)
  tree.mod <- prune.tree(tree.auto, best = i)
  yhat <- predict(tree.mod , newdata = df_test)
  MSE<-get_mse(yhat,df_test$mpg)
  print(MSE)
  tree.MSE<-append(tree.MSE,MSE)

}

plot(tree.size.par,tree.MSE,type='b',col='red',xlab='tree size',ylab='validation MSE')
title (" Validation MSE for different sizes of tree ")

prune.auto <- prune.tree(tree.auto, best = 6)
plot (prune.auto)
text (prune.auto , pretty = 0)

yhat_tree <- predict(prune.auto , newdata = df_test)
tree_MSE <- get_mse(yhat_tree,df_test$mpg)

plot(yhat_tree,df_test$mpg, xlab = "Predicted mpg", ylab = "Actual mpg", main = "Pruned Regression model")
abline(0, 1, col = "blue")
```

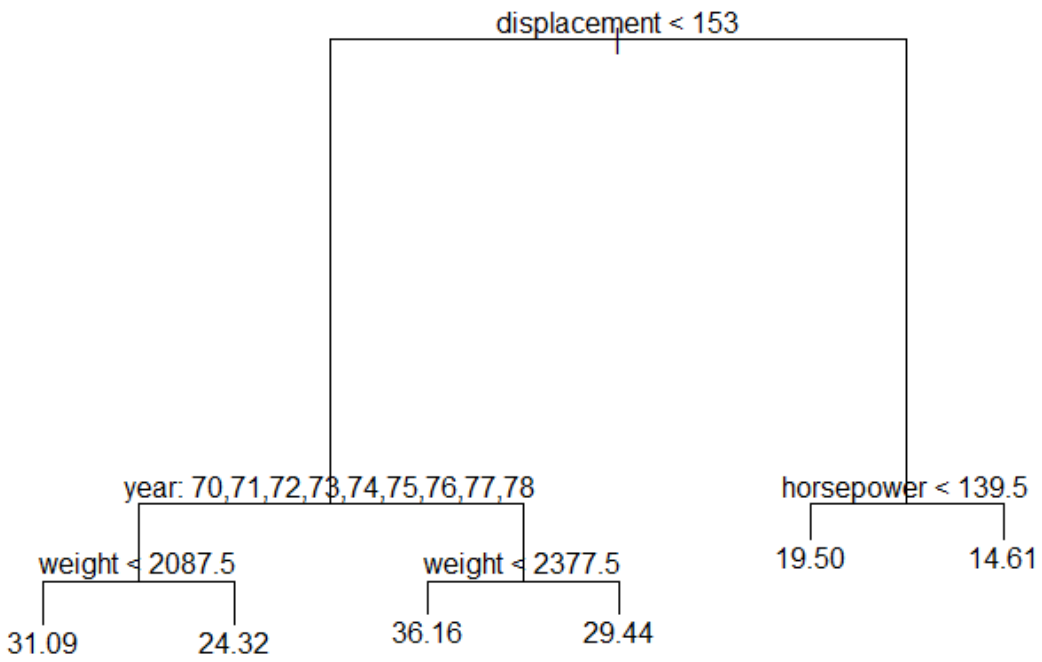
The best value for size of tree was derived using validation set approach. At best=6, the MSE was the lowest in the validation set. The full grown tree was pruned to have total number of terminal nodes=6 and then prediction on test data was made. The resulting MSE on test data was **11.67434.**

The Validation MSEs for different sizes of the tree were plotted.



It can be observed from the above chart that the MSE is lowest when the tree size=6. Hence we pruned our model to have size=6.

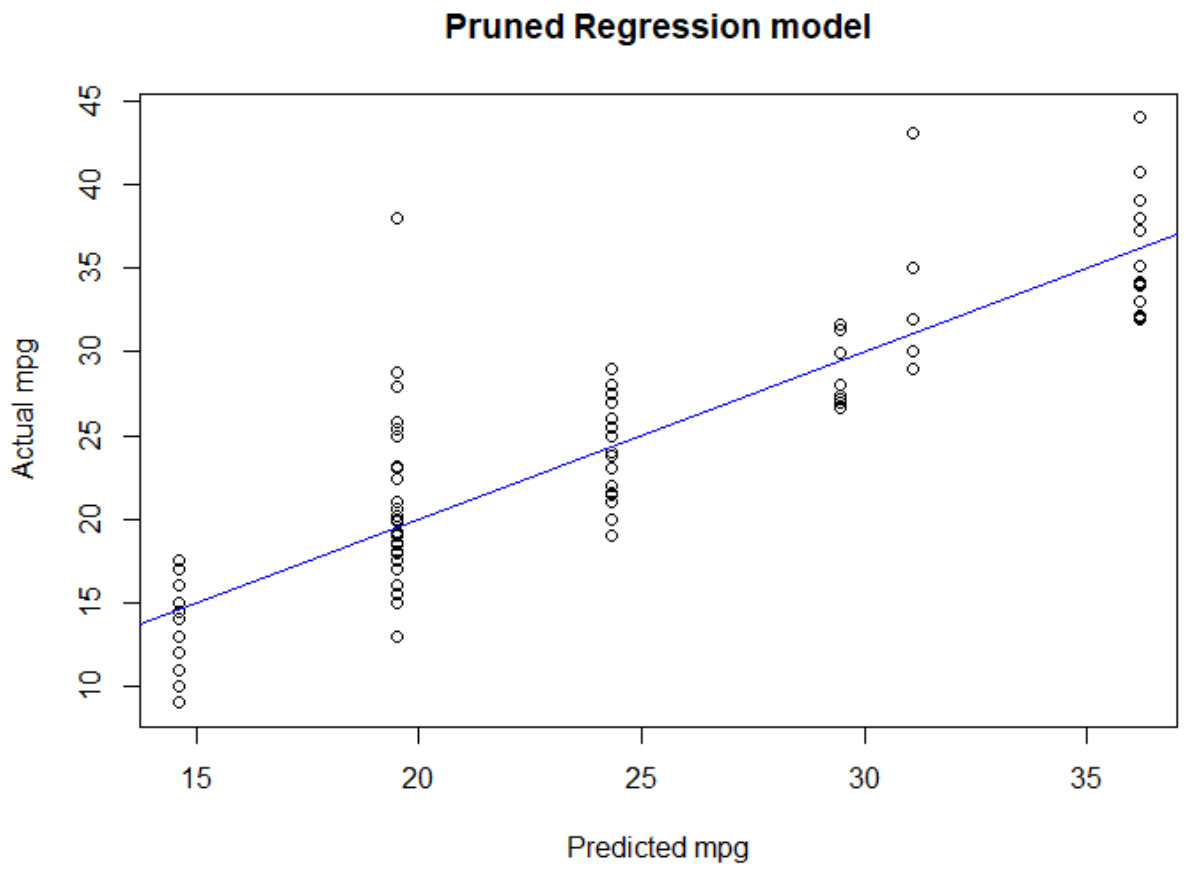
The final pruned tree was plotted.



From the above tree we can infer that the highest mean mpg among the data of 36.16 belongs to the cars with displacement less than 153, weight less than 2377.5 and manufacturing years not between 70 and 78. The lowest mean mpg (14.61) belongs to the cars with displacement more than 153 and whose horsepower is more than 139.5.

The MSE of 11.67 suggests that the test predictions are within sqrt(11.67) i.e. 3.41 of the actual mpg values.

```
plot(yhat_tree,df_test$mpg, xlab = "Predicted mpg", ylab = "Actual mpg", main = "Pruned Regression model")
abline(0, 1, col = "blue")
```

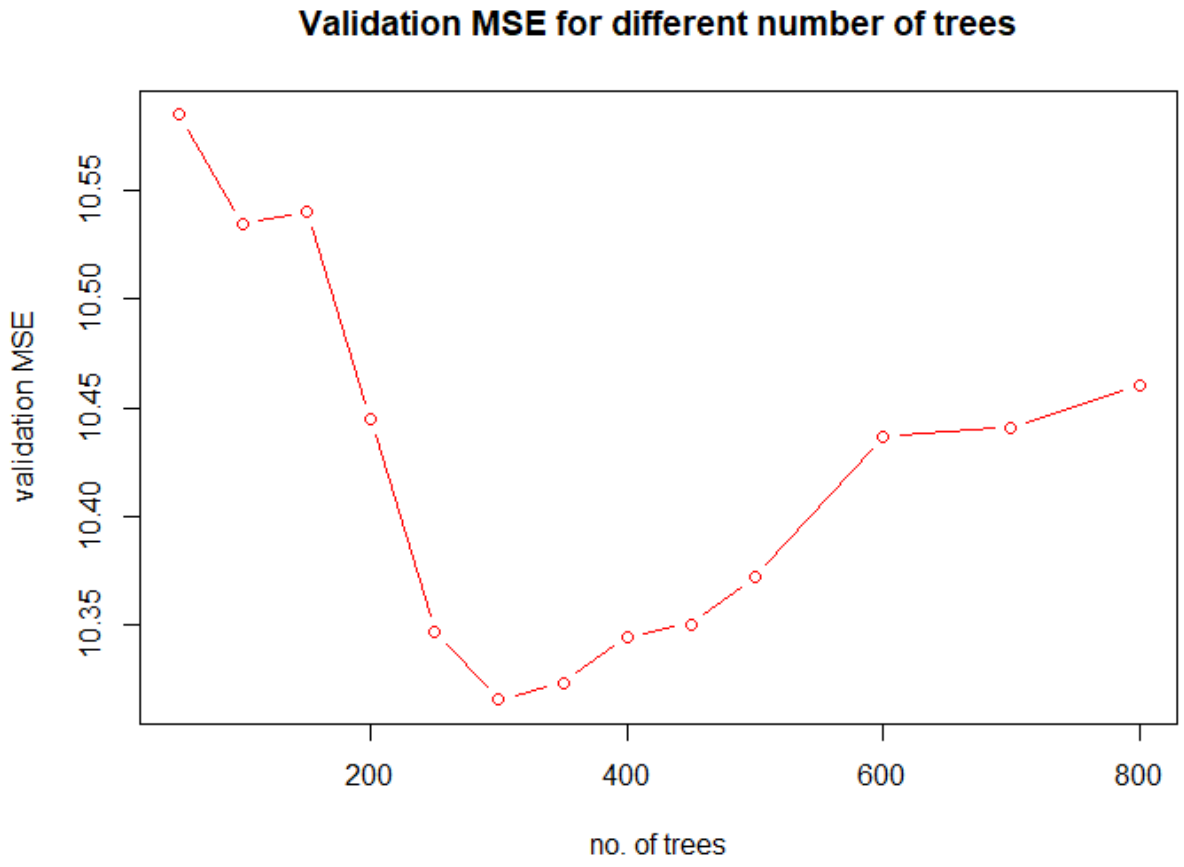


B.

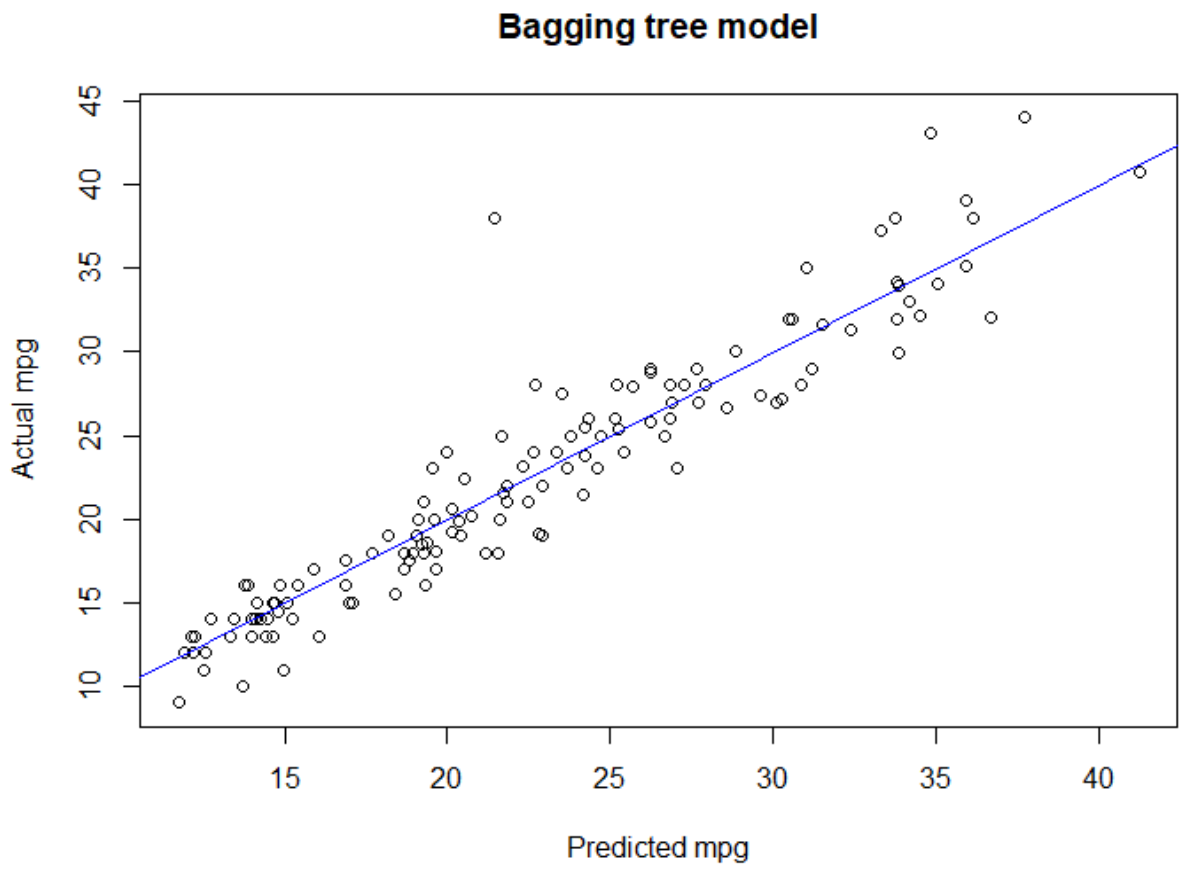
```
library(randomForest)
set.seed(101)

n.trees <- c(50,100,150,200,250,300,350,400,450,500,600,700,800)
bag.MSE <- c()
for (i in n.trees){
  set.seed(101)
  bag.mod <- randomForest (mpg ~ ., data = df_train[valid,] , mtry = ncol(df_train[valid,])-1,ntree=i, importance = TRUE)
  yhat <- predict(bag.mod , newdata = df_train[-valid,])
  MSE<-get_mse(yhat,df_train[-valid,]$mpg)
  bag.MSE<-append(bag.MSE,MSE)

}
plot(n.trees,bag.MSE,type='b',col='red',xlab='no. of trees',ylab='validation MSE')
title('validation MSE for different number of trees' )

set.seed(101)
bag.auto <- randomForest (mpg ~ ., data = df_train , mtry = ncol(df_train)-1,ntree=300, importance = TRUE)
yhat <- predict(bag.auto , newdata = df_test)
bag.MSE<-get_mse(yhat,df_test$mpg)

plot(yhat,df_test$mpg, xlab = "Predicted mpg", ylab = "Actual mpg", main = "Bagging tree model")
abline(0, 1, col = "blue")
```

The hyperparameter used to tune this model was number of trees. Different values for number of trees were tried ranging from 50 to 800. Since, this is a bagging tree model, all the features will be used in each tree and hence mtry is fixed. The best value for number of trees was derived using validation set approach. At ntree=300, the MSE was the lowest in the validation set.The model was re-trained on the entire training data using ntree=300.

The resulting MSE on test data was **6.77479.** This test error is clearly less than the error in a decision tree model.




```
plot(yhat,df_test$mpg, xlab = "Predicted mpg", ylab = "Actual mpg", main = "Bagging tree model")
abline(0, 1, col = "blue")
```

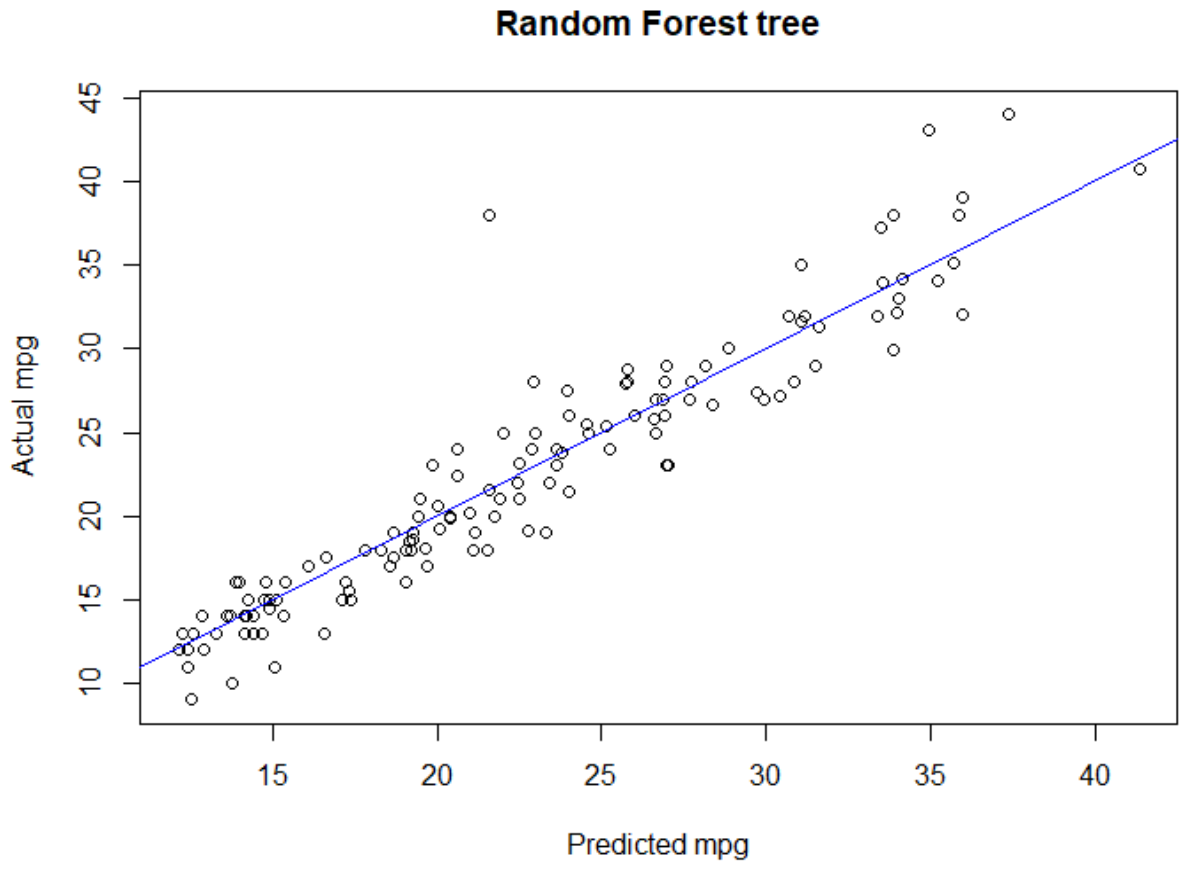


From the above chart, it is clear that the predicted values of mpg are close to the actual values of mpg.The MSE of 6.77 suggests that the test predictions are within sqrt(6.77) i.e. 2.60 of the actual mpg values.

C.

```
set.seed(101)

n.trees <- c(50,100,150,200,250,300,350,400,450,500,600,700,800)
n.var <- c(2,3,4,5,6)
rf.MSE <- c()
n.tree.var <-c()
for (i in n.trees){
  for(j in n.var){
  set.seed(101)
  rf.mod <- randomForest (mpg ~ ., data = df_train[valid,] , mtry = (ncol(df_train[valid,])-1)/j,ntree=i, importance = TRUE)
  yhat <- predict(rf.mod , newdata = df_train[-valid,])
  MSE<-get_mse(yhat,df_train[-valid,]$mpg)
  rf.MSE<-append(rf.MSE,MSE)
  n.tree.var <- append(n.tree.var,paste(i,j))

  }
}
print('the best combination of trees and variable subset is: ')
print(n.tree.var[which.min(rf.MSE)])

set.seed(101)
#bag.auto <- randomForest (mpg ~ ., data = df_train[valid,] , mtry = ncol(df_train[valid,])-1,sampsize=length(valid), importance = TRUE)
rf.auto <- randomForest (mpg ~ ., data = df_train , mtry = (ncol(df_train[valid,])-1)/2,ntree=150, importance = TRUE)
yhat <- predict(rf.auto , newdata = df_test)
rf.MSE<-get_mse(yhat,df_test$mpg)
```

The hyperparameters used to tune this model were number of trees and number of features. Different values for number of trees and number of features were tried ranging from 50 to 800 alongwith the fraction of features to be used. The best combination was derived using validation test approach. Since, this is a random forest model, a subset of the features can be used in each tree and hence mtry is not fixed. The best value for number of trees and fraction of features were derived using validation set approach. At ntree=50 and mtry=2, the MSE was the lowest in the validation set.The model was re-trained on the entire training data using ntree=50 and mtry=2. The resulting MSE on test data was **6.755885** which is slightly lower than the bagging test MSE. We can observe that random forest is able to achieve similar results with less number of trees than the bagging model. This test error is clearly less than the error in a decision tree model.



The MSE of 6.75 suggests that the test predictions are within sqrt(6.75) i.e. 2.59 of the actual mpg values.

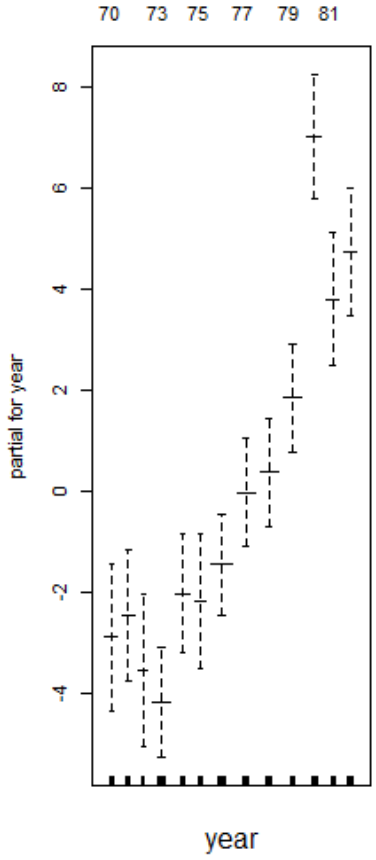
D.

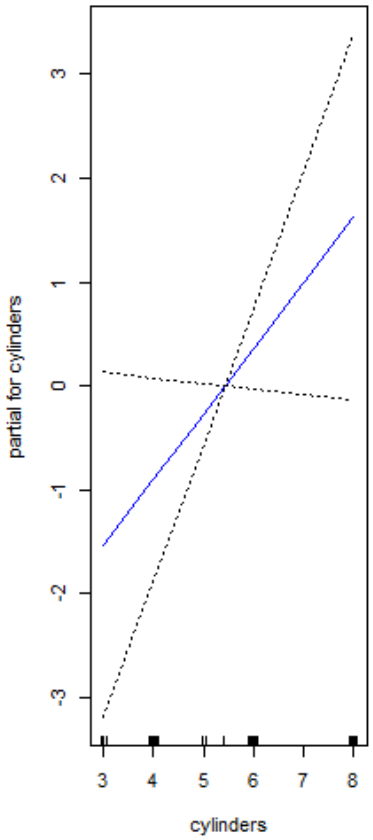
```
library(gam)

gam.m1 <- gam (mpg ~ year + cylinders + s(displacement,3) + s(horsepower,3) + s(weight,3) + s(acceleration,3) + origin ,
               data = df_train)
gam.m2 <- gam (mpg ~ year + s(cylinders,3) + displacement + horsepower + weight + acceleration + origin ,
               data = df_train)
gam.m3 <- gam (mpg ~ year + cylinders + displacement + s(horsepower,3) + s(weight,3) + s(acceleration,3) + origin ,
               data = df_train)
gam.m4 <- gam (mpg ~ year + cylinders + displacement + horsepower + s(weight,3) + s(acceleration,3) + origin ,
               data = df_train)
gam.m5 <- gam (mpg ~ year + cylinders + displacement + horsepower + weight + s(acceleration,3) + origin ,
               data = df_train)
par(mfrow=c(1,3))
plot (gam.m1, se = TRUE , col = " blue ")

gam.m1.MSE <- get_mse(predict(gam.m1,df_test),df_test$mpg)
gam.m1.MSE
gam.m2.MSE <- get_mse(predict(gam.m2,df_test),df_test$mpg)
gam.m2.MSE
gam.m3.MSE <- get_mse(predict(gam.m3,df_test),df_test$mpg)
gam.m3.MSE
gam.m4.MSE <- get_mse(predict(gam.m4,df_test),df_test$mpg)
gam.m4.MSE
gam.m5.MSE <- get_mse(predict(gam.m5,df_test),df_test$mpg)
gam.m5.MSE
```

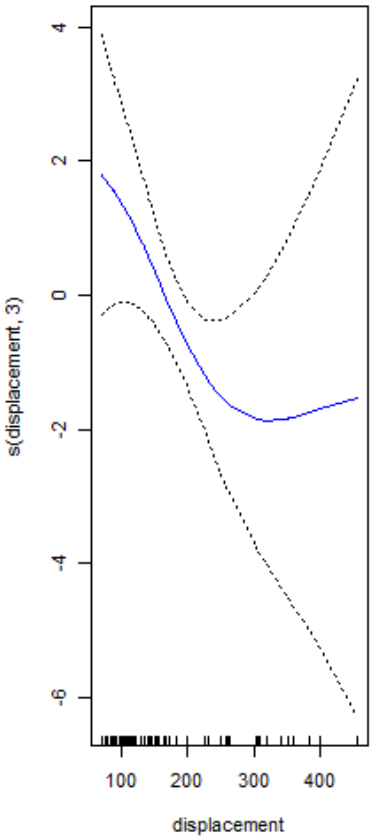
```
> gam.m1.MSE <- get_mse(predict(gam.m1,df_test),df_test$mpg)
> gam.m1.MSE
[1] 6.547371
> gam.m2.MSE <- get_mse(predict(gam.m2,df_test),df_test$mpg)
> gam.m2.MSE
[1] 9.38907
> gam.m3.MSE <- get_mse(predict(gam.m3,df_test),df_test$mpg)
> gam.m3.MSE
[1] 6.521332
> gam.m4.MSE <- get_mse(predict(gam.m4,df_test),df_test$mpg)
> gam.m4.MSE
[1] 6.659368
> gam.m5.MSE <- get_mse(predict(gam.m5,df_test),df_test$mpg)
> gam.m5.MSE
[1] 9.278731
> gam.m1
Call:
gam(formula = mpg ~ year + cylinders + s(displacement, 3) + s(horsepower,
    3) + s(weight, 3) + s(acceleration, 3) + origin, data = df_train)

Degrees of Freedom: 260 total; 232.9998 Residual
Residual Deviance: 1699.53
```

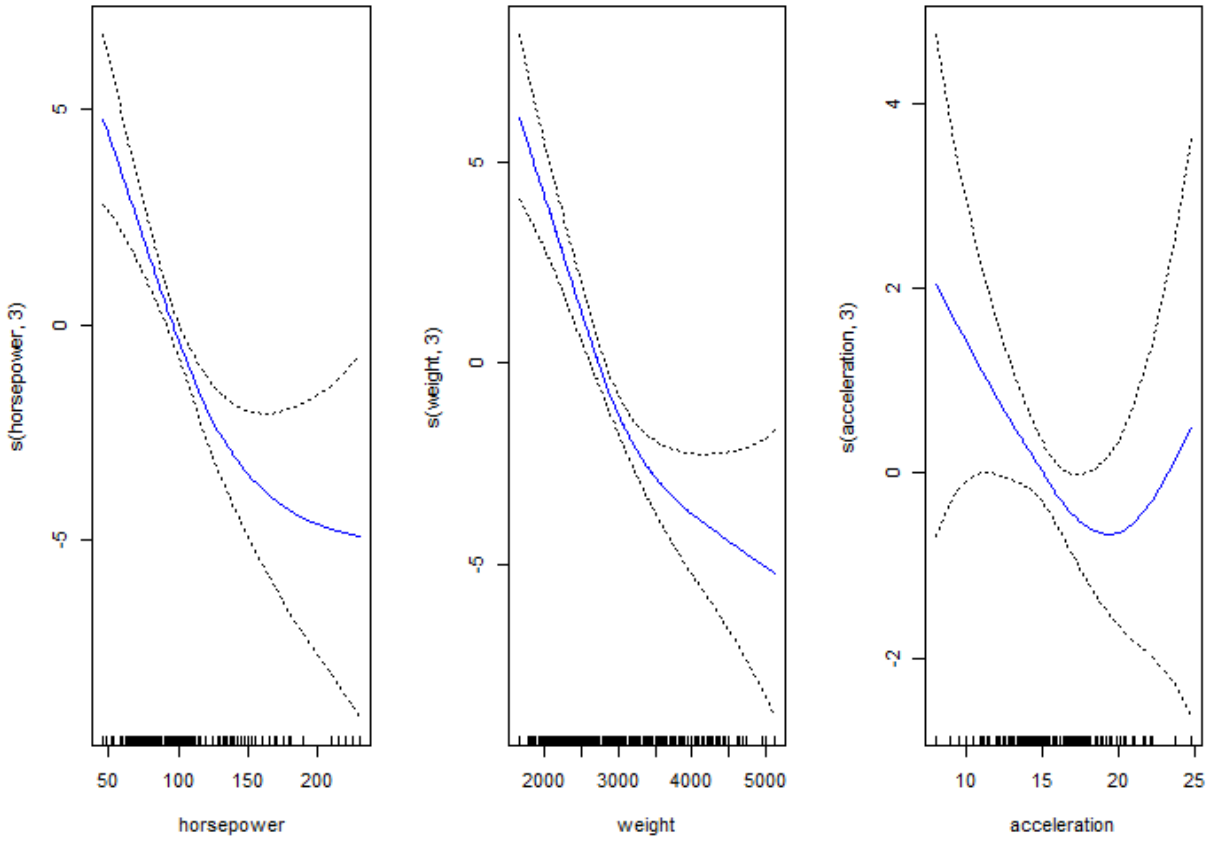
For GAMs, different additive models were tried with highest degree=3. It was observed that lowest test MSE was found in model1 which was **6.5473.** This result is even better than Random forest and bagging model but this comes at a cost of more complexity and is less intuitive than decision trees. It is hard to draw inference from the results of the GAM models if more complex additive functions are used. In this case, we have used spline functions but the individual contributions of the terms can be interpreted similar to a linear regression model.

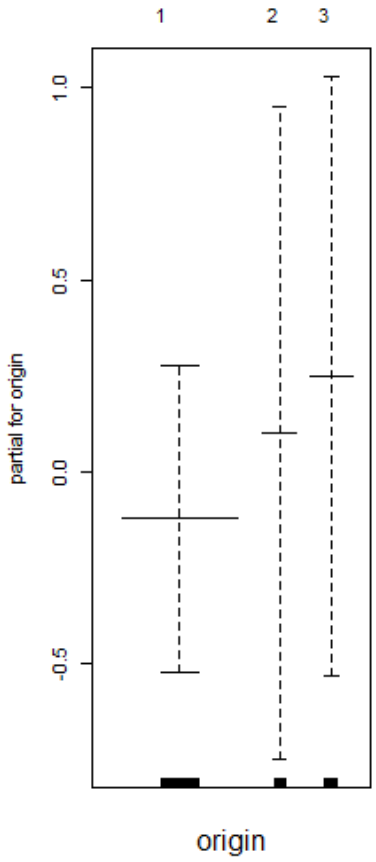
The fitted model 1 is visualized below:


70 73 75 77 79 81
partial for year
year


partial for cylinders
cylinders


s(displacement, 3)
displacement


s(horsepower, 3)
5
0
-5
50
100
150
200
horsepower
s(weight, 3)
5
0
-5
2000
3000
4000
5000
weight
s(acceleration, 3)
4
2
0
-2
10
15
20
25
acceleration



Few of the inferences that can be drawn from the plots of the fitted model:

1. Cars with origin 3 tends to have higher mpg values keeping all other variables constant.
2. Cars with higher weights tend to have less mpg values keeping all other variables constant. This aligns with our observations from the decision tree as well.
3. There is a general increasing trend in mpg with increase in number of cylinders keeping all other variables constant.

E. Considering accuracy, I would prefer random forest as it is easy to tune the parameters in a random forest model and it yields better results than other tree models. Comparing with GAMs, GAMs may yield better results but it's hard to determine which additive components are better. GAMs can also makes the model very complex. In terms of interpretability, I would prefer decision trees as they are the most interpretable among all the models used in this question. It's easy to see what values of variables would lead to what results. Random forest and bagging models on the other hand use many many trees and it is sort of a black box and it's hard to visualize a random forest or a bagging model.
Considering both, I would prefer GAM models as GAM yields comparable results to Random forest and it can be interpreted similar to a linear least square model.