HW4

Peiran Chen

5/13/2022

1

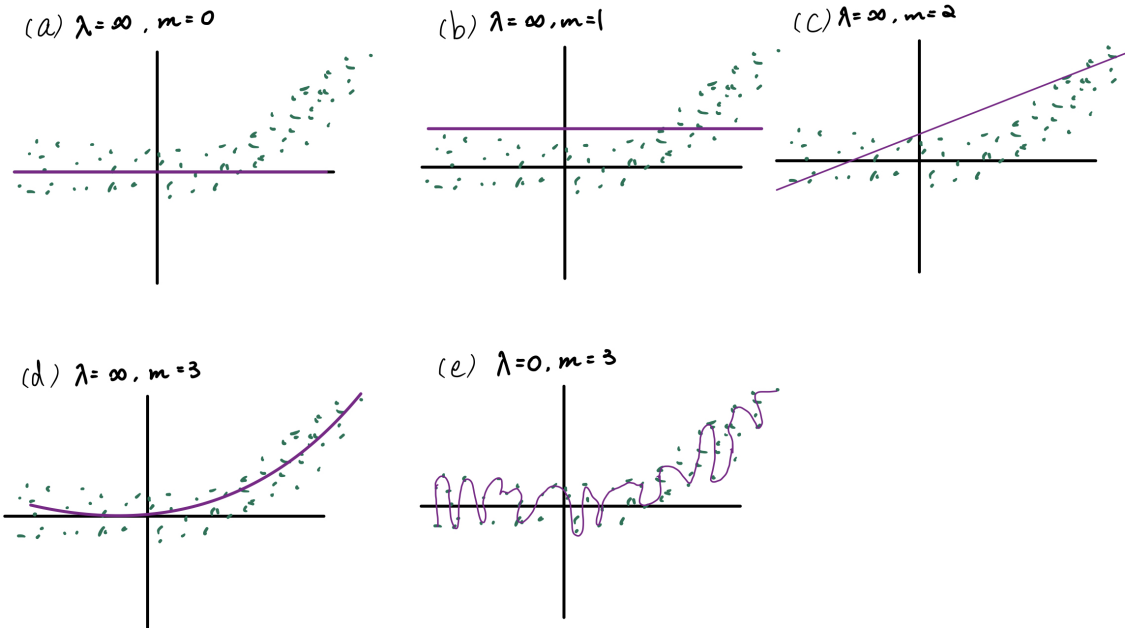


Figure 1: Question 1

For (a) to (d), when $\lambda = \infty$, we have a large penalty term that forces our function g to be smooth.

(a)

When $\lambda = \infty, m = 0$, $g^{(0)}(x) \to 0$. So,

$$\hat{g} = \arg\min_g \sum_{i=1}^{n} (y_i - g(x_i))^2 + \infty \int [g(x)]^2 dx = 0$$

.

(b)

When $\lambda = \infty, m = 1$, $g' = 0$. So,

$$\hat{g} = \arg\min_g \sum_{i=1}^{n} (y_i - g(x_i))^2 + \infty \int [g'(x)]^2 dx$$

. , it would be a horizontal line.

(c)

When $\lambda = \infty, m = 2$, $g'' = 0$.So,

$$\hat{g} = \arg\min_g \sum_{i=1}^{n} (y_i - g(x_i))^2 + \infty \int [g''(x)]^2 dx$$

. would be a straight line with slope.

(d)

When $\lambda = \infty, m = 3$, $g''' = 0$. So,

$$\hat{g} = \arg\min_g \sum_{i=1}^{n} (y_i - g(x_i))^2 + \infty \int [g'''(x)]^2 dx$$

. woule be a curve that is quadratic.

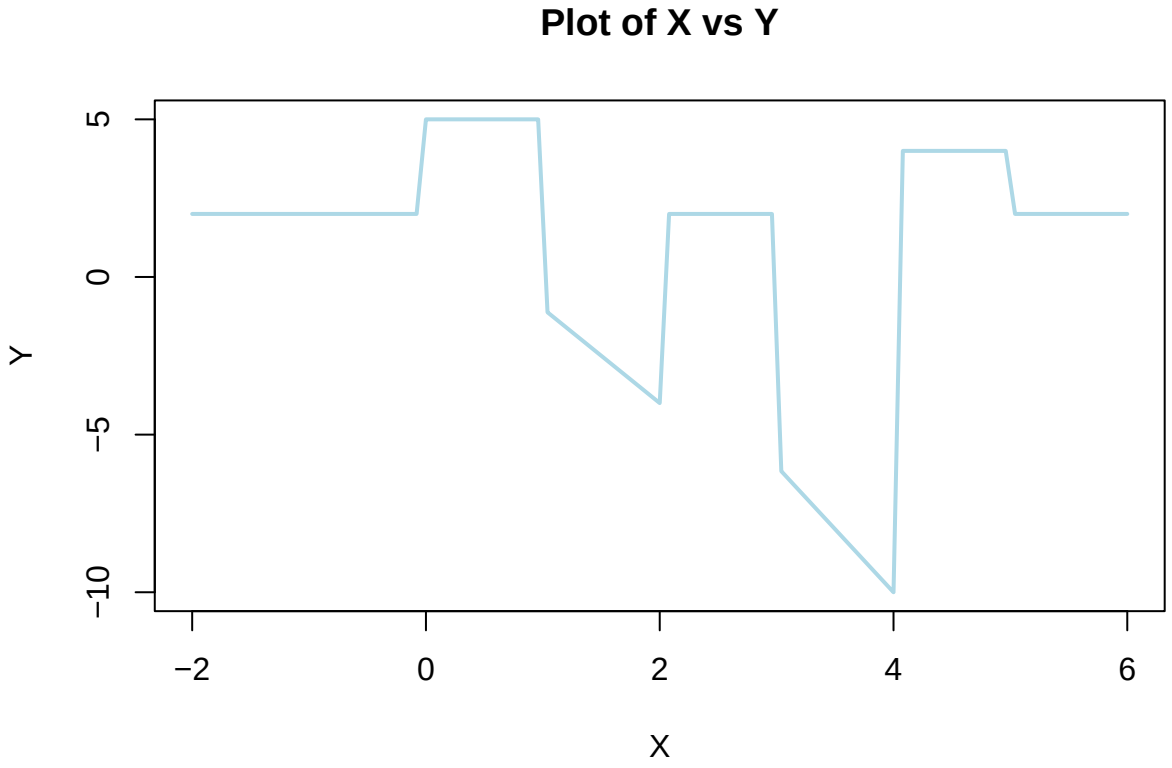
(e)

$$\hat{g} = \arg\min_g \sum_{i=1}^{n} (y_i - g(x_i))^2$$

. When $\lambda = 0$ the penalty term has no effect in our function, so we get a curve/spline that interpolates all the n points perfectly.

2.

```
curve(
  2 + 3 * (ifelse(x >= 0,
                  ifelse(x <= 2, 1, 0), 0) - (x + 1) * ifelse(x >= 1,
                  ifelse(x <= 2, 1, 0), 0)) - 2 * ((2 * x - 2) * I(x >=
                  3 & x <= 4) - I(x > 4 & x <= 5)) ,
  from = -2,
  to  = 6,
  xlab = "X",
  ylab = "Y",
  col = "lightblue",
  lwd = 2,
  main = "Plot of X vs Y"
)
```



3.

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

- We can first find a cubic polynomial for $X \leqslant \psi$

$$\begin{aligned} f_1(X) &= \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 (X - \psi)_+^3 \\ &= \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 \end{aligned}$$

- And we now have $f_1(X) = f(x)$. Then, for $X \geqslant \psi$.

$$\begin{aligned} f_2(X) &= \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 (X - \psi)^3 \\ &= \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 (X^3 - 3\psi X^2 + 3\psi^2 X - \psi^3) \\ &= (\beta_0 - \beta_4 \psi^3) + (\beta_1 + 3\psi^2) X + (\beta_2 - 3\psi\beta_4) X^2 + (\beta_3 + \beta_4) X^3 \end{aligned}$$

- We now have shown that $f_2(X) = f(X)$. Then, we just need to show that $f(X)$ is continuous at ψ.
- At $X = \psi$:

$$\begin{aligned} f_1(\psi) &= \beta_0 + \beta_1 \psi + \beta_2 \psi^2 + \beta_3 \psi^3 \\ f_2(\psi) &= (\beta_0 - \beta_4 \psi^3) + (\beta_1 + 3\psi^2)\psi + (\beta_2 - 3\psi\beta_4)\psi^2 + (\beta_3 + \beta_4)\psi^3 \\ &= \beta_0 + \beta_1 \psi + \beta_2 \psi^2 + \beta_3 \psi^3 \end{aligned}$$

Hence, we have

$$f_1(\psi) = f_2(\psi)$$

Therefore $f(X)$ is continuous at ψ.

- We now want to show that $f'(\psi) = f'(\psi)$ so that we can show $f'(X)$ is continuous at ψ.

$$\begin{aligned} f_1'(\psi) &= \beta_1 + 2\beta_2\psi + 3\beta_3\psi^2 \\ f_2'(\psi) &= \beta_1 + 3\psi^2\beta_4 + 2(\beta_2 - 3\beta_4\psi)\psi + 3(\beta_3 + \beta_4)\psi^2 \\ &= \beta_1 + 2\beta_2\psi + 3\beta_3\psi^2 \end{aligned}$$

Hence, we have

$$f_1'(\psi) = f_2'(\psi)$$

Therefore $f'(X)$ is continuous at ψ.

- We now want to show that $f''(\psi) = f''(\psi)$ so that we can show $f''(X)$ is continuous at ψ.

$$\begin{aligned} f_1''(\psi) &= 2\beta_2 + 6\beta_3\psi \\ f_2''(\psi) &= \frac{\partial}{\partial\psi} f_2'(\psi) \\ &= 2\beta_2 + 6\beta_3\psi \end{aligned}$$

Hence, we have

$$f_1''(\psi) = f_2''(\psi)$$

Therefore $f''(X)$ is continuous at ψ. Therefore, $f(x)$ is indeed a cubic spline.

4.

- To begin with, I first split the data into Training set and Validation set

```
data(Wage)

set.seed(435)
# Train-Test Split
wage_split <- initial_split(Wage, strata = age)

wage_train <- training(wage_split)
wage_val  <- testing(wage_split)
```

(a) Polynomial

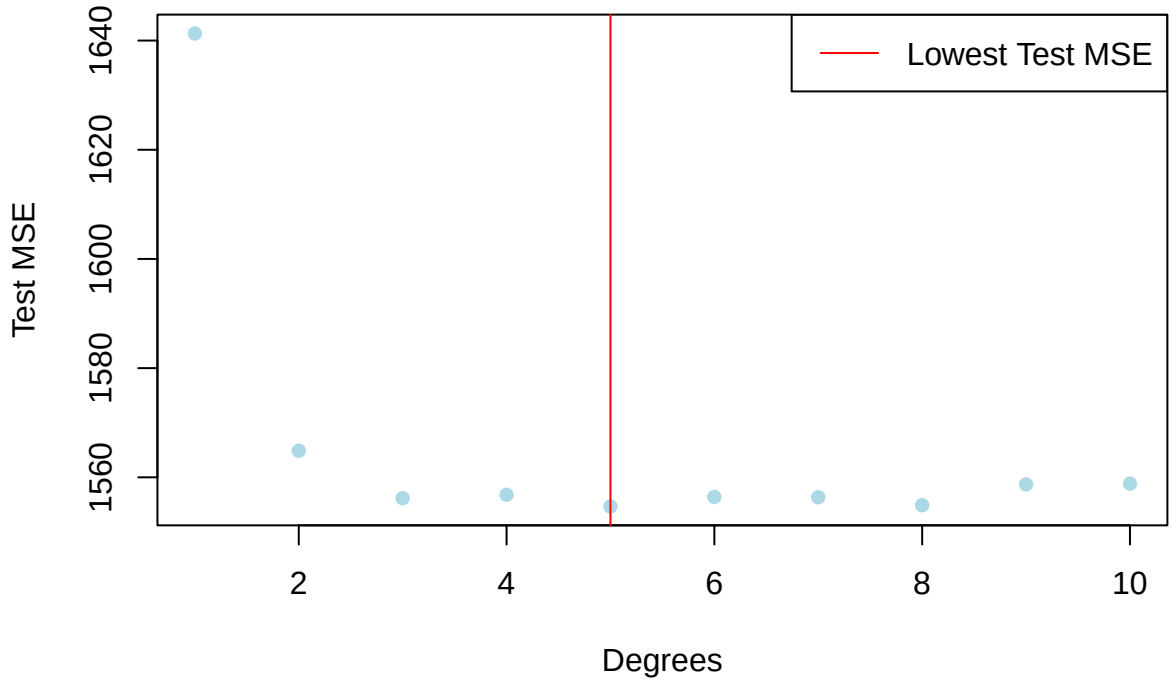
To fit a Polynomial Regression on our train data, we can use 10 fold CV on our train data to see which degree performs the best by comparing their test MSE.

```
set.seed(435)
Test_MSE <- rep(NA, 10)

for (i in 1:10) {
    fit <- glm(wage ~ poly(age, i), data = wage_train)
```

```
    Test_MSE[i] <- cv.glm(wage_train, fit, K = 10)$delta[1]
}


plot(c(1:10), Test_MSE,
     xlab = "Degrees",
     ylab = "Test MSE",
     pch = 16,
     col = "lightblue")
abline(v = which.min(Test_MSE), col = "red")
legend("topright", "Lowest Test MSE", lty = 1, lwd = 1, col = "red")
```




```
best.fit <- glm(wage ~ poly(age, which.min(Test_MSE)), data = wage_train)
```

- We see that at degree of 5, we have the optimal fit(lowest Test MSE) for Polynomial Regression.
- And now we can go back to the Validation set, we can calculate it's performance(Test MSE) as:

```
mean((wage_val$wage - predict(best.fit, data.frame(wage_val)))^2)
```

```
## [1] 1714.791
```

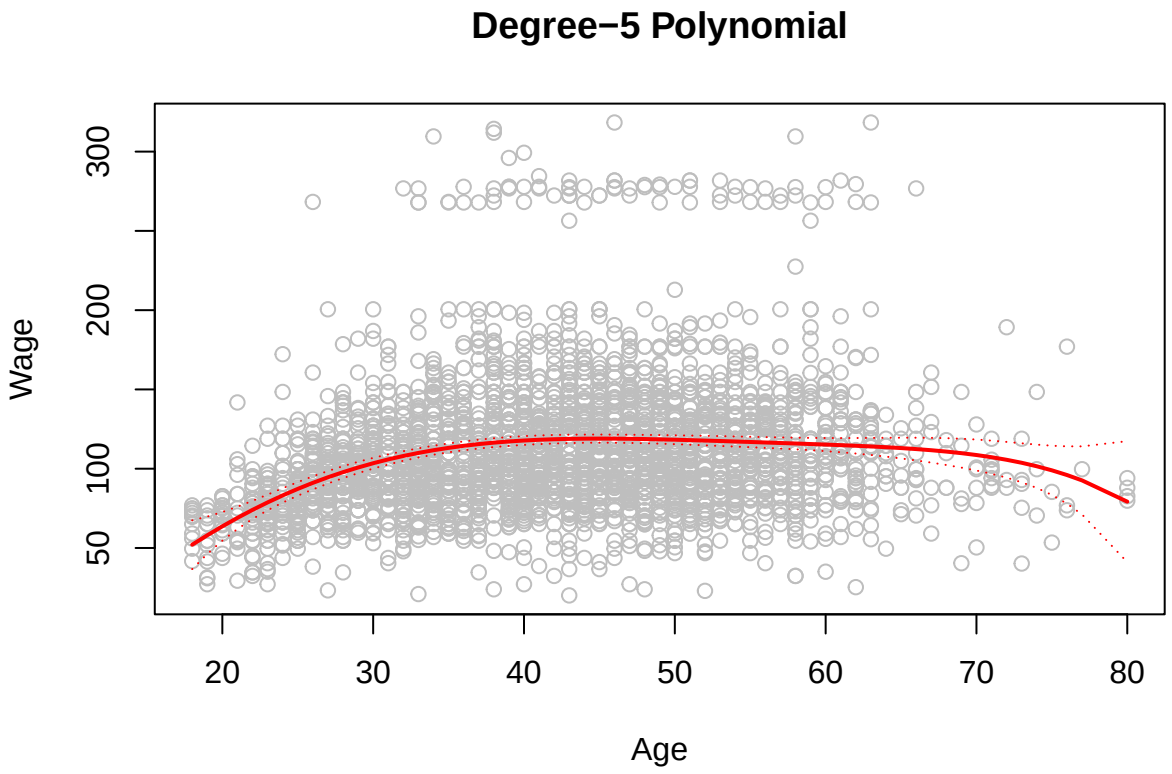
- Finally, we can plot our result and see it in a graphical representation alongside with confidence band.

```
plot(Wage$age, Wage$wage,xlab = "Age", ylab = "Wage", main = "Degree-5 Polynomial", col = "grey")
Wage_new <- Wage %>%
  arrange(age)

preds_poly <- predict(best.fit, data.frame(Wage_new[1:10]), se = TRUE)
```

```r
se.bands <- cbind(preds_poly$fit + 2 * preds_poly$se.fit,
                  preds_poly$fit - 2 * preds_poly$se.fit)

lines(Wage_new$age, preds_poly$fit,
      lwd = 2, col = "red")
matlines(Wage_new$age, se.bands, lwd = 1, col = "red", lty = 3)
```



- Then, we can use ANOVA to test the null hypothesis that a smaller model M_1 is sufficient to explain the data against the alternative hypothesis that a larger model M_2 is required.
- And we must let M_1, M_2 be nested models: the predictors in M_1 must be a subset of the predictors in M_2.
- Then, we can compare models from lienar to a degree 6 polynomial

```r
fit.1 <- lm(wage ~ age, data = wage_train)
fit.2 <- lm(wage ~ poly(age, 2), data = wage_train)
fit.3 <- lm(wage ~ poly(age, 3), data = wage_train)
fit.4 <- lm(wage ~ poly(age, 4), data = wage_train)
fit.5 <- lm(wage ~ poly(age, 5), data = wage_train)
fit.6 <- lm(wage ~ poly(age, 6), data = wage_train)
knitr::kable(cbind("Degree(s)" = c(1:6),anova(fit.1, fit.2, fit.3, fit.4, fit.5, fit.6)))
```

Degree(s)	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	2246	3679457	NA	NA	NA	NA
2	2245	3506486	1	172970.3513	111.4153417	0.0000000
3	2244	3487985	1	18501.8263	11.9175760	0.0005664

Degree(s)	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
4	2243	3483487	1	4497.4494	2.8969407	0.0888865
5	2242	3482845	1	642.4034	0.4137911	0.5201174
6	2241	3479113	1	3731.7207	2.4037121	0.1211889

- First row is showing the results for the smallest model, linear model. It only contains the RSS.
- Second row is comparing linear model with quadratic model and the p-value is essentially zero, indicating a linear fit is not sufficient.
- Third row also indicates that the quadratic model is also not enough, compared to the cubic model.
- Under significance level of 0.05, fourth, fifth, and sixth degree of Polynomial fit does not seem to be necessary.

```
best.fit <- glm(wage ~ poly(age, 3), data = wage_train)
poly_MSE <- mean((wage_val$wage - predict(best.fit, data.frame(wage_val)))^2)
```

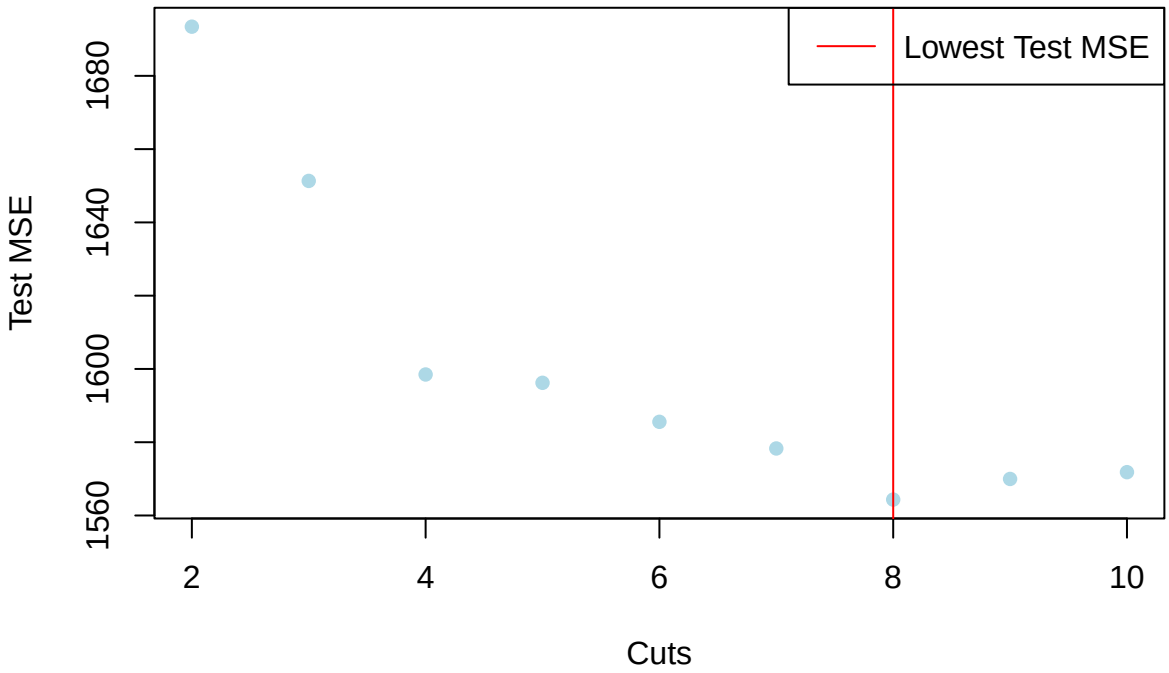
Hence, we would now use third degree of Polynomial fit, and it's Test MSE is 1717.6166721.

b Step Function

- Again, for the step function, we will also perform 10 fold CV on our Training set to see at what level of cuts will give us lowest Test MSE.

```
set.seed(435)
Test_MSE <- rep(NA, 10)
for (i in 2:10) {
    wage_train$age.cut <- cut(wage_train$age, i)
    fit <- glm(wage ~ age.cut, data = wage_train)
    Test_MSE[i] <- cv.glm(wage_train, fit, K = 10)$delta[1]
}

plot(2:10, Test_MSE[-1], xlab = "Cuts", ylab = "Test MSE", pch = 16, col = "lightblue")
abline(v = which.min(Test_MSE), col = "red")
legend("topright", "Lowest Test MSE", lty = 1, lwd = 1, col = "red")
```

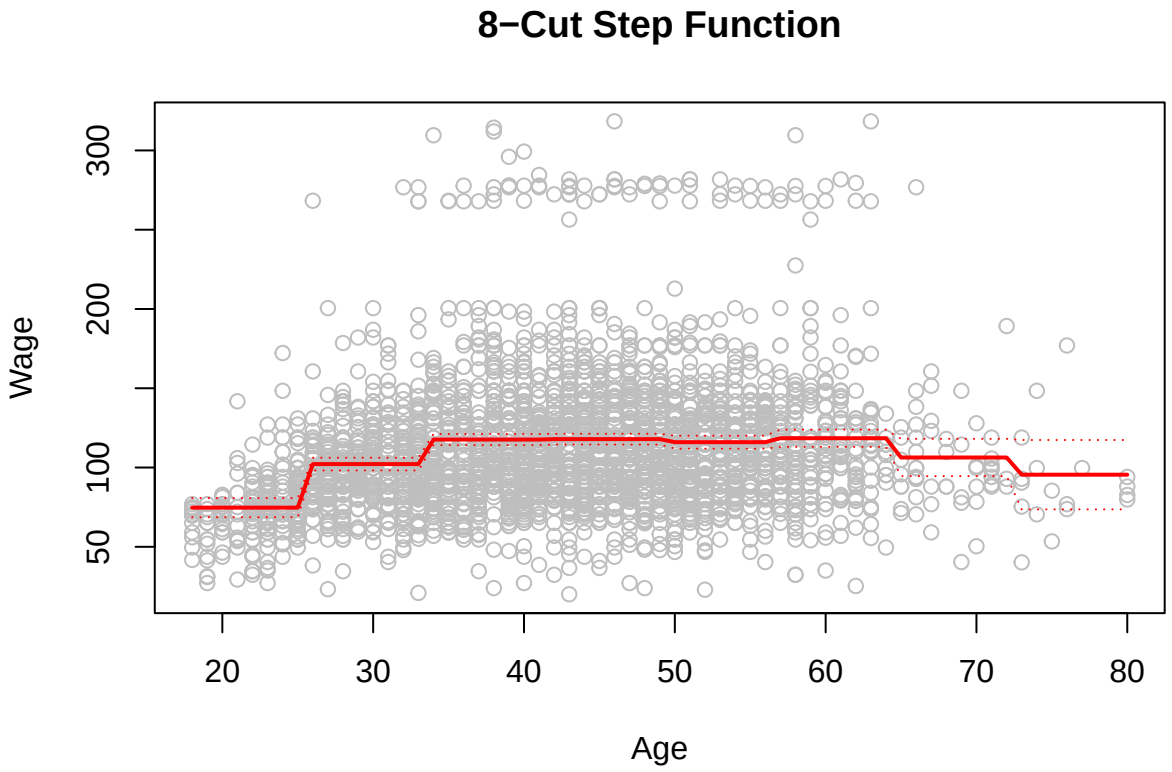


We can see that the Test MSE is at its minimum when we apply 8 cuts for our step function fit. Now, we can plot it and see the fit and it's correspond estimated 95% confidence interval.

```
plot(Wage$age, Wage$wage,xlab = "Age", ylab = "Wage", main = "8-Cut Step Function", col = "grey")
Wage_new <- Wage %>%
  arrange(age) %>%
  mutate(age.cut = cut(age, 8))

wage_train$age.cut <- cut(wage_train$age, 8)
best.fit <- glm(wage ~ age.cut, data = wage_train)
pred_step <- predict(best.fit, data.frame(Wage_new), se = TRUE)
se.bands <- cbind(pred_step$fit + 2 * pred_step$se.fit,
                  pred_step$fit - 2 * pred_step$se.fit)

lines(Wage_new$age, pred_step$fit,
      lwd = 2, col = "red")
matlines(Wage_new$age, se.bands, lwd = 1, col = "red", lty = 3)
```



- And we can calculate it's Test MSE as:

```
Wage_new <- wage_val %>%
  arrange(age) %>%
  mutate(age.cut = cut(age, 8))
step_MSE <-
  mean((Wage_new$wage - predict(best.fit, data.frame(Wage_new))) ^ 2)
```

Hence, we would now use 8 Cut Step function fit, and it's Test MSE is 1726.8131328.

c. Piecewise polynomial

- From part (a), we learned that polynomials that is greater than degree 3 is not need at $\alpha = 0.05$. Hence we can fit a piecewise 3 degree polynomial with knots at 25,50,75th quantiles.

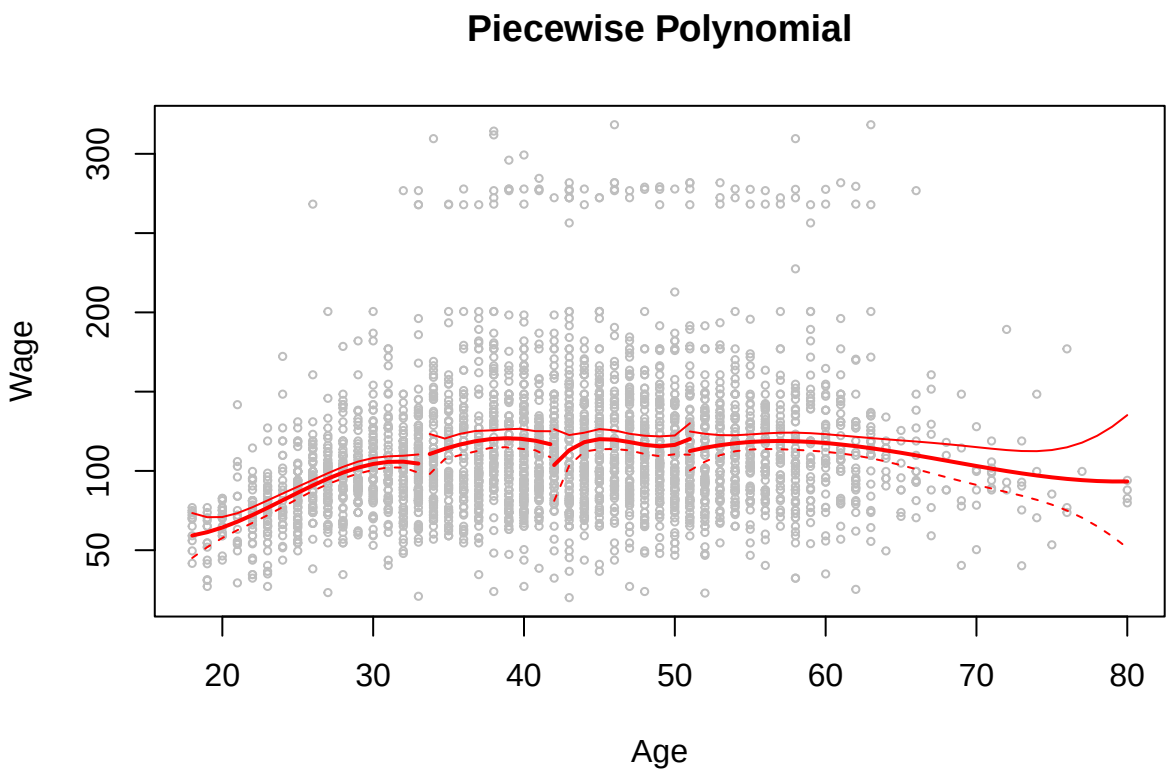
```
set.seed(435)
agelims <- range(Wage$age)
quantiles <- as.numeric(quantile(Wage$age, c(0.25, 0.5, 0.75)))
grid1 <- seq(from = agelims[1], to = quantiles[1])
grid2 <- seq(from = quantiles[1], to = quantiles[2])
grid3 <- seq(from = quantiles[2], to = quantiles[3])
grid4 <- seq(from = quantiles[3], to = agelims[2])

fit_1 <- lm(wage ~ poly(age, 3), data = wage_train[wage_train$age < quantiles[1],])
fit_2 <- lm(wage ~ poly(age, 3), data = wage_train[wage_train$age > quantiles[1] & wage_train$age <= qua
fit_3 <- lm(wage ~ poly(age, 3), data = wage_train[wage_train$age > quantiles[2] & wage_train$age <= qua
fit_4 <- lm(wage ~ poly(age, 3), data = wage_train[wage_train$age > quantiles[3],])
```

```
pred_1 <- predict(fit_1, newdata = list(age = grid1), se = TRUE)
bands1 <- cbind(pred_1$fit + 2* pred_1$se.fit,
                pred_1$fit - 2 * pred_1$se.fit)
pred_2 <- predict(fit_2, newdata = list(age = grid2), se = TRUE)
bands2 <- cbind(pred_2$fit + 2* pred_2$se.fit,
                pred_2$fit - 2 * pred_2$se.fit)
pred_3 <- predict(fit_3, newdata = list(age = grid3), se = TRUE)
bands3 <- cbind(pred_3$fit + 2* pred_3$se.fit,
                pred_3$fit - 2 * pred_3$se.fit)
pred_4 <- predict(fit_4, newdata = list(age = grid4), se = TRUE)
bands4 <- cbind(pred_4$fit + 2* pred_4$se.fit,
                pred_4$fit - 2 * pred_4$se.fit)

plot(Wage$age, Wage$wage, xlim = agelims, cex = .5, col = "grey", xlab = "Age", ylab = "Wage", main = "

lines(grid1, pred_1$fit, lwd=2, col="red")
lines(grid2, pred_2$fit, lwd=2, col="red")
lines(grid3, pred_3$fit, lwd=2, col="red")
lines(grid4, pred_4$fit, lwd=2, col="red")
matlines(grid1, bands1, lwd=1, col="red", ity = "dashed")
matlines(grid2, bands2, lwd=1, col="red", ity = "dashed")
matlines(grid3, bands3, lwd=1, col="red", ity = "dashed")
matlines(grid4, bands4, lwd=1, col="red", ity = "dashed")
```




```
wage_val_1 <- wage_val %>%
  arrange(age) %>%
  filter(age < quantiles[1])
```

```
wage_val_2 <- wage_val %>%
  arrange(age) %>%
  filter(age < quantiles[2] & age > quantiles[1])



wage_val_3 <- wage_val %>%
  arrange(age) %>%
  filter(age < quantiles[3] & age > quantiles[2])

wage_val_4 <- wage_val %>%
  arrange(age) %>%
  filter(age < agelims[2] & age > quantiles[3])

pp_MSE <-
  sum((
    wage_val_1$wage + wage_val_2$wage + wage_val_3$wage + wage_val_4$wage  -
      predict(fit_1, data.frame(wage_val_1)) - predict(fit_2, data.frame(wage_val_2))
    - predict(fit_3, data.frame(wage_val_3)) - predict(fit_4,
                                                       data.frame(wage_val_4))
  ) ^ 2) / (nrow(wage_val))
```

Hence, our Piecewise Polynomial fit have a Test MSE of 1624.2603302.

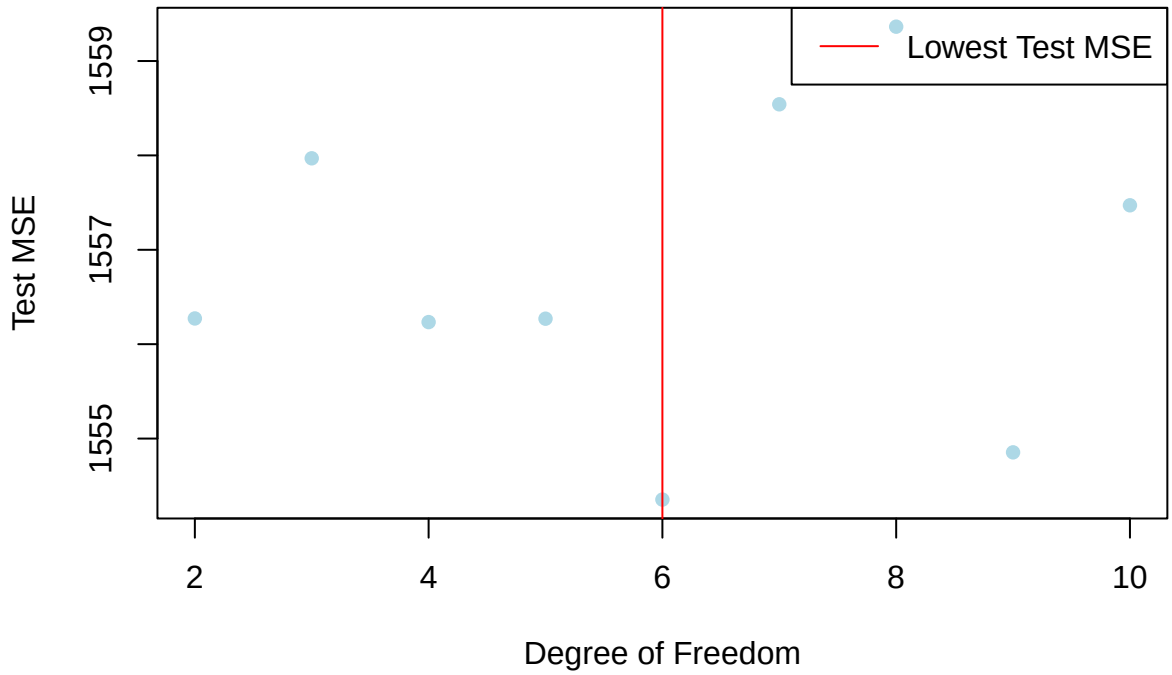
d. Cubic Spline

- Again, for the Cubic Spline, we will also perform 10 fold CV on our Training set to see at what degree of freedom will give us lowest Test MSE.

```
set.seed(435)

Test_MSE <- rep(NA, 10)
for (i in 2:10) {
    fit <- glm(wage ~ bs(age, df = i), data = wage_train)
    Test_MSE[i] <- cv.glm(wage_train, fit, K = 10)$delta[1]
}

plot(2:10, Test_MSE[-1], xlab = "Degree of Freedom", ylab = "Test MSE", pch = 16, col = "lightblue")
abline(v = which.min(Test_MSE), col = "red")
legend("topright", "Lowest Test MSE", lty = 1, lwd = 1, col = "red")
```




```
attr(bs(wage_train$age, df = 6), "knots")
```

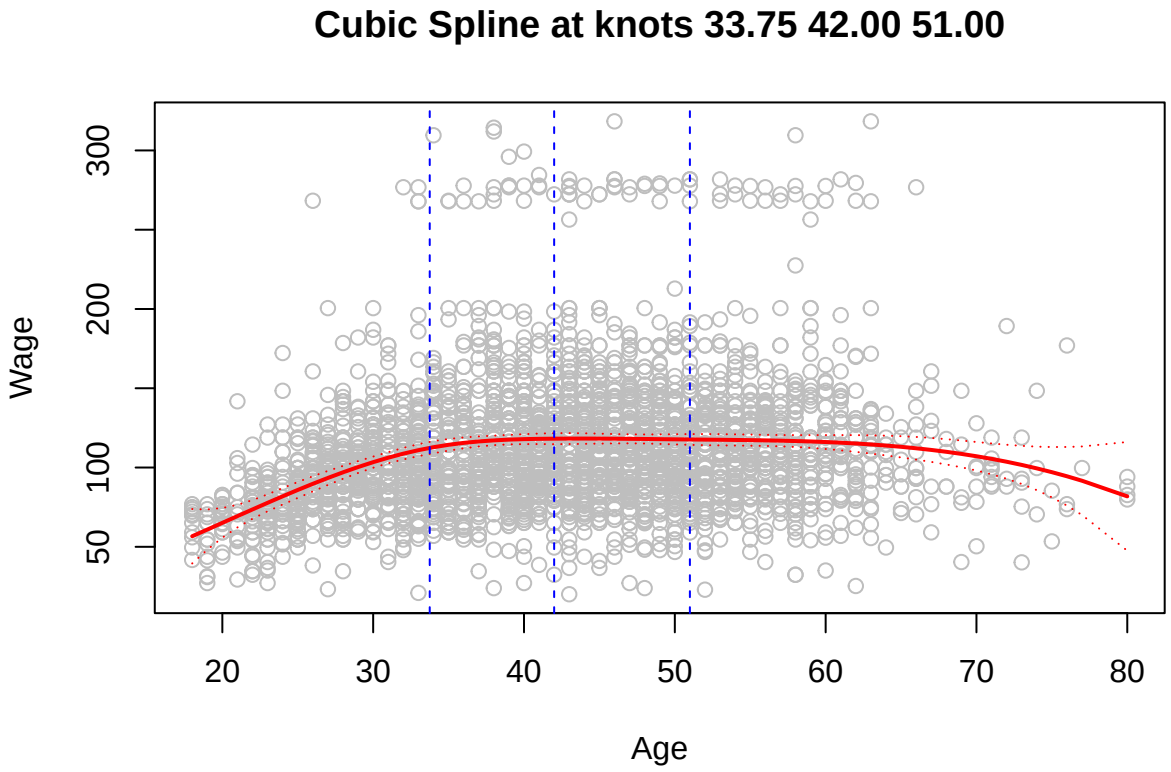
```
##   25%   50%   75%
## 33.75 42.00 51.00
```

We can see that the Test MSE is at its minimum when R chooses degree of freedom of 6 with knots at 33.75 42.00 51.00. for our Cubic Spline fit. Now, we can plot it and see the fit and it's correspond estimated 95% confidence interval.

```
plot(Wage$age, Wage$wage,xlab = "Age", ylab = "Wage", main = "Cubic Spline at knots 33.75 42.00 51.00",
Wage_new <- Wage %>%
  arrange(age)

cs.fit <- lm(wage ~ bs(age, df = 6), data = wage_train)
preds_cs <- predict(cs.fit, data.frame(Wage_new[1:10]), se = TRUE)
se.bands <- cbind(preds_cs$fit + 2 * preds_cs$se.fit,
                  preds_cs$fit - 2 * preds_cs$se.fit)

lines(Wage_new$age, preds_cs$fit,
      lwd = 2, col = "red")
matlines(Wage_new$age, se.bands, lwd = 1, col = "red", lty = 3)
abline(v = c(33.75, 42.00, 51.00), col = "blue", lty = "dashed")
```




```
cs_MSE <- mean((wage_val$wage - predict(cs.fit, data.frame(wage_val)))^2)
```

Hence, Cubic Spline with knots at 33.75 42.00 51.00's Test MSE is 1726.8131328.

e. Smoothing Spline

- When fitting Smoothing Spline, I'm using leave-one-out CV to select the best lambda.

```
fit.smooth_spline <- smooth.spline(wage_train$age, wage_train$wage, cv = TRUE)
DF <- fit.smooth_spline$df
plot(Wage$age, Wage$wage,xlab = "Age", ylab = "Wage", main = "Smoothing Spline", col = "grey")
Wage_new <- Wage %>%
  arrange(age)

lines(fit.smooth_spline,
      lwd = 2, col = "red")
```


```
ss_MSE <- mean((wage_val$wage - predict(fit.smooth_spline, wage_val$age)$y)^2)
```

Hence, our Smoothing Spline fit with degree of freedom of 5.6340873's Test MSE is 1713.3676722.

conclusion

So far, we've done fit to predict **Wage** using **Age**. And our Test MSE's are

```
kable(tibble(
  " " = c("Polynomial", "Step Function", "Piecewise Polynomial", "Cubic Spline", "Smoothing Spline"),
  "Tunning" = c("degree 3", "8 Cut", "25,50,75th Quantile, degree 3", "DF = 6", "DF = 5.6"),
  "Test MSE" = c(poly_MSE, step_MSE, pp_MSE, cs_MSE, ss_MSE)
))
```

	Tunning	Test MSE
Polynomial	degree 3	1717.617
Step Function	8 Cut	1726.813
Piecewise Polynomial	25,50,75th Quantile, degree 3	1624.260
Cubic Spline	DF = 6	1715.849
Smoothing Spline	DF = 5.6	1713.368

By the above table, it's not hard to see that Piecewise degree 3 Polynomial with knots at 25,50,75th Quantile yields the best results on the test set.

5.

We would first remove the name variable and split the Train and test data at the proportion of 2 : 1. So we will have about 66.7% of the data being training set. And it is in the typical range where it's not like lower sample sizes that can reduce the training time but may introduce more bias than necessary. And Increasing the training sample can increase performance but at the risk of overfitting(introduces more variance).

```
# Remove the name column
set.seed(435)
Auto_new <- Auto %>%
  select(-name)

train <- sample(1 : nrow(Auto_new), 2*nrow(Auto_new) / 3)
```

a.

- Our unpruned tree looks like:

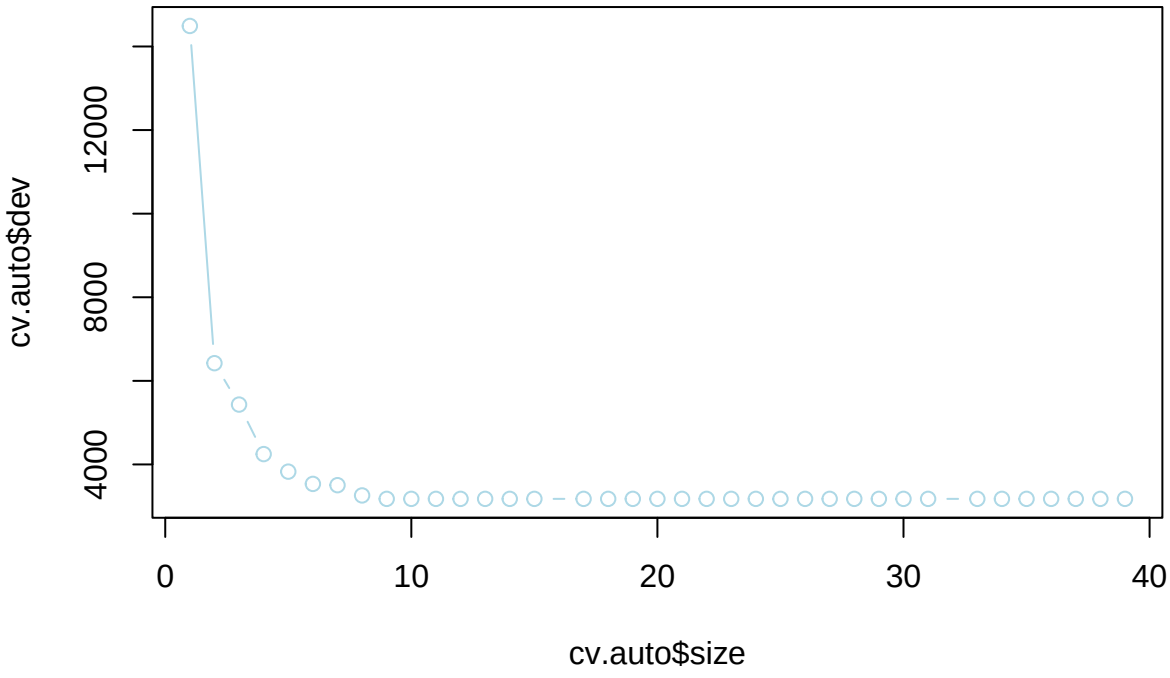
```
tree.auto <- tree(mpg~ .,Auto_new, subset = train,
                  control = tree.control(nobs = length(train),
                                         mindev = 0))
```

- And to prune the tree, we can perform 10 Fold Cross Validation

```
set.seed(435)

cv.auto <- cv.tree(tree.auto, K = 10)

plot(cv.auto$size, cv.auto$dev, type = "b", col =  "lightblue")
```

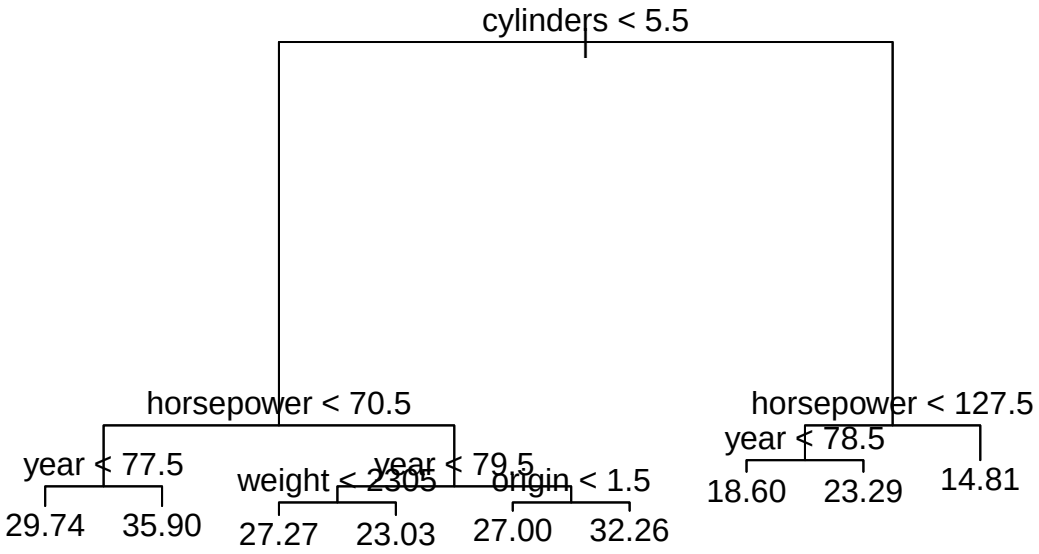


```
knitr::kable(tibble("number of terminal nodes" = cv.auto$size,
             "Test RSS" = cv.auto$dev))
```

number of terminal nodes	Test RSS
39	3176.700
38	3176.700
37	3176.700
36	3176.700
35	3176.700
34	3176.700
33	3176.700
31	3176.700
30	3176.700
29	3176.700
28	3176.700
27	3176.700
26	3176.700
25	3176.700
24	3176.700
23	3176.700
22	3176.700
21	3176.700
20	3176.700
19	3176.700
18	3176.700
17	3176.700
15	3176.700
14	3176.700
13	3176.700
12	3176.700
11	3176.700
10	3176.700
9	3176.700
8	3258.857
7	3503.548
6	3534.163
5	3827.913
4	4247.601
3	5432.722
2	6422.702
1	14490.879

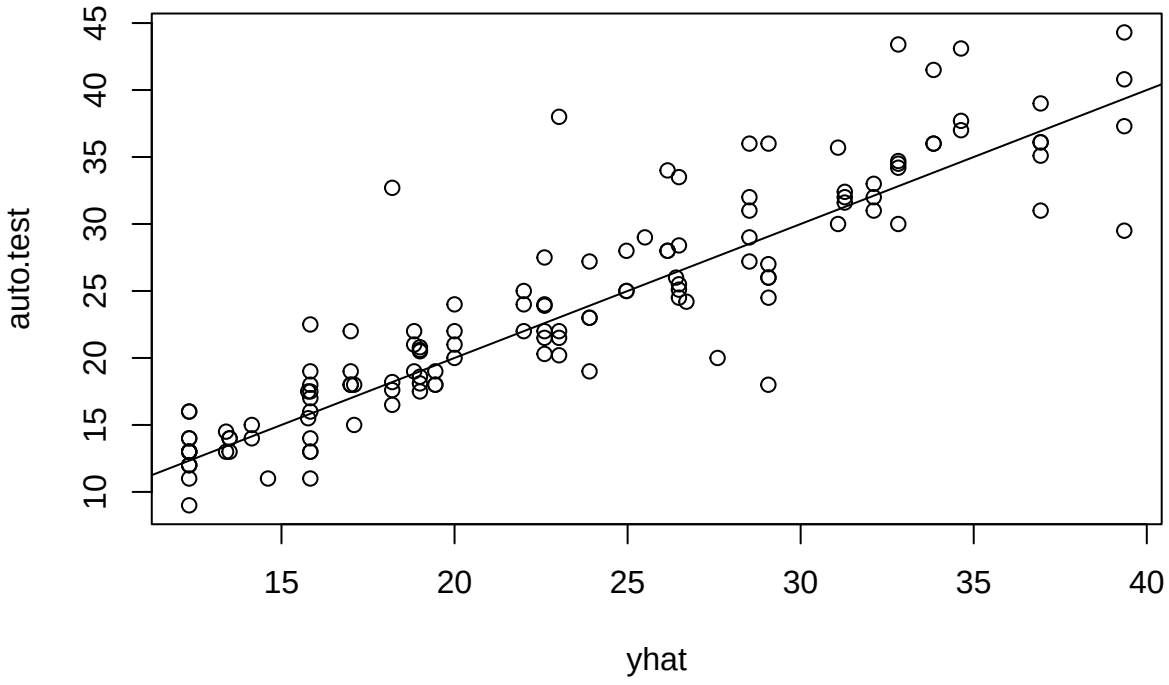
Hence, we shall pick the number of terminal nodes with lowest test RSS, and we can see that after nodes with 9, the Test RSS does not decrease anymore, hence we shall prune our tree to 9 terminal nodes.

```
prune.auto <- prune.tree(tree.auto, best = 9)
plot(prune.auto)
text(prune.auto, pretty = 0)
```




```
yhat <- predict(tree.auto, newdata = Auto_new[-train, ])
auto.test <- Auto_new[-train, "mpg"]

plot(yhat, auto.test)
abline(0, 1)
```




```
tree_MSE <- mean((yhat - auto.test)^2)
```

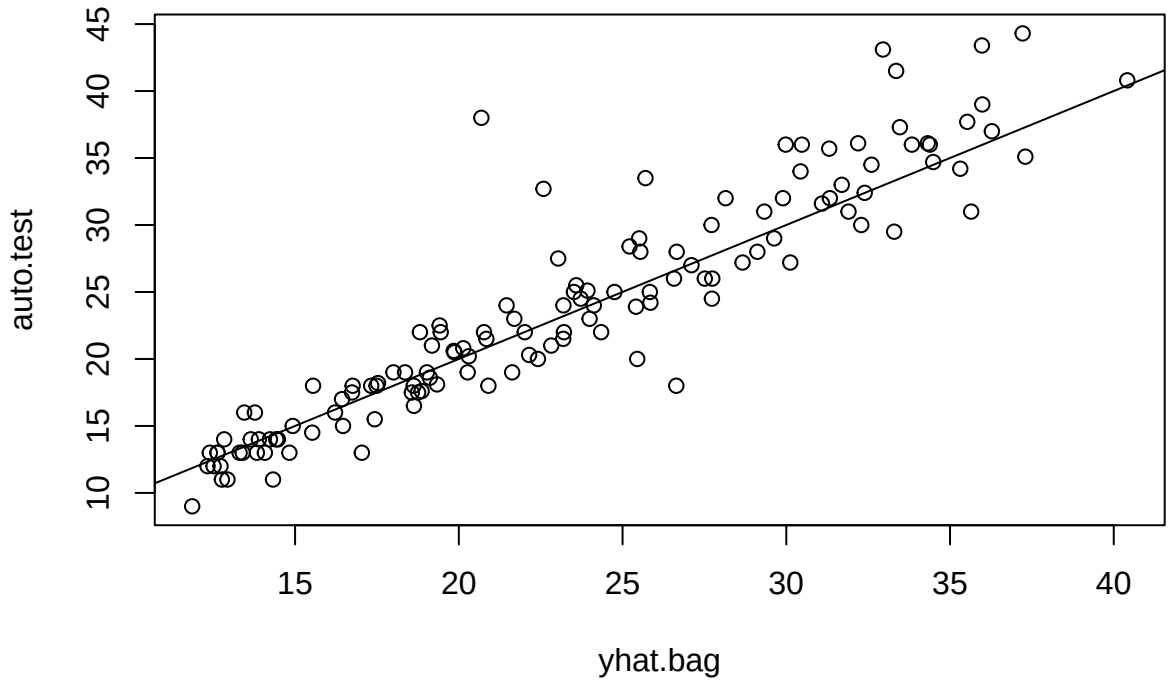
And my test MSE is 13.584193 for my regression tree to predict a car's gas mileage.

b

- Since Bagging is just a special case of RandomForest, we can perform it by setting $mtry = p$.

```r
set.seed(435)
bag.auto <- randomForest(mpg ~ ., data = Auto_new, subset = train, mtry = 7, importance = TRUE)

yhat.bag <- predict(bag.auto, newdata = Auto_new[-train, ])
plot(yhat.bag, auto.test)
abline(0, 1)
```




```r
bag_MSE <- mean((yhat.bag - auto.test)^2)

bag.auto
```

```
##
## Call:
##  randomForest(formula = mpg ~ ., data = Auto_new, mtry = 7, importance = TRUE,      subset = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##           Mean of squared residuals: 6.882735
##                     % Var explained: 87.43
```

And the test set MSE associated with the bagged regression tree is 10.2513024, about 75% of that obtained using an optimally-pruned single tree. And our tuning parameters are:

- $mtry = p = 7$: number of variables to randomly sample as candidates at each split.
- number of samples trained on is $\frac{2}{3}$ of the total data
- $ntree = 25$: number of trees

c.

- This time, we would fit a randomForest at our data.

- By default, random forest use $p/3$ variables when building a random forest of regression trees, here we use $mtry = 3$

```
set.seed(435)

rf.auto <- randomForest(mpg ~., data = Auto_new, subset = train,
                        mtry = 3)

yhat.rf <- predict(rf.auto, newdata = Auto_new[-train, ])
rf_MSE <- mean((yhat.rf - auto.test)^2)
```

- The test set MSE is 11.4253798. And we did not see an improvement over bagging in this case.
- I want to further tune the randomForest to make it performs better.
- I will use ranger(an implementation of Brieman's random forest algorithm) to perform grid search.

```
hyper_grid <- expand.grid(
  mtry = seq(1, 7, by = 1),
  num.trees = seq(5, 500, by = 25),
  nodesize = seq(3, 9, by = 1),
  OOB_RMSE = 0
)

for(i in 1:nrow(hyper_grid)) {

  # train model
  model <- ranger(
    formula         = mpg ~ .,
    data            = Auto_new[train, ],
    num.trees       = hyper_grid$num.trees[i],
    min.node.size   = hyper_grid$nodesize[i],
    mtry            = hyper_grid$mtry[i],
    seed            = 435
  )

  hyper_grid$OOB_RMSE[i] <- sqrt(model$prediction.error)
}

kable(hyper_grid %>%
  dplyr::arrange(OOB_RMSE) %>%
  head(10))
```

mtry	num.trees	nodesize	OOB_RMSE
3	255	3	2.535101
3	205	6	2.535878
3	205	5	2.536647
3	180	3	2.536653
3	180	5	2.537001
3	305	3	2.537327
3	255	5	2.537406
3	255	4	2.537437
3	330	3	2.538051

mtry	num.trees	nodesize	OOB_RMSE
3	205	4	2.538158

- Hence, we would now use the tuning parameters with best result from above and fit a new randomForest

```
set.seed(435)
rf2.auto <- randomForest(mpg ~., data = Auto_new, subset = train,
                         mtry = 3,ntree = 255, nodesize = 3)

yhat.rf2 <- predict(rf2.auto, newdata = Auto_new[-train, ])
rf2_MSE <- mean((yhat.rf2 - auto.test)^2)
```

Our new Test Set MSE is 11.323277, which is slightly lower than the first randomForest model we fit, but still worse then Bagged regression tree.

And my new randomForest have following tuning parameters:

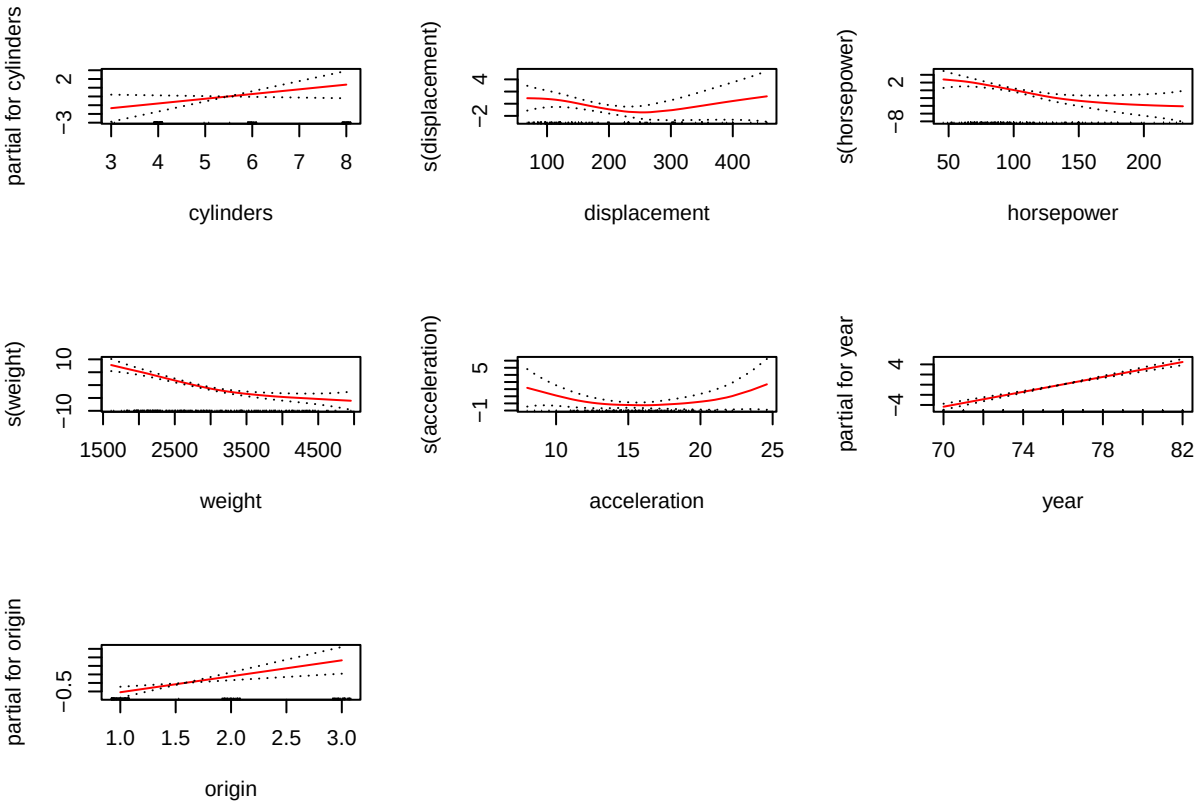
- $mtry = 3$: the number of variables to randomly sample as candidates at each split
- $ntree = 255$: number of trees
- $nodesize = 3$: minimum number of samples within the terminal nodes

d.

- Now, we want to fir a **GAM** to predict **mpg** using natural spline functions of **year**.

```
gam2.auto <-
  gam(mpg ~ cylinders + s(displacement) + s(horsepower) + s(weight) + s(acceleration) + year + origin,
    data = Auto_new,
    subset = train
  )

par(mfrow = c(3, 3))
plot.Gam(gam2.auto, se = TRUE, col = "red")
```



We plot the fitted function and pointwise standard errors, and saw that the functions are linear in **year, origin,** and **cylinders**. We can now perform a series of ANOVA tests in order to determine which of these three models is best:

- $\mathcal{M}_1$: a GAM that excludes **year, cylinders,** and **origin**.
- $\mathcal{M}_2$: a GAM that uses linear functions of **year, origin,** and **cylinders**.
- $\mathcal{M}_3$: a GAM that uses spline function for **year, cylinders,** and natural spline of **origin**.

```
gam1.auto <-
  gam(mpg ~ s(displacement) + s(horsepower) + s(weight) + s(acceleration),
    data = Auto_new,
    subset = train
  )
gam3.auto <-
  gam(mpg ~ s(cylinders) + s(displacement) + s(horsepower) + s(weight) +
        s(acceleration)+ s(year) + ns(origin),
    data = Auto_new,
    subset = train
  )

kable(tibble(" " = c("M1", "M2", "M3"),
             format(anova(gam1.auto, gam2.auto, gam3.auto), digits = 3)))
```

	Resid. Df	Resid. Dev	Df	Deviance	Pr(>Chi)
M1	244	3175	NA	NA	NA
M2	241	1697	3	1479	7.89e-56

	Resid. Df	Resid. Dev	Df	Deviance	Pr(>Chi)
M3	235	1342	6	354	1.77e-11

By ANOVA, we find that there's compelling evidence that a GAM with a lienar function of inclusion of **year, origin,** and **cylinders** is better than a GAM that does not include these predictors at all.(p-value = 7.89e-56) Then, there's also evidence that a GAM that uses spline function for **year, cylinders,** and natural spline of **origin** is better than a GAM that uses linear function of these predictors.(p-value = 1.77e-11). In other words, based on the result of this ANOVA, $\mathcal{M}_3$ is preferred.

```
gam3.auto %>%
  tidy() %>%
  mutate(
    p.value = scales::pvalue(p.value),
  ) %>%
  kable()
```

term	df	sumsq	meansq	statistic	p.value
s(cylinders)	1.0000	8561.833507	8561.833507	1498.817257	<0.001
s(displacement)	1.0000	1369.356525	1369.356525	239.716784	<0.001
s(horsepower)	1.0000	320.208250	320.208250	56.055009	<0.001
s(weight)	1.0000	184.052229	184.052229	32.219811	<0.001
s(acceleration)	1.0000	9.465584	9.465584	1.657026	0.199
s(year)	1.0000	1480.763665	1480.763665	259.219492	<0.001
ns(origin)	1.0000	69.706422	69.706422	12.202665	<0.001
Residuals	234.9993	1342.408189	5.712393	NA	NA

As we can see from the above table, all the predictors except s(acceleration) has a p-value of < 0.001. We can drop the acceleration term.

```
gam4.auto <-
  gam(mpg ~ s(cylinders) + s(displacement) + s(horsepower) + s(weight)+ s(year) + ns(origin) + accelera
    data = Auto_new,
    subset = train
  )

gam5.auto <- gam(mpg ~ s(cylinders) + s(displacement) + s(horsepower) + s(weight)+ s(year) + ns(origin)
    data = Auto_new,
    subset = train
  )


kable(tibble(" " = c("M3", "M4", "M5"),
             format(anova(gam3.auto, gam4.auto, gam5.auto), digits = 3)))
```

	Resid. Df	Resid. Dev	Df	Deviance	Pr(>Chi)
M3	235	1342	NA	NA	NA
M4	238	1383	-3	-40.710	0.068
M5	239	1383	-1	-0.125	0.882

With high p-value for comparing $\mathcal{M}_4$ with $\mathcal{M}_3$ and $\mathcal{M}_5$ with $\mathcal{M}_4$, we see that it's not necessary to drop acceleration or to make it linear. Hence, based on the results of this ANOVA, $\mathcal{M}_3$ is preferred.

```
yhat.gam3 <- predict(gam3.auto, newdata = Auto_new[-train, ])
gam3_MSE <- mean((yhat.gam3 - auto.test)^2)
```

And our Test Set MSE is 10.1188454.

e.

```
kable(tibble(
  "MEthods" = c("Regression Tree", "Bagged Regression", "Random Forest", "GAM"),
   "Test MSE" = c(tree_MSE, bag_MSE, rf2_MSE, gam3_MSE)
))
```

MEthods	Test MSE
Regression Tree	13.58419
Bagged Regression	10.25130
Random Forest	11.32328
GAM	10.11885

As we can see, GAM provides the lowest test MSE, however, if we take into account of the interpretability, I would prefer Regression Tree, because it's the most intuitive and easiest to explain to other people alongside with a comparable test MSE. If we don't take consideration into interpretability, I think either GAM or bagged regression will provide great result in terms of accuracy.