# DATA 558: HW Assignment 4

Hriday Baghar

May 25, 2022

## Instructions:

You may discuss the homework problems in small groups, but you must write up the final solutions and code yourself. Please turn in your code for the problems that involve coding. However, code without written answers will receive no credit. To receive credit, you must explain your answers and show your work. All plots should be appropriately labeled and legible, with axis labels, legends, etc., as needed.

*On this assignment, some of the problems involve random number generation. Be sure to set a random seed (using the command* `set.seed()`*) before you begin.*

```
library(ISLR2)
library(splines)
library(ggplot2)
library(tree)
library(dplyr)
library(gam)
library(randomForest)
```

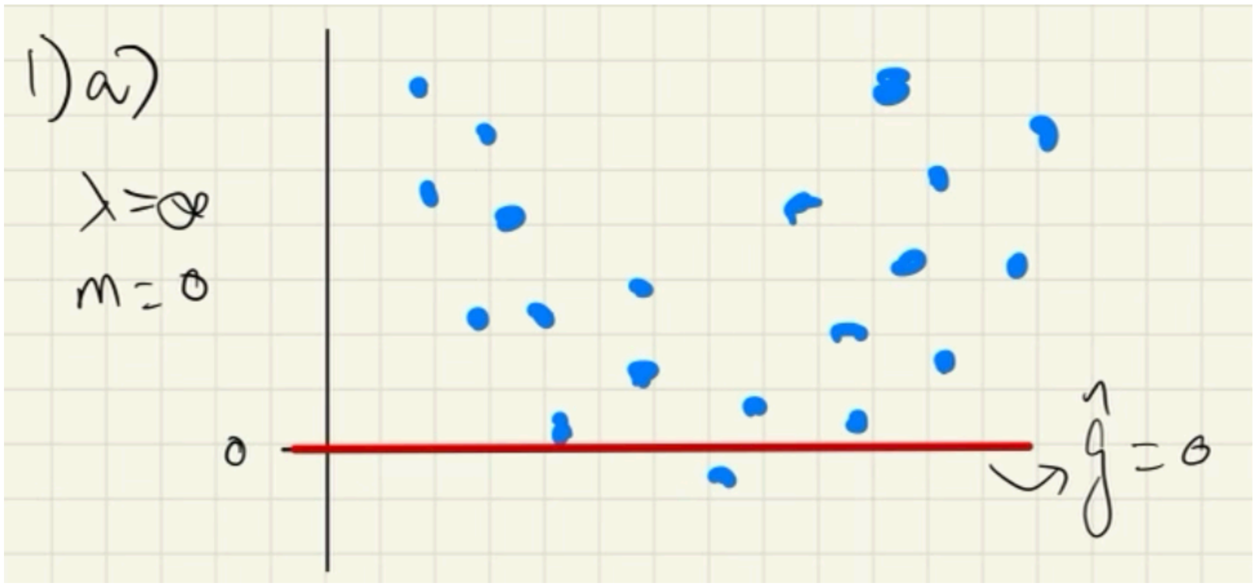
## 1. Suppose that a curve $\hat{g}$ is computed to smoothly fit a set of $n$ points using the following formula:

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

**where $g^{(m)}$ represents the $m$th derivative of $g$ (and $g^{(0)} = g$). Provide example sketches of $\hat{g}$ in each of the following scenarios.**

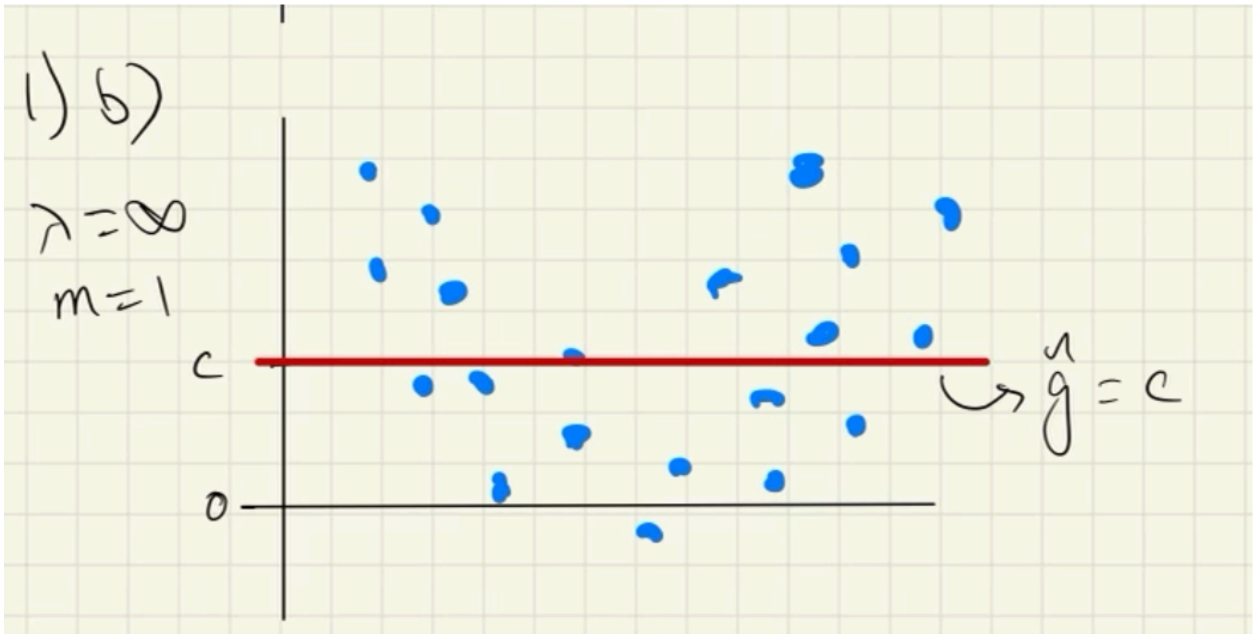
Since the penalty is a positive term, the minimum possible value is 0.

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



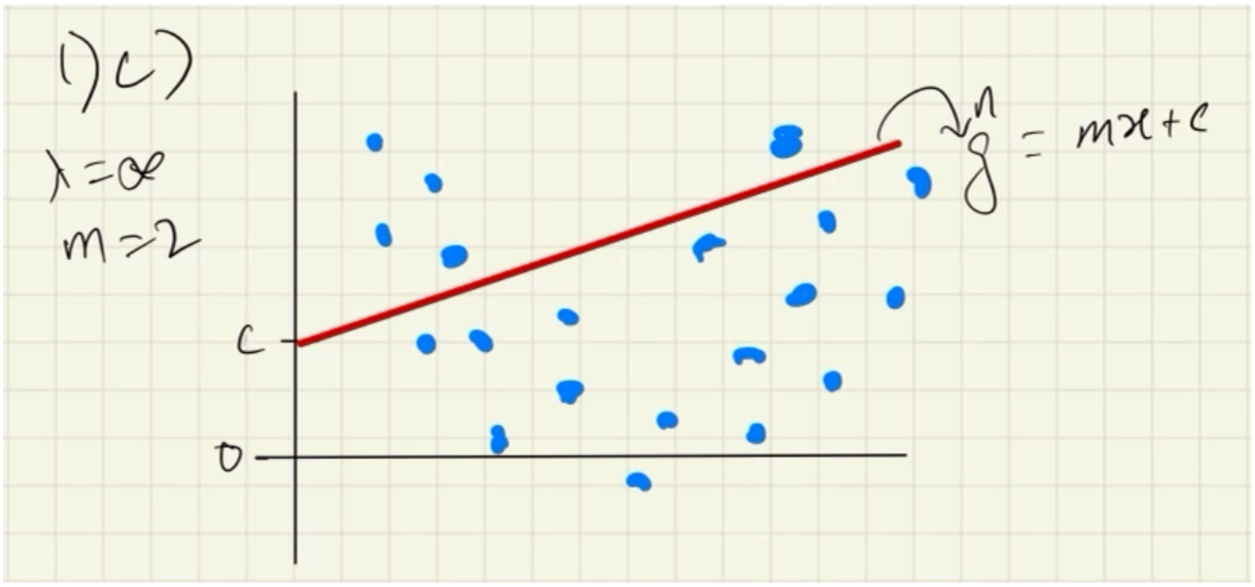
Since $m = 0$ and $\lambda = \infty$, we get the minimum when $\hat{g} = 0$ because the residuals are much smaller than the penalty term.

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



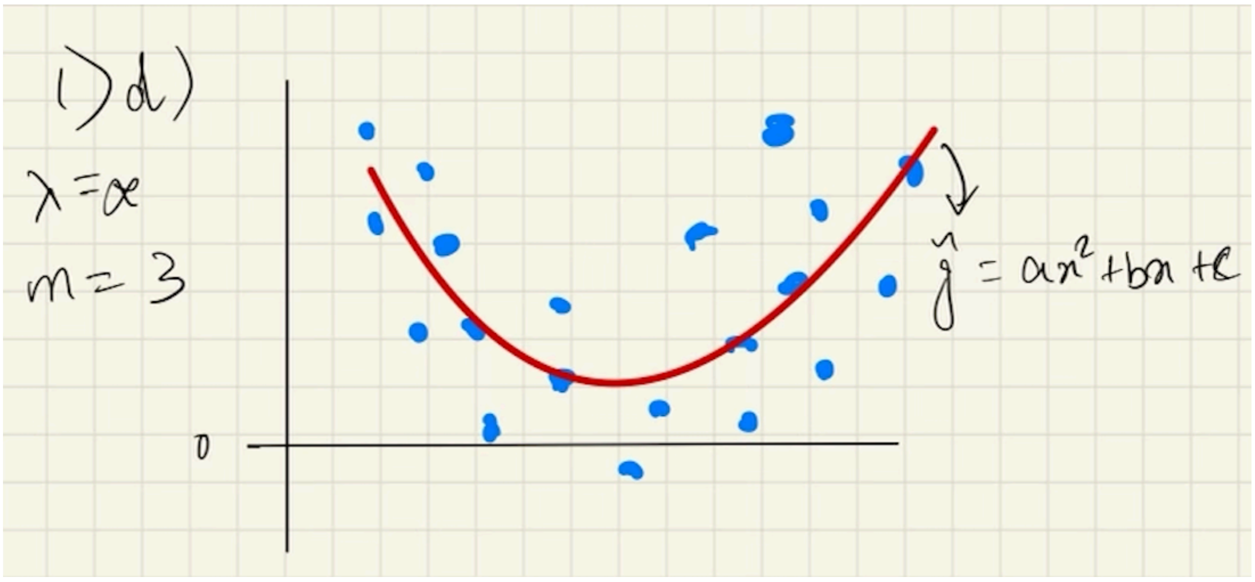
Since $m = 1$ and $\lambda = \infty$, we get the minimum when $\hat{g}' = 0$. We can get this when $\hat{g} = c$ i.e. some constant value because the residuals are much smaller than the penalty term.

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



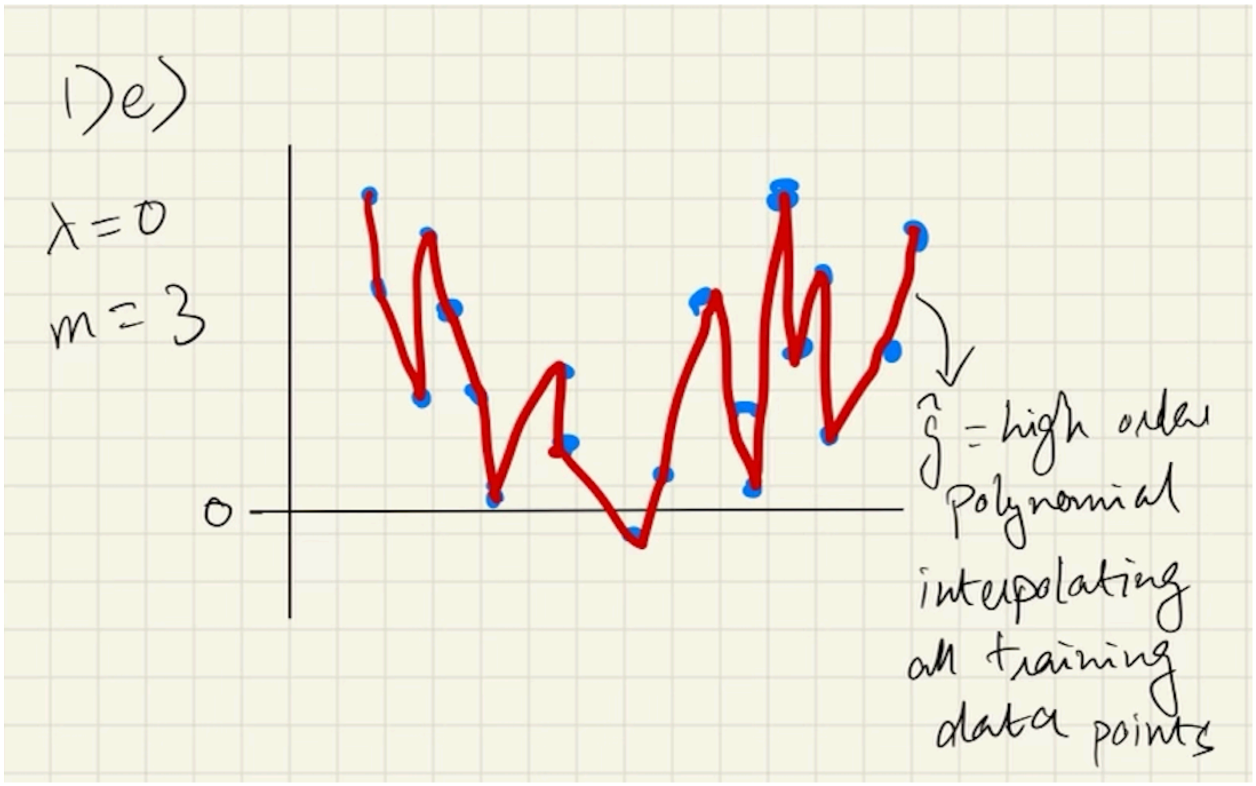
Since $m = 2$ and $\lambda = \infty$, we get the minimum when $\hat{g}'' = 0$. We can get this when $\hat{g} = mx + c$ i.e. a line of slope-intercept form because the residuals are much smaller than the penalty term.

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



Since $m = 3$ and $\lambda = \infty$, we get the minimum when $\hat{g}''' = 0$. We can get this when $\hat{g} = ax^2 + bx + c$ i.e. a quadratic polynomial because the residuals are much smaller than the penalty term.

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



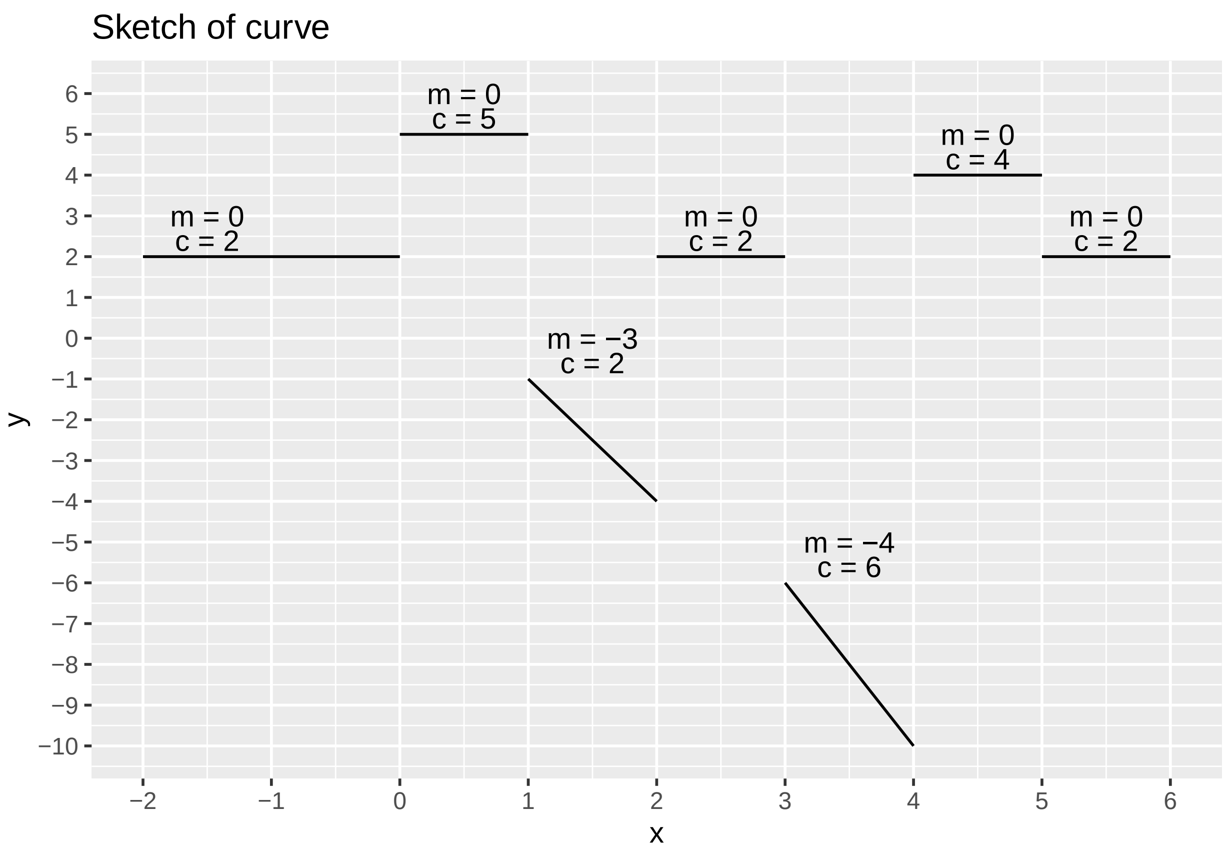
Since $\lambda = 0$, we get the minimum when the residuals are zero. This happens when the fitted values perfectly interpolate the training data i.e. the spline has a knot at each training data point.

**2. Suppose we fit a curve with basis functions $b_1(X) = I(0 \leq X \leq 2) - (X + 1)I(1 \leq X \leq 2)$, $b_2(X) = (2X - 2)I(3 \leq X \leq 4) - I(4 < X \leq 5)$. We fit the linear regression model**

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

**and obtain coefficient estimates $\hat{\beta}_0 = 2$, $\hat{\beta}_1 = 3$, $\hat{\beta}_2 = -2$. Sketch the estimated curve between $X = -2$ and $X = 6$. Note the intercepts, slopes, and other relevant information.**

```r
seg.df <- data.frame(x=c(-2,0,1,2,3,4,5),xend=c(0,1,2,3,4,5,6),
                     y=c(2,5,-1,2,-6,4,2),yend=c(2,5,-4,2,-10,4,2))

seg.df$slope <- (seg.df$yend - seg.df$y)/(seg.df$xend - seg.df$x)

ggplot(data = seg.df, aes(x,y))+
  geom_segment(aes(xend=xend,yend=yend))+
  geom_text(data = seg.df, aes(x=x+0.5,y=y+1,label=paste0("m = ",slope)))+
  geom_text(data = seg.df, aes(x=x+0.5,y=y+0.4,label=paste0("c = ",-slope*xend+yend)))+
  scale_x_continuous(breaks=seq(-2,6))+
  scale_y_continuous(breaks = seq(-10,10))+
  ggtitle("Sketch of curve")
```



**3. Prove that any function of the form**

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

**is a cubic spline with a knot at $\psi$.**

We know

$$f(X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 (X - \psi)_+^3 = \begin{cases} \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 & \text{if } X \leq \psi \\ \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 (X - \psi)^3 & \text{if } X > \psi \end{cases}$$

We need to prove the following about $f(X)$ to show that it is a cubic spline:

1. It is continuous

Plugging in the value $X = \psi$ gives us:

$$f(\psi) = \begin{cases} \beta_0 + \beta_1 \psi + \beta_2 \psi^2 + \beta_3 \psi^3 \\ \beta_0 + \beta_1 \psi + \beta_2 \psi^2 + \beta_3 \psi^3 + \beta_4 (\psi - \psi)^3 \end{cases}$$

Since we can see that at $X = \psi$ both cases have the same value, we can conclude that $f(X)$ is continuous

2. It is piecewise cubic with a knot at $\psi$

By definition, we can see that $f(X)$ is a function of polynomials upto the third degree. We also see that the definition of the polynomial function defining $f(X)$ changes at $X > \psi$, which is why it is piecewise with a knot at $\psi$

3. Has a continuous 1st derivative

We compute the first derivative and plug in $X = \psi$ in the two cases to check if it is continuous

$$f'(X) = \begin{cases} \beta_1 + 2\beta_2 X + 3\beta_3 X^2 & \text{if } X \leq \psi \\ \beta_1 + 2\beta_2 X + 3\beta_3 X^2 + 3\beta_4 (X-\psi)^2 \times 1 & \text{if } X > \psi \end{cases}$$

We see that on plugging in $X = \psi$ both cases will have the same value

$$f'(\psi) = \beta_1 + 2\beta_2\psi + 3\beta_3\psi^2$$

which proves that the first derivative is continuous.

4. Has a continuous 2nd derivative

We compute the second derivative and plug in $X = \psi$ in the two cases to check if it is continuous

$$f''(X) = \begin{cases} 2\beta_2 + 3 \times 2\beta_3 X & \text{if } X \leq \psi \\ 2\beta_2 + 3 \times 2\beta_3 X + 3 \times 2\beta_4 (X-\psi) & \text{if } X > \psi \end{cases}$$

We see that on plugging in $X = \psi$ both cases will have the same value

$$f''(\psi) = 2\beta_2 + 6\beta_3\psi$$

which proves that the second derivative is continuous.

## 4. For this problem, we will use the `Wage` data set that is part of the `ISLR` package. Split the data into a training set and a test set, and then fit using the following models to predict `Wage` using `Age` on the training set. Make some plots, and comment on your results.

```
str(Wage)
```

```
## 'data.frame':    3000 obs. of  11 variables:
##  $ year      : int  2006 2004 2003 2003 2005 2008 2009 2008 2006 2004 ...
##  $ age       : int  18 24 45 43 50 54 44 30 41 52 ...
##  $ maritl    : Factor w/ 5 levels "1. Never Married",..: 1 1 2 2 4 2 2 1 1 2 ...
##  $ race      : Factor w/ 4 levels "1. White","2. Black",..: 1 1 1 3 1 1 4 3 2 1 ...
##  $ education : Factor w/ 5 levels "1. < HS Grad",..: 1 4 3 4 2 4 3 3 3 2 ...
##  $ region    : Factor w/ 9 levels "1. New England",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ jobclass  : Factor w/ 2 levels "1. Industrial",..: 1 2 1 2 2 2 1 2 2 2 ...
##  $ health    : Factor w/ 2 levels "1. <=Good","2. >=Very Good": 1 2 1 2 1 2 2 1 2 2 ...
##  $ health_ins: Factor w/ 2 levels "1. Yes","2. No": 2 2 1 1 1 1 1 1 1 1 ...
##  $ logwage   : num  4.32 4.26 4.88 5.04 4.32 ...
##  $ wage      : num  75 70.5 131 154.7 75 ...
```

```
set.seed(42)
train <- sample(nrow(Wage), size = 0.5*nrow(Wage), replace = FALSE)
```

Defining function to report test MSE, plot test data and model fit.

```r
plot.fit <- function(model, subtitle=""){
  agelims <- range(Wage[train, "age"])
  age.grid <- seq(from = agelims[1], to = agelims[2])
  preds <- predict(model, newdata = list(age = age.grid), se = TRUE)

  preds.df <- data.frame(age = age.grid, lower = preds$fit - 2*preds$se.fit,
                         upper = preds$fit + 2*preds$se.fit,
                         preds = preds$fit)

  test.preds <- predict(model, newdata = Wage[-train,])
  mse <- mean((test.preds - Wage[-train,"wage"])^2)
  print(paste0("Test MSE = ", mse))

  ggplot()+
    geom_point(data = Wage[-train,], aes(x=age, y=wage), shape=1, color = "darkgray") +
    geom_line(data = preds.df, aes(x=age, y = lower), linetype = 2, color = "blue")+
    geom_line(data = preds.df, aes(x=age, y = upper), linetype = 2, color = "blue")+
    geom_line(data = preds.df, aes(x=age, y = preds), color = "blue")+
    labs(title = "Plot of Test Data and Model Fit", subtitle = subtitle)
}
```

**(a) polynomial**

```r
poly.fit <- lm(wage ~ poly(age,4), data = Wage, subset = train)
summary(poly.fit)
```

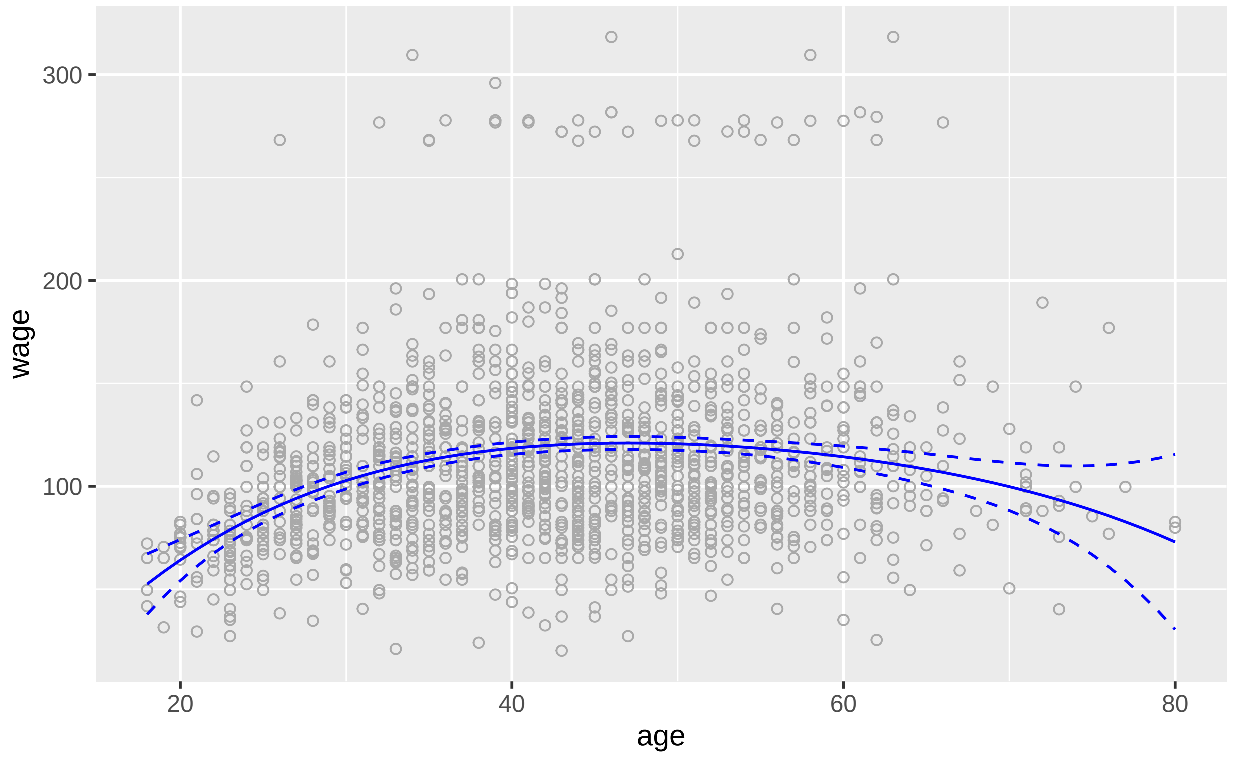
```
##
## Call:
## lm(formula = wage ~ poly(age, 4), data = Wage, subset = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -97.011 -25.721  -4.666  15.221 197.713
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      111.66       1.03 108.384  < 2e-16 ***
## poly(age, 4)1    429.10      57.64   7.445 1.63e-13 ***
## poly(age, 4)2   -556.48      59.71  -9.320  < 2e-16 ***
## poly(age, 4)3    104.71      58.85   1.779   0.0754 .
## poly(age, 4)4    -46.42      56.85  -0.817   0.4143
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.86 on 1495 degrees of freedom
## Multiple R-squared:  0.09476,    Adjusted R-squared:  0.09234
## F-statistic: 39.12 on 4 and 1495 DF,  p-value: < 2.2e-16
```

```r
plot.fit(poly.fit, "Polynomial model of degree 4")
```

```
## [1] "Test MSE = 1602.96868348542"
```

Plot of Test Data and Model Fit

Polynomial model of degree 4



The model summary shows us that a polynomial fit of order 2 might be sufficient to represent the non-linear relationship in the data. The cubic term has a p-value close to 0.05, however, the 4th power is evidently not statistically significant. This model does very well to represent the relationship for most of the range of the data (even for when the number of data points reduces considerably) as we will see.

**(b) step function**

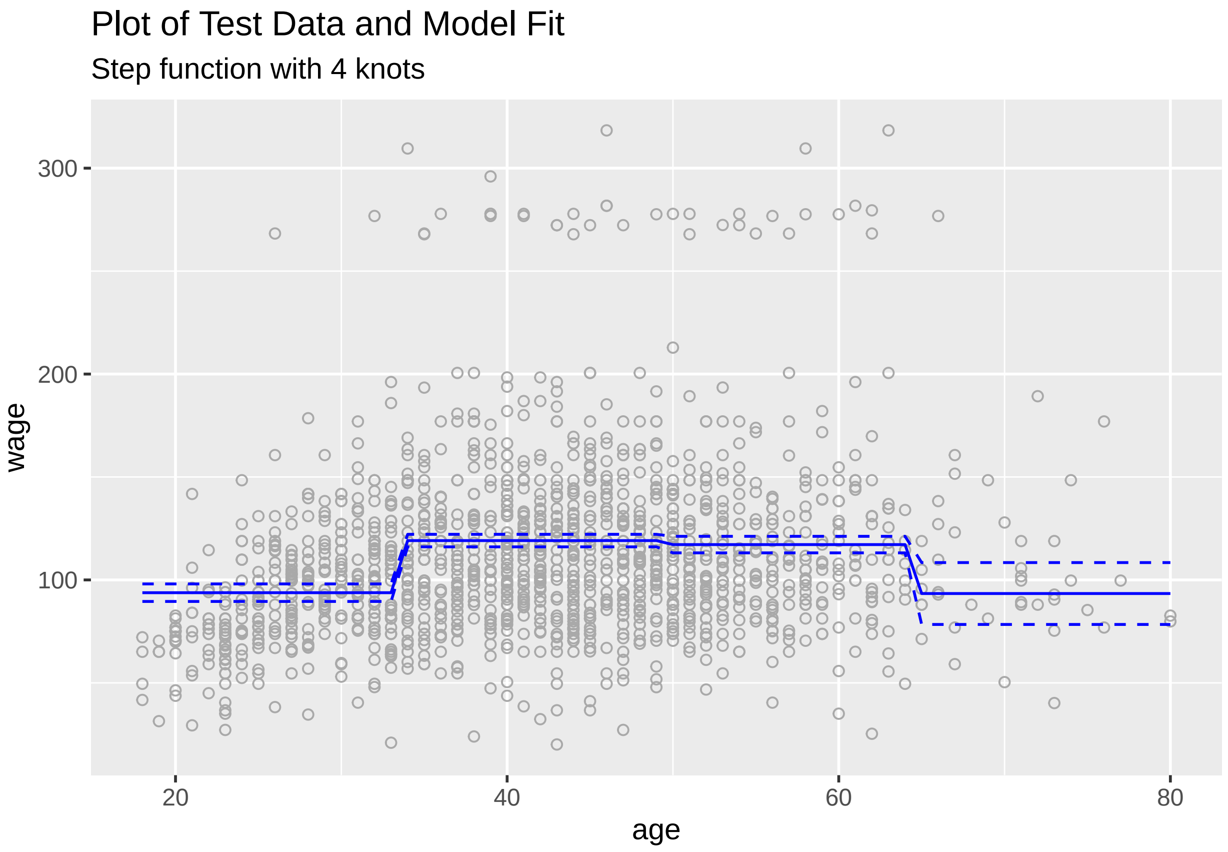
```
step.fit <- lm(wage ~ cut(age, 4), data = Wage, subset = train)
summary(step.fit)
```

```
##
## Call:
## lm(formula = wage ~ cut(age, 4), data = Wage, subset = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -95.15 -25.50  -5.42  16.04 195.22
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)               93.7973     2.1289  44.060  < 2e-16 ***
## cut(age, 4)(33.5,49]      25.3078     2.6178   9.668  < 2e-16 ***
## cut(age, 4)(49,64.5]      23.3662     2.9278   7.981 2.87e-15 ***
## cut(age, 4)(64.5,80.1]    -0.3967     7.8070  -0.051    0.959
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 40.45 on 1496 degrees of freedom
## Multiple R-squared:  0.0671, Adjusted R-squared:  0.06522
## F-statistic: 35.86 on 3 and 1496 DF,  p-value: < 2.2e-16
```

```
plot.fit(step.fit, "Step function with 4 knots")
```

```
## [1] "Test MSE = 1636.50237303064"
```



From the summary table we see that the coefficient of the 4th cut in this model is not statistically significant. This is likely because of the sample size in the cut being smaller than the rest. It does not however limit our ability to use coefficient estimates to make predictions.

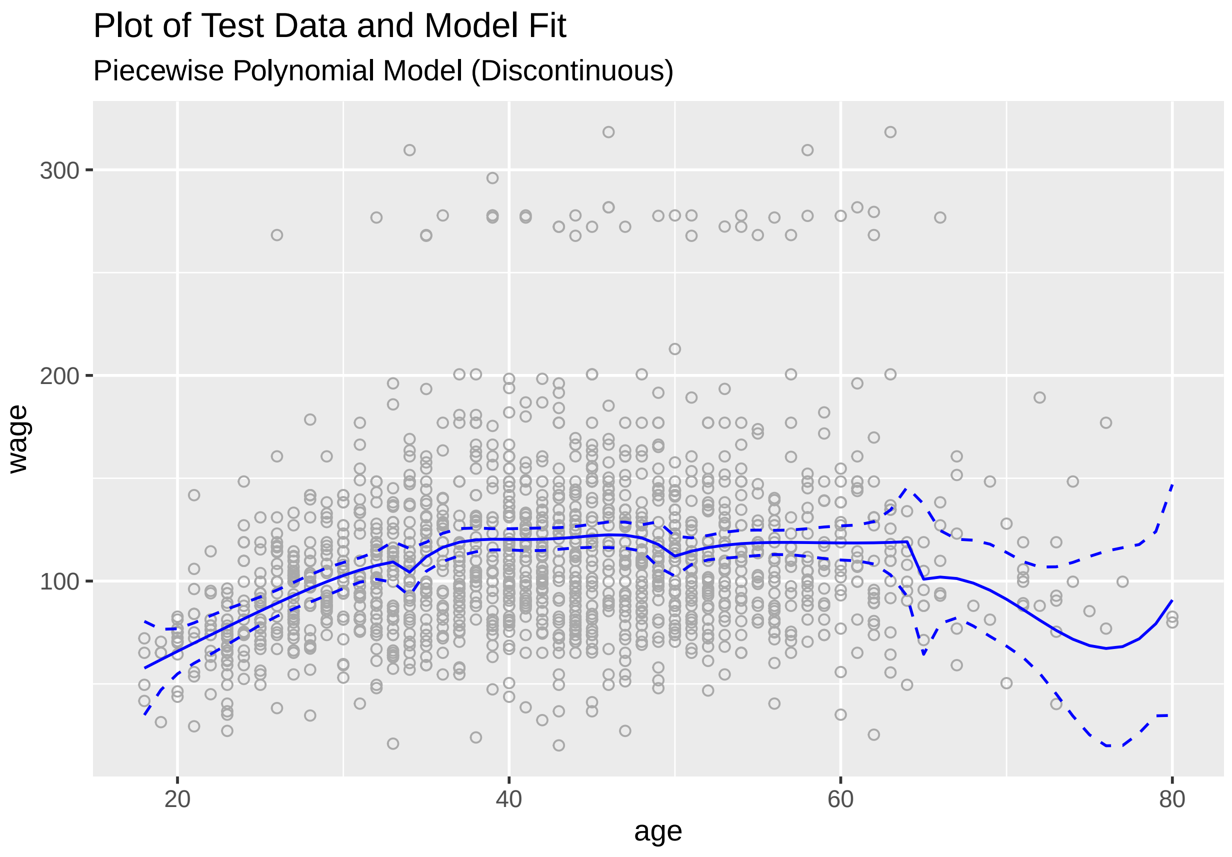
**(c) piecewise polynomial**

```
piecewise.poly.fit <- lm(wage ~ poly(age,4):cut(age,4),
                         data = Wage, subset = train)
summary(piecewise.poly.fit)
```

```
##
## Call:
## lm(formula = wage ~ poly(age, 4):cut(age, 4), data = Wage, subset = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -97.025 -24.814  -4.655  15.460 194.299
##
## Coefficients:
##                                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)                                -96.33     206.57  -0.466    0.641
## poly(age, 4)1:cut(age, 4)(17.9,33.5] -16351.30   17736.76  -0.922    0.357
## poly(age, 4)2:cut(age, 4)(17.9,33.5] -14887.93   17824.87  -0.835    0.404
## poly(age, 4)3:cut(age, 4)(17.9,33.5]  -6214.13   10451.44  -0.595    0.552
## poly(age, 4)4:cut(age, 4)(17.9,33.5]  -1049.66    2743.68  -0.383    0.702
## poly(age, 4)1:cut(age, 4)(33.5,49]    -8763.66   12137.54  -0.722    0.470
## poly(age, 4)2:cut(age, 4)(33.5,49]   -29965.18   29966.42  -1.000    0.317
## poly(age, 4)3:cut(age, 4)(33.5,49]   -13280.02   15535.34  -0.855    0.393
## poly(age, 4)4:cut(age, 4)(33.5,49]   -12587.09   12557.30  -1.002    0.316
## poly(age, 4)1:cut(age, 4)(49,64.5]    14407.41   14955.92   0.963    0.336
## poly(age, 4)2:cut(age, 4)(49,64.5]    -9505.86   12550.20  -0.757    0.449
## poly(age, 4)3:cut(age, 4)(49,64.5]     4245.58    6142.05   0.691    0.490
## poly(age, 4)4:cut(age, 4)(49,64.5]    -1024.48    4000.89  -0.256    0.798
## poly(age, 4)1:cut(age, 4)(64.5,80.1]  -7032.84   38102.33  -0.185    0.854
## poly(age, 4)2:cut(age, 4)(64.5,80.1]  17172.50   44382.40   0.387    0.699
## poly(age, 4)3:cut(age, 4)(64.5,80.1] -12637.31   26024.63  -0.486    0.627
## poly(age, 4)4:cut(age, 4)(64.5,80.1]   3793.38    6716.40   0.565    0.572
##
## Residual standard error: 39.88 on 1483 degrees of freedom
## Multiple R-squared:  0.101,  Adjusted R-squared:  0.09127
## F-statistic: 10.41 on 16 and 1483 DF,  p-value: < 2.2e-16
```

```
plot.fit(piecewise.poly.fit, "Piecewise Polynomial Model (Discontinuous)")
```

```
## [1] "Test MSE = 1611.85162991468"
```



(The different polynomials are joined together as they are a part of a single geom_line layer in the ggplot call. A closer look at the knots shows that it is indeed two polynomials joined together by a line segment)

From the results we see that the confidence interval becomes wider at the upper end of the age scale (similar to other model fits). The discontinuity at the cutpoints is noticeable, which means there will be irregularity in predictions for values of age that are close to the cutpoints. We also notice that the polynomial for the last cutpoint (64.5, 80.1] takes on a very unpredictable shape and has a huge confidence interval, making predictions in this range less reliable.

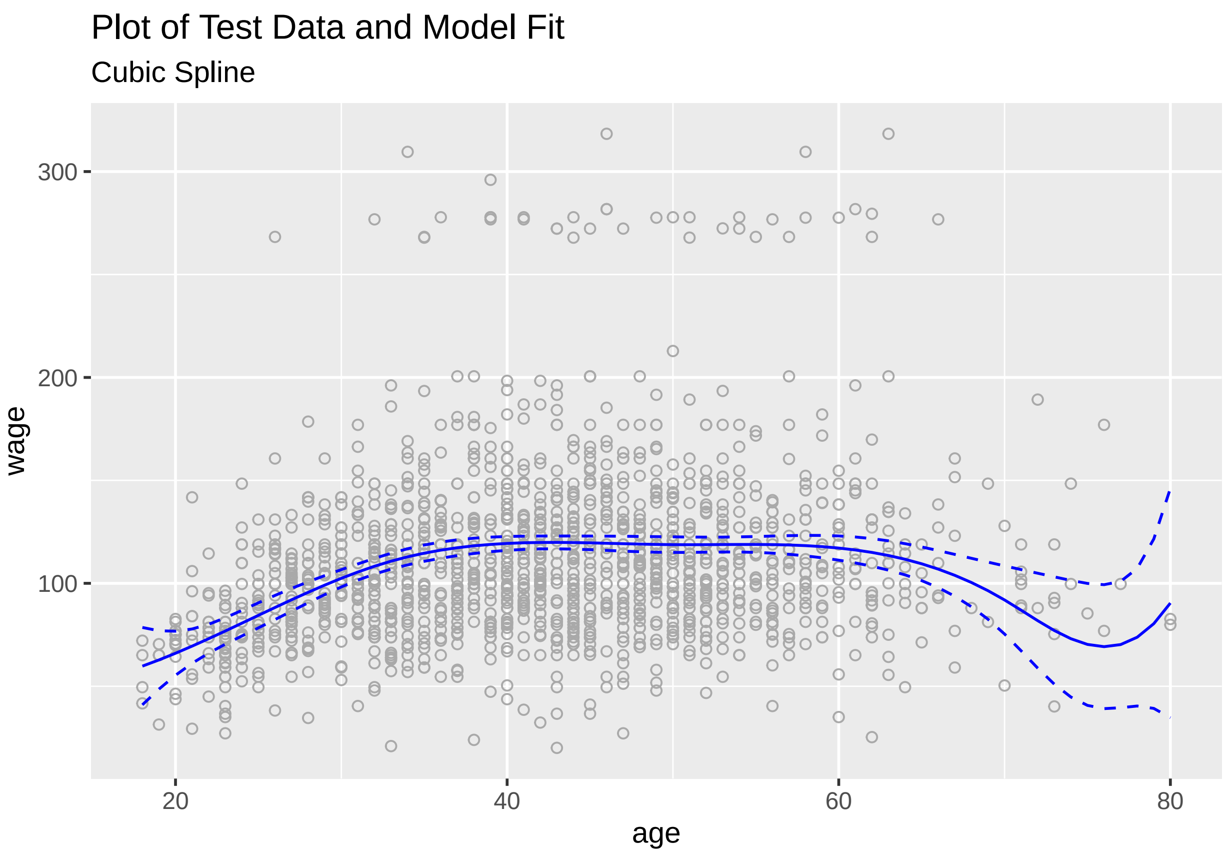
### (d) cubic spline

```
cubic.spline.fit <- lm(wage ~ bs(age, knots = c(17,33,50,70)),
                       data = Wage, subset = train)
summary(cubic.spline.fit)
```

```
##
## Call:
## lm(formula = wage ~ bs(age, knots = c(17, 33, 50, 70)), data = Wage,
##     subset = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -95.823 -24.816  -4.308  15.239 196.115
##
## Coefficients: (1 not defined because of singularities)
##                                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)                            90.43      27.86   3.246   0.0012 **
## bs(age, knots = c(17, 33, 50, 70))1   -30.66      29.34  -1.045   0.2963
## bs(age, knots = c(17, 33, 50, 70))2   -16.28      28.86  -0.564   0.5727
## bs(age, knots = c(17, 33, 50, 70))3    37.38      28.37   1.318   0.1878
## bs(age, knots = c(17, 33, 50, 70))4    23.31      28.85   0.808   0.4192
## bs(age, knots = c(17, 33, 50, 70))5    36.05      29.20   1.234   0.2172
## bs(age, knots = c(17, 33, 50, 70))6   -39.67      38.14  -1.040   0.2985
## bs(age, knots = c(17, 33, 50, 70))7       NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.83 on 1493 degrees of freedom
## Multiple R-squared:  0.09726,    Adjusted R-squared:  0.09363
## F-statistic: 26.81 on 6 and 1493 DF,  p-value: < 2.2e-16
```

```
plot.fit(cubic.spline.fit, "Cubic Spline")
```

```
## Warning in predict.lm(model, newdata = list(age = age.grid), se = TRUE):
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = Wage[-train, ]): prediction from a rank-
## deficient fit may be misleading

## [1] "Test MSE = 1606.95901627553"
```



The cubic spline does a better job of addressing the discontinuities in the piecewise polynomial model, however, it does not fare much better in terms of fit for ages > 70 than the piecewise polynomial model.

**(e) smoothing spline**

```
(smooth.spline.fit1 <- smooth.spline(x = Wage[train, "age"],
                                  y = Wage[train, "wage"],
                                  df = 20))
```

```
## Call:
## smooth.spline(x = Wage[train, "age"], y = Wage[train, "wage"],
##     df = 20)
##
## Smoothing Parameter  spar= 0.4148406  lambda= 0.0001170571 (10 iterations)
## Equivalent Degrees of Freedom (Df): 20.00071
## Penalized Criterion (RSS): 63045.4
## GCV: 1606.382
```

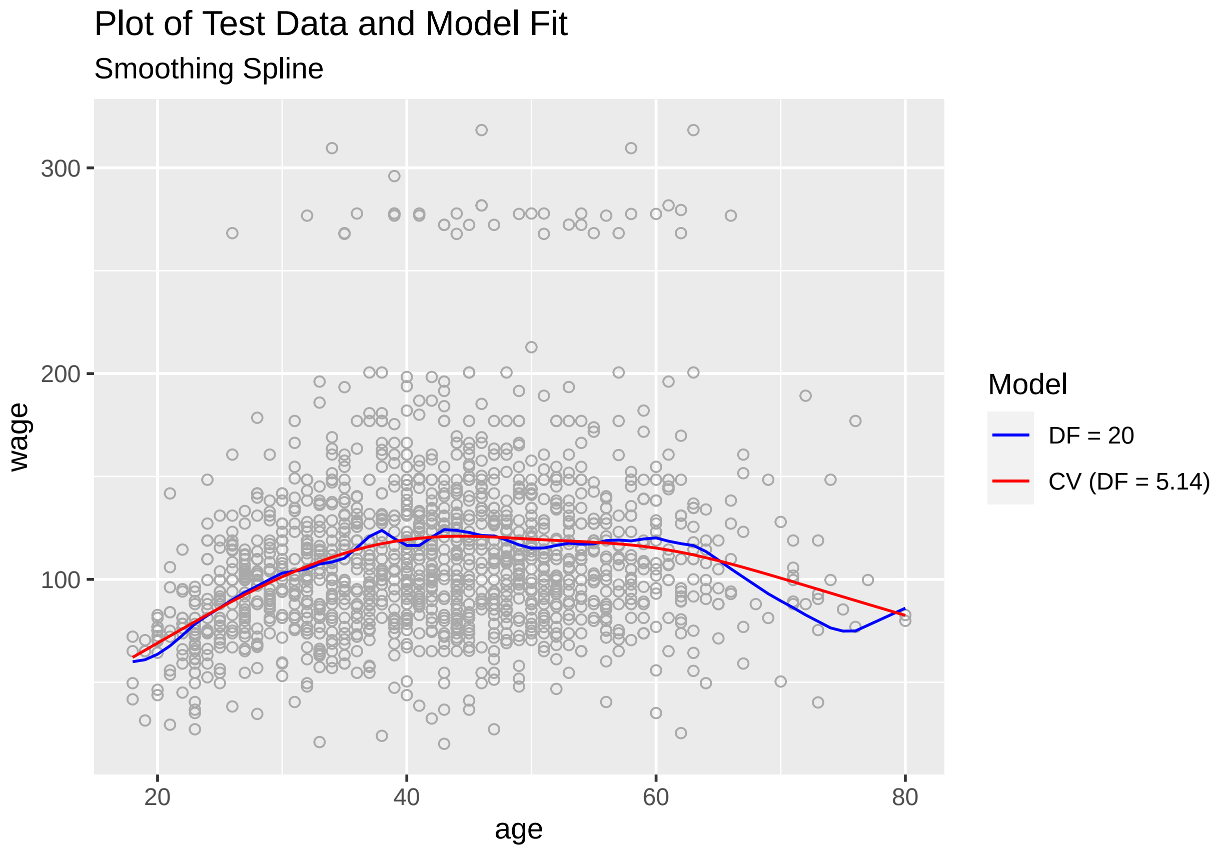
```
(smooth.spline.fit2 <- smooth.spline(x = Wage[train, "age"],
                                  y = Wage[train, "wage"],
                                  cv = TRUE))
```

```
## Warning in smooth.spline(x = Wage[train, "age"], y = Wage[train, "wage"], :
## cross-validation with non-unique 'x' values seems doubtful
```

```
## Call:
## smooth.spline(x = Wage[train, "age"], y = Wage[train, "wage"],
##     cv = TRUE)
```

```
##
## Smoothing Parameter  spar= 0.7766741  lambda= 0.04811751 (9 iterations)
## Equivalent Degrees of Freedom (Df): 5.143873
## Penalized Criterion (RSS): 89101.51
## PRESS(l.o.o. CV): 1589.636
```

```
spline1 <- data.frame(age = smooth.spline.fit1$x, wage = smooth.spline.fit1$y,
                      Model = rep("DF = 20", length(smooth.spline.fit1$x)))

spline2 <- data.frame(age = smooth.spline.fit2$x, wage = smooth.spline.fit2$y,
                      Model = rep("CV (DF = 5.14)", length(smooth.spline.fit2$x)))

ggplot()+
  geom_point(data = Wage[-train,], aes(x=age, y=wage), shape=1, color = "darkgray") +
  geom_line(data =spline1, aes(x=age, y = wage, color = Model), linetype = 1)+
  geom_line(data =spline2, aes(x=age, y = wage, color = Model), linetype = 1)+
  labs(title = "Plot of Test Data and Model Fit", subtitle = "Smoothing Spline")+
  scale_color_manual(values = c("DF = 20" = "blue", "CV (DF = 5.14)" = "red"))
```



```
mse.best.spline <- mean((predict(smooth.spline.fit2, x = Wage[-train, "age"])$y -
                          Wage[-train,"wage"])^2)

print(paste0("MSE for best spline (DF = 5.14) = ",round(mse.best.spline,2)))
```

```
## [1] "MSE for best spline (DF = 5.14) = 1600.85"
```

We fit two splines to show the difference in model fit. The model with DF=20 has a very wiggly fit and takes an unpredictable shape for age > 70. The spline fitted using cross-validation does a much better job of following some kind of predictable behavior for all ranges of age.

**(d) Which approach yields the best results on the test set?**

The smoothing spline produces the the lowest test error with an MSE of 1600.85.

## 5. Use the `Auto` data set to predict a car's `mpg`. (You should remove the `name` variable before you begin.)

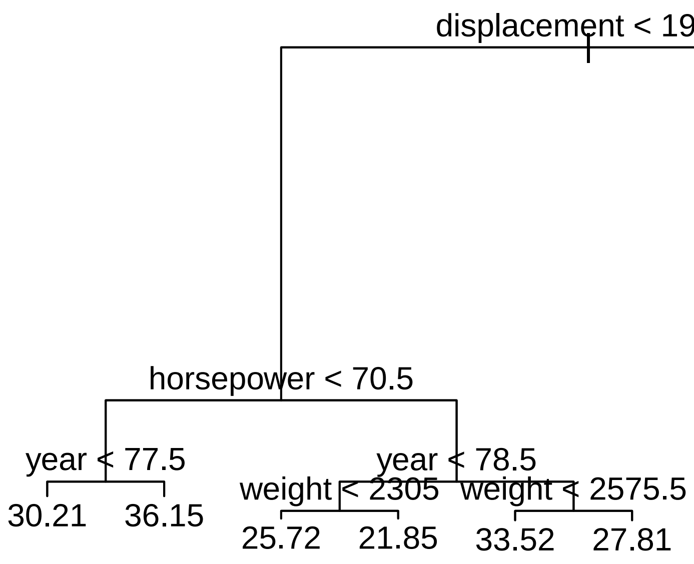
**(a) First, try using a regression tree. You should grow a big tree, and then consider pruning the tree. How accurately does your regression tree predict a car's gas mileage? Make some figures, and comment on your results.**

```r
set.seed(42)
train <- sample(nrow(Auto), 0.5*nrow(Auto), replace = FALSE)

data <- Auto |> select(-name) |> mutate(origin = as.factor(origin))

tree.auto <- tree(mpg ~ ., data = data, subset = train)
summary(tree.auto)
```

```
##
## Regression tree:
## tree(formula = mpg ~ ., data = data, subset = train)
## Variables actually used in tree construction:
## [1] "displacement" "horsepower"   "year"         "weight"
## Number of terminal nodes:  9
## Residual mean deviance:  9.185 = 1718 / 187
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -7.7200 -1.8350 -0.2083  0.0000  1.8780 14.1900
```

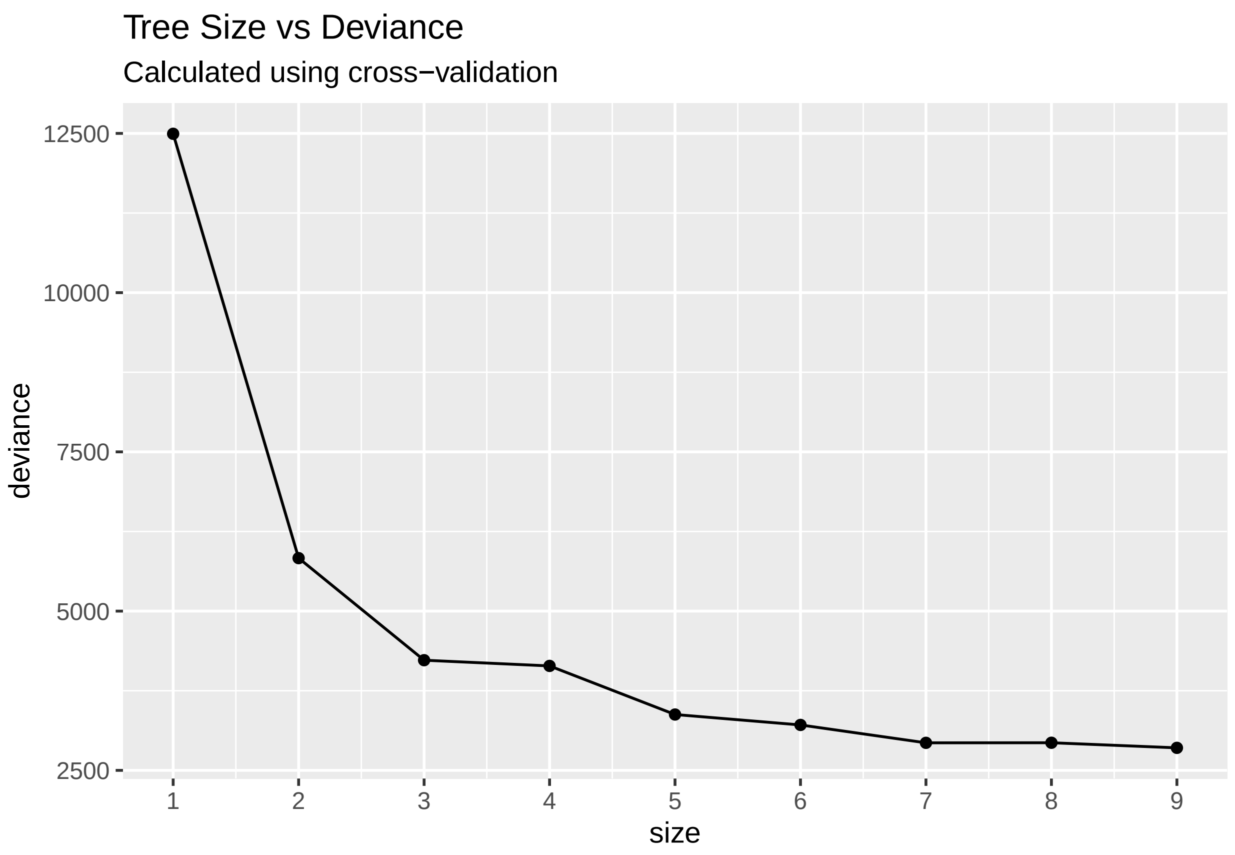
```r
plot(tree.auto)
text(tree.auto, pretty=0)
title("Decision Tree Before Any Pruning")
```

## Decision Tree Before Any Pruning



Using cross validation to select the best tree:

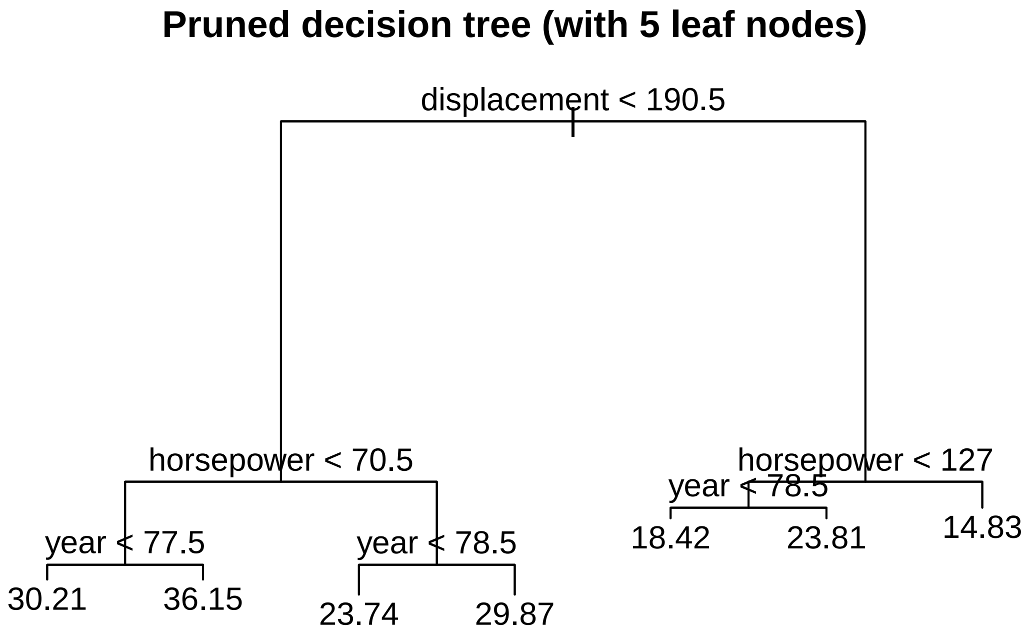
```
cv.auto <- cv.tree(tree.auto)
cv.results <- data.frame(size = cv.auto$size, deviance = cv.auto$dev)

ggplot(data = cv.results, aes(x=size, y=deviance))+
  geom_point()+
  geom_line()+
  scale_x_continuous(breaks = seq(0,9))+
  ggtitle("Tree Size vs Deviance", subtitle = "Calculated using cross-validation")
```



Using cross-validation we can see that a tree size of 9 provides the best results.

Let us try a pruned tree:

```
prune.auto <- prune.tree(tree.auto, best = 7)
plot(prune.auto)
text(prune.auto, pretty=0)
title("Pruned decision tree (with 5 leaf nodes)")
```



Now, we compare the results

```
pred.full <- predict(tree.auto, newdata = data[-train,])
mse.full <- mean((pred.full - data[-train, "mpg"])^2)
print(paste("MSE on test set for 9-node tree", round(mse.full,2), sep = " = "))
```

```
## [1] "MSE on test set for 9-node tree = 9.29"
```

```
pred.prune <- predict(prune.auto, newdata = data[-train,])
mse.prune <- mean((pred.prune - data[-train, "mpg"])^2)
print(paste("MSE for test set pruned tree", round(mse.prune,2), sep = " = "))
```

```
## [1] "MSE for test set pruned tree = 11.88"
```

We see that a tree with 9 nodes gives a lower estimated test MSE than a pruned tree with 5 nodes. Cross validation did indeed help us identify the best tree size. The pruned tree does help more with interpretability but comes at the cost of accuracy.

**(b) Fit a bagged regression tree model to predict a car's** `mpg` **How accurately does this model predict gas mileage? What tuning parameter value(s) did you use in fitting this model?**

Selecting best ntree and nodesize parameter for bagging:

```
ntree <- c(5,10,25,50,100,200,500)

set.seed(42)
val <- sample(train, length(train)/2, replace=FALSE)

nodesize <- seq(5,15)

mse <- matrix(rep(0, length(ntree)*length(nodesize)),
              nrow = length(ntree), ncol = length(nodesize))

for(i in 1:length(ntree)){
  for(j in 1:length(nodesize)){
    set.seed(42)
    fit <-  randomForest(mpg ~., data = data, subset = setdiff(train,val),
                         mtry = ncol(data)-1, importance = TRUE,
                         ntree = ntree[i], nodesize = nodesize[j])
    mse[i,j] <- mean((predict(fit, newdata = data[val,]) - data[val, "mpg"])^2)
    # print(paste("ntree =",ntree[i],", nodesize=",nodesize[j],", MSE =",mse[i,j]))
  }
}

min <- which(mse == min(mse), arr.ind = TRUE)
print(paste("Best ntree =",ntree[min[1]]))
```

```
## [1] "Best ntree = 10"
```

```
print(paste("Best nodesize =",nodesize[min[2]]))
```

```
## [1] "Best nodesize = 9"
```

```
set.seed(42)

bag.auto <- randomForest(mpg ~., data = data, subset = train,
                         mtry = ncol(data)-1, importance = TRUE,
                             ntree = ntree[min[1]], nodesize = nodesize[min[2]])
bag.auto
```

```
##
## Call:
##    randomForest(formula  =  mpg  ~  .,  data  =  data,  mtry  =  ncol(data)  -
1, importance = TRUE, ntree = ntree[min[1]], nodesize = nodesize[min[2]],     subset = train)
##                Type of random forest: regression
##                      Number of trees: 10
## No. of variables tried at each split: 7
##
##           Mean of squared residuals: 12.32329
##                     % Var explained: 80.41
```

```
pred.bag <- predict(bag.auto, newdata = data[-train,])
mse.bag <- mean((pred.bag - data[-train, "mpg"])^2)
print(paste("MSE on test set for bagged tree", round(mse.bag,2), sep = " = "))
```

```
## [1] "MSE on test set for bagged tree = 6.86"
```

Tuning parameters used (by searching using CV):

- Number of trees (ntrees): 10
- Minimum size of terminal nodes (nodesize): 9

**(c) Fit a random forest model to predict a car's `mpg` How accurately does this model predict gas mileage? What tuning parameter value(s) did you use in fitting this model?**

```
mtry <- seq(1,ncol(data)/2)

mse <- matrix(rep(0, length(ntree)*length(mtry)),
              nrow = length(ntree), ncol = length(mtry))

for(i in 1:length(ntree)){
  for(j in 1:length(mtry)){
    set.seed(42)
    fit <-  randomForest(mpg ~., data = data, subset = setdiff(train,val),
                             mtry = mtry[j], importance = TRUE,
                             ntree = ntree[i], nodesize = 10)
    mse[i,j] <- mean((predict(fit, newdata = data[val,]) - data[val, "mpg"])^2)
    # print(paste("ntree =",ntree[i],", mtry=",mtry[j],", MSE =",mse[i,j]))
  }
}

min <- which(mse == min(mse), arr.ind = TRUE)
print(paste("Best ntree =",ntree[min[1]]))
```

```
## [1] "Best ntree = 500"
```

```
print(paste("Best mtry =",mtry[min[2]]))
```

```
## [1] "Best mtry = 3"
```

```
set.seed(42)

rf.auto <- randomForest(mpg ~., data = data, subset = train,
                         mtry = mtry[min[2]], importance = TRUE,
                         ntree = ntree[min[1]])
rf.auto
```

```
##
## Call:
##  randomForest(formula = mpg ~ ., data = data, mtry = mtry[min[2]],      importance = TRUE, ntree =
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 9.994926
##                     % Var explained: 84.11
```

```
pred.rf <- predict(rf.auto, newdata = data[-train,])
mse.rf <- mean((pred.rf - data[-train, "mpg"])^2)
print(paste("MSE on test set for random forest", round(mse.rf,2), sep = " = "))
```

```
## [1] "MSE on test set for random forest = 6.22"
```

Tuning parameters used (by searching using CV):

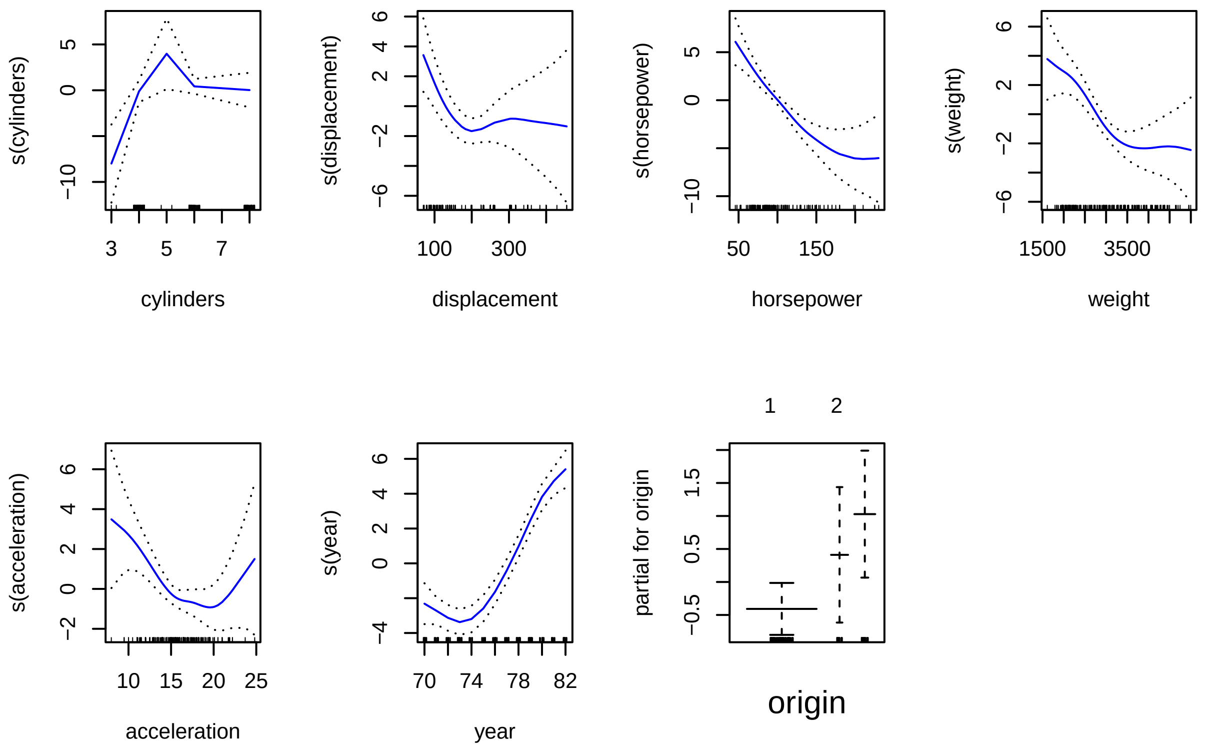
- Number of trees (ntrees): 500
- Variables tried at each split (mtry): 3

**(d) Fit a generalized additive model (GAM) model to predict a car's `mpg` How accurately does your GAM model predict a car's gas mileage? Make some figures to help visualize the fitted functions in your GAM model, and comment on your results.**

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

```
##
## Call: gam(formula = mpg ~ s(cylinders) + s(displacement) + s(horsepower) +
##     s(weight) + s(acceleration) + s(year) + origin, data = data,
##     subset = train)
## Deviance Residuals:
##    Min     1Q Median     3Q    Max
## -6.912 -1.475 -0.213  1.285 10.431
##
## (Dispersion Parameter for gaussian family taken to be 7.4469)
##
##     Null Deviance: 12329.15 on 195 degrees of freedom
## Residual Deviance: 1258.518 on 168.9998 degrees of freedom
## AIC: 976.7011
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##                  Df Sum Sq Mean Sq  F value    Pr(>F)
## s(cylinders)      1 6954.8  6954.8 933.9301 < 2.2e-16 ***
## s(displacement)   1  782.3   782.3 105.0550 < 2.2e-16 ***
## s(horsepower)     1  628.1   628.1  84.3405 < 2.2e-16 ***
## s(weight)          1   66.4    66.4   8.9161  0.003247 **
## s(acceleration)    1   63.0    63.0   8.4622  0.004114 **
## s(year)            1 1269.9  1269.9 170.5322 < 2.2e-16 ***
## origin             2   37.3    18.7   2.5074  0.084505 .
## Residuals        169 1258.5     7.4
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                  Npar Df  Npar F     Pr(F)
## (Intercept)
## s(cylinders)           3  6.1057 0.0005733 ***
## s(displacement)        3  7.4771 9.922e-05 ***
## s(horsepower)          3  3.0731 0.0292482 *
## s(weight)              3  5.9820 0.0006721 ***
## s(acceleration)        3  3.3629 0.0200859 *
## s(year)                3 13.4307 6.662e-08 ***
## origin
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
par(mfrow=c(2,4))
plot(gam.fit, se=T, col="blue")
```



Let us calculate MSE:

```
pred.gam <- predict(gam.fit, newdata = data[-train,])
mse.gam <- mean((data[-train,"mpg"] - pred.gam)^2)
print(paste("MSE on test set for GAM", round(mse.gam,2), sep = " = "))
```

```
## [1] "MSE on test set for GAM = 6.33"
```

The summary table of the GAM shows us important information:

- All variables other than origin proved statistically significant in the model fit (as seen in ANOVA for parametric effects)
- The non-linear relationships chosen by the splines were also statistically significant (as seen in ANOVA for non-parametric effects)

**(e) Considering both accuracy and interpretability of the fitted model, which of the models in (a)–(d) do you prefer? Justify your answer.**

The Random Forest we generated in (c) gives us the lowest estimated test MSE. However, the GAM in (d) and is much more interpretable in terms of how each predictor is related to the response `mpg`, and has a very slight difference in MSE.

As compared to the bagged tree and random forest models, is able to clearly show how the relation varies with the value of the predictors. For the former models, the best we can do is get a sense of feature importance, but even then we will not know how the feature affects the response for different ranges of values.

On the other hand, GAM gives us some clear insights:

- Cars from `origin=1` (USA) have the lowest MPG
- `mpg` has generally increased with `year`
- increasing `weight` and `horsepower` decrease `mpg` - this decrease slows down when `weight` reaches the 3500 range
- When `cylinders>4`, more cylinders reduce `mpg`. We can also see clearly that even though there is evidence that `cylinder=3` has the lowest `mpg` there are only 3 observations in that group.

Given the fact that the GAM offers us the ability to understand underlying relationships in data much more clearly without compromising too much on prediction error, it is a great choice. That being said, if prediction error is a priority without much concern for interpretability, random forest may be a better bet.