Homework #4

```
#  include=FALSE}
rm(list=ls())

pp <- function(...) {
    print(paste0(...))
}
```

1. In this problem, we will make use of the Auto data set, which is part of the ISLR2 package.

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.*

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 mth derivative of g (and $g^{(0)} = g$). Provide example sketches of $\hat{g}$ in each of the following scenarios.

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

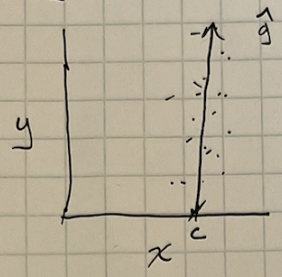
$$Y = \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \epsilon,$$

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.

```
b1 <- function(x) {
    0 + (0 <= x && x <= 2) - (x + 1)*(1 <= x && x <= 2)
}

b2 <- function(x) {
    (2*x - 2)*(3 <= x && x <= 4) - (4 < x && x <= 5)
}
```

1a) $\lambda = \infty$, $m = 0$

Assume g is a constant function $\Rightarrow g^{(0)} = 0$

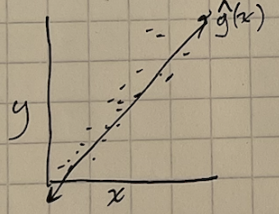


1b) ~~$\lambda = \infty$, $m = 0$~~ ~~Assume $g = ax + b$~~ ~~$\Rightarrow g = a$~~ (1)

Same as 1a)

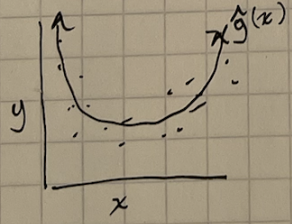
1c) $\lambda = \infty$, $m = 2$

let $g(x) = ax + b \Rightarrow g^{(2)}(x) = 0$



1d) $\lambda = \infty$ $m = 3$

let $g(x) = ax^2 + bx + c \Rightarrow g^{(3)}(x) = 0$



1e) $\lambda = 0$, $m = 3$ g can be anything. Let it be linear

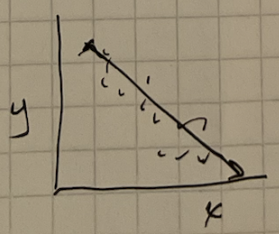
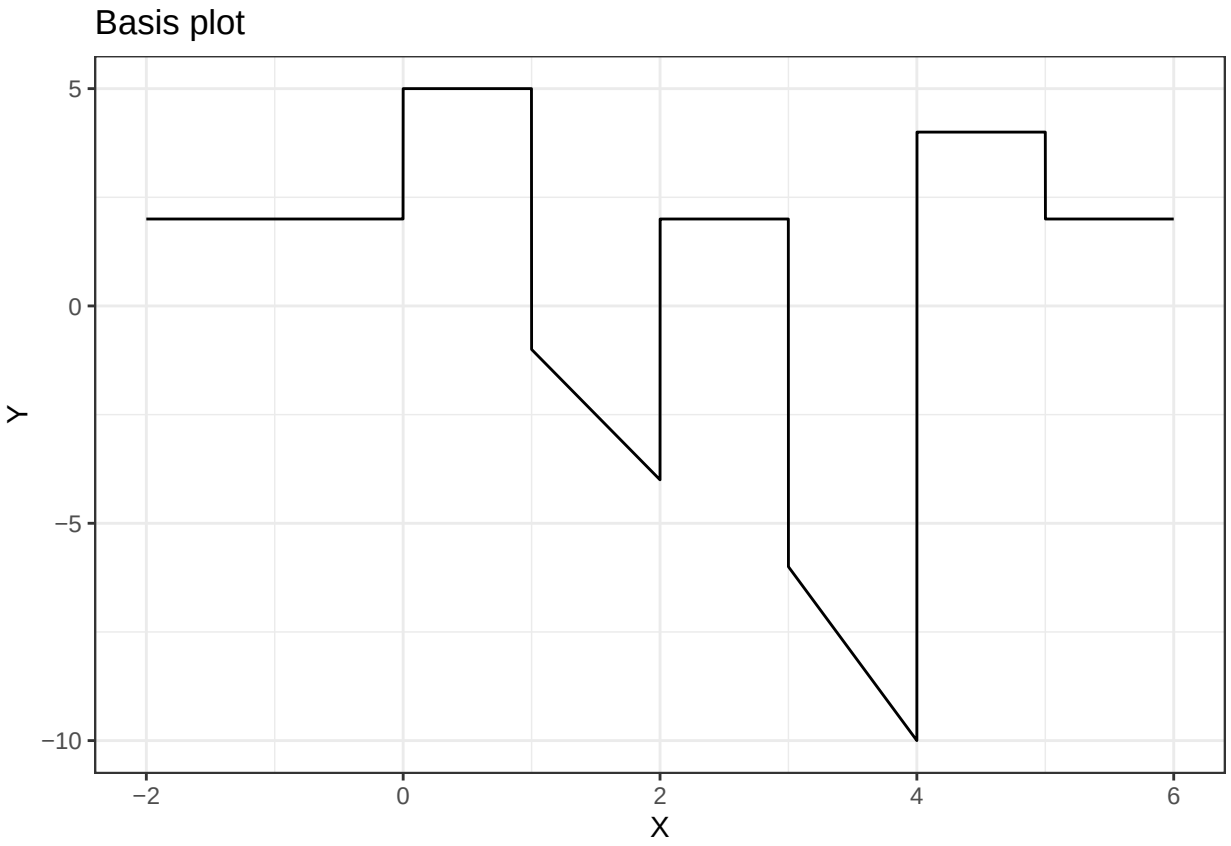


Figure 1: Problem 1

```
f <- function(x) {
    2 + 3*b1(x) - 2*b2(x)
}

xs <- seq(from=-2, to=6, by=0.001)
ys <- map_dbl(xs, f)

df <- cbind(data.frame(xs), data.frame(ys))

ggplot(data=df, aes(x=xs, y=ys)) +
    geom_line() +
    labs(title="Basis plot",x="X", y = "Y") +
    theme_bw()
```




```
yint <- f(0)
```

The plot estimates the curve using by subdividing each integer into 1,000 x-values. The y-intercept is 5.000. The slope is non-zero only where $1 \leq x \leq 2$, where slope is -2, and $3 \leq X \leq 4$, where slope is -4

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 ψ.

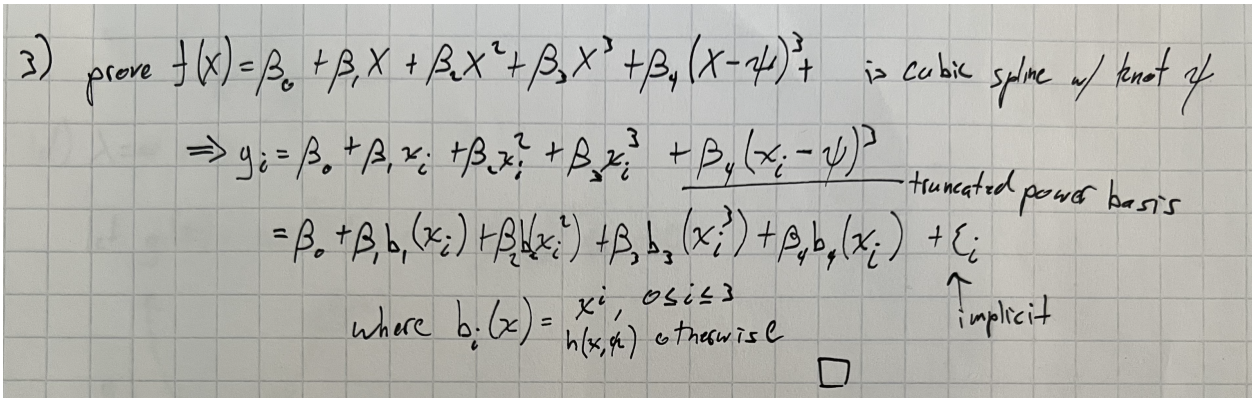


Figure 2: Problem 1

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.

(a) polynomial

```
library(ISLR2)

splitData <- function(df, trainProportion) {
    df <- df %>% mutate(id = row_number())

    set.seed(1)
    train <- sample_frac(df, trainProportion)
    test <- anti_join(df, train, by='id')

    df <- df %>% mutate(id=NULL)

    return(list(train=train, test=test))
}

calcError <- function(ys, ys_hat) {
    mean((ys - ys_hat)^2)
}

myPred <- function(model, df, ysLabel) {

    preds <- predict(model, newdata=df, se=TRUE)
    df[paste0("predicted_", ysLabel)] <- preds$fit
    df["seTop"] <- preds$fit + 2*preds$se.fit
    df["seBot"] <- preds$fit - 2*preds$se.fit
    return(df)
}

drawPlot <- function(df, title) {
    ggplot(data=df, aes(x=age, y=wage)) +
        geom_point(size=0.25) +
        geom_line(data=df, aes(x=age, y=predicted_wage)) +
```

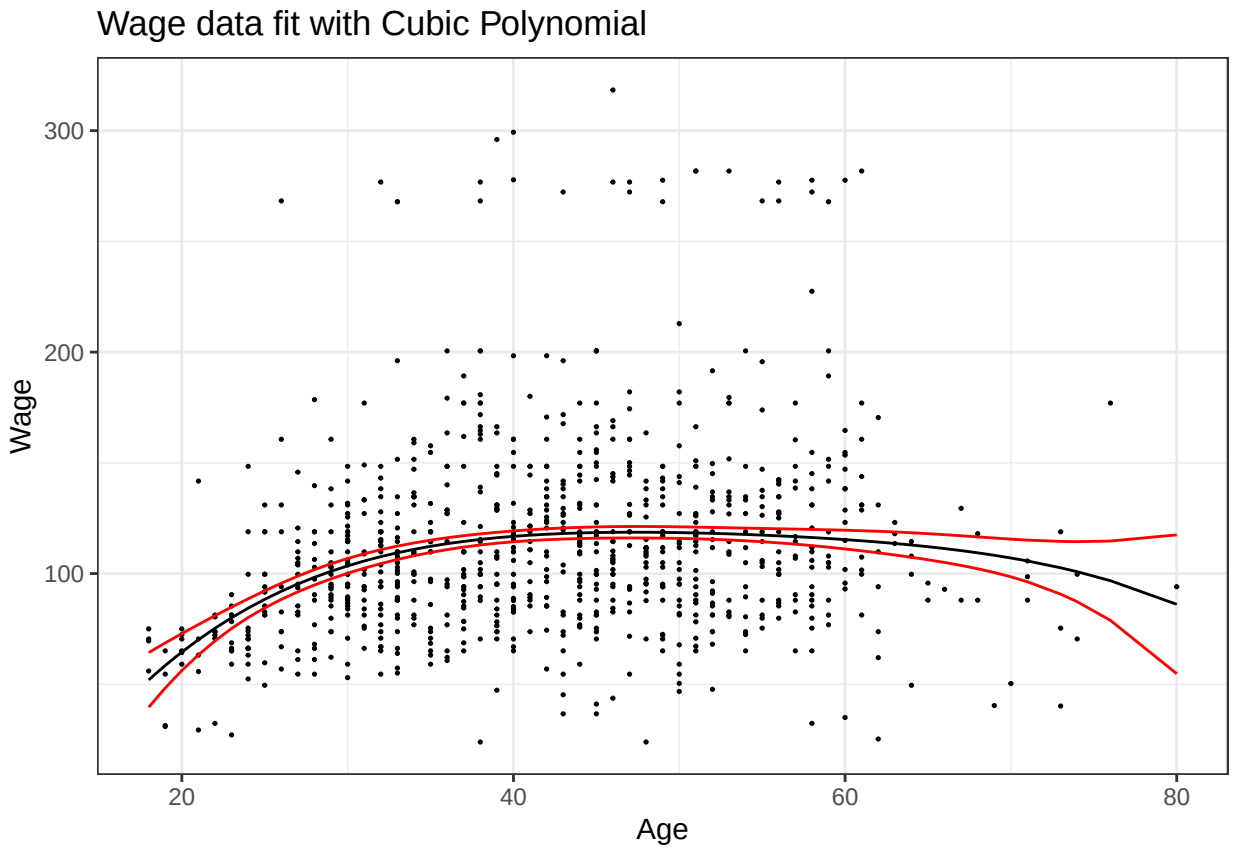
```
        geom_line(data=df, aes(x=age, y=seTop), color="red") +
        geom_line(data=df, aes(x=age, y=seBot), color="red") +
        labs(title=title, x="Age", y = "Wage") +
        theme_bw()
}

dfPoly <- Wage

split <- splitData(dfPoly, 0.7)

# Now make a polynomial model!
pmodel <- lm(wage ~ poly(age, 4), data=split$train)
summary(pmodel)
```

```
##
## Call:
## lm(formula = wage ~ poly(age, 4), data = split$train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -98.008 -24.238  -4.367  15.274 204.751
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    111.0861     0.8489 130.852  < 2e-16 ***
## poly(age, 4)1  363.8027    38.9036   9.351  < 2e-16 ***
## poly(age, 4)2 -392.3496    38.9036 -10.085  < 2e-16 ***
## poly(age, 4)3  116.8166    38.9036   3.003  0.00271 **
## poly(age, 4)4  -60.8166    38.9036  -1.563  0.11814
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 38.9 on 2095 degrees of freedom
## Multiple R-squared:  0.08739,    Adjusted R-squared:  0.08565
## F-statistic: 50.15 on 4 and 2095 DF,  p-value: < 2.2e-16
```

```
testedDf <- myPred(pmodel, split$test, "wage")
pp("MSE: ", calcError(testedDf$wage, testedDf$predicted_wage))
```

```
## [1] "MSE: 1780.86124237119"
```

```
drawPlot(testedDf, "Wage data fit with Cubic Polynomial")
```



(b) step function

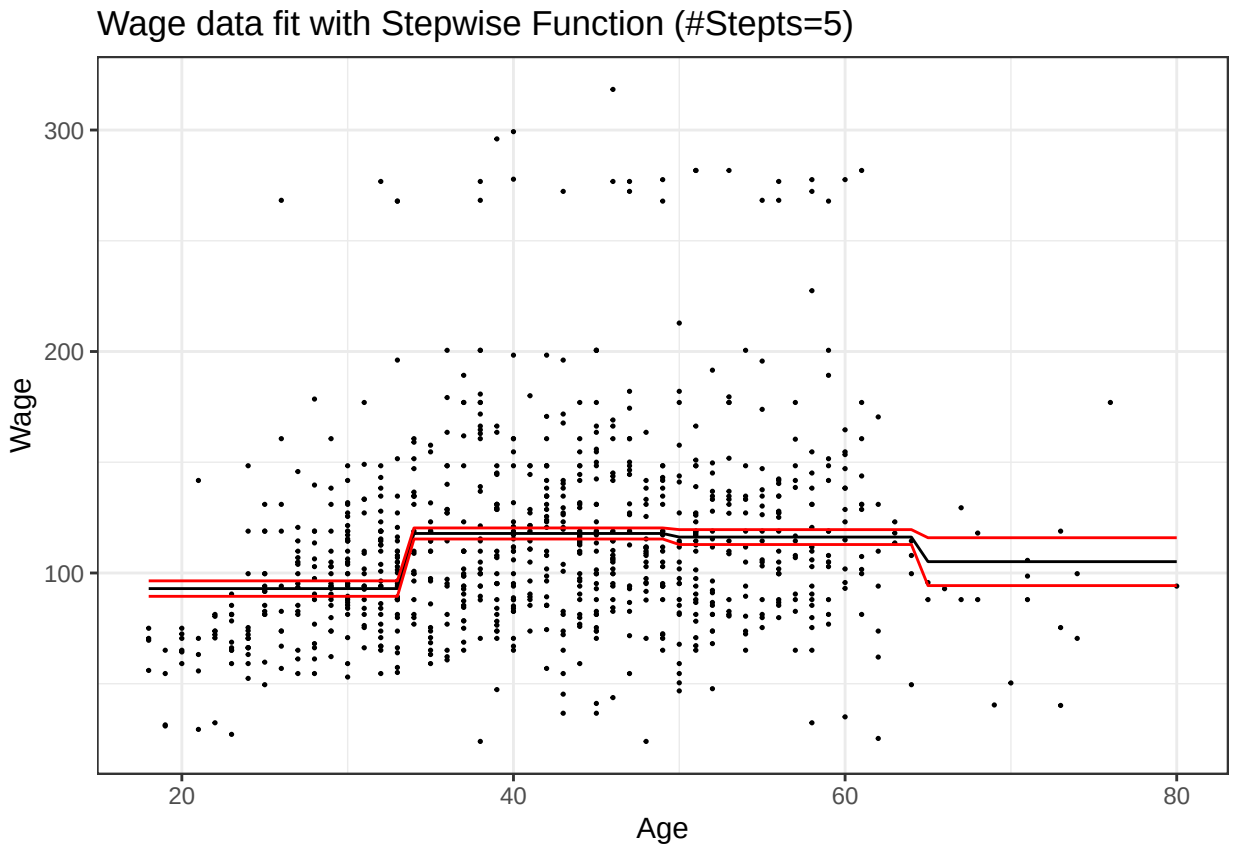
```
pmodel <- lm(wage ~ cut(age, 4), data=split$train)
summary(pmodel)
```

```
##
## Call:
## lm(formula = wage ~ cut(age, 4), data = split$train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -97.755 -24.205  -5.192  16.865 202.128
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)               92.953      1.748  53.184   <2e-16 ***
## cut(age, 4)(33.5,49]      24.888      2.148  11.584   <2e-16 ***
## cut(age, 4)(49,64.5]      23.261      2.425   9.592   <2e-16 ***
## cut(age, 4)(64.5,80.1]    12.157      5.681   2.140   0.0325 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.35 on 2096 degrees of freedom
## Multiple R-squared:  0.0657, Adjusted R-squared:  0.06436
## F-statistic: 49.13 on 3 and 2096 DF,  p-value: < 2.2e-16
```

```
testedDf <- myPred(pmodel, split$test, "wage")
pp("MSE: ", calcError(testedDf$wage, testedDf$predicted_wage))
```

```
## [1] "MSE: 1837.4269012853"
```

```
drawPlot(testedDf, "Wage data fit with Stepwise Function (#Stepts=5)")
```



(c) piecewise polynomial

```
pmodel <- lm(wage ~ cut(age, 4) + poly(age, 4), data=split$train)
summary(pmodel)
```

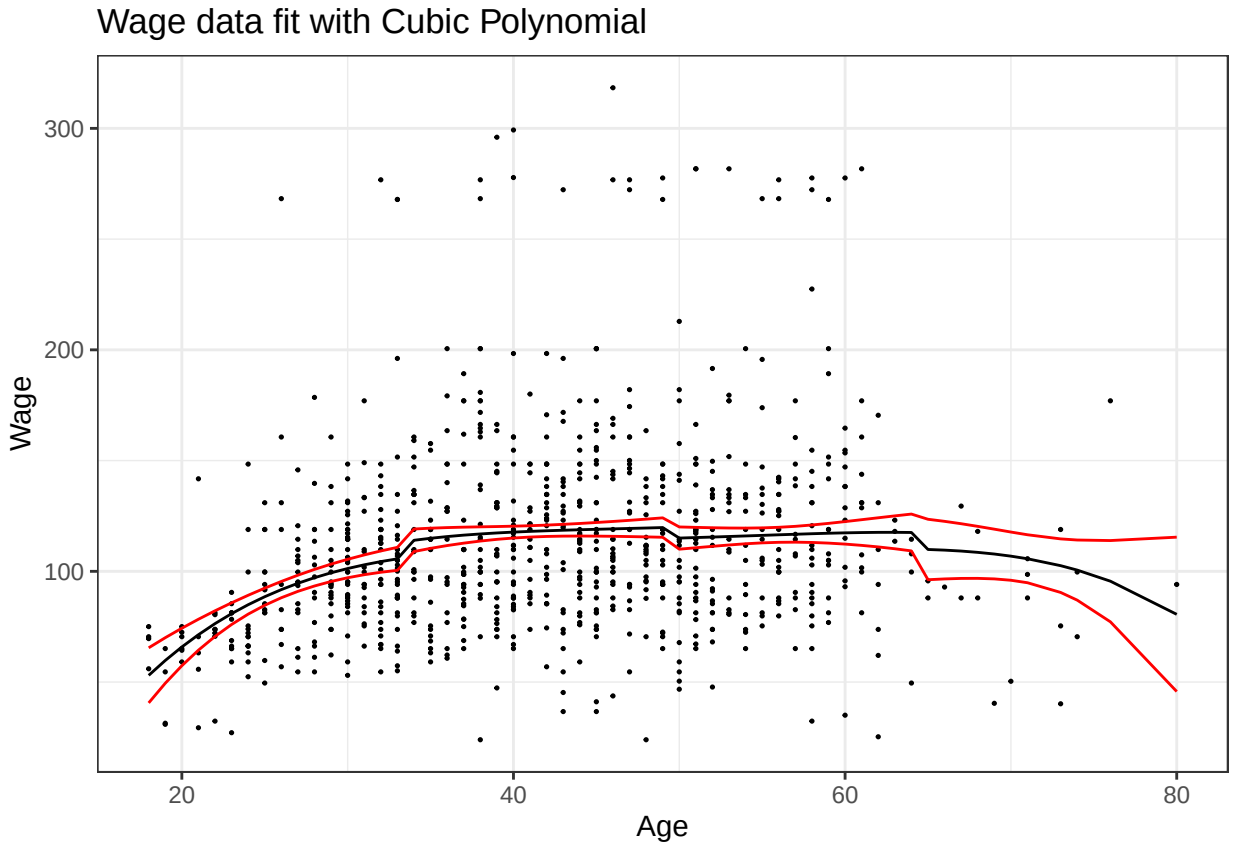
```
##
## Call:
## lm(formula = wage ~ cut(age, 4) + poly(age, 4), data = split$train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -98.465 -23.483  -5.059  15.375 200.735
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              107.145      3.771  28.409  < 2e-16 ***
## cut(age, 4)(33.5,49]       7.313      4.435   1.649 0.099345 .
## cut(age, 4)(49,64.5]       2.362      6.301   0.375 0.707742
## cut(age, 4)(64.5,80.1]    -5.148     11.530  -0.446 0.655289
## poly(age, 4)1             356.627    107.163   3.328 0.000890 ***
## poly(age, 4)2            -269.106     73.075  -3.683 0.000237 ***
## poly(age, 4)3             102.682     51.563   1.991 0.046566 *
## poly(age, 4)4            -106.108     44.573  -2.381 0.017375 *
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 38.89 on 2092 degrees of freedom
## Multiple R-squared:  0.08952,    Adjusted R-squared:  0.08648
## F-statistic: 29.39 on 7 and 2092 DF,  p-value: < 2.2e-16
```

```
testedDf <- myPred(pmodel, split$test, "wage")
pp("MSE: ", calcError(testedDf$wage, testedDf$predicted_wage))
```

```
## [1] "MSE: 1779.95654597275"
```

```
drawPlot(testedDf, "Wage data fit with Cubic Polynomial")
```



(d) cubic spline

```
library(splines)

pmodel <- lm(wage ~ bs(age, knots=c(30, 45, 60)), data=split$train)
summary(pmodel)
```

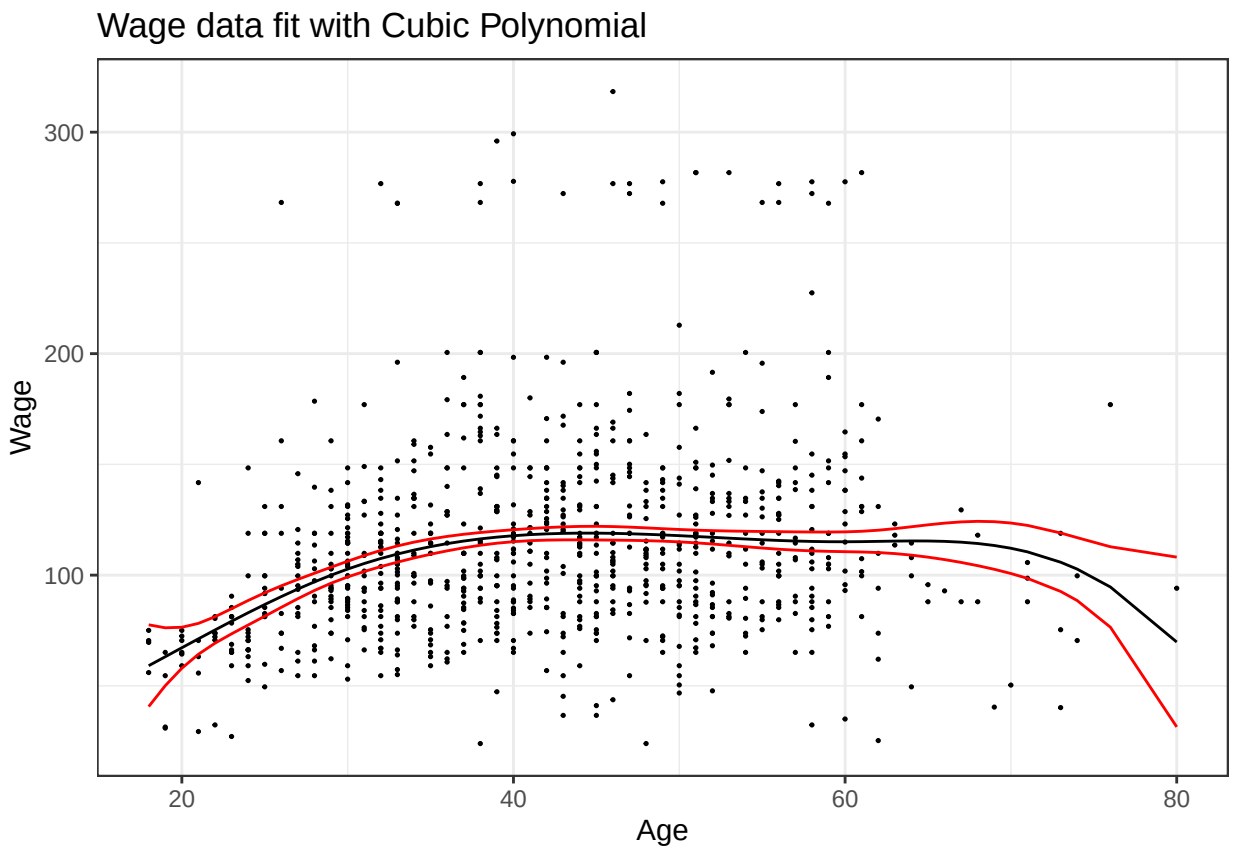
```
##
## Call:
## lm(formula = wage ~ bs(age, knots = c(30, 45, 60)), data = split$train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -98.772 -23.968  -4.741  15.384 202.984
```

```
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       59.107      9.232   6.402 1.89e-10 ***
## bs(age, knots = c(30, 45, 60))1   16.142     13.517   1.194 0.232540
## bs(age, knots = c(30, 45, 60))2   54.230      9.152   5.925 3.63e-09 ***
## bs(age, knots = c(30, 45, 60))3   63.364     10.926   5.799 7.67e-09 ***
## bs(age, knots = c(30, 45, 60))4   50.831     10.745   4.731 2.39e-06 ***
## bs(age, knots = c(30, 45, 60))5   63.098     16.822   3.751 0.000181 ***
## bs(age, knots = c(30, 45, 60))6   10.694     21.067   0.508 0.611773
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 38.9 on 2093 degrees of freedom
## Multiple R-squared:  0.08856,    Adjusted R-squared:  0.08594
## F-statistic: 33.89 on 6 and 2093 DF,  p-value: < 2.2e-16
```

```
testedDf <- myPred(pmodel, split$test, "wage")
pp("MSE: ", calcError(testedDf$wage, testedDf$predicted_wage))
```

```
## [1] "MSE: 1787.57488349256"
```

```
drawPlot(testedDf, "Wage data fit with Cubic Polynomial")
```



(e) smoothing spline

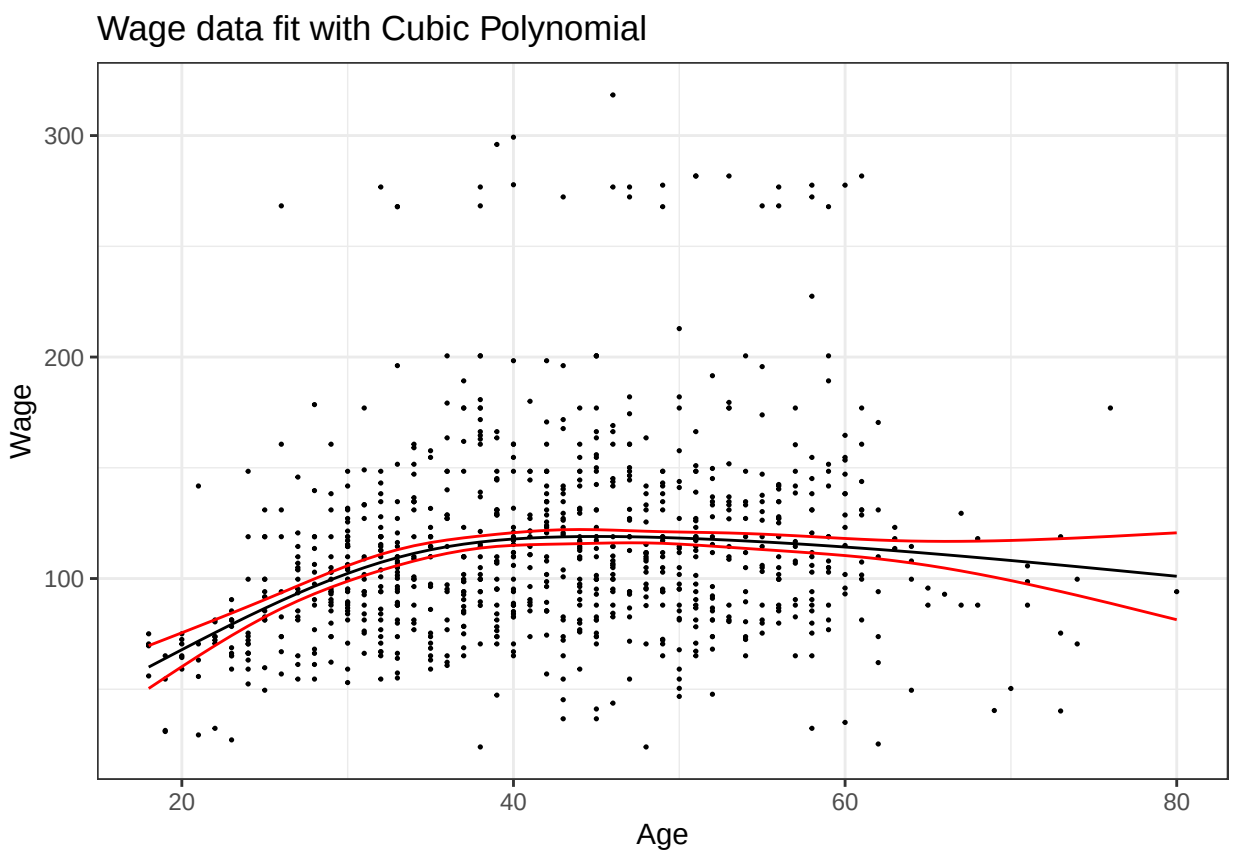
```
pmodel <- lm(wage ~ ns(age, df=4), data=split$train)
summary(pmodel)
```

```
##
## Call:
## lm(formula = wage ~ ns(age, df = 4), data = split$train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -98.736 -23.899  -4.537  15.235 205.777
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         60.026      4.838  12.407  < 2e-16 ***
## ns(age, df = 4)1    59.778      4.771  12.528  < 2e-16 ***
## ns(age, df = 4)2    36.396      5.012   7.261 5.39e-13 ***
## ns(age, df = 4)3    94.964     12.013   7.905 4.28e-15 ***
## ns(age, df = 4)4    14.690     10.104   1.454    0.146
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 38.9 on 2095 degrees of freedom
## Multiple R-squared:  0.08733,    Adjusted R-squared:  0.08559
## F-statistic: 50.12 on 4 and 2095 DF,  p-value: < 2.2e-16
```

```
testedDf <- myPred(pmodel, split$test, "wage")
pp("MSE: ", calcError(testedDf$wage, testedDf$predicted_wage))
```

```
## [1] "MSE: 1783.36240553229"
```

```
drawPlot(testedDf, "Wage data fit with Cubic Polynomial")
```

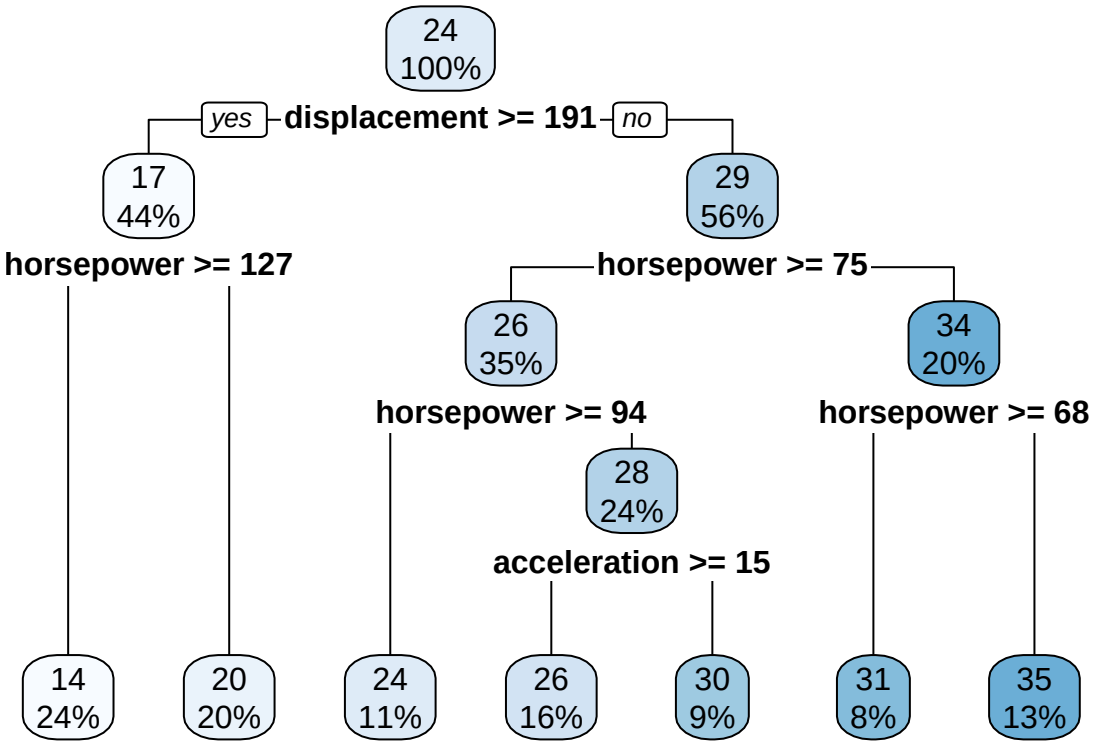


Using RSS as the performance metric,

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.

```
library(tree)
library(rpart)
library(rpart.plot)

dfAuto <- Auto
dfAuto <- subset(dfAuto, select=-c(name))
split <- splitData(dfAuto, 0.7)

set.seed(1)
tmodel <- rpart(mpg ~ cylinders + displacement + horsepower + weight + acceleration + factor(origin), da

rpart.plot(tmodel)
```



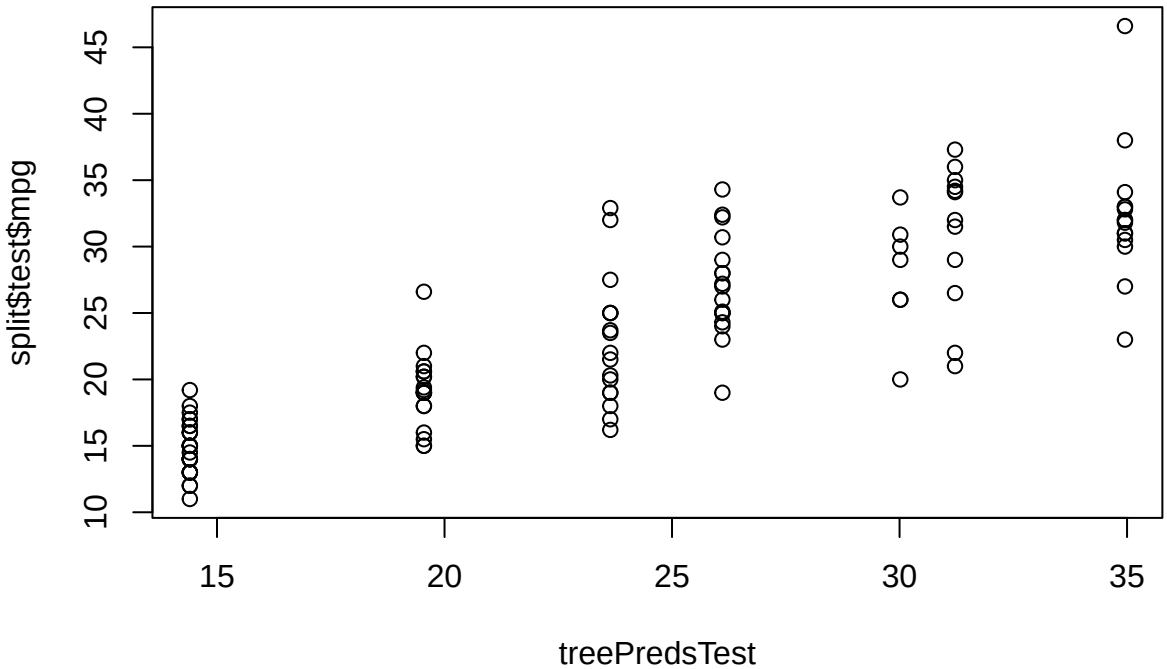

```
treePredsTrain <- predict(tmodel, split$train)
treePredsTest <- predict(tmodel, split$test)

pp("MSE train: ", calcError(split$train$mpg, treePredsTrain))
```

```
## [1] "MSE train: 13.8631774814616"
```

```
pp("MSE test: ", calcError(split$test$mpg, treePredsTest))
```

```
## [1] "MSE test: 15.42665537053"
```

```
plot(treePredsTest, split$test$mpg)
```



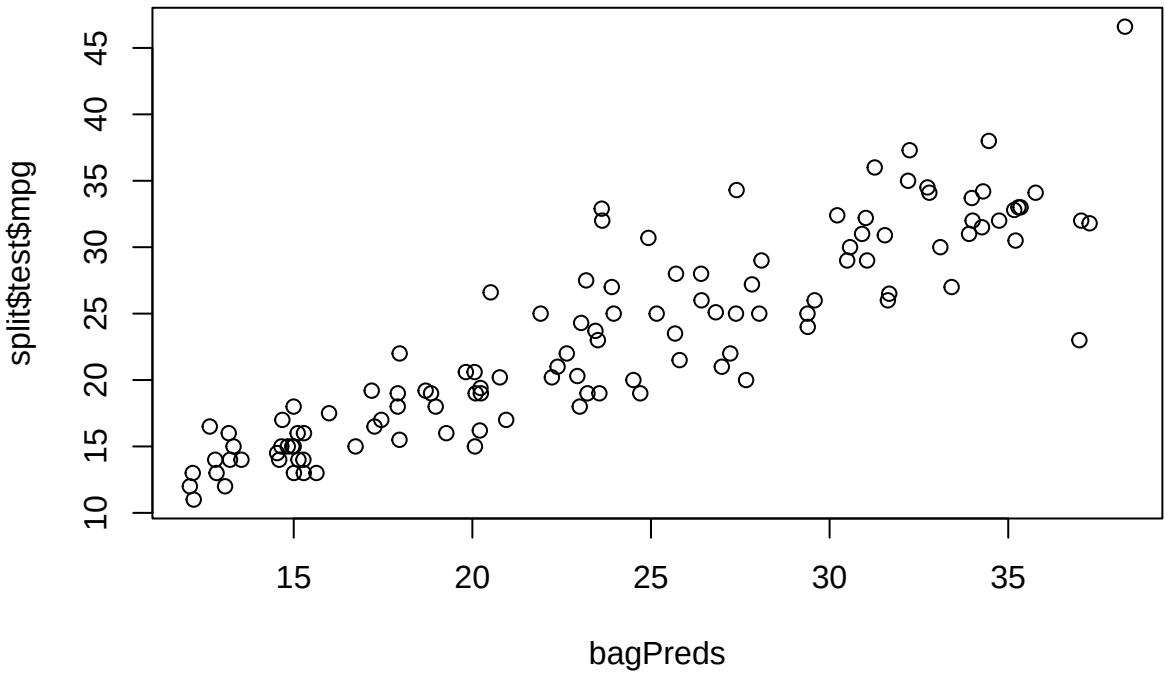
I considered pruning the tree further, but MSE between train and test is relatively small, so it's possible that different samples of data would yield similar, albeit slightly fluctuating results. Overall, the model performs surprisingly well. MSE is high relative to the range of the model. The tree is already quite simple with using the top four relevant parameters.

(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?

```
library(randomForest)

set.seed(1)
bag.auto <- randomForest(mpg ~ cylinders + displacement + horsepower + weight + acceleration + origin,
                         data=split$train,
                         mtry=6, importance=TRUE)

bag.auto
```

```
##
## Call:
##  randomForest(formula = mpg ~ cylinders + displacement + horsepower +      weight + acceleration + o
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##           Mean of squared residuals: 15.04435
##                     % Var explained: 76.14
```

```
bagPreds <- predict(bag.auto, split$test)

plot(bagPreds, split$test$mpg)
```




```
pp("MSE test: ", calcError(split$test$mpg, bagPreds))
```

```
## [1] "MSE test: 12.179646943562"
```

Hyperparameters: 500 trees, $m = p$. The model had a slightly higher training MSE, implying a more general fit. This happened to lead to lower testing error for this particular testing set relative to the single decision tree. Overall, the results are surprisingly comparable to the simple tree.

(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?

```
library(randomForest)

set.seed(1)
bag.auto <- randomForest(mpg ~ cylinders + displacement + horsepower + weight + acceleration + origin,
                         data=split$train,
                         mtry=3, importance=TRUE)

bag.auto
```

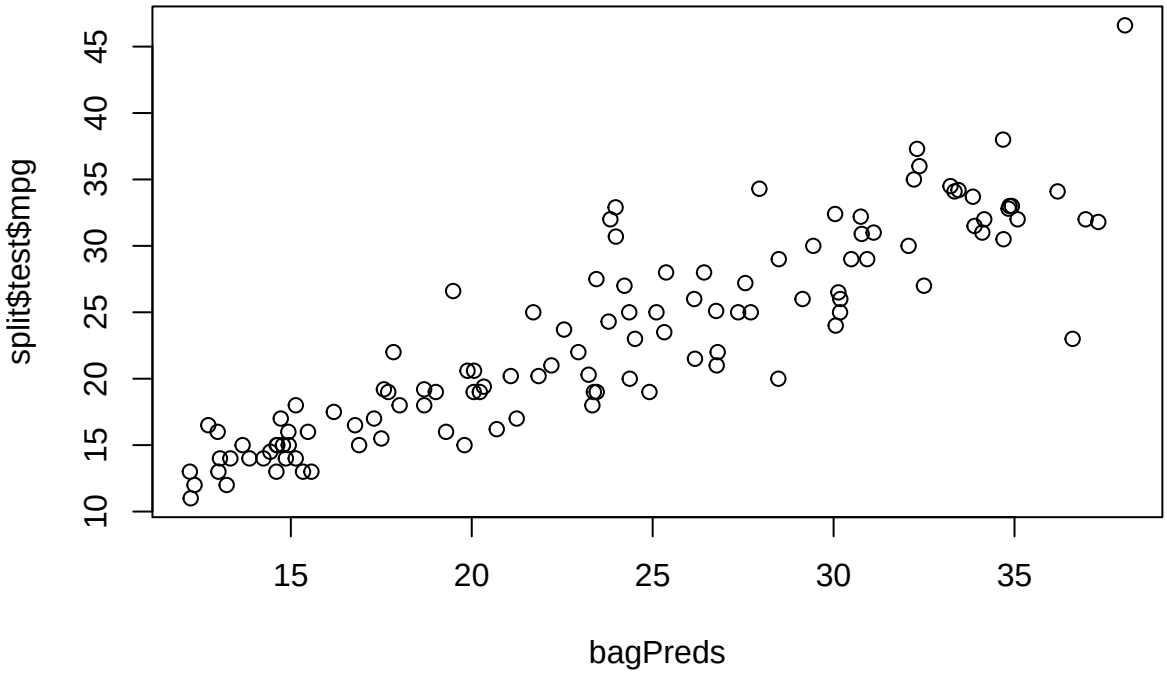
```
##
## Call:
##  randomForest(formula = mpg ~ cylinders + displacement + horsepower +      weight + acceleration + o
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 14.6604
##                     % Var explained: 76.75
```

```
importance(bag.auto)
```

```
##               %IncMSE IncNodePurity
## cylinders    12.41019     2901.5786
```

```
## displacement 23.39558     5463.6541
## horsepower   25.21255     3236.4577
## weight       17.58922     4001.5744
## acceleration 17.18069     1068.9828
## origin       10.22236      223.9051
```

```
bagPreds <- predict(bag.auto, split$test)

plot(bagPreds, split$test$mpg)
```




```
pp("MSE test: ", calcError(split$test$mpg, bagPreds))
```

```
## [1] "MSE test: 11.8225232952906"
```

The only difference between C and B is the number of variables considered at teach tree split. Using only the three most important variables improves training MSE while not penalizing test MSE, implying that we're closer to the optimal combination of flexibility and generalizability. Like each of the previous models, we see increased error variance as precitions increase in value.

(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.

```
library(gam)

gmodel <- gam(mpg ~ s(cylinders) + s(displacement) + s(horsepower) + s(weight) + s(acceleration) + facto
              data=split$train)

summary(gmodel)
```

```
##
## Call: gam(formula = mpg ~ s(cylinders) + s(displacement) + s(horsepower) +
##     s(weight) + s(acceleration) + factor(origin), data = split$train)
## Deviance Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -8.9106 -2.1824 -0.2299  1.7978 17.3201
##
## (Dispersion Parameter for gaussian family taken to be 13.781)
##
##     Null Deviance: 17273.82 on 273 degrees of freedom
## Residual Deviance: 3459.043 on 250.9999 degrees of freedom
## AIC: 1520.338
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##                   Df  Sum Sq Mean Sq  F value    Pr(>F)
## s(cylinders)       1 10306.1 10306.1 747.8469 < 2.2e-16 ***
## s(displacement)    1  1169.8  1169.8  84.8850 < 2.2e-16 ***
## s(horsepower)      1   456.5   456.5  33.1236 2.512e-08 ***
## s(weight)          1   209.9   209.9  15.2293 0.0001224 ***
## s(acceleration)    1   184.1   184.1  13.3625 0.0003126 ***
## factor(origin)     2   174.7    87.3   6.3382 0.0020636 **
## Residuals        251  3459.0    13.8
## ---
## 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  9.5617 5.308e-06 ***
## s(displacement)        3  1.7539  0.156532
## s(horsepower)          3 11.1302 6.963e-07 ***
## s(weight)              3  0.4436  0.722020
## s(acceleration)        3  5.1357  0.001832 **
## factor(origin)
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
pp("MSE test: ", calcError(split$test$mpg, predict(gmodel, split$test)))
```

```
## [1] "MSE test: 12.4491962496014"
```

We see high significance for all predictors.

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

I'd go with the GAM, simply because it makes use more predictors with better performance than any of the tree-based models.