Stats 101C HW 6

Anna Piskun

11/29/2020

Problem 5.4.2

2. We will now derive the probability that a given observation is part of a bootstrap sample. Suppose that we obtain a bootstrap sample from a set of n observations.

(a) What is the probability that the first bootstrap observation is not the jth observation from the original sample? Justify your answer.

The probability that the first bootstrap observation is not the jth observation from the original sample is 1 - 1/n. We find this by noting that n is the sample size and thus there are n samples. Since we are discussing the first observation and bootstrap sampling works by choosing observations with replacement, we are equally likely to pick each observation on the first try giving us 1 - 1/n.

(b) What is the probability that the second bootstrap observation is not the jth observation from the original sample?

As we mentioned earlier, since the mechanics of bootstrap sampling requires picked observations to be replaced, the probability that the second bootstrap observation is not the jth observation from the original sample is still 1 - 1/n.

(c) Argue that the probability that the jth observation is not in the bootstrap sample is (1 — 1/n)^n.

From parts a and b, we can conclude that the probability that any bootstrap observation is not the jth observation is 1-1/n (again due to drawing with replacement). Thus, if we want to find the probability that the jth observation is not in the bootstrap sample as a whole, we must take a total of n bootstrap observations from our original sample. This gives us a probability of (1-1/n)^n since we can assume that each observation is independent due to the replacement factor found in bootstrapping.

(d) When n = 5, what is the probability that the jth observation is in the bootstrap sample?

When n = 5, the probability that the jth observation is in the boostrap sample is 0.67232.

```
#probability that the jth observation is NOT in the bootstrap sample
(1 - 1/5)^5
```

```
## [1] 0.32768
```

```
#we find the probability that the jth observation is IN the bootstrap sample
#by subtracting the value found above from 1

1-(1 - 1/5)^5
```

```
## [1] 0.67232
```

(e) When n = 100, what is the probability that the jth observation is in the bootstrap sample?

When n = 100, the probability that the jth observation is in the bootstrap sample is 0.6339677.

```
#probability that the jth observation is NOT in the bootstrap sample
(1 - 1/100)^100
```

```
## [1] 0.3660323
```

```
#we find the probability that the jth observation is IN the bootstrap sample
#by subtracting the value found above from 1

1-(1 - 1/100)^100
```

```
## [1] 0.6339677
```

(f) When n = 10, 000, what is the probability that the jth observation is in the bootstrap sample?

When n = 10,000, the probability that the jth observation is in the boostrap sample is 0.632139.

```
#probability that the jth observation is NOT in the bootstrap sample
(1 - 1/10000)^10000
```

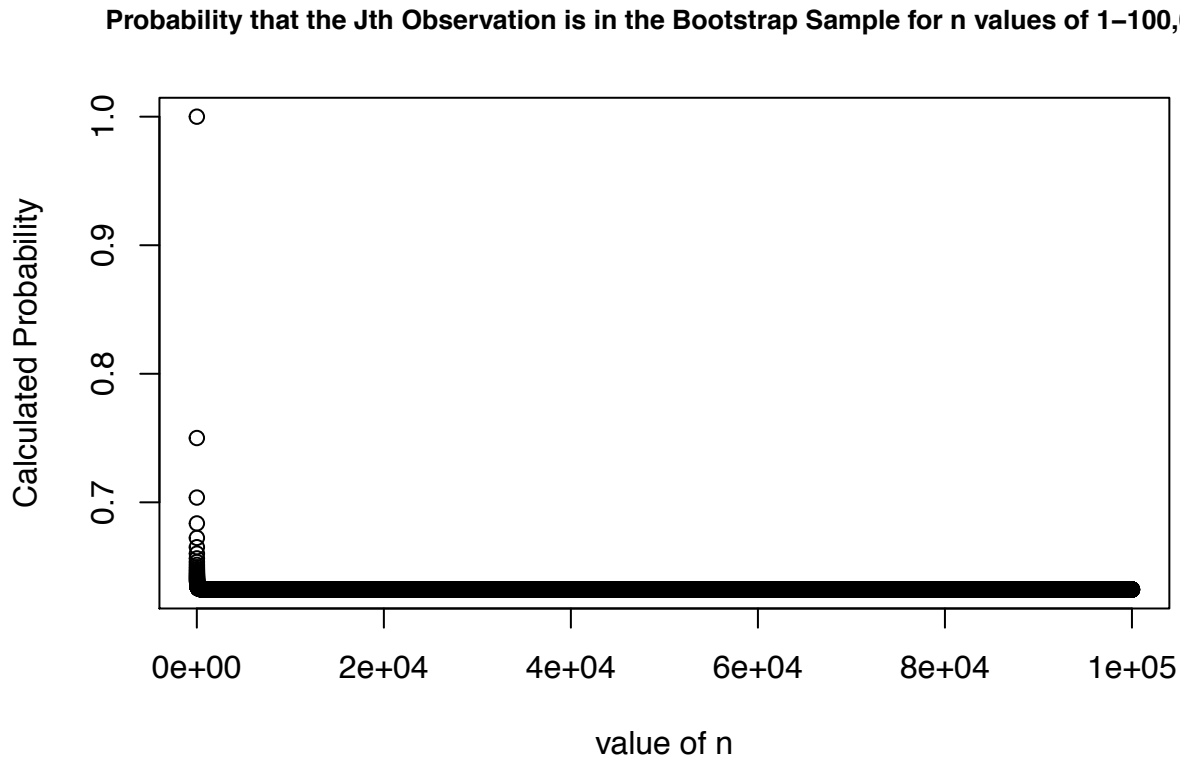
```
## [1] 0.367861
```

```
#we find the probability that the jth observation is IN the bootstrap sample
#by subtracting the value found above from 1

1-(1 - 1/10000)^10000
```

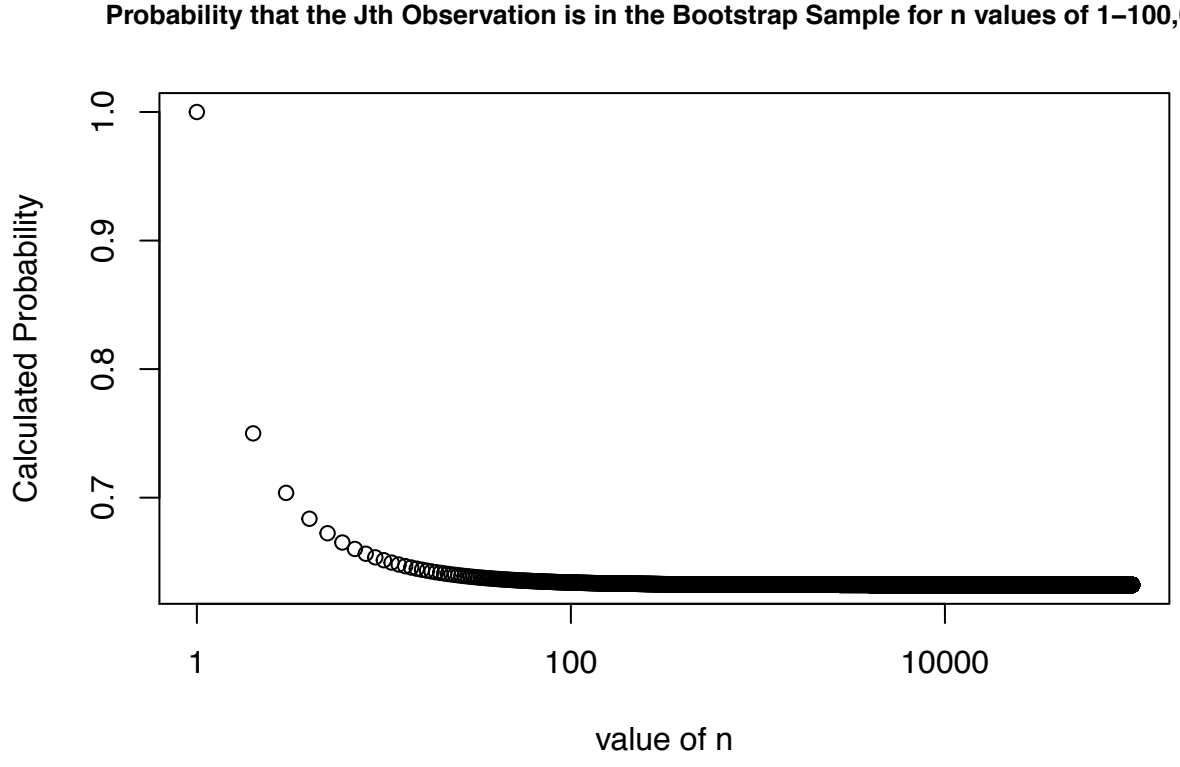
```
## [1] 0.632139
```

(g) Create a plot that displays, for each integer value of n from 1 to 100,000, the probability that the jth observation is in the bootstrap sample. Comment on what you observe.

```
x <- seq(1, 100000)
y <-function(n){
 z <-1 -(1-1/n)^n
return(z)
  }

plot(x, y(x), xlab="value of n", ylab="Calculated Probability", main = "Probability that the Jth Observa
```




```
plot(x, y(x), xlab="value of n", ylab="Calculated Probability", main = "Probability that the Jth Observa
```



We notice from the plot that the probabilities seem to converge to a clear horizontal asymptote. Using a log transformation of x to better visualize this asymptote, we find that the probabilities converge around 0.6 and a n value of around 100.

(h) We will now investigate numerically the probability that a bootstrap sample of size n = 100 contains the jth observation. Here j = 4. We repeatedly create bootstrap samples, and each time we record whether or not the fourth observation is contained in the bootstrap sample. Comment on the results obtained.

```
store <- rep(NA, 10000)

for(i in 1:10000){
  store[i] <- sum(sample(1:100, rep=TRUE)==4)>0}

mean(store)
```

```
## [1] 0.6378
```

Using the given code, every time I re-run and create a new bootstrap sample, I get a value of approximately 0.63. This tells us that each time we created a bootstrap sample, it sampled 0-100 with replacement and found that around 63% of the time the sample contained the number 4. This supports the earlier conclusion found by looking at the plots, that the probabilities do in fact converge at a probability of around 0.63 with an n value of 100.

Problem 8.4.10

```
library(ISLR)
attach(Hitters)
nrow(Hitters)
```

```
## [1] 322
```

10. We now use boosting to predict Salary in the Hitters data set.

(a) Remove the observations for whom the salary information is unknown, and then log-transform the salaries.

```
data_clean <- na.omit(Hitters)
nrow(data_clean)
```

```
## [1] 263
```

```
data_clean$Salary <- log(data_clean$Salary)
```

(b) Create a training set consisting of the first 200 observations, and a test set consisting of the remaining observations.

```
i.train <- 1:200
hit.train <- data_clean[i.train,]
hit.test <- data_clean[-i.train,]
head(hit.train)
```

```
##                   AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun
## -Alan Ashby         315   81     7   24  38    39    14   3449   835      69
## -Alvin Davis        479  130    18   66  72    76     3   1624   457      63
## -Andre Dawson       496  141    20   65  78    37    11   5628  1575     225
## -Andres Galarraga   321   87    10   39  42    30     2    396   101      12
## -Alfredo Griffin    594  169     4   74  51    35    11   4408  1133      19
## -Al Newman          185   37     1   23   8    21     2    214    42       1
##                   CRuns CRBI CWalks League Division PutOuts Assists Errors
```

```
## -Alan Ashby          321  414    375      N        W     632      43     10
## -Alvin Davis         224  266    263      A        W     880      82     14
## -Andre Dawson        828  838    354      N        E     200      11      3
## -Andres Galarraga     48   46     33      N        E     805      40      4
## -Alfredo Griffin     501  336    194      A        W     282     421     25
## -Al Newman            30    9     24      N        E      76     127      7
##                     Salary NewLeague
## -Alan Ashby       6.163315         N
## -Alvin Davis      6.173786         A
## -Andre Dawson     6.214608         N
## -Andres Galarraga 4.516339         N
## -Alfredo Griffin  6.620073         A
## -Al Newman        4.248495         A
```

```
nrow(hit.train)
```

```
## [1] 200
```

(c) Perform boosting on the training set with 1,000 trees for a range of values of the shrinkage parameter lambda. Produce a plot with different shrinkage values on the x-axis and the corresponding training set MSE on the y-axis.

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

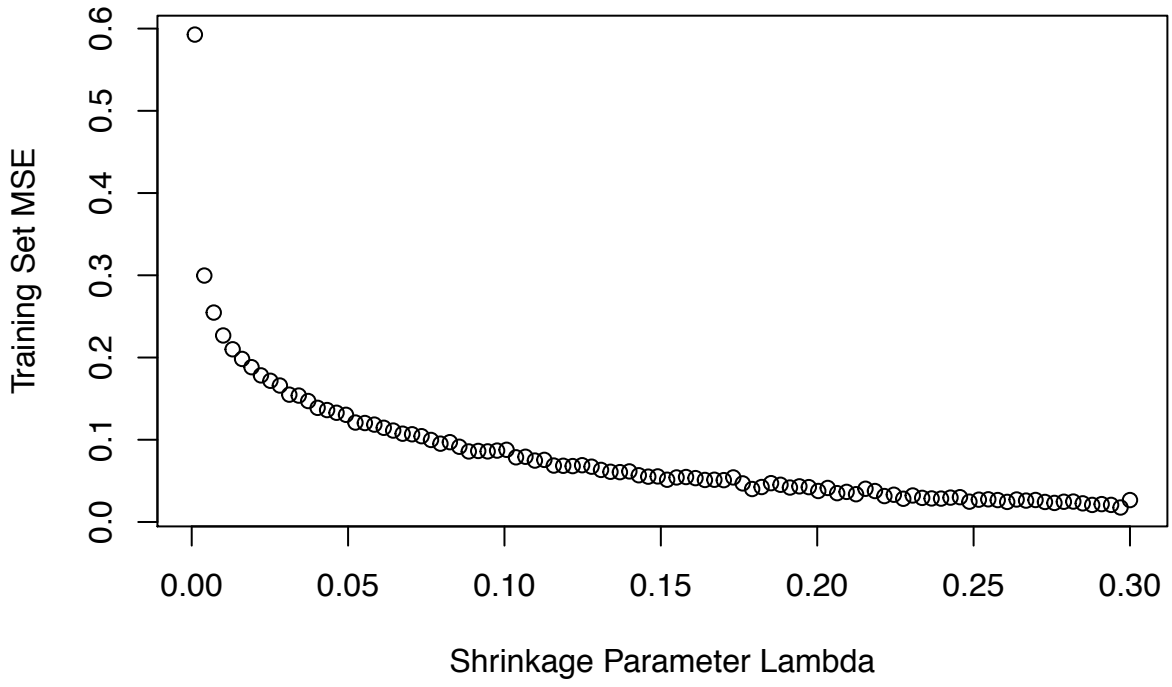
```
set.seed(123)

lambda <- seq(0.001, 0.3, length = 100)
tr_errors <- length(lambda)
n <- nrow(Hitters)
n_train <- nrow(hit.train)
n_val <- n - n_train

for (i in 1:tr_errors) {
  boost_hitters <- gbm(Salary ~., data = hit.train, n.trees = 1000, distribution = "gaussian", shrinkag
  yhat.train <- predict(boost_hitters, hit.train)
  tr_errors[i] <- sum((hit.train$Salary - yhat.train)^2)/n_val
}
```

```
## Using 1000 trees...
```

```
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
```

```
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
```

```
plot(lambda, tr_errors, xlab = "Shrinkage Parameter Lambda", ylab = "Training Set MSE")
```

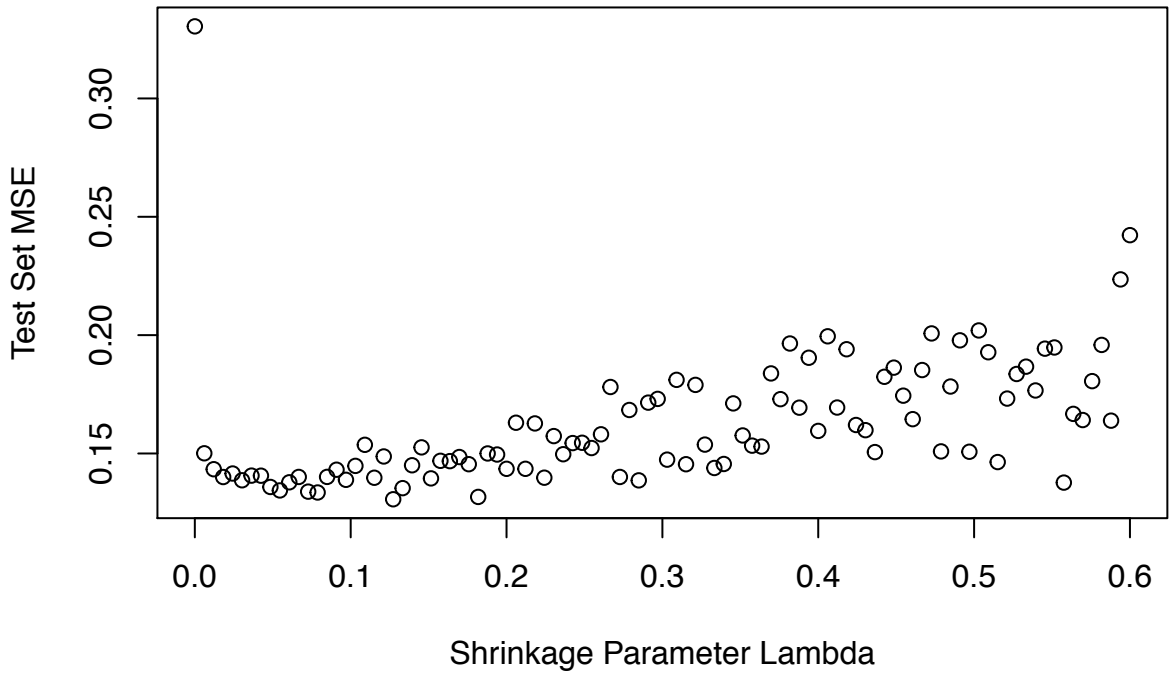


(d) Produce a plot with different shrinkage values on the x-axis and the corresponding test set MSE on the y-axis.

```
set.seed(123)
lambda <- seq(0.00001, 0.6, length = 100)
test_errors <- length(lambda)
n <- nrow(Hitters)
n_train <- nrow(hit.train)
n_val <- n - n_train

for (i in 1:test_errors) {
  boost_hitters <- gbm(Salary ~., data = hit.train, n.trees = 1000, distribution = "gaussian", shrinkag
  yhat.test <- predict(boost_hitters, newdata = hit.test)
  test_errors[i] <- sum((hit.test$Salary - yhat.test)^2)/n_val
}
```

```
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
```

```
plot(lambda, test_errors, xlab = "Shrinkage Parameter Lambda", ylab = "Test Set MSE")
```



(e) Compare the test MSE of boosting to the test MSE that results from applying two of the regression approaches seen in Chapters 3 and 6.

```
set.seed(123)

# fit a linear model

ml <- lm(Salary~., data = hit.train)

predictions <- predict(ml, newdata = hit.test)

MSE.lm <- sum((hit.test$Salary - predictions)^2)/n_val

MSE.lm
```

```
## [1] 0.2539602
```

```
#fit ridge regression model

library(glmnet)
```

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

```
## Loaded glmnet 4.0-2
```

```
x <- model.matrix(Salary~., hit.train)[,-1]
z <- model.matrix(Salary~., hit.test)[, -1]
y <- hit.train$Salary

i.exp <- seq(10, -2, length = 100)
grid <- 10^i.exp
```

```
ridge.mod <- glmnet(x, y, family = "gaussian", alpha = 0,
                    lambda = grid, standardize = TRUE)
set.seed(123)
cv.output <- cv.glmnet(x, y, family = "gaussian", alpha = 0,
                    lambda = grid, standardize = TRUE,
                    nfolds = 10)

best.lambda.cv <- cv.output$lambda.min

ridge.predictions <- predict(ridge.mod, newx = z, s = best.lambda.cv)

MSE.ridge <- sum((hit.test$Salary - ridge.predictions)^2)/n_val

MSE.ridge
```

```
## [1] 0.232007
```

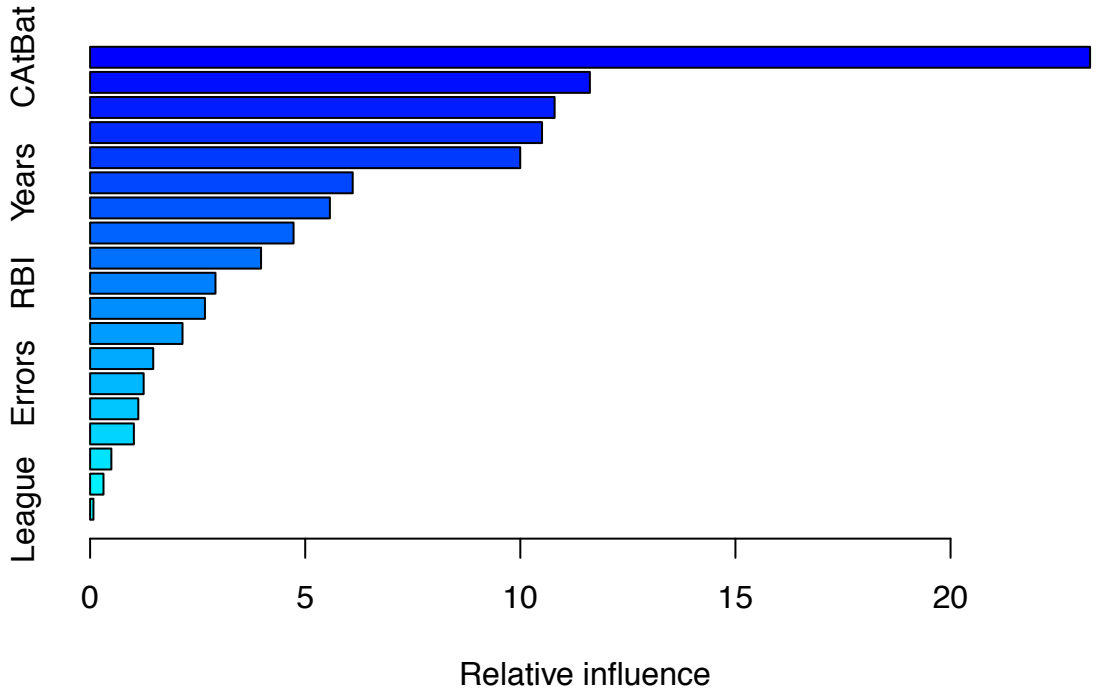
```
#find lowest MSE for boosted model
min(test_errors)
```

```
## [1] 0.1306555
```

Compared to a linear model and ridge regression model, we find that the boosted model has the lowest test MSE of 0.1306555, while the test MSEs for the LM and RR models were 0.2539602 and 0.232007, respectively.

(f) Which variables appear to be the most important predictors in the boosted model?

```
set.seed(123)
hitters_boosted <- gbm(Salary ~., data = hit.train, n.trees = 1000, distribution = "gaussian", shrinkag
summary(hitters_boosted)
```




```
##                var     rel.inf
## CAtBat       CAtBat 23.24340007
```

```
## CHits         CHits 11.61638774
## CRBI           CRBI 10.79677884
## CRuns         CRuns 10.50428420
## CWalks       CWalks  9.99732450
## Years         Years  6.10365648
## CHmRun       CHmRun  5.57340954
## Hits           Hits  4.72926355
## Walks         Walks  3.97325791
## RBI             RBI  2.91374353
## PutOuts     PutOuts  2.66899852
## AtBat         AtBat  2.14770896
## HmRun         HmRun  1.46724078
## Errors       Errors  1.24450076
## Runs           Runs  1.12000154
## Assists     Assists  1.01691360
## Division   Division  0.49421707
## NewLeague NewLeague  0.31121428
## League       League  0.07769812
```

In the boosted models, the most important variables appear to be *League*, *Errors*, *RBI*, *Years*, and *CAtBat*

(g) Now apply bagging to the training set. What is the test set MSE for this approach?

```
set.seed(123)
library(randomForest)
```

```
## randomForest 4.6-14
```

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
bag_hitters <- randomForest(Salary ~ ., data = hit.train, mtry = 19, ntree = 1000, importance = T)
bag_hitters <- predict(bag_hitters, hit.test, n.trees = 1000)
MSE.bag <- sum((hit.test$Salary - bag_hitters)^2)/n_val
MSE.bag
```

```
## [1] 0.1185516
```

The test MSE for this approach is 0.1185516, which is smaller compared to the test MSE for boosting.