HW1

Terence Zhi Liu

2/2/2017

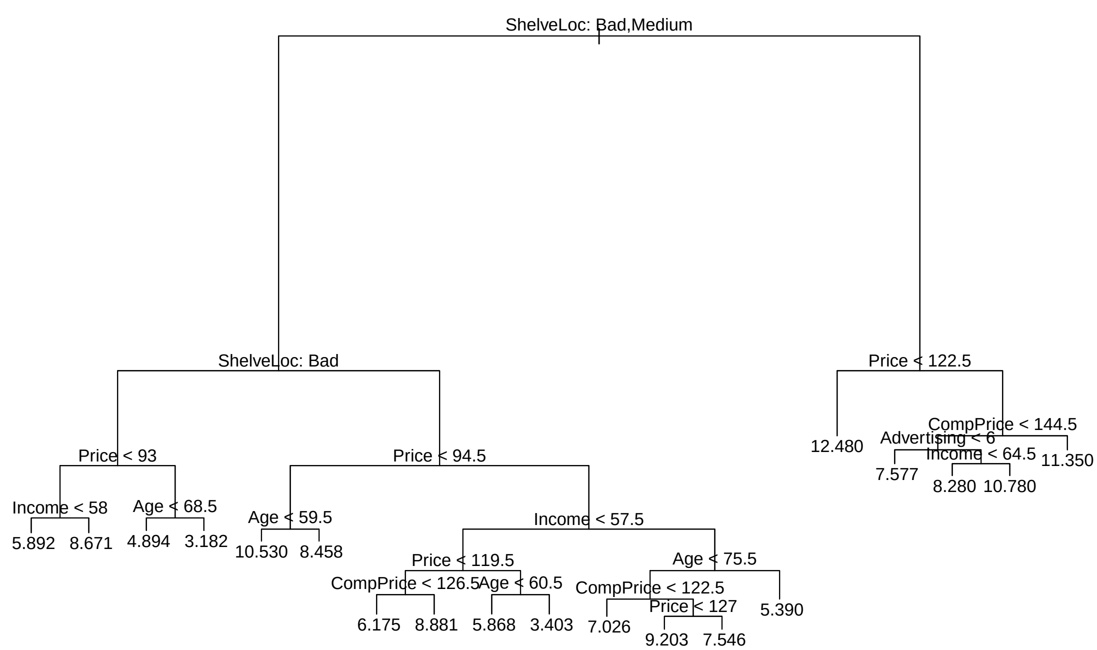
1.

(a) We split the data.

```
set.seed(1)
carseats = read.csv('Carseats.txt')[-1] # get rid of index column
trainIndex <- createDataPartition(carseats$Sales, p = 0.6, list = F)
carseatsTrain = carseats[trainIndex, ]
carseatsTest = carseats[-trainIndex, ]
```

(b) We fit a regression tree to the training set.

```
reg.tree = tree(Sales ~ ., carseatsTrain)
plot(reg.tree)
text(reg.tree, pretty = 0)
```




```
summary(reg.tree)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = carseatsTrain)
```

```
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Income"      "Age"         "CompPrice"
## [6] "Advertising"
## Number of terminal nodes:  19
## Residual mean deviance:  2.259 = 501.5 / 222
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -4.5240 -0.8714 -0.0700  0.0000  1.0080  4.0320
```

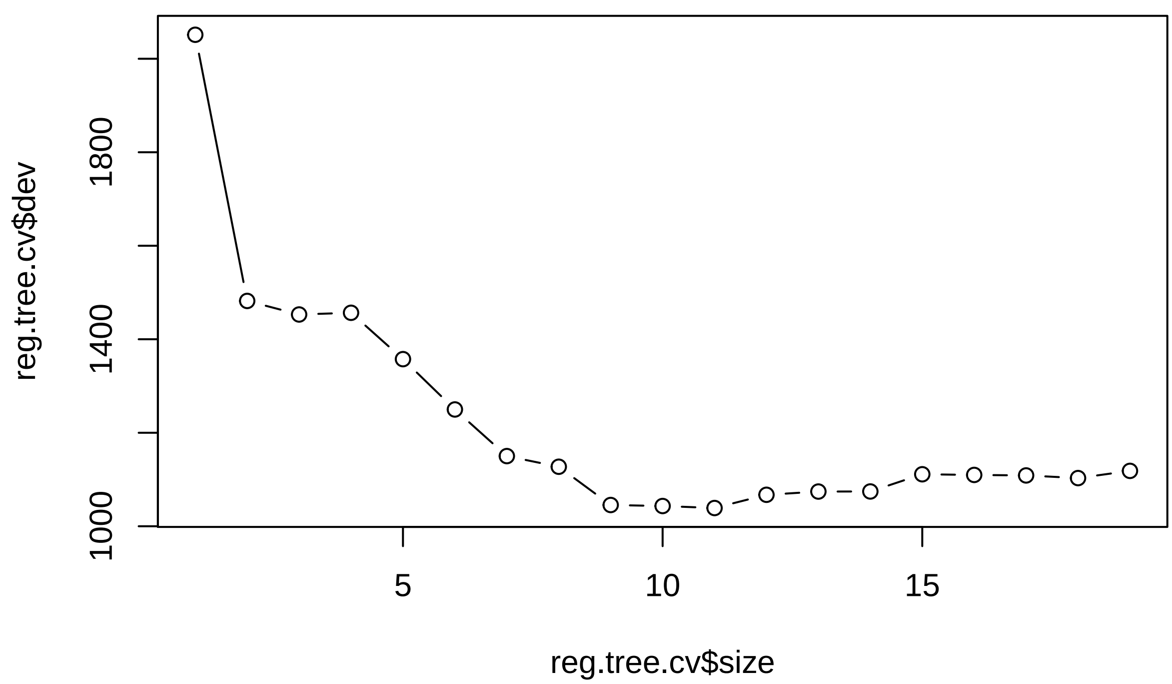
It shows the MSE of the model, 2.26. Now we predict the test set.

```
sales_predicted = predict(reg.tree, newdata = carseatsTest[-1])
MSE = mean((carseatsTest$Sales - sales_predicted)^2)
```

The test set MSE is 4.6872892.

(c) We do CV trees.

```
set.seed(1)
reg.tree.cv = cv.tree(reg.tree)
plot(reg.tree.cv$size, reg.tree.cv$dev, type = 'b')
```



Looks like 11 terminal nodes yield the lowest deviance/MSE. Prune it.

```
reg.tree.prune = prune.tree(reg.tree, best = 11)
sales_predicted = predict(reg.tree.prune, newdata = carseatsTest[-1])
MSE = mean((carseatsTest$Sales - sales_predicted)^2)
```

The test set MSE is 4.985042, which is slightly bigger than the full tree. But this might vary depending on random sampling.

(d) We do bagging.

```
set.seed(1)
reg.bag = randomForest(Sales ~ ., carseatsTrain, mtry = 10, # all variables
                       importance = T)
sales_predicted = predict(reg.bag, newdata = carseatsTest[-1])
MSE = mean((carseatsTest$Sales - sales_predicted)^2)
```

The test set MSE is 2.6440175, much smaller than a single tree.

```
importance(reg.bag)
```

```
##                %IncMSE IncNodePurity
## CompPrice   20.3084581    172.697550
## Income      15.7224836    152.623672
## Advertising 18.8442050    118.506620
## Population   0.1466463     57.456619
## Price       57.3829865    478.068737
## ShelveLoc   67.0722017    759.111978
## Age         17.6662540    172.597449
## Education    2.4831451     50.209699
## Urban       -1.7014201      9.911654
## US           3.6955812      7.925052
```

ShelveLoc is the most important variable.

(e) We do random forest.

```
set.seed(1)
reg.rf = randomForest(Sales ~ ., carseatsTrain, mtry = 4, importance = T)
sales_predicted = predict(reg.rf, newdata = carseatsTest[-1])
MSE = mean((carseatsTest$Sales - sales_predicted)^2)
```

THe test set MSE is 2.5566931.

```
importance(reg.rf)
```

```
##                  %IncMSE IncNodePurity
## CompPrice   16.150237761     184.18659
## Income      12.376003379     189.24614
## Advertising 12.356530644     136.78742
## Population  -1.160936685      94.17359
## Price       43.223647447     431.37145
## ShelveLoc   51.989877639     596.89551
## Age         17.266027223     219.75209
## Education    0.001116039      74.99521
## Urban       -1.412973353      12.67855
## US           2.646111366      16.03370
```

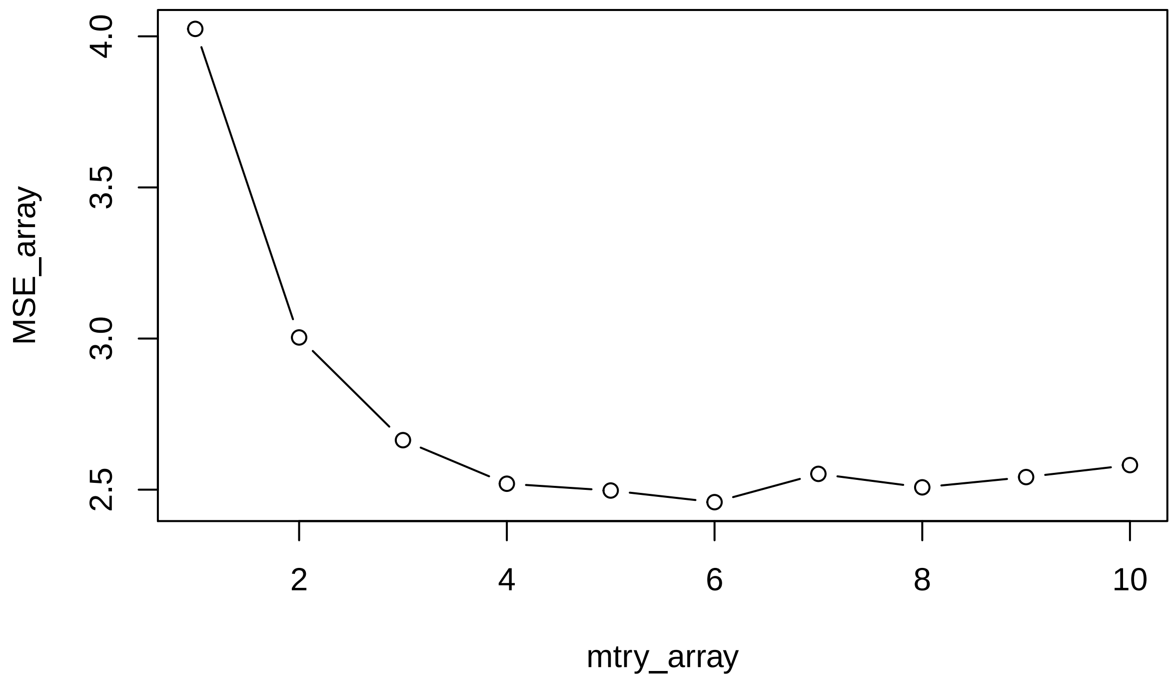
Still the ShelveLoc is the most important variable.

Now let's try an array of `mtrys`.

```
set.seed(1)
mtry_array = seq(1, 10)
MSE_array = numeric(length(mtry_array))
for (i in seq_along(mtry_array)) {
  reg.rf = randomForest(Sales ~ ., carseatsTrain, mtry = mtry_array[i])
```

```r
  sales_predicted = predict(reg.rf, newdata = carseatsTest[-1])
  MSE_array[i] = mean((carseatsTest$Sales - sales_predicted)^2)
}

plot(mtry_array, MSE_array, 'b')
```



mtry = 6 seems to yield the least MSE.

2.

(a) We clean and transform the data.

```r
hitters = read.csv('Hitters.txt')[-1] # get rid of name column
hitters = hitters[!is.na(hitters['Salary']), ]
hitters$Salary = log(hitters$Salary)
```

(b) We split the data.

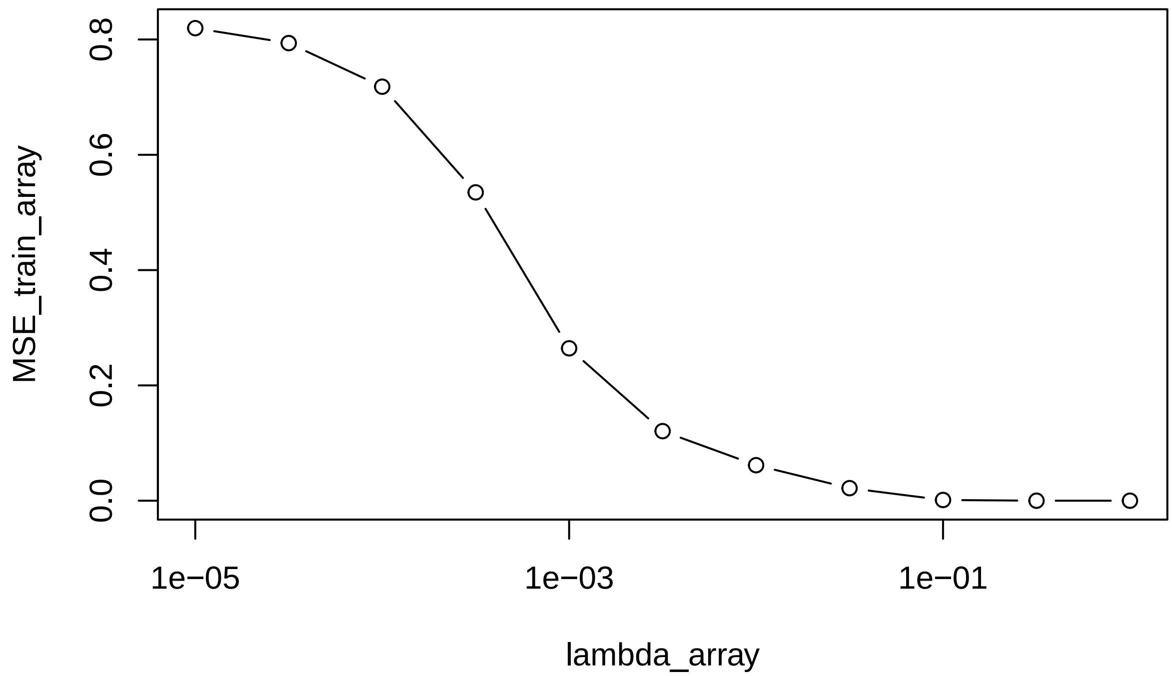
```r
hittersTrain = hitters[1:200, ]
hittersTest = hitters[200:dim(hitters)[1], ]
```

(c) Training set MSE:

```r
set.seed(1)
lambda_array = 10^(seq(-5, 0, 0.5))
MSE_train_array = numeric(length(lambda_array))
```

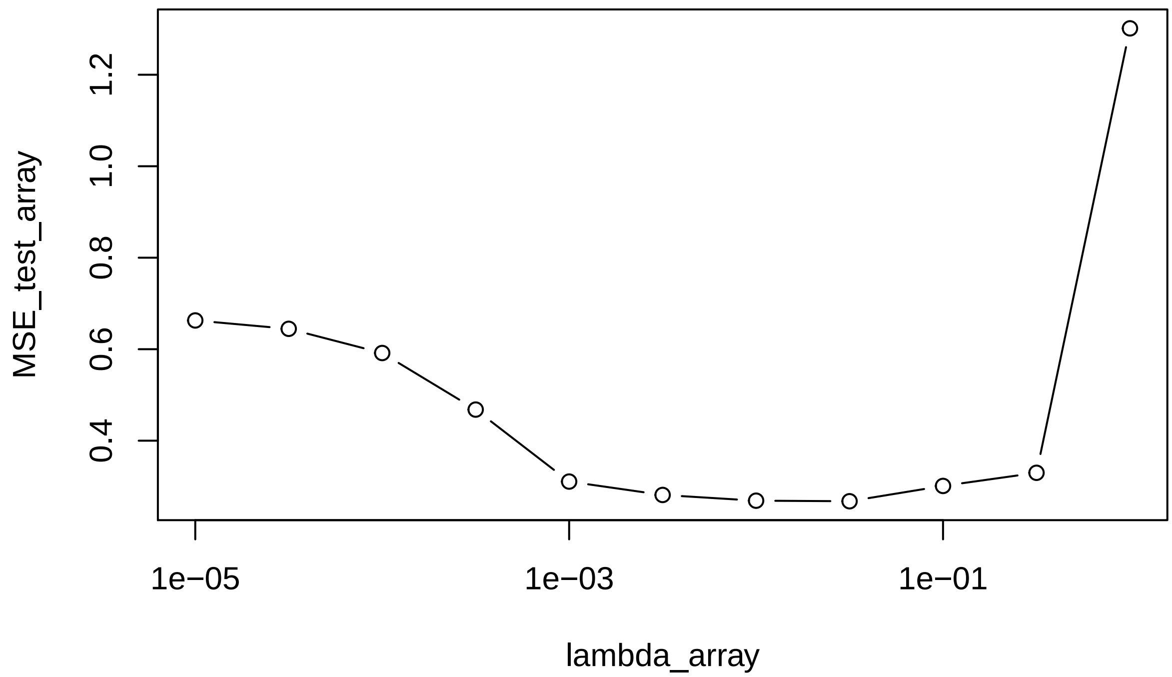
```r
MSE_test_array = numeric(length(lambda_array))
for (i in seq_along(lambda_array)) {
  boost = gbm(Salary ~ .,data = hittersTrain, distribution = "gaussian",
              n.trees = 1000, interaction.depth = 4, shrinkage = lambda_array[i])
  salary_predicted = predict(boost,
              newdata = hittersTest[!(names(hittersTest) == "Salary")], n.trees = 1000)
  MSE_train_array[i] = mean((hittersTrain$Salary - boost$fit)^2)
  MSE_test_array[i] = mean((hittersTest$Salary - salary_predicted)^2)
}

plot(lambda_array, MSE_train_array, 'b', log='x')
```



(d) Test set MSE:

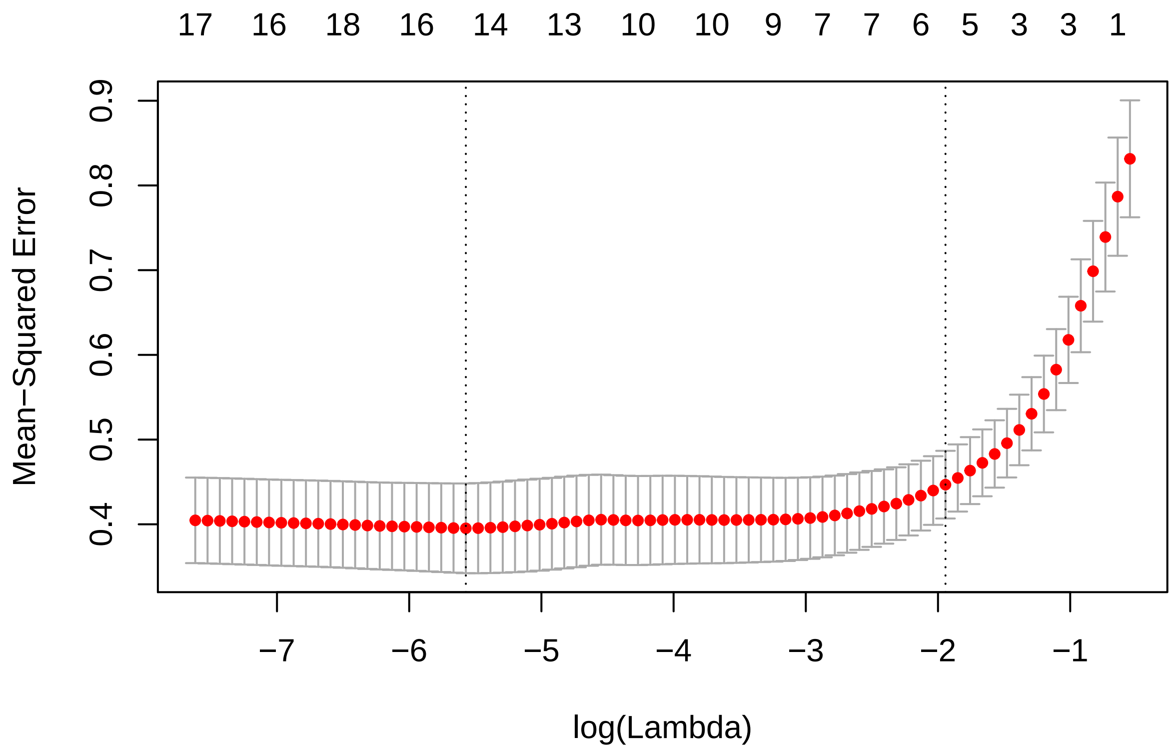
```r
plot(lambda_array, MSE_test_array, 'b', log='x')
```



The shrinkage value that yields the minimum MSE 2.4586619 is $\lambda = 0.0031623$.

(e) Take the Lasso with CV.

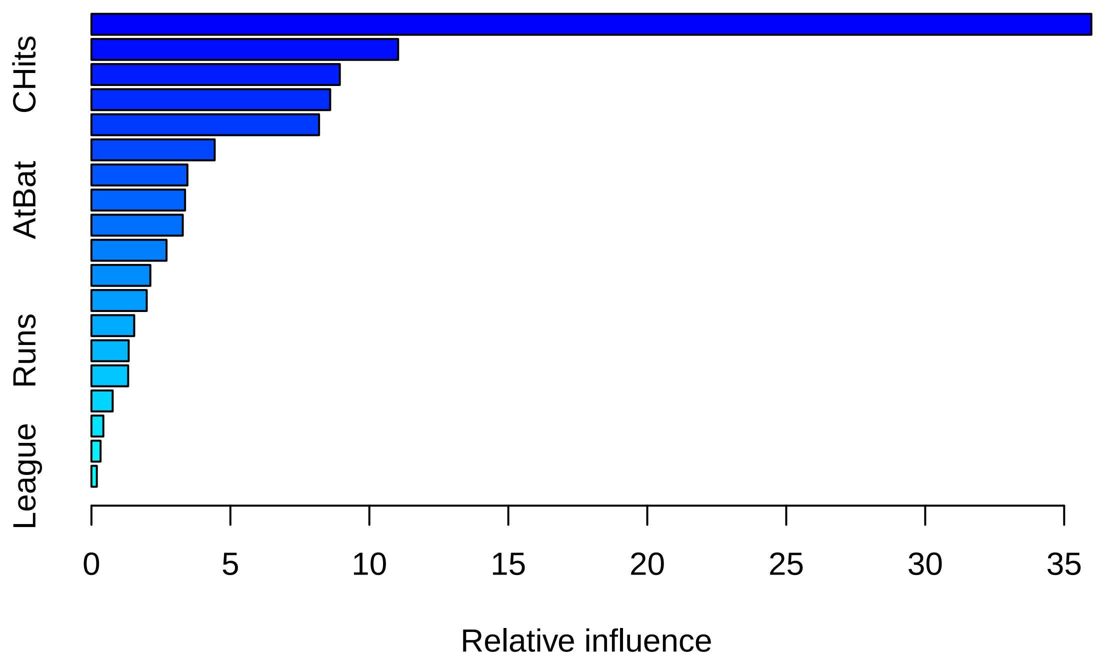
```
set.seed(1)
lasso.cv = cv.glmnet(model.matrix(Salary ~ ., hittersTrain)[, -1],
                     hittersTrain$Salary, alpha = 1)
salary_predicted = predict(lasso.cv,
                           newx = model.matrix(Salary ~ ., hittersTest)[, -1],
                           s = lasso.cv$lambda.min)
MSE = mean((hittersTest$Salary - salary_predicted)^2)
plot(lasso.cv)
```



The test set MSE is 0.4699547, bigger than that of boosting.

(f) Apply the boosting model with the optimal $\lambda = 0.0031623$.

```
set.seed(1)
boost = gbm(Salary ~ .,data = hittersTrain, distribution = "gaussian",
              n.trees = 1000, interaction.depth = 4, shrinkage = 0.0031623)
summary(boost)
```




```
##                 var    rel.inf
## CAtBat       CAtBat 35.9789134
## CWalks       CWalks 11.0357557
## CHits         CHits  8.9370523
## CRBI           CRBI  8.5903126
## CRuns         CRuns  8.1900969
## Years         Years  4.4344475
## Walks         Walks  3.4529951
## AtBat         AtBat  3.3687560
## CHmRun       CHmRun  3.2865178
## PutOuts     PutOuts  2.7017775
## Hits           Hits  2.1195298
## RBI             RBI  1.9891216
## Assists     Assists  1.5386959
## Runs           Runs  1.3404422
## Errors       Errors  1.3207729
## HmRun         HmRun  0.7638603
## Division   Division  0.4292169
## NewLeague NewLeague  0.3278123
## League       League  0.1939233
```

`CAtBat` is the most important predictor.

(g) Do bagging.

```
set.seed(1)
reg.bag = randomForest(Salary ~ ., hittersTrain, mtry = 19) # all variables
salary_predicted = predict(reg.bag,
```

```
                       newdata = hittersTest[!(names(hittersTest) == "Salary")])
MSE = mean((hittersTest$Salary - salary_predicted)^2)
```

The test set MSE is 0.2280919.