Final Question

Huafeng Zhang (217107707)

2019.7.26

1. load data, remove na, log-transform

```
library(ISLR)
hit <- Hitters #322 obs
hit <- na.omit(hit) #263 obs
hit$log.Salary <- log(hit$Salary)
#names(hit)
```

2. split to train/test

```
tr.hit <- hit[1:200,]
te.hit <- hit[201:263,]
```

3. Random Forest

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.6.1

## randomForest 4.6-14

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

```
set.seed(345)

#25 trees
rf25 <- randomForest(log.Salary ~ . - Salary, data = tr.hit, ntree = 25, Importance = T)
ps25 <- predict(rf25, newdata = te.hit)
mse25 <- mean((te.hit$log.Salary-ps25)^2)

#100 trees
#rf100 <- grow(rf25, 75)
rf100 <- randomForest(log.Salary ~ . - Salary, data = tr.hit, ntree = 100, Importance = T)
ps100 <- predict(rf100, newdata = te.hit)
mse100 <- mean((te.hit$log.Salary-ps100)^2)

#500 trees
#rf500 <- grow(rf100, 400)
rf500 <- randomForest(log.Salary ~ . - Salary, data = tr.hit, ntree = 500, Importance = T)
ps500 <- predict(rf500, newdata = te.hit)
```

```
mse500 <- mean((te.hit$log.Salary-ps500)^2)

#1000 trees
#rf1000 <- grow(rf500, 500)
rf1000 <- randomForest(log.Salary ~ . - Salary, data = tr.hit, ntree = 1000, Importance = T)
ps1000 <- predict(rf1000, newdata = te.hit)
mse1000 <- mean((te.hit$log.Salary-ps1000)^2)

print(paste('MSE of 25 trees is', mse25))
```

```
## [1] "MSE of 25 trees is 0.227011143697994"
```

```
print(paste('MSE of 100 trees is', mse100))
```

```
## [1] "MSE of 100 trees is 0.228650313687036"
```

```
print(paste('MSE of 500 trees is', mse500))
```

```
## [1] "MSE of 500 trees is 0.216076387586862"
```

```
print(paste('MSE of 1000 trees is', mse1000))
```

```
## [1] "MSE of 1000 trees is 0.217779438384326"
```

###Given the 4 results, using 500 trees gives the best result with the smallest MSE of 0.216.

4. identify the most important variables

```
print(rf500) #No. of variables tried at each split: 6
```

```
##
## Call:
##  randomForest(formula = log.Salary ~ . - Salary, data = tr.hit,      ntree = 500, Importance = T)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##           Mean of squared residuals: 0.2135731
##                     % Var explained: 74.33
```

```
importance(rf500, type=2)
```

```
##           IncNodePurity
## AtBat         6.0610657
## Hits          5.8677837
```

```
## HmRun          1.9309252
## Runs           4.5674917
## RBI            4.8661721
## Walks          5.0550154
## Years          7.2652886
## CAtBat        34.7331263
## CHits         24.5292756
## CHmRun         5.5484426
## CRuns         26.1100611
## CRBI          15.1346342
## CWalks        16.3724500
## League         0.1046373
## Division       0.1991141
## PutOuts        2.7902861
## Assists        1.4517977
## Errors         1.2341805
## NewLeague      0.2172757
```

The most important variables (with the largest values) associated with predicting Salary are 'CAtBat', 'CHits', 'CRuns', 'CWalks', and 'CRBI'.

5. Boosting and plot of learning rates vs. train MSEs

```r
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.6.1
```

```
## Loaded gbm 2.1.5
```

```r
set.seed(567)

#0.01 learning rate
b01 <- gbm(formula = log.Salary ~ . - Salary, data = tr.hit,
           distribution = "gaussian", n.trees = 1000,
           interaction.depth = 1, shrinkage = 0.01, verbose = F)
trps01 <- predict(b01, n.trees = b01$n.trees, newdata = tr.hit)
trmse01 <- mean((tr.hit$log.Salary-trps01)^2)

#0.001 learning rate
b001 <- gbm(log.Salary ~ . - Salary, data = tr.hit,
           distribution = "gaussian", n.trees = 1000,
           interaction.depth = 1, shrinkage = 0.001, verbose = F)
trps001 <- predict(b001, n.trees = b001$n.trees, newdata = tr.hit)
trmse001 <- mean((tr.hit$log.Salary-trps001)^2)

#0.1 learning rate
b1 <- gbm(log.Salary ~ . - Salary, data = tr.hit,
           distribution = "gaussian", n.trees = 1000,
           interaction.depth = 1, shrinkage = 0.1, verbose = F)
trps1 <- predict(b1, n.trees = b1$n.trees, newdata = tr.hit)
```

```r
trmse1 <- mean((tr.hit$log.Salary-trps1)^2)

#0.2 learning rate
b2 <- gbm(log.Salary ~ . - Salary, data = tr.hit,
          distribution = "gaussian", n.trees = 1000,
          interaction.depth = 1, shrinkage = 0.2, verbose = F)
trps2 <- predict(b2, n.trees = b2$n.trees, newdata = tr.hit)
trmse2 <- mean((tr.hit$log.Salary-trps2)^2)

print(paste('train MSE of 0.01 learning rate at d=1 is', trmse01))
```

```
## [1] "train MSE of 0.01 learning rate at d=1 is 0.138358148208026"
```

```r
print(paste('train MSE of 0.001 learning rate at d=1 is', trmse001))
```

```
## [1] "train MSE of 0.001 learning rate at d=1 is 0.361502516103347"
```

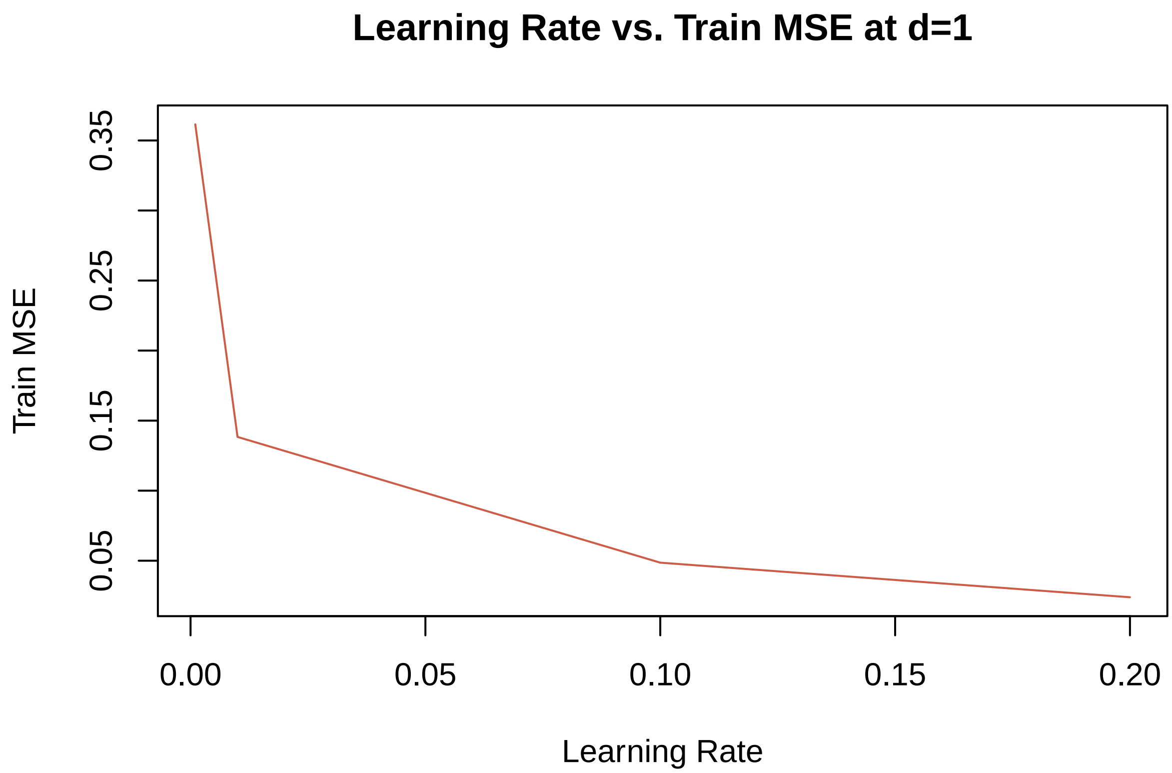
```r
print(paste('train MSE of 0.1 learning rate at d=1 is', trmse1))
```

```
## [1] "train MSE of 0.1 learning rate at d=1 is 0.0486036557696533"
```

```r
print(paste('train MSE of 0.2 learning rate at d=1 is', trmse2))
```

```
## [1] "train MSE of 0.2 learning rate at d=1 is 0.0239267402963262"
```

```r
#plot (train)
plot(c(0.001,0.01,0.1,0.2),
     c(trmse001,trmse01,trmse1,trmse2),
     type = 'l', col = 'coral3',
     main = 'Learning Rate vs. Train MSE at d=1',
     xlab = 'Learning Rate',
     ylab = 'Train MSE')
```



6. plot of learning rates vs. test MSEs

```
#interaction.depth = 1
library(gbm)
set.seed(567)

#0.01 learning rate
ps01 <- predict(b01, n.trees = b01$n.trees, newdata = te.hit)
mse01 <- mean((te.hit$log.Salary-ps01)^2)

#0.001 learning rate
ps001 <- predict(b001, n.trees = b001$n.trees, newdata = te.hit)
mse001 <- mean((te.hit$log.Salary-ps001)^2)

#0.1 learning rate
ps1 <- predict(b1, n.trees = b1$n.trees, newdata = te.hit)
mse1 <- mean((te.hit$log.Salary-ps1)^2)

#0.2 learning rate
ps2 <- predict(b2, n.trees = b2$n.trees, newdata = te.hit)
mse2 <- mean((te.hit$log.Salary-ps2)^2)

print(paste('MSE of 0.01 learning rate at d=1 is', mse01))
```

```
## [1] "MSE of 0.01 learning rate at d=1 is 0.281627516952774"
```

```r
print(paste('MSE of 0.001 learning rate at d=1 is', mse001))
```

```
## [1] "MSE of 0.001 learning rate at d=1 is 0.337744738172609"
```

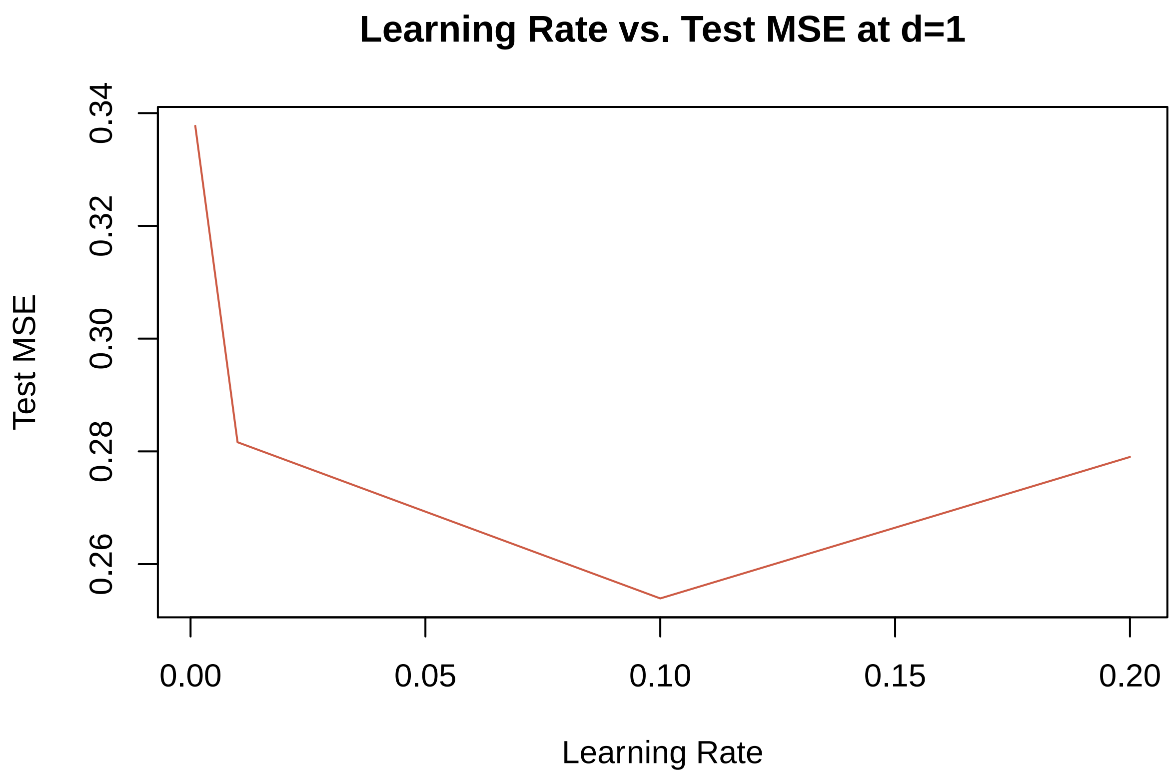
```r
print(paste('MSE of 0.1 learning rate at d=1 is', mse1))
```

```
## [1] "MSE of 0.1 learning rate at d=1 is 0.253919209796006"
```

```r
print(paste('MSE of 0.2 learning rate at d=1 is', mse2))
```

```
## [1] "MSE of 0.2 learning rate at d=1 is 0.279011819536214"
```

```r
#plot (test)
plot(c(0.001,0.01,0.1,0.2),
     c(mse001,mse01,mse1,mse2),
     type = 'l', col = 'coral3',
     main = 'Learning Rate vs. Test MSE at d=1',
     xlab = 'Learning Rate',
     ylab = 'Test MSE')
```



7. try different interaction depths

```
#interaction.depth = 2
set.seed(666)

#0.01 learning rate
b01 <- gbm(formula = log.Salary ~ . - Salary, data = tr.hit,
           distribution = "gaussian", n.trees = 1000,
           interaction.depth = 2, shrinkage = 0.01, verbose = F)
ps01 <- predict(b01, n.trees = b01$n.trees, newdata = te.hit)
mse01 <- mean((te.hit$log.Salary-ps01)^2)

#0.001 learning rate
b001 <- gbm(log.Salary ~ . - Salary, data = tr.hit,
           distribution = "gaussian", n.trees = 1000,
           interaction.depth = 2, shrinkage = 0.001, verbose = F)
ps001 <- predict(b001, n.trees = b001$n.trees, newdata = te.hit)
mse001 <- mean((te.hit$log.Salary-ps001)^2)

#0.1 learning rate
b1 <- gbm(log.Salary ~ . - Salary, data = tr.hit,
           distribution = "gaussian", n.trees = 1000,
           interaction.depth = 2, shrinkage = 0.1, verbose = F)
ps1 <- predict(b1, n.trees = b1$n.trees, newdata = te.hit)
mse1 <- mean((te.hit$log.Salary-ps1)^2)

#0.2 learning rate
b2 <- gbm(log.Salary ~ . - Salary, data = tr.hit,
           distribution = "gaussian", n.trees = 1000,
           interaction.depth = 2, shrinkage = 0.2, verbose = F)
ps2 <- predict(b2, n.trees = b2$n.trees, newdata = te.hit)
mse2 <- mean((te.hit$log.Salary-ps2)^2)

print(paste('MSE of 0.01 learning rate at d=2 is', mse01))
```

```
## [1] "MSE of 0.01 learning rate at d=2 is 0.276960632061591"
```

```
print(paste('MSE of 0.001 learning rate at d=2 is', mse001))
```

```
## [1] "MSE of 0.001 learning rate at d=2 is 0.314604324482154"
```

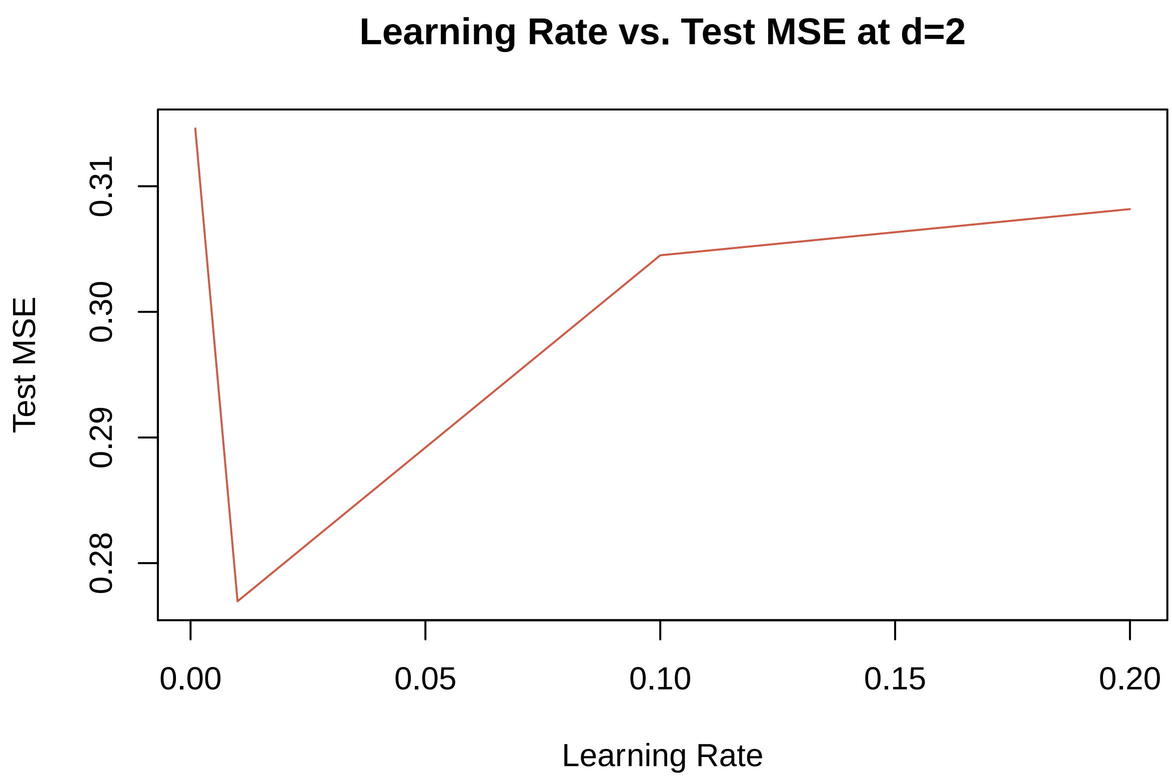
```
print(paste('MSE of 0.1 learning rate at d=2 is', mse1))
```

```
## [1] "MSE of 0.1 learning rate at d=2 is 0.304501837582825"
```

```
print(paste('MSE of 0.2 learning rate at d=2 is', mse2))
```

```
## [1] "MSE of 0.2 learning rate at d=2 is 0.308179134663943"
```

```r
#plot (test)
plot(c(0.001,0.01,0.1,0.2),
     c(mse001,mse01,mse1,mse2),
     type = 'l', col = 'coral3',
     main = 'Learning Rate vs. Test MSE at d=2',
     xlab = 'Learning Rate',
     ylab = 'Test MSE')
```




```r
#interaction.depth = 4
set.seed(233)

#0.01 learning rate
b01 <- gbm(formula = log.Salary ~ . - Salary, data = tr.hit,
           distribution = "gaussian", n.trees = 1000,
           interaction.depth = 4, shrinkage = 0.01, verbose = F)
ps01 <- predict(b01, n.trees = b01$n.trees, newdata = te.hit)
mse01 <- mean((te.hit$log.Salary-ps01)^2)

#0.001 learning rate
b001 <- gbm(log.Salary ~ . - Salary, data = tr.hit,
           distribution = "gaussian", n.trees = 1000,
           interaction.depth = 4, shrinkage = 0.001, verbose = F)
ps001 <- predict(b001, n.trees = b001$n.trees, newdata = te.hit)
mse001 <- mean((te.hit$log.Salary-ps001)^2)
```

```r
#0.1 learning rate
b1 <- gbm(log.Salary ~ . - Salary, data = tr.hit,
          distribution = "gaussian", n.trees = 1000,
          interaction.depth = 4, shrinkage = 0.1, verbose = F)
ps1 <- predict(b1, n.trees = b1$n.trees, newdata = te.hit)
mse1 <- mean((te.hit$log.Salary-ps1)^2)

#0.2 learning rate
b2 <- gbm(log.Salary ~ . - Salary, data = tr.hit,
          distribution = "gaussian", n.trees = 1000,
          interaction.depth = 4, shrinkage = 0.2, verbose = F)
ps2 <- predict(b2, n.trees = b2$n.trees, newdata = te.hit)
mse2 <- mean((te.hit$log.Salary-ps2)^2)

print(paste('MSE of 0.01 learning rate at d=4 is', mse01))
```

```
## [1] "MSE of 0.01 learning rate at d=4 is 0.280471018407855"
```

```r
print(paste('MSE of 0.001 learning rate at d=4 is', mse001))
```

```
## [1] "MSE of 0.001 learning rate at d=4 is 0.307862913776067"
```

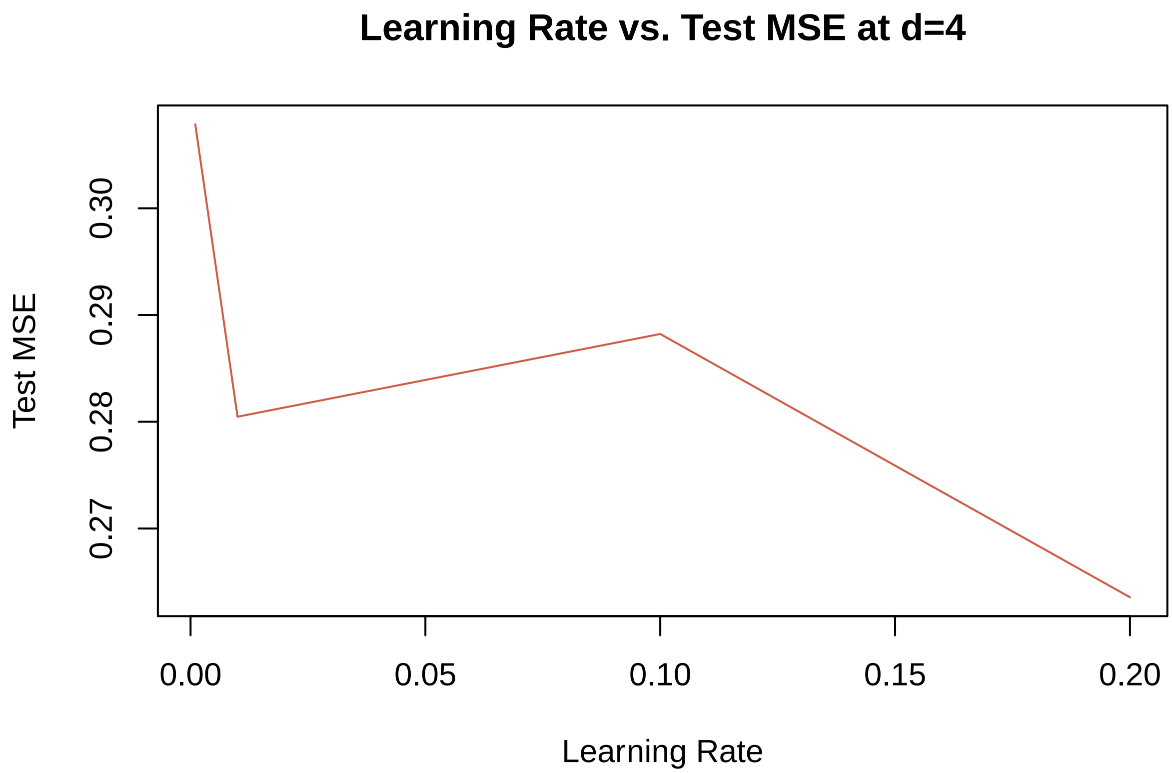
```r
print(paste('MSE of 0.1 learning rate at d=4 is', mse1))
```

```
## [1] "MSE of 0.1 learning rate at d=4 is 0.288221061450338"
```

```r
print(paste('MSE of 0.2 learning rate at d=4 is', mse2))
```

```
## [1] "MSE of 0.2 learning rate at d=4 is 0.263558950473661"
```

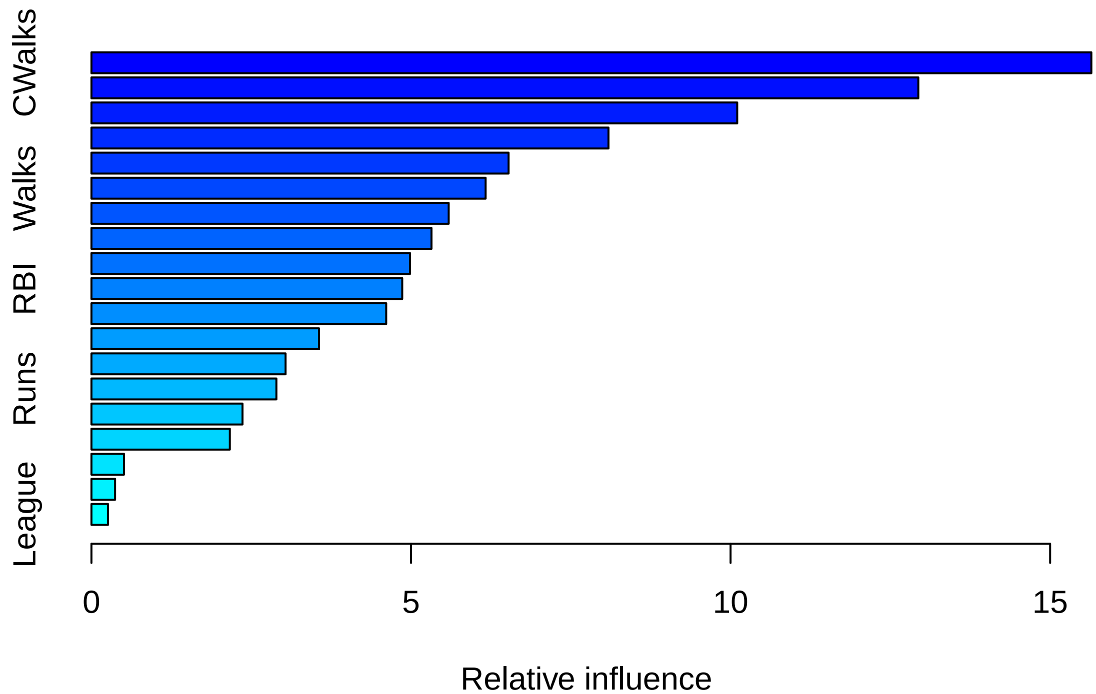
```r
#plot (test)
plot(c(0.001,0.01,0.1,0.2),
     c(mse001,mse01,mse1,mse2),
     type = 'l', col = 'coral3',
     main = 'Learning Rate vs. Test MSE at d=4',
     xlab = 'Learning Rate',
     ylab = 'Test MSE')
```



For interaction depth of 1, $\lambda = 0.1$ gives the smallest test MSE of 0.253919209796006.
For interaction depth of 2, $\lambda = 0.01$ gives the smallest test MSE of 0.276960632061591.
For interaction depth of 4, $\lambda = 0.2$ gives the smallest test MSE of 0.263558950473661.

8. most important variables from boosted tree

```
set.seed(567) #the same seed as above for d=1
b1 <- gbm(log.Salary ~ . - Salary, data = tr.hit,
           distribution = "gaussian", n.trees = 1000,
           interaction.depth = 1, shrinkage = 0.1, verbose = F)
summary(b1)
```




```
##                 var    rel.inf
## CWalks       CWalks 15.6449722
## CAtBat       CAtBat 12.9375051
## CRBI           CRBI 10.1040118
## PutOuts     PutOuts  8.0896231
## CRuns         CRuns  6.5267876
## Walks         Walks  6.1674450
## CHmRun       CHmRun  5.5898294
## Years         Years  5.3205366
## Hits           Hits  4.9848118
## RBI             RBI  4.8630142
## Assists     Assists  4.6119239
## HmRun         HmRun  3.5604525
## AtBat         AtBat  3.0374210
## Runs           Runs  2.8937549
## Errors       Errors  2.3637443
## CHits         CHits  2.1652687
## Division   Division  0.5091896
## NewLeague NewLeague  0.3701619
## League       League  0.2595463
```

Given the result, 'CWalks', 'CAtBat', and 'CRBI' are the most important variables.

9. compare test MSEs from RF and boosting

In random forest, using 500 trees gives the best result with the smallest MSE of 0.216, which is better than any of the test MSE from boosting.