BUAN 6356 - Homework 4

Group No.9 (Shubhi Kala, Spoorthi Thatipally, Hao-Yu Lin, Loc Nguyen, Tatsat Joshi)

4/15/2020

Contents

Predict salaries of the players

```
if(!require("pacman")) install.packages("pacman")
pacman::p_load(rpart, caret, leaps, ISLR, data.table, tree, gbm, magrittr,
               dplyr, randomForest, moments)
search()
theme_set(theme_classic())
```

Solution to Q1

1. Remove the observation with unknown salary information

```
#Using na.omit for removing missing salary
Hitters.clean = na.omit(Hitters, cols=Salary)

#Total number of observations in Hitters dataset
dim(Hitters)
```

```
## [1] 322  20
```

```
#Total number of observations after removing missing values from Salary
dim(Hitters.clean)
```

```
## [1] 263  20
```

```
# Number of unknown observations removed
sum(is.na(Hitters$Salary))
```

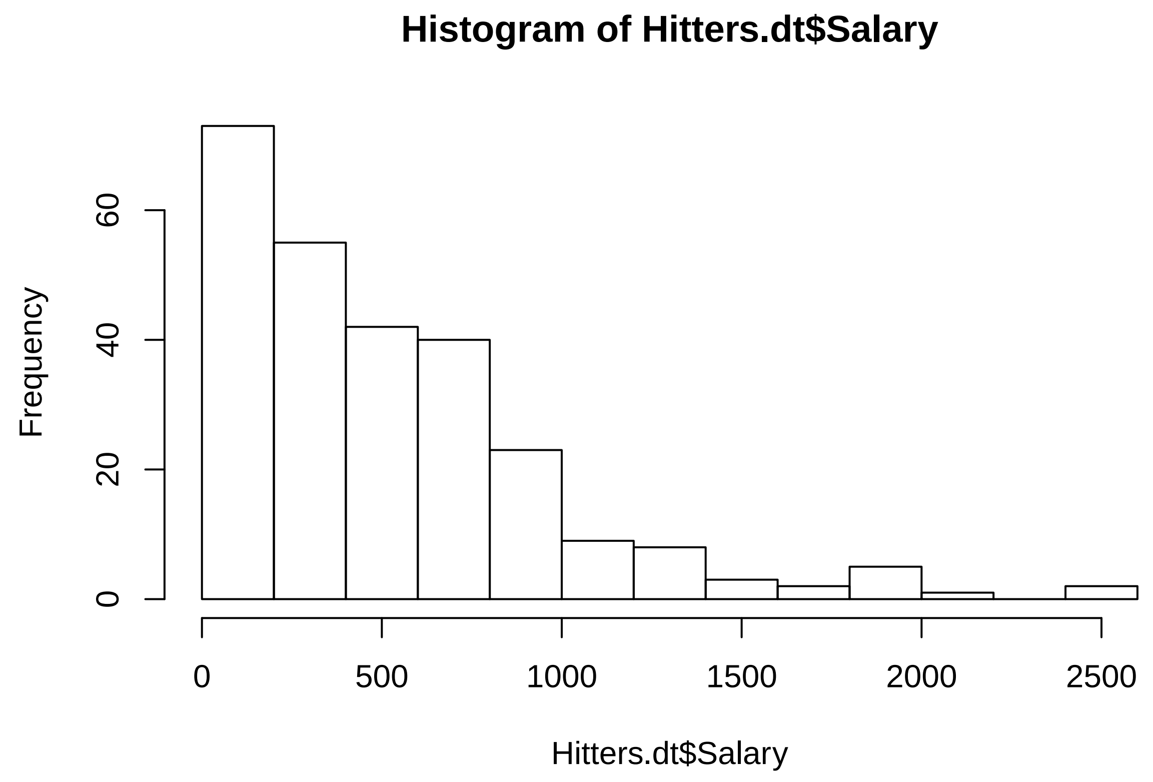
```
## [1] 59
```

- Out of the total 322 observations, there are 59 observations that are having unknown values in Salary variable, Hence 59 observations are removed from the dataset.
- na.omit() method removes the missing values from the dataset and returns the clean dataset.

Solution to Q2

2. Transform the salaries using a (natural) log transformation

```
Hitters.dt <- data.frame(Hitters.clean)

# Data Transformation using natural log
Hitters.dt$logSalary = log(Hitters.dt$Salary)

# Plotting histogram to show right skewness before transformation
hist(Hitters.dt$Salary) #positively skewed data #right skewed
```

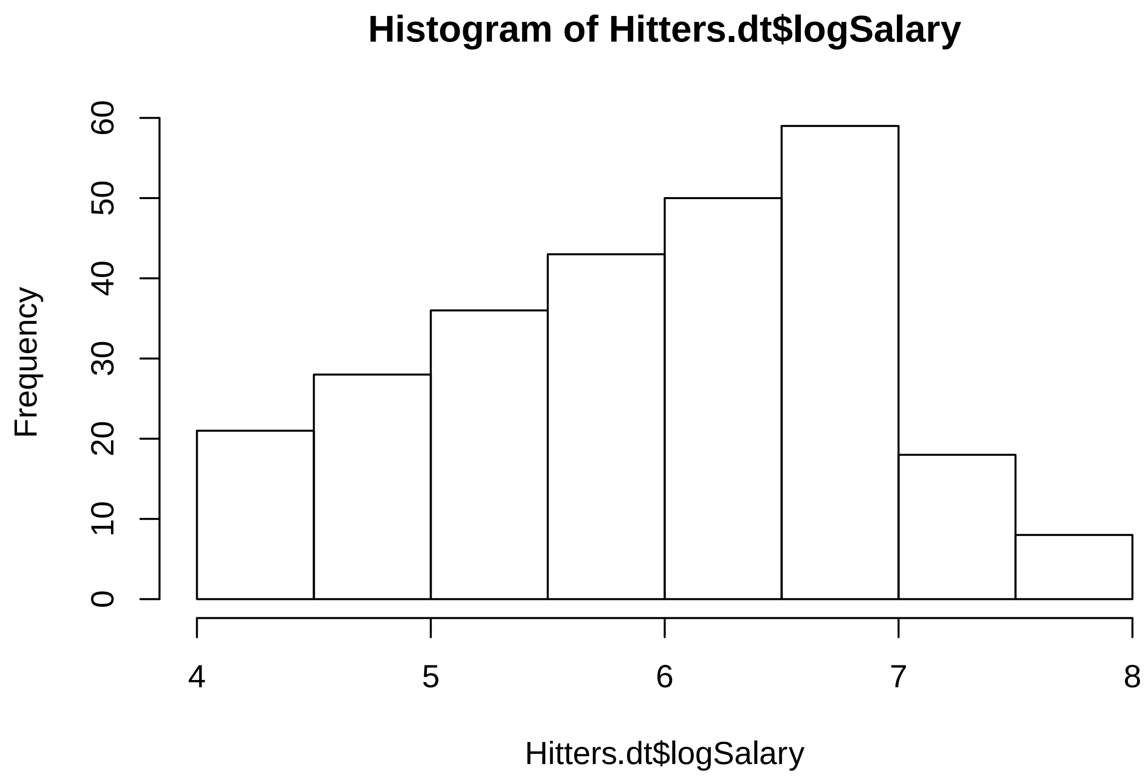


```r
skewness(Hitters.dt$Salary)
```

```
## [1] 1.57989
```

- Histogram showing the rightly skewed salary variable before log transformation.
- Skewness before log transformation is 1.58.
- If the skewness is less than -1(negatively skewed) or greater than 1(positively skewed), the data are highly skewed.

```r
# Plotting the histogram to show reduction in the skeweness
hist(Hitters.dt$logSalary)#less skewed, more bell curve normal distribution
```



```r
skewness(Hitters.dt$logSalary)
```

```
## [1] -0.1809668
```

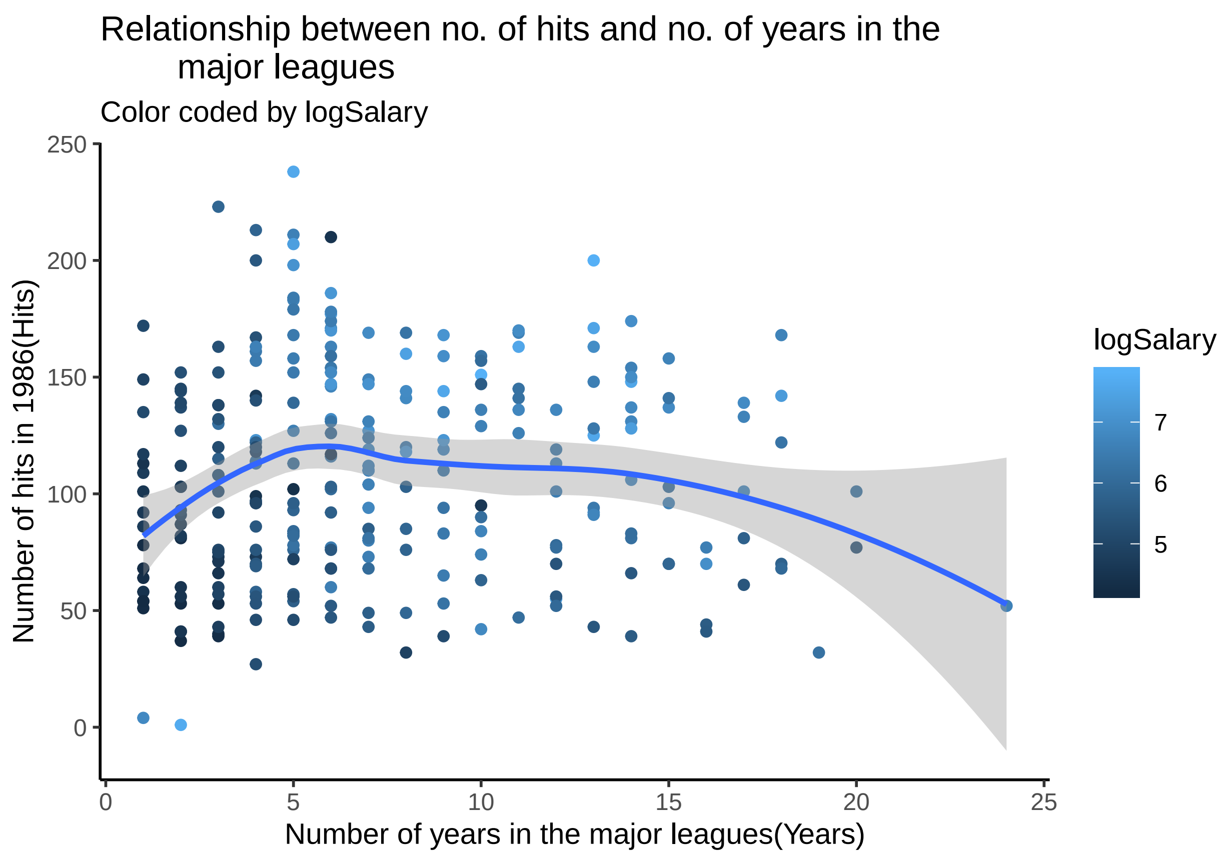
- Histogram showing slightly left skewed, more bell curve following normal distribution of the logSalary variable after log transformation.
- Skewness after log transformation is -0.18.
- If the skewness is between -0.5 and 0.5, the data are fairly symmetrical.
- Data is transformed using natural logaritm. They are handy for reducing the skewness in data.
- Improves the performance by transforming highly skewed predictor like here highly right-skewed variables (such as salary) is transformed by taking a log transform.

Solution to Q3

3. Scatterplot with Hits on the y-axis and Years on the x-axis and color coded using logSalary

```r
# Using ggplot for plotting the scatter plot
ggplot(data=Hitters.dt, mapping=aes(x=Years, y=Hits) )+
  geom_point(aes(color = logSalary)) +
  labs(
    x = "Number of years in the major leagues(Years)",
    y = "Number of hits in 1986(Hits)",
    color = "logSalary",
    title = "Relationship between no. of hits and no. of years in the
        major leagues",
    subtitle = "Color coded by logSalary"
  ) +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



- Scatterplot of Number of Hits in the year 1986 against Numner of years in the major leagues. A smooth curve shows that when the player plays for large number of years in the major leagues the number of hits played by him goes on decreasing.
- In a same range of year, higher Hits tends to have higher salary, and within same range of hit, higher year tend to have higher salary.
- In short, both Years and Hits have positive effect on salary and Hits tend to have stronger effect.
- The plot also shows that in the initial years like 1 or 2 the players earn less but their salary goes on increasing with the experience.

Solution to Q4

4. Run a linear regression model of Log Salary on all the predictors

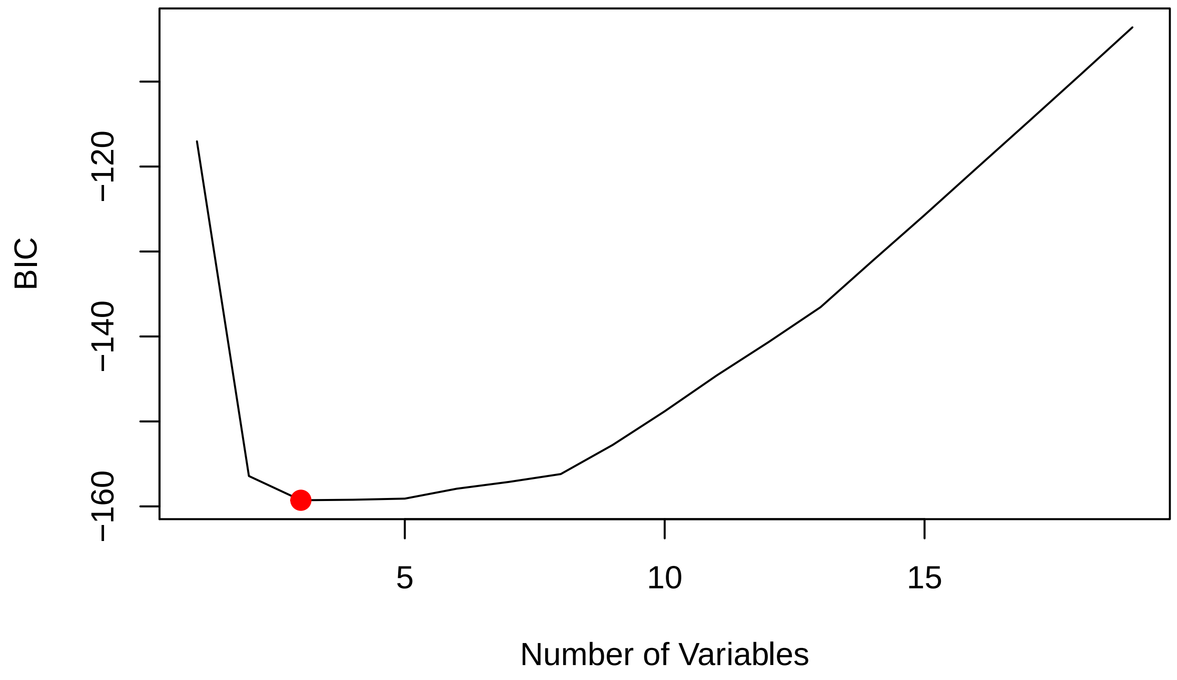
```
Hitter.new <-  Hitters.dt[,-19]

search <- regsubsets(logSalary ~ ., data = Hitter.new, nbest = 1,
                     nvmax = dim(Hitter.new)[2], method = "exhaustive")

sum <- summary(search)

# show bic values
sum$bic
```

```
##  [1] -117.0304 -156.4291 -159.2777 -159.2182 -159.0885 -157.9207 -157.1229
##  [8] -156.1954 -152.7649 -148.8061 -144.5962 -140.6541 -136.5480 -131.0939
## [15] -125.7112 -120.1995 -114.7125 -109.1859 -103.6145
```

```
# looking for model with the smallest statistic
plot(sum$bic, xlab = "Number of Variables", ylab = "BIC", type = "l")
bic_min = which.min(sum$bic)
bic_min
```

```
## [1] 3
```

```
points(bic_min, sum$bic[bic_min], col = "red", cex = 2, pch = 20)
```



```r
data.frame(
  Adj.R2 = which.max(sum$adjr2),
  CP = which.min(sum$cp),
  BIC = which.min(sum$bic)
)
```

```
##   Adj.R2 CP BIC
## 1     13  9   3
```

```r
coef(search, bic_min)
```

```
## (Intercept)        Hits       Walks       Years
## 4.231360675 0.006811902 0.006582034 0.094467566
```

- According to BIC values, -159.277 is the smallest, hence the best performer is the model with 3 variables - Hits, Walks and Years.
- The regsubsets() function has a built-in plot() command which can be used to display the selected variables for the best model with a given number of predictors, ranked according to a chosen statistic.

Solution to Q5

5. Create a training data set consisting of 80 percent of the observations, and a test data set consisting of the remaining observations.

```r
set.seed(42)
train.index <- sample(c(1:dim(Hitter.new)[1]), dim(Hitter.new)[1]*0.8)
train.df <- Hitter.new[train.index, ]
test.df <- Hitter.new[-train.index, ]

dim(train.df)
```

```
## [1] 210  20
```

```r
dim(test.df)
```

```
## [1] 53 20
```

- Training dataset contains 80% of the total observations = 210
- Test dataset contains remaining 20% of the total observations = 53

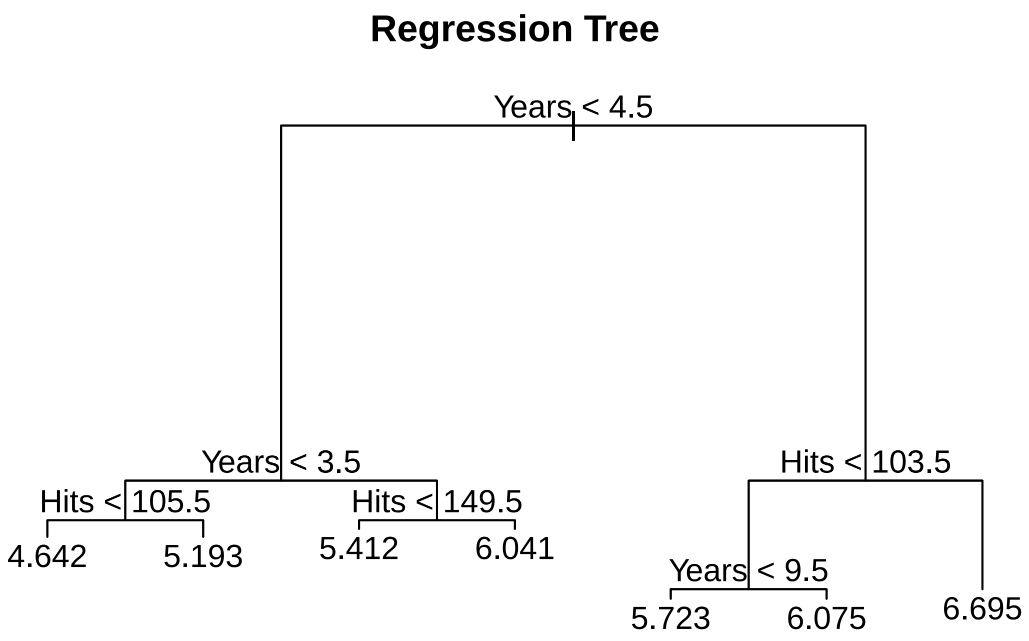
Solution to Q6

6. Regression tree of log Salary using only Years and Hits variables from the training data set

```r
tree.hitters <- tree(logSalary ~ Years + Hits, data = train.df)
summary(tree.hitters)
```

```
##
## Regression tree:
## tree(formula = logSalary ~ Years + Hits, data = train.df)
## Number of terminal nodes:  7
## Residual mean deviance:  0.2436 = 49.45 / 203
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.19500 -0.29810 -0.03641  0.00000  0.22790  2.18200
```

```r
plot(tree.hitters)
text(tree.hitters, pretty = 0)
title("Regression Tree")
```

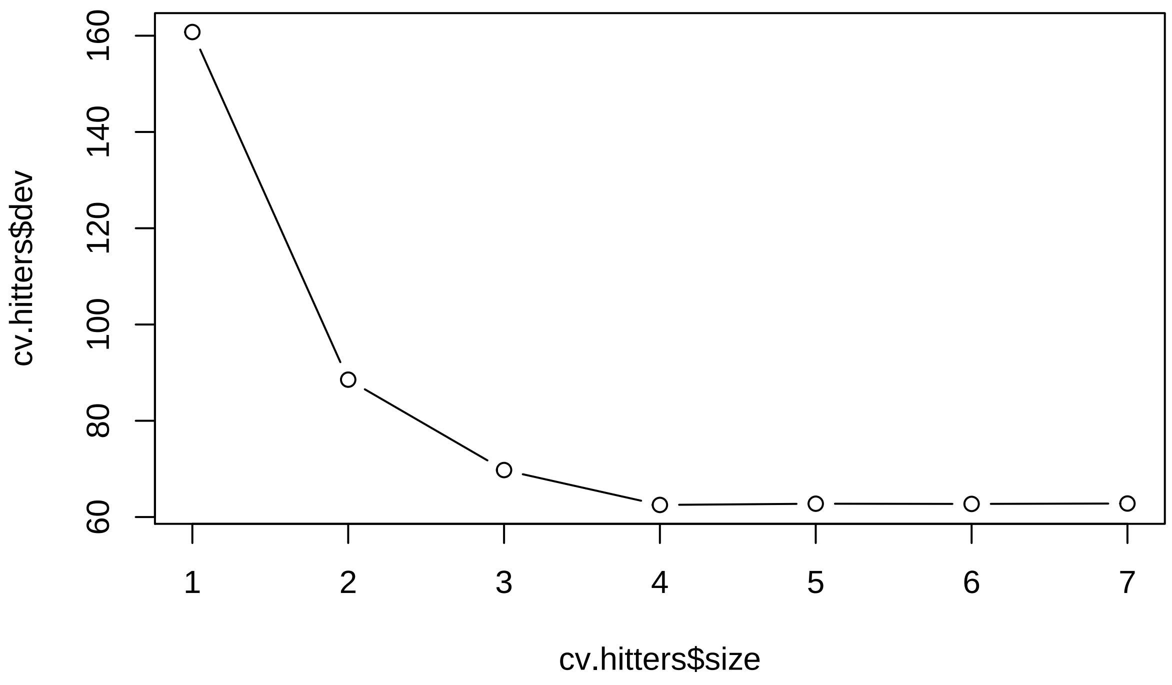


```r
cv.hitters <- cv.tree(tree.hitters)
cv.hitters
```

```
## $size
## [1] 7 6 5 4 3 2 1
##
## $dev
## [1]  62.80880  62.72547  62.76649  62.51527  69.76195  88.54011 160.76154
##
## $k
## [1]      -Inf  1.753076  1.953384  3.373620  8.071311 22.103227 72.312399
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

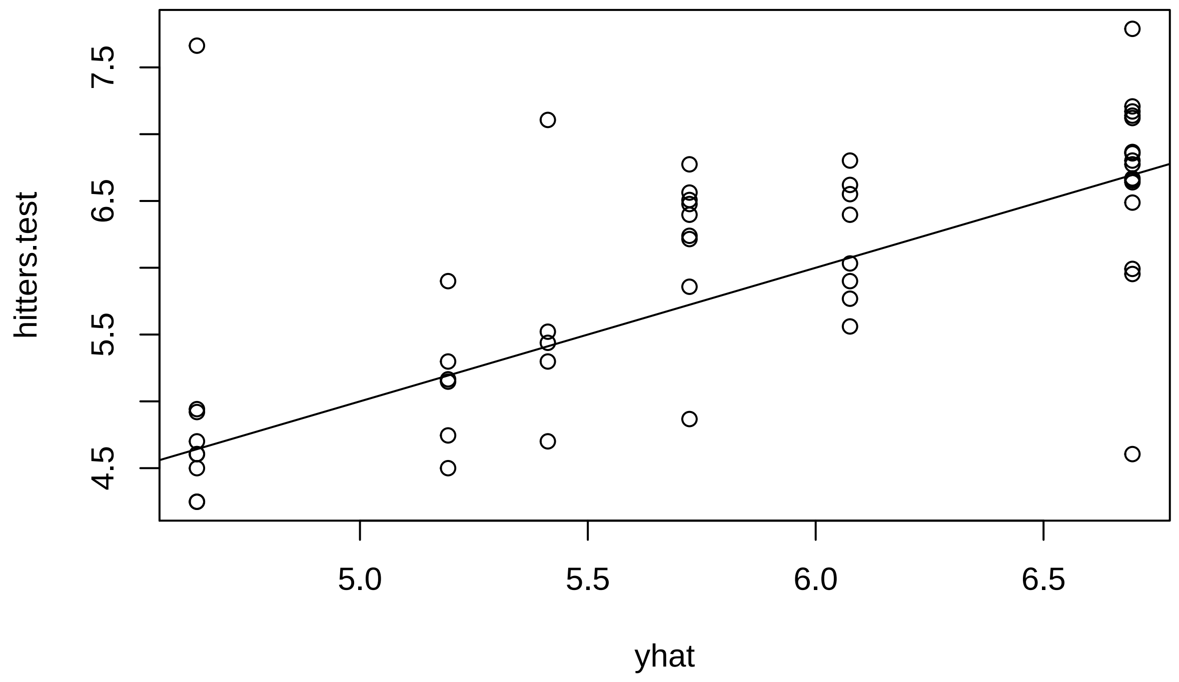
- we use the cross-validation function 'cv.tree()' to determine the optimal level of tree complexity, i.e., the best tree size; cost complexity pruning is used in order to select a sequence of trees for consideration. The 'cv.tree()' function reports the number of terminal nodes of each tree considered (size) as well as the corresponding error rate (dev) and the value of the cost-complexity parameter used.

```
plot(cv.hitters$size,cv.hitters$dev,type = 'b')
```



* 'dev' corresponds to the cross-validation error rate in this instance. This is the plot of error rate as a function of the size.

```
yhat <- predict(tree.hitters, newdata = test.df)

hitters.test <- test.df$logSalary

plot(yhat,hitters.test)
abline(0,1)
```



```r
mean((yhat-hitters.test)^2) # prediction - actual -> mean squared error
```

```
## [1] 0.5297053
```

```r
# Get the players who are likely to receive highest salaries according to this model
for(i in 1:nrow(Hitters.clean)){
  if((Hitters.clean$Years[i] <= 4.5) && (Hitters.clean$Hits[i]<= 103.5)){
    print(row.names(Hitters.clean)[i])
  }
}
```

```
## [1] "-Andres Galarraga"
## [1] "-Al Newman"
## [1] "-Argenis Salazar"
## [1] "-Andres Thomas"
## [1] "-Barry Bonds"
## [1] "-Bob Melvin"
## [1] "-BillyJo Robidoux"
## [1] "-Bill Schroeder"
## [1] "-Curt Ford"
## [1] "-Carmelo Martinez"
## [1] "-Curt Wilkerson"
## [1] "-Dave Anderson"
## [1] "-Daryl Boston"
## [1] "-Dan Gladden"
## [1] "-Donnie Hill"
## [1] "-Dan Pasqua"
## [1] "-Dale Sveum"
## [1] "-Glenn Braggs"
## [1] "-Harold Reynolds"
## [1] "-Herm Winningham"
## [1] "-John Cangelosi"
## [1] "-Jack Howell"
## [1] "-John Kruk"
## [1] "-Jeff Reed"
## [1] "-John Russell"
## [1] "-Joel Skinner"
## [1] "-Jose Uribe"
## [1] "-Kal Daniels"
## [1] "-Kevin Mitchell"
## [1] "-Kurt Stillwell"
## [1] "-Larry Sheets"
## [1] "-Mike Aldrete"
## [1] "-Mike Diaz"
## [1] "-Mariano Duncan"
## [1] "-Mike Kingery"
## [1] "-Mike LaValliere"
## [1] "-Mark Salas"
## [1] "-Mike Schmidt"
## [1] "-Mickey Tettleton"
## [1] "-Milt Thompson"
## [1] "-Marvell Wynne"
## [1] "-Rey Quinones"
## [1] "-Ronn Reynolds"
## [1] "-Rafael Santana"
## [1] "-Rick Schu"
## [1] "-Ruben Sierra"
## [1] "-Scott Bradley"
## [1] "-Steve Jeltz"
## [1] "-Steve Lombardozzi"
## [1] "-Tom Foley"
## [1] "-Terry Kennedy"
## [1] "-Tim Teufel"
```

- For a regression tree, the predicted response for an observation is given by the mean response of the training observations that belong to the same terminal node.
- In order to build a regression tree, you first use recursive binary splititng to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
- At a given internal node, the label of the form a < b (Years< 4.5) indicates the left-hand branch issued from that split, and the right-hand branch corresponds to a >= b (Years > 4.5). The above tree has 8 terminal nodes. Based on above tree plot, 'Years' is the most important factor in determining 'Salary', and players with less experience earn lower salaries. To predict the 'log(Salary)' of a new payer, we only need to check which region this new player belongs to. For instance, a new player with 'Years' < 3.5 and 'Hits' < 40.5 will have a predicted 'log(Salary)' 5.511.
- According to the Regression tree, the Rule which leads to highest salary is Years >= 4.5 AND Hits >= 103.5.

- If we want to predict the log salary of a new player who had been in the league for 5 seasons and had 120 hits the previous season, we would start at the top of the tree and compare 5 years to 4.5 years. Seeing that 5 is not less than 4.5, we choose the right branch. Reaching the next split, we see that 110 hits is greater than 103.5 we choose the right branch at this split, arriving at our log salary estimate of 6.7 for our new player (which corresponds to a salary point estimate of $812,405.80).
- Players who have more experience and more hits tend to make more than players who have less experience and less hits.
- The details of players who are likely to get the highest salaries is listed above.
- Here, the MSE of the model is 0.53.

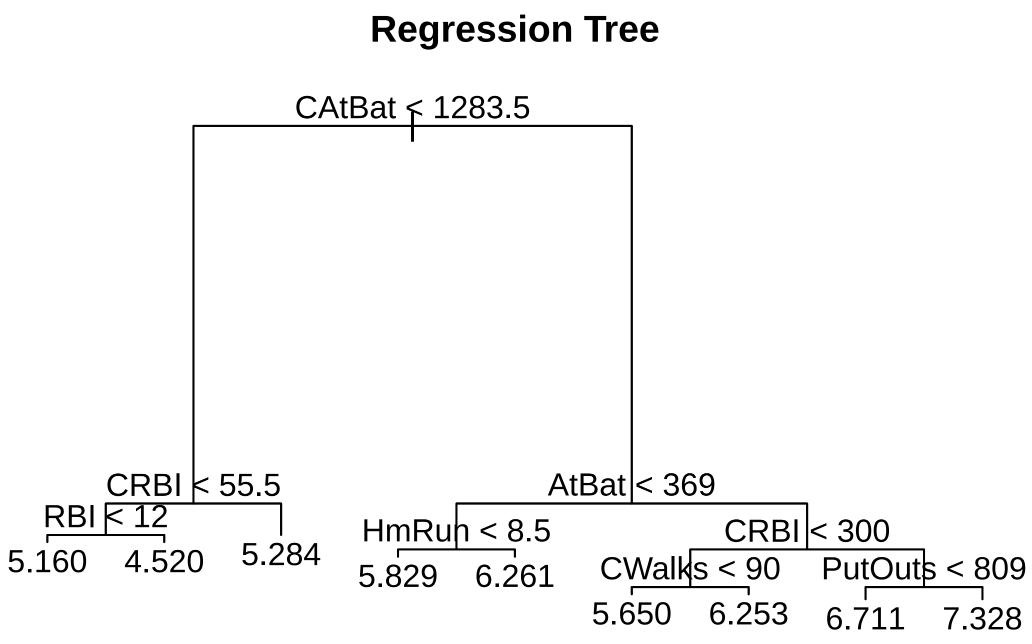
Solution to Q7

7. Create regression tree using all variables in the dataset and perform boosting.

```
tree.hitters <- tree(logSalary ~ ., data = train.df)
summary(tree.hitters)
```

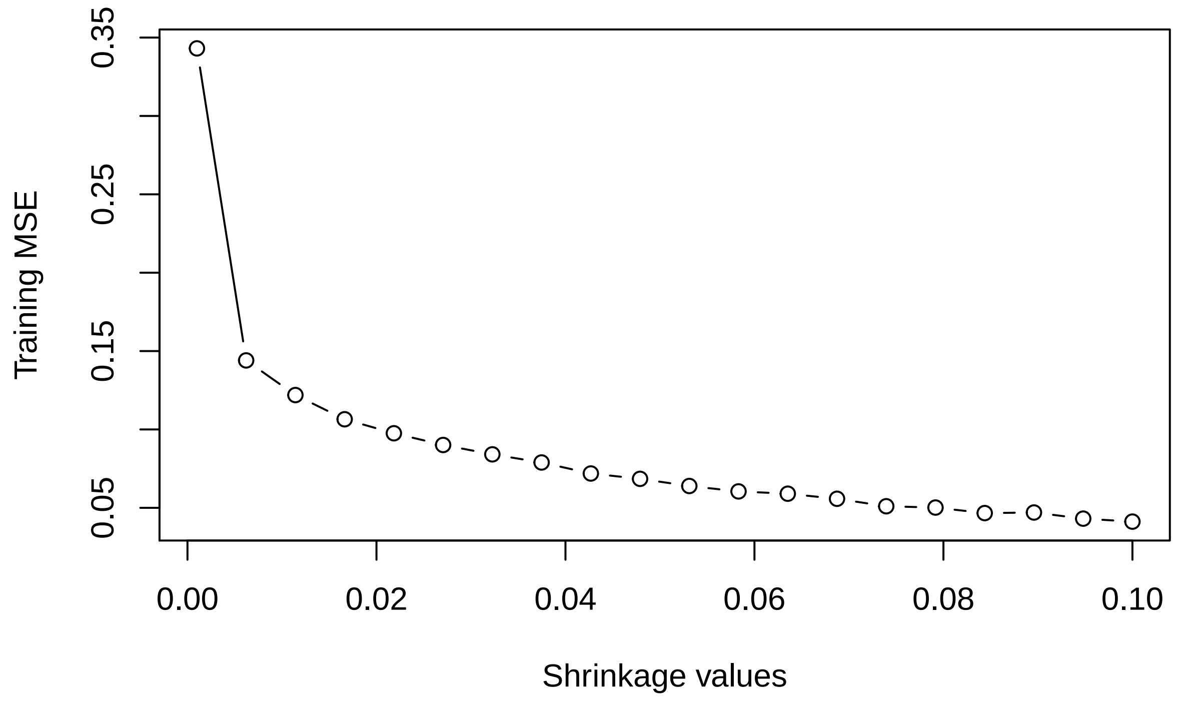
```
##
## Regression tree:
## tree(formula = logSalary ~ ., data = train.df)
## Variables actually used in tree construction:
## [1] "CAtBat"  "CRBI"    "RBI"     "AtBat"   "HmRun"   "CWalks"  "PutOuts"
## Number of terminal nodes:  9
## Residual mean deviance:  0.1545 = 31.06 / 201
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -1.753000 -0.207600 -0.005399  0.000000  0.221500  1.664000
```

```
plot(tree.hitters)
text(tree.hitters, pretty = 0)
title("Regression Tree")
```



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.

```r
set.seed(42)
pows = seq(0.001, 0.1, length.out = 20)
lambdas = pows
train.err = c()
test.err = c()
gbm_models <- list()

for (i in 1:length(lambdas)) {
  gbm_models[[i]] = gbm(logSalary ~ ., data = train.df, distribution = "gaussian",
                        n.trees = 1000, shrinkage = lambdas[i])

  pred.train = predict(gbm_models[[i]], train.df, n.trees = 1000)
  train.err[i] = mean((pred.train - train.df$logSalary)^2)

  pred.test = predict(gbm_models[[i]], test.df, n.trees = 1000)
  test.err[i] = mean((pred.test - test.df$logSalary)^2)
}

plot(lambdas, train.err, type = "b", xlab = "Shrinkage values", ylab = "Training MSE")
```



```r
min(train.err)
```

```
## [1] 0.04120374
```

```r
lambdas[which.min(train.err)]
```

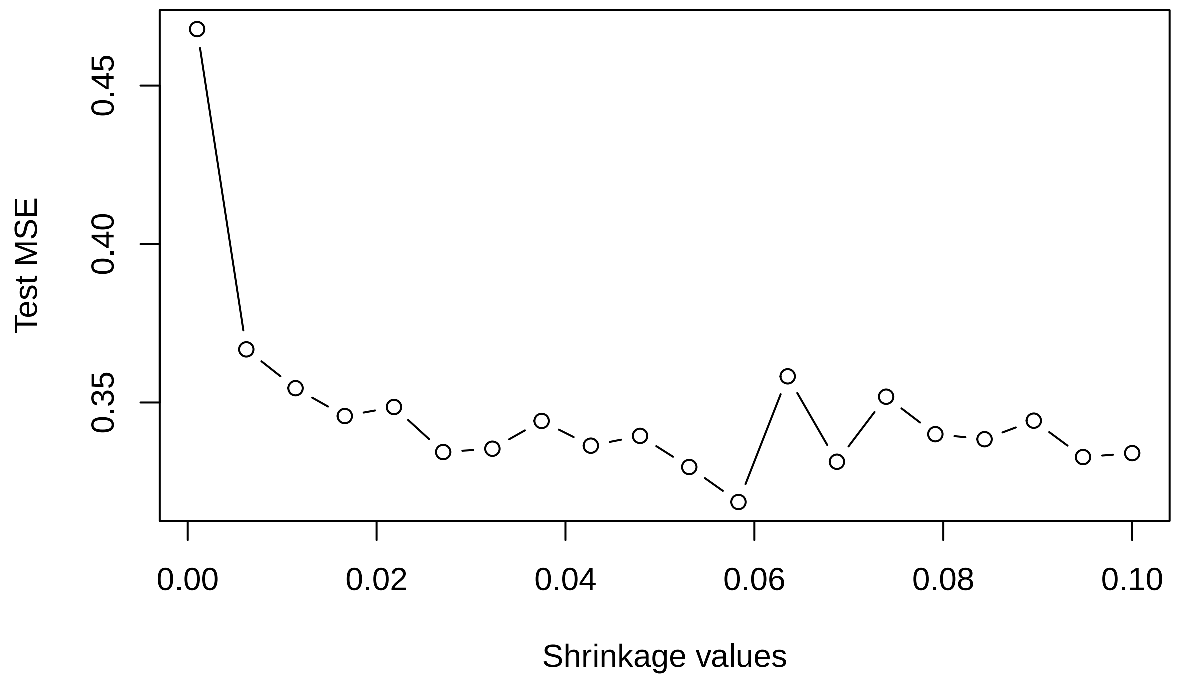
```
## [1] 0.1
```

- The boosted model with shrinkage parameter equals to 0.1 seems to be the best model when it has the lowest MSE = 0.04 in the training set.

Solution to Q8

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

```r
set.seed(42)

plot(lambdas, test.err, type = "b", xlab = "Shrinkage values", ylab = "Test MSE")
```



```r
min(test.err)
```

```
## [1] 0.3185998
```

```r
lambdas[which.min(test.err)]
```

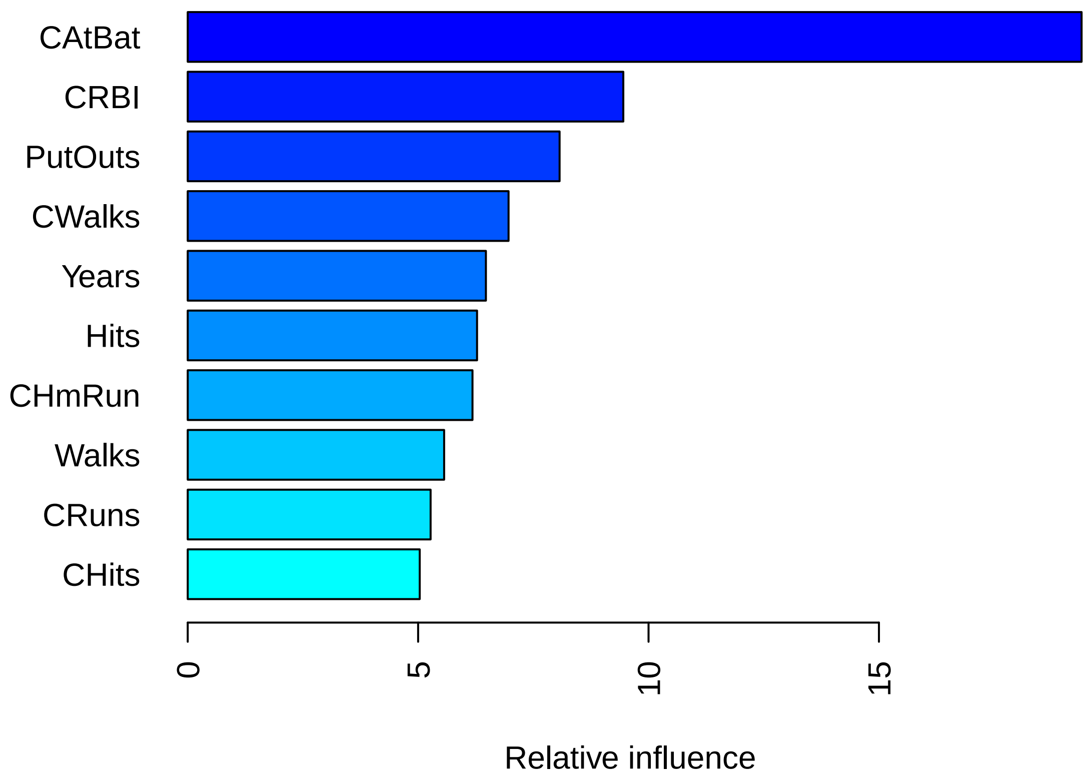
```
## [1] 0.05831579
```

- The boosted model with shrinkage parameter equals to 0.0583 seems to be the best model when it has the lowest MSE = 0.318 in the test set.

Solution to Q9

9. Which variables appear to be the most important predictors in the boosted model?

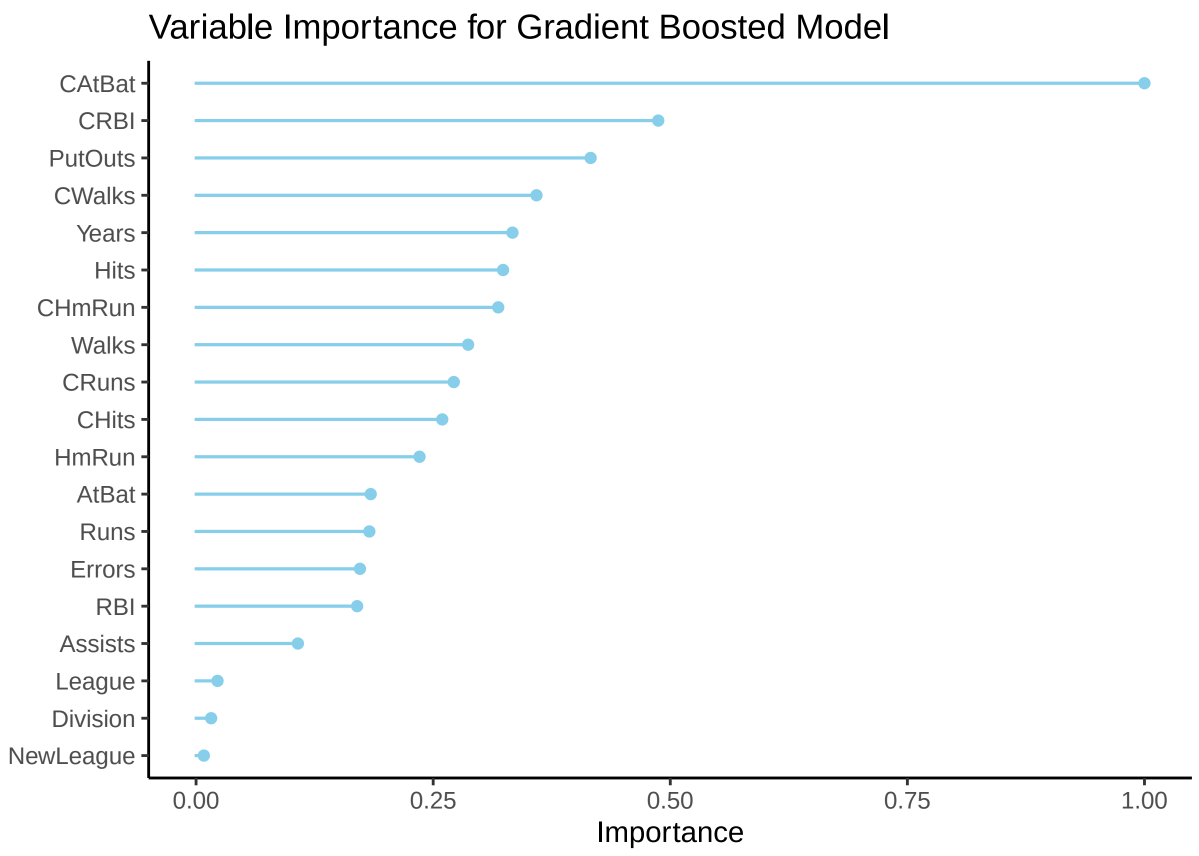
```r
set.seed(42)

fit.boost <- gbm(logSalary ~., data = train.df, shrinkage = 0.1, n.trees = 1000,
                 distribution = "gaussian")

par(mar = c(5, 8, 1, 1))
summary(fit.boost,
        cBars = 10,
        method = relative.influence, # also can use permutation.test.gbm
        las = 2
)
```



```
##                 var    rel.inf
## CAtBat       CAtBat 19.3977234
## CRBI           CRBI  9.4525291
## PutOuts     PutOuts  8.0698170
## CWalks       CWalks  6.9623492
## Years         Years  6.4710730
## Hits           Hits  6.2786735
## CHmRun       CHmRun  6.1797697
## Walks         Walks  5.5641246
## CRuns         CRuns  5.2705615
## CHits         CHits  5.0348289
## HmRun         HmRun  4.5699787
## AtBat         AtBat  3.5713740
## Runs           Runs  3.5438697
## Errors       Errors  3.3518180
## RBI             RBI  3.2945031
## Assists     Assists  2.0822365
## League       League  0.4383285
## Division   Division  0.3075536
## NewLeague NewLeague  0.1588880
```

```r
var_imp <- relative.influence(fit.boost,
                              n.trees = 1000,
                              scale. = TRUE)

data_frame(variable = names(var_imp),
           importance = var_imp) %>%
  mutate(variable = reorder(variable, importance)) %>%
  ggplot(aes(variable, importance)) +
  geom_col(width = 0.01,
           col = 'skyblue',
           alpha = 0.6) +
  geom_point(col = 'skyblue') +
  coord_flip() +
  labs(y = 'Importance',
       x = '',
       title = 'Variable Importance for Gradient Boosted Model')
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```



- From the above variable importance plots, we can infer that CAtBat, CRBI and PutOuts are the most importance predictors in the same order for the boosted model.
- It appears that the number of times at bat during their career (CAtBat) is by far the most important factor followed by the number of runs battles during their career(CRBI) and Number of put outs in 1986(PutOuts)

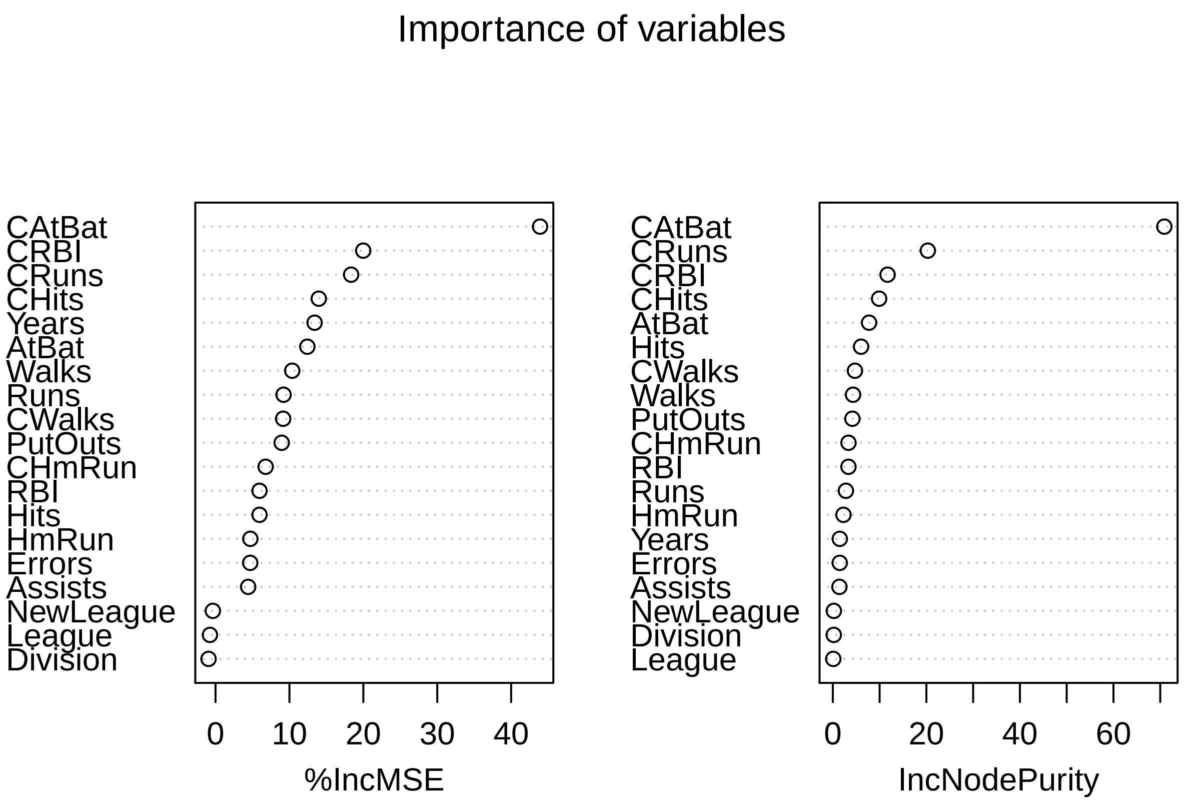
Solution to Q10

10. Apply bagging to the training set. What is the test set MSE for this approach?

```r
set.seed(42)
num_var <- dim(train.df)[2] - 1
bag.hitters <- randomForest(logSalary ~ ., data = train.df, mtry = num_var,
                            ntree = 1000, importance = TRUE)
yhat.bag <- predict(bag.hitters, newdata = test.df)
mean((yhat.bag - test.df$logSalary)^2)
```

```
## [1] 0.2487804
```

```r
varImpPlot(bag.hitters, main = "Importance of variables")
```



- The MSE for bagging approach is 0.249, which is lower than the MSE for boosting approach which was 0.319 for the test dataset.
- According to the plot, CatBat, CRBI and CRuns are the most important variables in this order.