# p8106_hw4_yg2625

*Yue Gu*

*April 21, 2019*

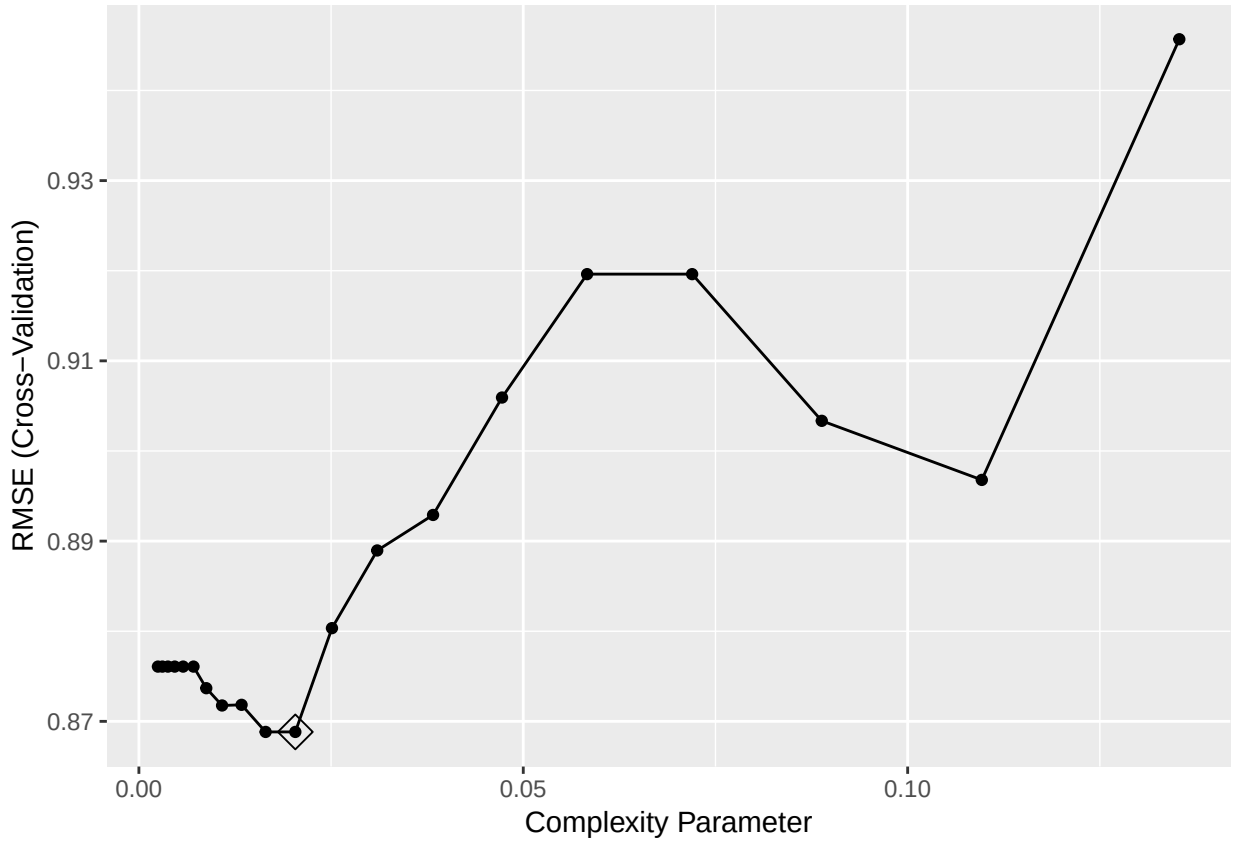
**1. This problem involves the Prostate data in the lasso2 package (see L5.Rmd). Use set.seed() for reproducible results.**

### load data

```
data("Prostate")
pros_data = Prostate%>%
  janitor::clean_names()
```

## (a) Fit a regression tree with lpsa as the response and the other variables as predictors. Use cross-validation to determine the optimal tree size. Which tree size corresponds to the lowest cross-validation error? Is this the same as the tree size obtained using the 1 SE rule?

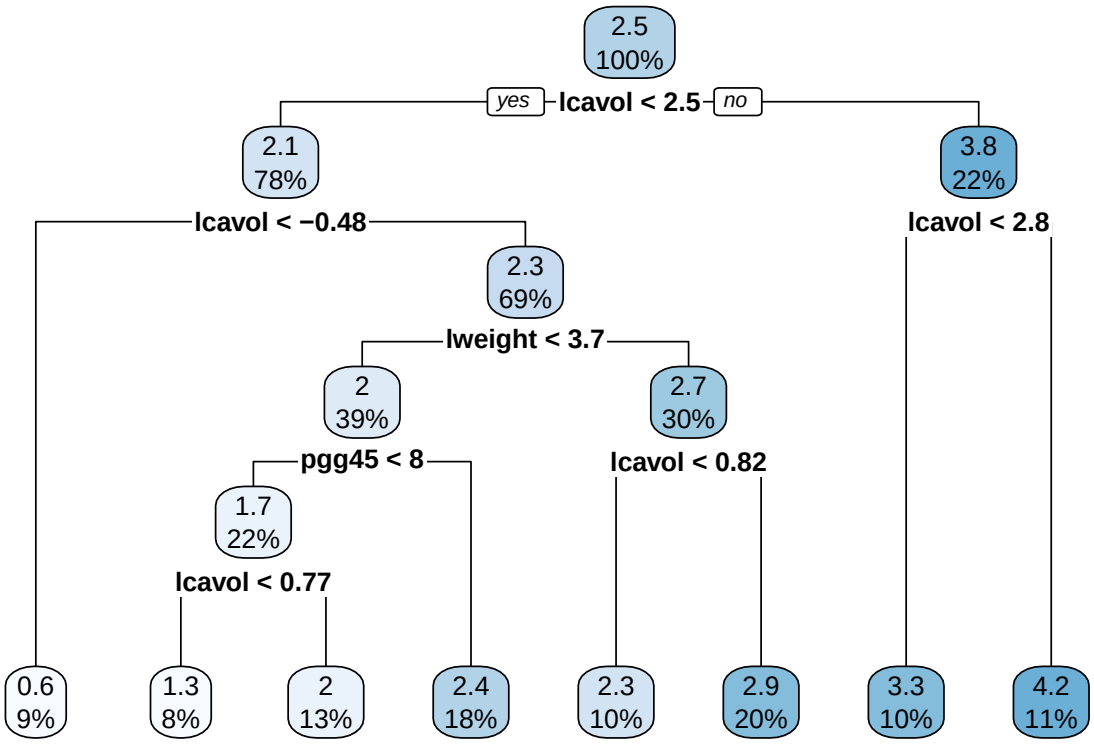
```
# use cross-validation through caret
ctrl <- trainControl(method = "cv")

# tune over cp, method = "rpart"
rpart.fit1 <- train(lpsa ~ ., pros_data,
                    method = "rpart",
                    tuneGrid = data.frame(cp = exp(seq(-6,-2, length = 20))),
                    trControl = ctrl)
ggplot(rpart.fit1, highlight = TRUE)
```




```
# cptable showed that the optimal tree size is 8
rpart.fit1$finalModel$cptable
```

```
##           CP nsplit rel error
## 1 0.34710828      0 1.0000000
## 2 0.18464743      1 0.6528917
## 3 0.05931585      2 0.4682443
## 4 0.03475635      3 0.4089284
## 5 0.03460901      4 0.3741721
## 6 0.02156368      5 0.3395631
## 7 0.02146995      6 0.3179994
## 8 0.00000000      7 0.2965295
```

```
rpart.plot(rpart.fit1$finalModel)
```



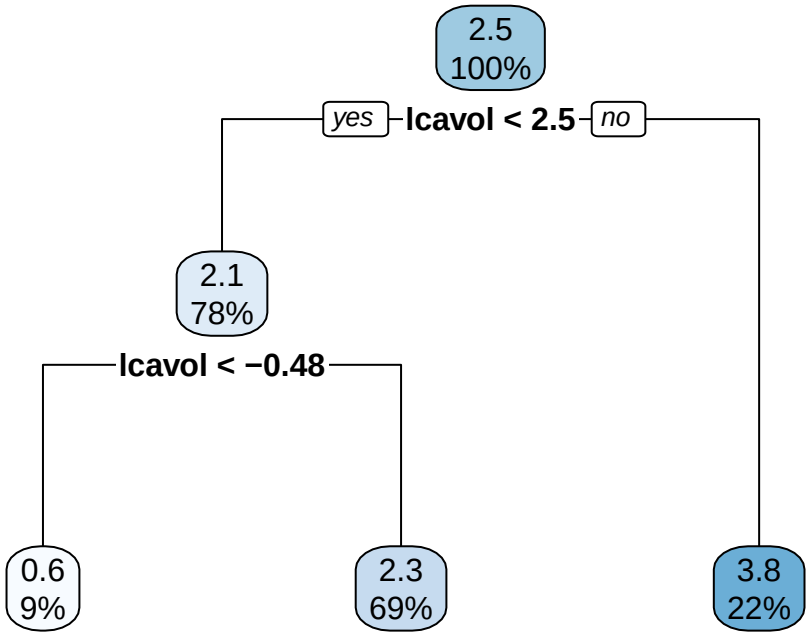

```
# use 1SE through caret
rpart.fit2 <- train(lpsa ~ ., pros_data,
                    method = "rpart",
                    tuneGrid = data.frame(cp = exp(seq(-6,-2, length = 20))),
                    trControl = trainControl(method = 'cv',
                                             number = 10,
                                             selectionFunction = 'oneSE'))
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
```

```
# cptable showed that the optimal tree size is 3
rpart.fit2$finalModel$cptable
```

```
##           CP nsplit rel error
## 1 0.34710828      0 1.0000000
## 2 0.18464743      1 0.6528917
## 3 0.08882807      2 0.4682443
```

```
rpart.plot(rpart.fit2$finalModel)
```




Based on the result, cross-validation showed that the optimal tree size is 8 while 1SE obtained optimal tree size as 3. Hence, 1SE rule generates tree with smaller size.

## (b) Create a plot of the final tree you choose. Pick one of the terminal nodes, and interpret the information displayed.

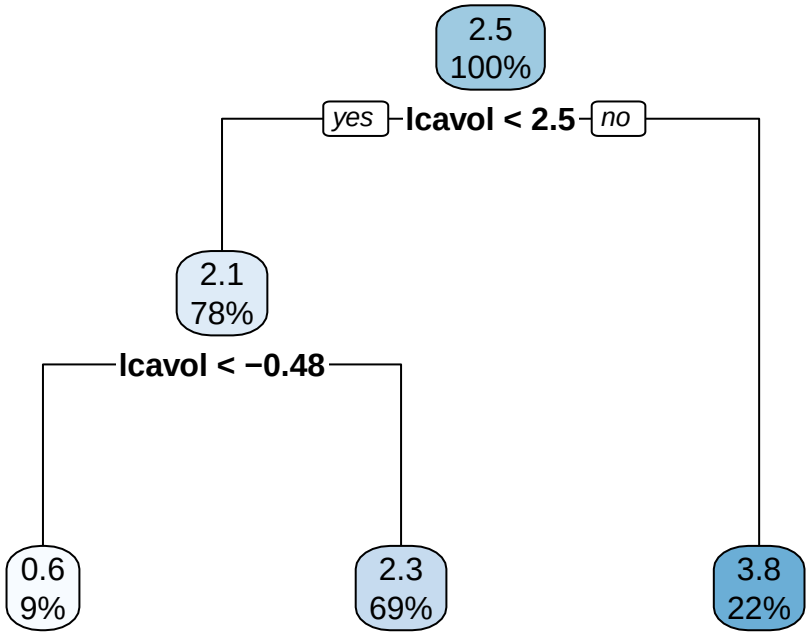
```
resamp = resamples(list(minErr = rpart.fit1, min_1se = rpart.fit2))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: minErr, min_1se
## Number of resamples: 10
##
## MAE
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## minErr  0.6029257 0.6833061 0.7132861 0.7481058 0.7878006 1.0108929    0
## min_1se 0.5640925 0.6790613 0.7081498 0.7305603 0.7886243 0.8963087    0
##
## RMSE
##              Min.   1st Qu.    Median      Mean   3rd Qu.     Max. NA's
## minErr  0.6940977 0.7916380 0.8659767 0.8688293 0.9202064 1.116127    0
## min_1se 0.7171773 0.8096023 0.8513418 0.8740592 0.9689131 1.015270    0
```

```
##
## Rsquared
##                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## minErr  0.10742511 0.4435906 0.5507225 0.4869429 0.5676906 0.6732530    0
## min_1se 0.08493423 0.4242442 0.4933295 0.4436730 0.5761213 0.5981874    1
```

Since two regression generates similar RMSE. Following principle of parsimony, we choose the simpler model with tree using 1SE principle. And the plot is shown below:

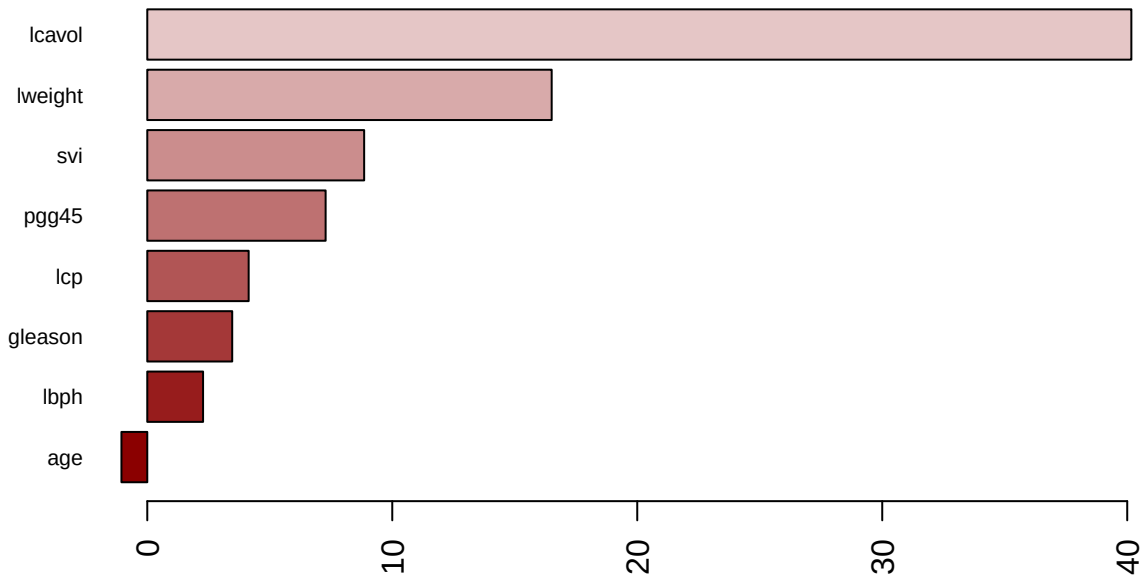
```
rpart.plot(rpart.fit2$finalModel)
```




**Interpretation:** The regression tree pruned by 1SE rule has size as 3. When log(cancer volume)(lcavol) is greater than 2.5, the log(prostate specific antigen) is predicted to be 3.8, which contains 22% of the trainning observations.

## (c) Perform bagging and report the variable importance.

```
bagging <- ranger(lpsa ~., pros_data,
                          mtry = 8, splitrule = "variance",
                          min.node.size = 30,
                          importance = "permutation",
                          scale.permutation.importance = TRUE)

barplot(sort(ranger::importance(bagging), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("darkred","white","darkblue"))(19))
```




Based on the output, variable importance ranking: lcavol>lweight>svi>pgg45>lcp>gleason>lbph>age.

## (d) Perform random forests and report the variable importance.

```
set.seed(1)
rf.grid <- expand.grid(mtry = 1:8,
                       splitrule = "variance",
                       min.node.size = 1:30)

rf.fit <- train(lpsa ~ ., pros_data,
                method = "ranger",
                tuneGrid = rf.grid,
                trControl = ctrl)
# get best tuning parameter alpha = 5
rf.fit$bestTune
```

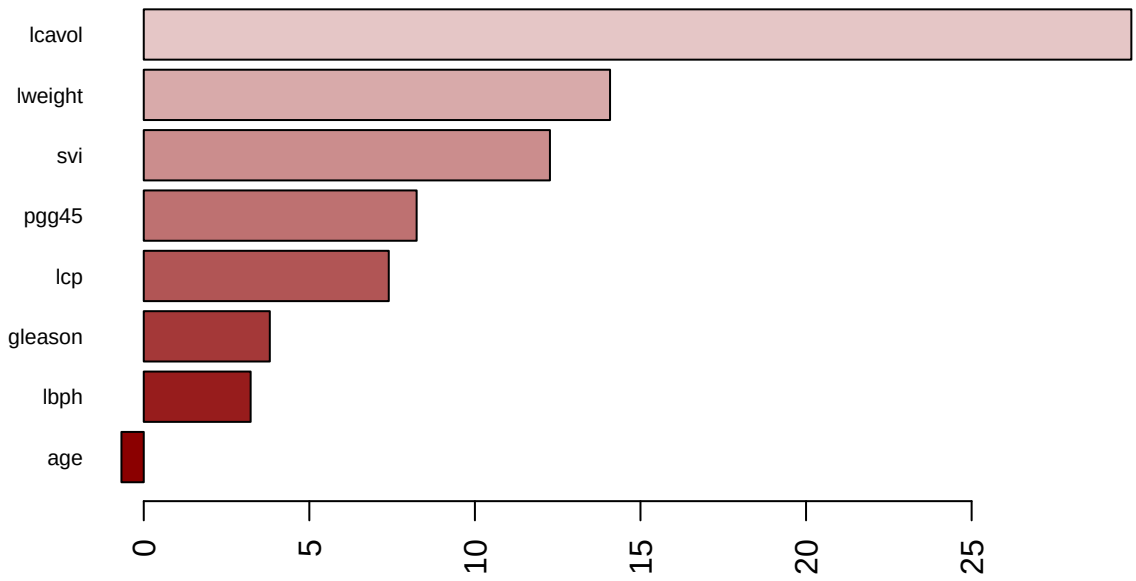
```
##     mtry splitrule min.node.size
## 141    5  variance            21
```

```
# random forests plot
ggplot(rf.fit, highlight = TRUE)
```

```
## Warning: The shape palette can deal with a maximum of 6 discrete values
## because more than 6 becomes difficult to discriminate; you have 8.
## Consider specifying shapes manually if you must have them.
```

```
## Warning: Removed 60 rows containing missing values (geom_point).
```


```
# fit random forest model using best tuning parameter
rf <- ranger(lpsa ~., pros_data,
                        mtry = 5, splitrule = "variance",
                        min.node.size = 30,
                        importance = "permutation",
                        scale.permutation.importance = TRUE)

barplot(sort(ranger::importance(rf), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("darkred","white","darkblue"))(19))
```



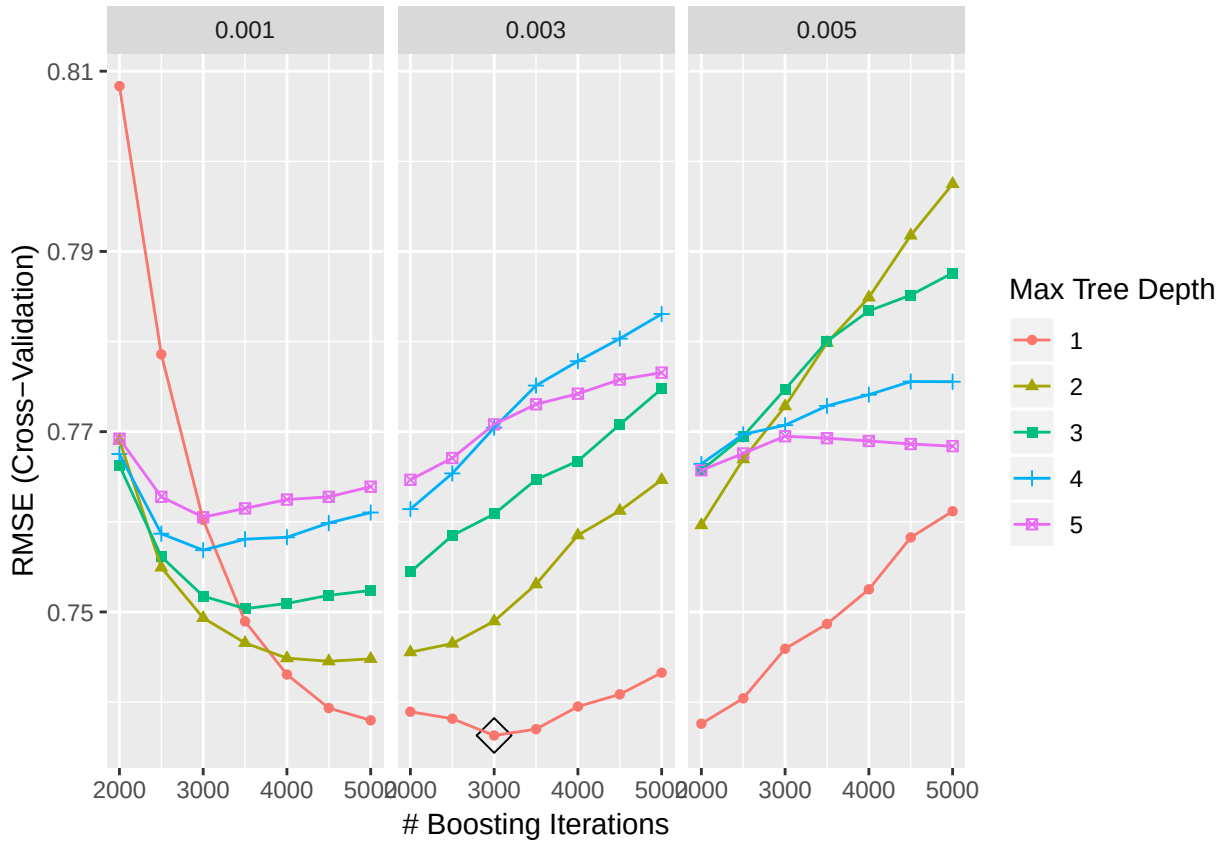

```
summary(rf.fit)
```

```
##                             Length Class         Mode
## predictions                 97     -none-        numeric
## num.trees                    1     -none-        numeric
## num.independent.variables    1     -none-        numeric
## mtry                         1     -none-        numeric
## min.node.size                1     -none-        numeric
## prediction.error             1     -none-        numeric
## forest                       8     ranger.forest list
## splitrule                    1     -none-        character
## treetype                     1     -none-        character
## r.squared                    1     -none-        numeric
## call                         9     -none-        call
## importance.mode              1     -none-        character
## num.samples                  1     -none-        numeric
## replace                      1     -none-        logical
## xNames                       8     -none-        character
## problemType                  1     -none-        character
## tuneValue                    3     data.frame    list
## obsLevels                    1     -none-        logical
## param                        0     -none-        list
```

Based on the output, variable importance ranking: lcavol>lweight>svi>pgg45>lcp>lbph>gleason>age.

## (e) Perform boosting and report the variable importance.

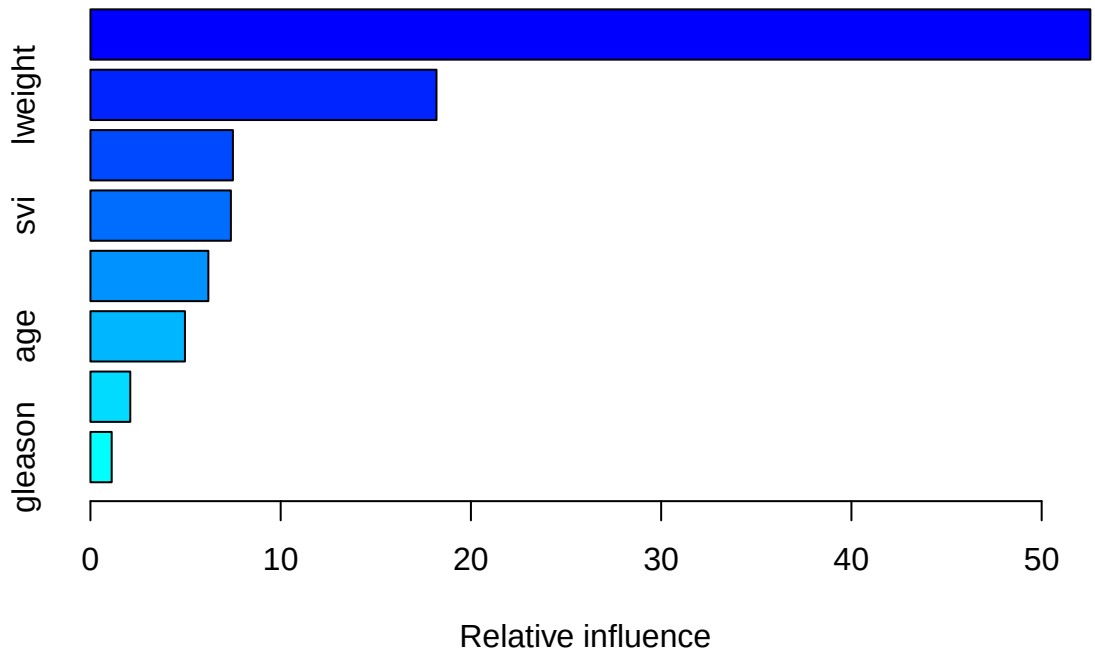
```
set.seed(1)
gbm.grid <- expand.grid(n.trees = c(2000,2500,3000,3500,4000,4500,5000),
                        interaction.depth = 1:5,
                        shrinkage = c(0.001,0.003,0.005),
                        n.minobsinnode = 1)

gbm.fit <- train(lpsa ~ ., pros_data,
                 tuneGrid = gbm.grid,
                 trControl = ctrl,
                 method = "gbm",
                 verbose = FALSE)
# boosting plot
ggplot(gbm.fit, highlight = TRUE)
```




```
# best tunning parameter alph = 38
gbm.fit$bestTune
```

```
##    n.trees interaction.depth shrinkage n.minobsinnode
## 38    3000                 1     0.003              1
```

```
# summary output
summary(gbm.fit)
```




```
##             var   rel.inf
## lcavol   lcavol 52.559738
## lweight lweight 18.189924
## lcp         lcp  7.489362
## svi         svi  7.382924
## pgg45     pgg45  6.198291
## age         age  4.968294
## lbph       lbph  2.094587
## gleason gleason  1.116880
```

Based on the result, variable importance ranking: lcavol>lweight>svi>lcp>pgg45>age>lbph>gleason.

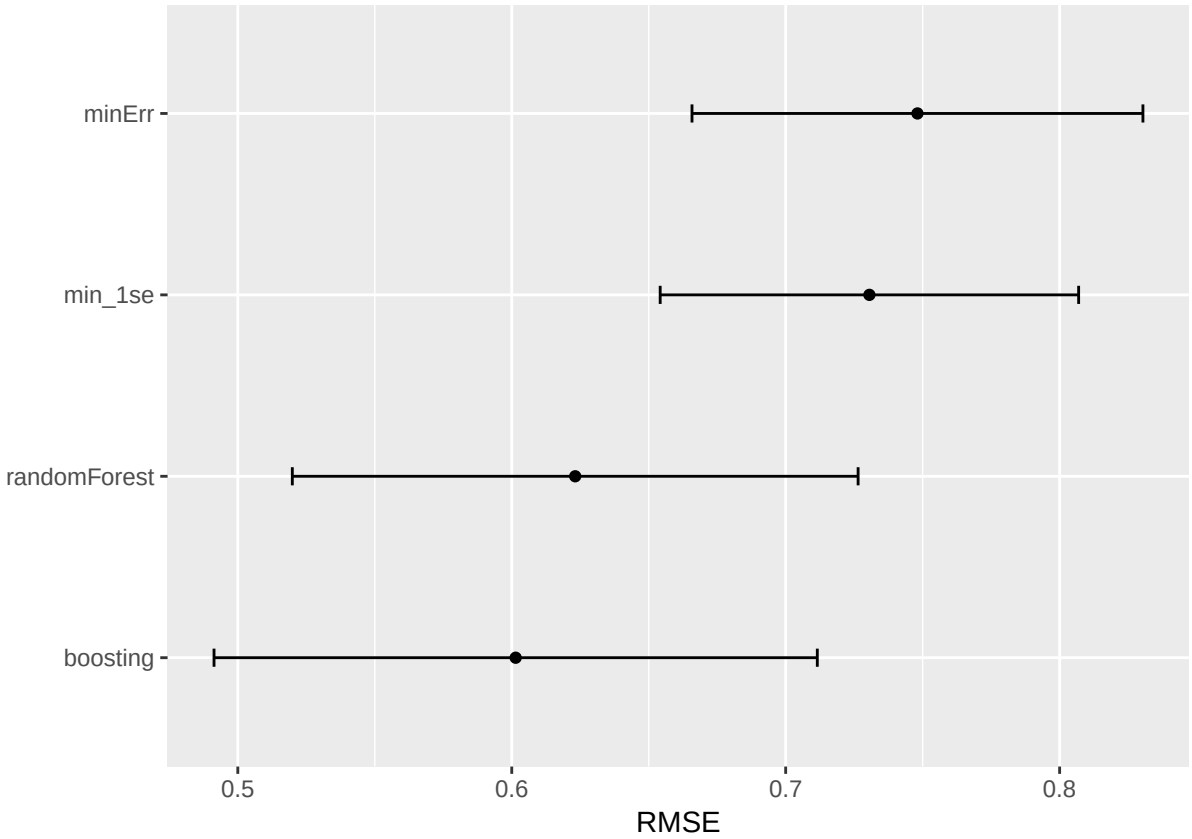
## (f) Which of the above models will you select to predict PSA level? Explain.

```
resamp2 = resamples(list(minErr = rpart.fit1,
                         min_1se = rpart.fit2,
                         randomForest = rf.fit,
                         boosting = gbm.fit))

summary(resamp2)
```

```
##
## Call:
## summary.resamples(object = resamp2)
```

```
##
## Models: minErr, min_1se, randomForest, boosting
## Number of resamples: 10
##
## MAE
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## minErr       0.6029257 0.6833061 0.7132861 0.7481058 0.7878006 1.0108929
## min_1se      0.5640925 0.6790613 0.7081498 0.7305603 0.7886243 0.8963087
## randomForest 0.4391181 0.5799877 0.5927113 0.6231634 0.6411043 0.9684338
## boosting     0.4199595 0.4948608 0.5521303 0.6014267 0.6682515 0.8755513
##              NA's
## minErr          0
## min_1se         0
## randomForest    0
## boosting        0
##
## RMSE
##                   Min.   1st Qu.    Median      Mean   3rd Qu.     Max.
## minErr       0.6940977 0.7916380 0.8659767 0.8688293 0.9202064 1.116127
## min_1se      0.7171773 0.8096023 0.8513418 0.8740592 0.9689131 1.015270
## randomForest 0.5182232 0.6775767 0.7640668 0.7500340 0.8232725 1.078782
## boosting     0.5134918 0.6555081 0.7108809 0.7362931 0.7893717 1.001967
##              NA's
## minErr          0
## min_1se         0
## randomForest    0
## boosting        0
##
## Rsquared
##                    Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## minErr       0.10742511 0.4435906 0.5507225 0.4869429 0.5676906 0.6732530
## min_1se      0.08493423 0.4242442 0.4933295 0.4436730 0.5761213 0.5981874
## randomForest 0.23616569 0.6109251 0.6770639 0.6289637 0.7185763 0.7342918
## boosting     0.37577693 0.5852946 0.6866416 0.6439764 0.7348701 0.7897067
##              NA's
## minErr          0
## min_1se         1
## randomForest    0
## boosting        0
```

```
ggplot(resamp2, "RMSE") +
  labs(y = 'RMSE')
```




Based on the output, simple linear regression trees method including minimum CV error and 1SE principle methods generates model with larger RMSE than ensemble methods(bagging, random forest, boosting). And bagging being a special case of random forest selecting 8 predictors(mtry = 8), violated principle of parsimony compared to random forest and boosting methods. Boosting generates model with smaller RMSE than randomForest. Hence, boosting model is most preferrable.