Consider the `Boston` data set from *R*. This example shows how to build tree models to predict the median price of houses `medv` in the Boston area. All other variables in the data set are considered predictors. It is of interest to identify which variables are most useful to predict the price of houses. Divide the dataset into a training (50%) and a test set as needed.

Single trees

a) Use function `tree` from library `tree` to fit a regression tree. Use the full dataset. Plot the resulting tree. Plot the regions defined by the terminal nodes.

b) Fit a regression tree on the training set. Plot the resulting tree. Prune the tree to five terminal nodes. Use cross validation to find the best number of terminal nodes.

c) Which predictors are found most important? Report the test MSPE. Fit a tree with 5 splits. Compare MSPE.

Bagging

d) Use function `randomForest` from library `randomForest` to fit regression trees with bagging. How many trees are bagged?

e) Which predictors are found most important? Report the test MSPE. Fit regression trees limiting the number of trees to 25. Compare MSPE.

f) Fit a random forest model to the training set. Compare the test MSPE with that of the bagging tree model.

Boosting

g) Use function `gbm` from library `bgm` to fit 5000 boosted regression trees. Limit the depth of each tree to 4 using the option `interaction.depth=4`. Use option `distribution="gaussian"` since this is a *regression* tree.

h) Which predictors are found most important? Report the test MSPE.

```
library(MASS)      # Boston dataset
library(tree)      # tree()

dim(Boston)
# [1] 506  14
# medv is response, p=13 predictors

# tree - 2 predictors, full dataset
#===================================================================
tree0=tree(medv~lstat+rm,Boston)

# scatterplot on predictors space
plot(lstat~rm,Boston,pch=19,cex=0.6,col="red")
# regions and predicted averages
partition.tree(tree0,add = T,col="blue")
# tree plot
plot(tree0)
text(tree0,cex=0.75)

# inequality at split is for left arm
# $16950 is house prediction for rm < 6.94, and 14.4 < lstat < 19.83

# all predictors - train set
#===================================================================
set.seed(1)
n = nrow(Boston)
train = sample(1:n,n/2)   # 253 train rows
dtrain = Boston[train,]
dtest = Boston[-train,]

tree1=tree(medv~.,Boston,subset=train)
summary(tree1)

# Regression tree:
# tree(formula = medv ~ ., data = Boston, subset = train)
# Variables actually used in tree construction:
# [1] "lstat" "rm"    "dis"
# Number of terminal nodes:  8
# Residual mean deviance:  12.65 = 3099 / 245
# Distribution of residuals:
#      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
# -14.10000  -2.04200  -0.05357   0.00000   1.96000  12.60000

# "lstat" "rm"    "dis"     best classifiers
# RSS is 3099
# tree with 8 terminal nodes
# 253 - 8 = 245 dof
```

```
plot(tree1)
text(tree1,cex=0.75)
# partition.tree()  does not apply for 3 classifiers

names(tree1)
# [1] "frame"   "where"   "terms"   "call"    "y"       "weights"

tree1$frame
#       var   n        dev     yval splits.cutleft splits.cutright
# 1   lstat 253 20894.6572 22.67312         <9.715          >9.715
# 2      rm 103  7764.5843 30.13204         <7.437          >7.437
# 4      rm  89  3310.1604 27.57640        <6.7815         >6.7815
# 8     dis  61  1994.6223 25.52131        <2.6221         >2.6221
# 16 <leaf>   5   615.7800 37.40000
# 17     rm  56   610.3336 24.46071        <6.4755         >6.4755
# 34 <leaf>  31   136.3555 22.54194
# 35 <leaf>  25   218.3200 26.84000
# 9  <leaf>  28   496.6496 32.05357
# 5  <leaf>  14   177.8436 46.37857
# 3   lstat 150  3464.7147 17.55133         <21.49          >21.49
# 6   lstat 120  1593.6987 19.16333         <14.48          >14.48
# 12 <leaf>  62   398.4892 21.04032
# 13 <leaf>  58   743.2822 17.15690
# 7  <leaf>  30   311.8897 11.10333

# 22.67312 is mean response (medv) in training set
# dev = deviance (square distance to the mean of that region)
# <leaf> rows are terminal nodes
#        sum of deviance of terminal nodes is 3099
#        sum of deviance decreases with large n. splits
# y val  of terminal nodes are means of regions
# leftmost column is order of splitting
# 1st row splits into rows 2 and 3
# 2nd row splits into row 4 and 5
# n number of obs in terminal nodes
# prunning tree to 5 terminal nodes
#================================================================

pruned1=prune.tree(tree1,best=5)
pruned1$frame
#      var   n        dev     yval splits.cutleft splits.cutright
# 1  lstat 253 20894.6572 22.67312         <9.715          >9.715
# 2     rm 103  7764.5843 30.13204         <7.437          >7.437
# 4     rm  89  3310.1604 27.57640        <6.7815         >6.7815
# 8 <leaf>  61  1994.6223 25.52131
# 9 <leaf>  28   496.6496 32.05357
# 5 <leaf>  14   177.8436 46.37857
# 3  lstat 150  3464.7147 17.55133         <21.49          >21.49
# 6 <leaf> 120  1593.6987 19.16333
# 7 <leaf>  30   311.8897 11.10333

summary(pruned1)
# Regression tree:
# snip.tree(tree = tree1, nodes = c(6L, 8L))
# Variables actually used in tree construction:
# [1] "lstat" "rm"
# Number of terminal nodes:  5
# Residual mean deviance:  18.45 = 4575 / 248
# Distribution of residuals:
#     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
# -9.56300 -2.86300 -0.06333  0.00000  2.69700 24.48000

# Residual mean deviance 18.45, larger than 12.65 of non-pruned tree

plot(pruned1)
text(pruned1)

# regions
plot(rm~lstat,Boston,pch=19,cex=0.6,col="red")
partition.tree(pruned1,add = T,col="blue")
# most important predictor must go on X-axis

# this way is wrong!
plot(lstat~rm,Boston,pch=19,cex=0.6,col="red")
partition.tree(pruned1,add = T,col="blue")
```

```
# cross validation - best n. terminal nodes
#==================================================================
cv.boston=cv.tree(tree1)
plot(cv.boston)                            # 8 nodes is best tree

# test error rate
y.test= Boston[-train,"medv"]             # y values in test set
newval= Boston[-train,]
yhat  = predict(tree1,newval)

# plot means of terminal regions (yhat) vs y
plot(yhat~y.test,pch=19,cex=0.5,ylim=c(10,50))
abline(0,1)
grid()

# identify houses with large residuals

res = y.test - yhat
# yhat is vector
# vector has no rownames
a = rownames(as.matrix(yhat))   # as.matrix required

text(yhat~y.test,labels=ifelse(res>10,a,""),pos=1,offset=0.25,cex=0.4)

# houses with res>5
a[res>5]
#  [1] "8"   "9" "124" "149" "180" "183" "185" "209" "210" "215"
# [11] "223" "264" "267" "292" "369" "370" "371" "409" "413" "474"

text(yhat~y.test,labels=ifelse(res>15|res<(-15),a,""),pos=1,offset=0.25,cex=0.4)

# n. of predictions = n. of regions
unique(yhat)
# 26.84000 22.54194 32.05357 17.15690 11.10333 21.04032 37.40000 46.37857

# test MSE
mspe = mean((yhat-y.test)^2)   # 25.05
sqrt(mspe)                     #[1] 5.004557

# predictions are within $5005 of true median home value
```

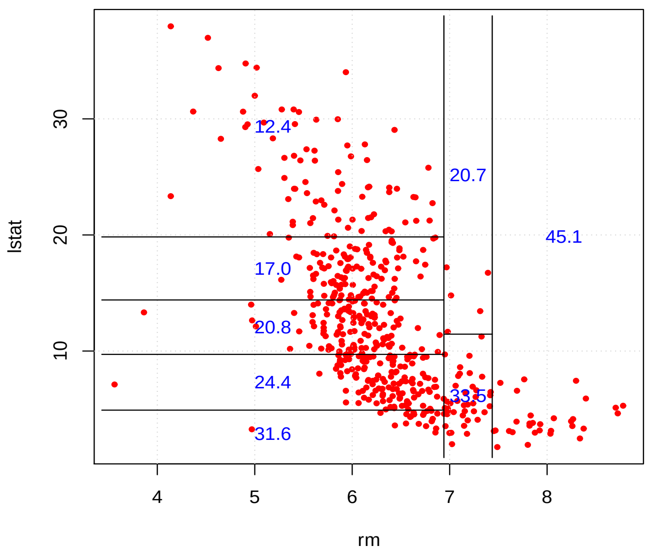

Figure 1: Regions for the tree with 2 predictors *lstat* and *rm*, full dataset

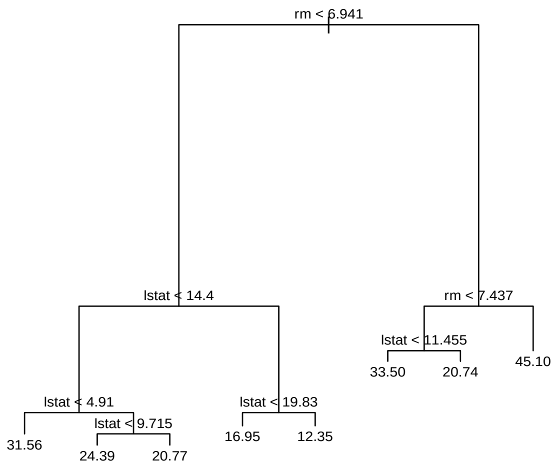

Figure 2: Tree with 2 predictors *lstat* and *rm*, full dataset

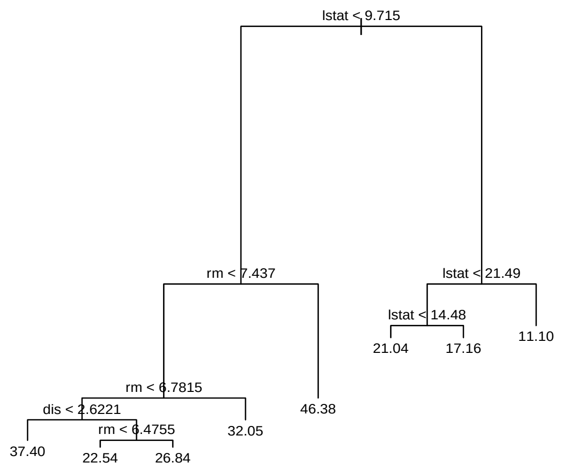

Figure 3: Tree1 considering all predictors (but using only 3, lstat, rm, dis), training set

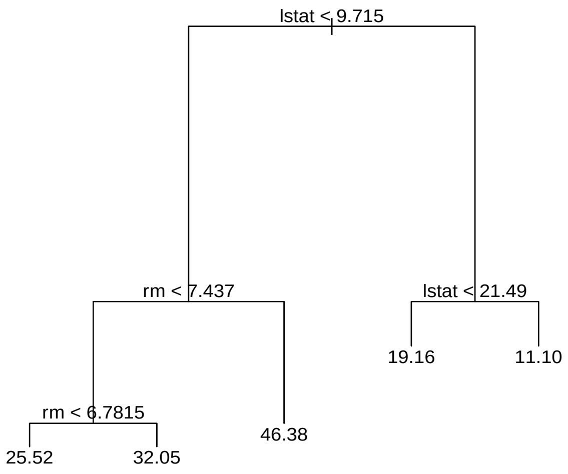

Figure 4: Pruned Tree with 5 terminal nodes, training set

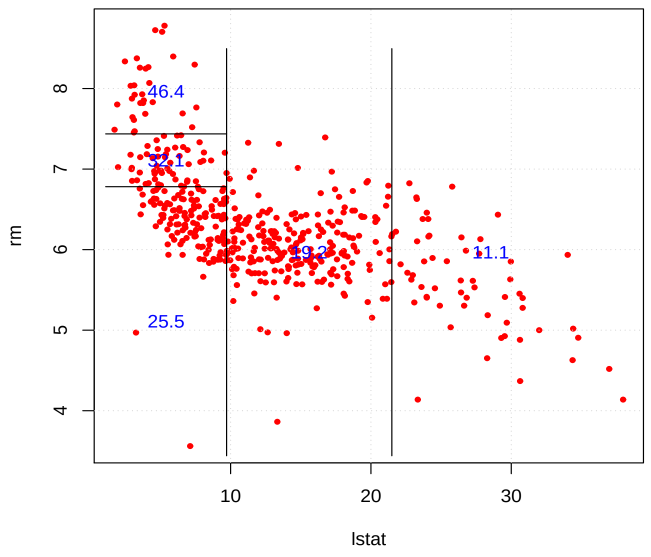

Figure 5: Regions of pruned tree with 5 terminal nodes, training set

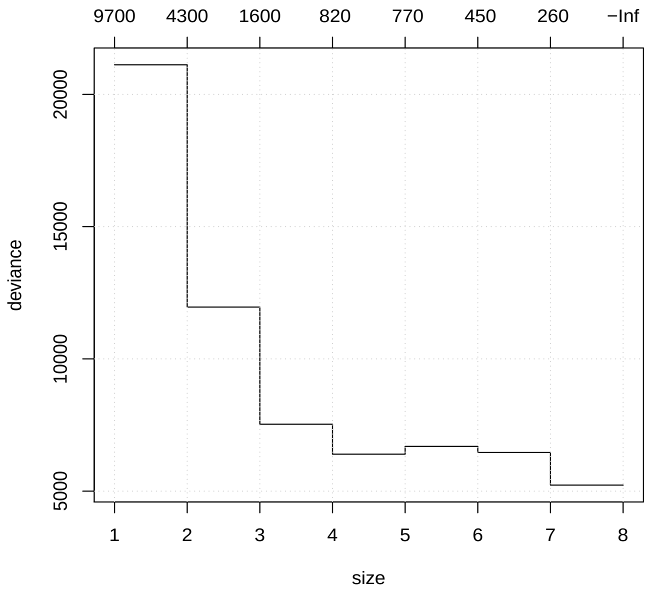

Figure 6: MSPE for different number of terminal nodes, from plotting output of cv.tree()

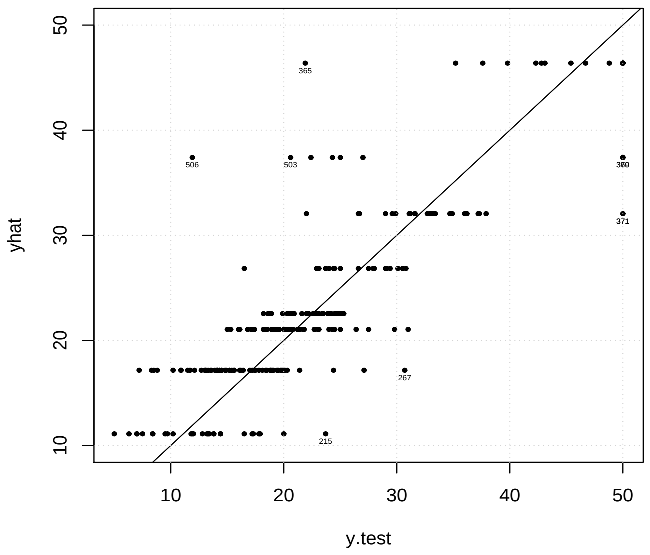

Figure 7: Output of cv.tree() from Tree with all predictors , training set

```
# bagging.r
# all 13 predictors should be considered at each split (mtry=p)

library(MASS)      # Boston dataset
library(randomForest)

dim(Boston)  # [1] 506  14
# medv is response, p=13 predictors
n = nrow(Boston)
train = sample(1:n,n/2)   # 253 train rows

# Bagging and Random Forests
#==================================================================
set.seed(1)
bag1=randomForest(medv~.,data=Boston,subset=train,mtry=13,importance = T)
bag1
# randomForest(formula = medv ~ ., data = Boston, mtry = 13, importance = T,subset = train)
#                Type of random forest: regression
#                      Number of trees: 500
# No. of variables tried at each split: 13
#           Mean of squared residuals: 10.54712
#                     % Var explained: 87.23

# 10.54 is train MSE
# p = 13 predictors
# importance of predictors needed for:  importance(bag1)

summary(bag1)
#                 Length Class  Mode
# call              6    -none- call
# type              1    -none- character
# predicted       253    -none- numeric
# mse             500    -none- numeric
# rsq             500    -none- numeric
# oob.times       253    -none- numeric
# importance       26    -none- numeric
# importanceSD     13    -none- numeric
# localImportance   0    -none- NULL
# proximity         0    -none- NULL
# ntree             1    -none- numeric
# mtry              1    -none- numeric
# forest           11    -none- list
# coefs             0    -none- NULL
# y               253    -none- numeric
# test              0    -none- NULL
# inbag             0    -none- NULL
# terms             3    terms  call
# 500 train MSEs

# times train obs was OOB
table(bag1$oob.times)
# 144 153 158 160 161 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182
#   1   1   1   1   1   3   1   2   3   2   4   2   2   3   9   6   9   9   6  11  12  10  11   7   8
# 186 187 188 189 190 191 192 193 194 195 196 197 198 200 201 202 203 204 205 206 207 211 212 215 216
#   6   9   8   9   6   8   5   1   7   6   3   5   2   1   3   5   2   2   2   2   1   1   1   1   1

head(bag1$predicted)
#      175      410      307      222      391      344
# 21.90482 16.87879 34.48303 17.70609 14.03278 29.22515
head(bag1$y)
#  175  410  307  222  391  344
# 22.6 27.5 33.4 21.7 15.1 23.9
head(Boston[train,"medv"])
# [1] 22.6 27.5 33.4 21.7 15.1 23.9


# test set performance
y.test=Boston[-train,"medv"]         # y values in test set
yhat.bag = predict(bag1,newdata=Boston[-train,])

# residuals and  row numbers
res = y.test - yhat.bag
a = rownames(as.matrix(yhat.bag))   # as.matrix is required

# plot means of terminal regions (yhat) vs y
plot(yhat.bag~y.test,pch=19,cex=0.5,ylim=c(10,50))
abline(0,1)
grid()
text(yhat.bag~y.test,labels=ifelse(res>5,a,""),pos=1,offset=0.25,cex=0.4)
# dots seem to cluster around 45 degree line

# MSEP
mean((yhat.bag-y.test)^2)    #[1] 12.81118    (this value changes)
# half of CV best tree MSEP -good!

# limit n. of trees to 25
bag2=randomForest(medv~.,data=Boston,subset=train,mtry=13,ntree=25)
yhat.bag = predict(bag2,newdata=Boston[-train,])
mean((yhat.bag-y.test)^2)    # [1] 14.29294    (this value changes)
```

```
# random forest (mtry < p)
#=================================================================
set.seed(1)
forest1=randomForest(medv~.,data=Boston,subset=train,mtry=6,importance=T)
forest1
# randomForest(formula = medv ~ ., data = Boston, mtry = 6, importance = T,subset = train)
#                Type of random forest: regression
#                      Number of trees: 500
# No. of variables tried at each split: 6
#           Mean of squared residuals: 11.64401
#                     % Var explained: 85.9

# test set performance
yhat.rf = predict(forest1,newdata=Boston[-train,])
mean((yhat.rf-y.test)^2)      #[1] 11.20823
# little improvement over bagging

# importance of each predictor

importance(forest1)
#            %IncMSE IncNodePurity
# crim    12.4532926    1025.78688
# zn       2.7950457      52.75363
# indus   10.8424427     982.29679
# chas     0.9750709      67.04497
# nox     11.9825575    1264.03961
# rm      34.0973035    6667.19323
# age      9.6228463     458.13995
# dis     14.4846189    1280.22577
# rad      4.0478715     106.03032
# tax      7.6757169     481.81702
# ptratio 12.6021811    1030.07392
# black    6.7831663     373.05591
# lstat   27.4485257    6825.52957

# IncMSE - avg increase in MSE when predictor is excluded from model
# IncNodePurity - avg increase in RSS from splits using this predictor

# plot these two columns - for convenience
varImpPlot(forest1,main="")

# rm & lstat most important predictors
```

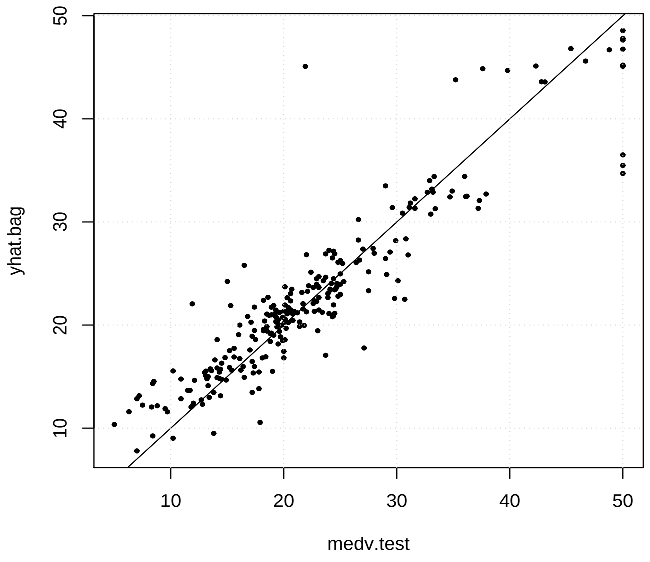

Figure 8: Bagging trees predictions vs observed y values

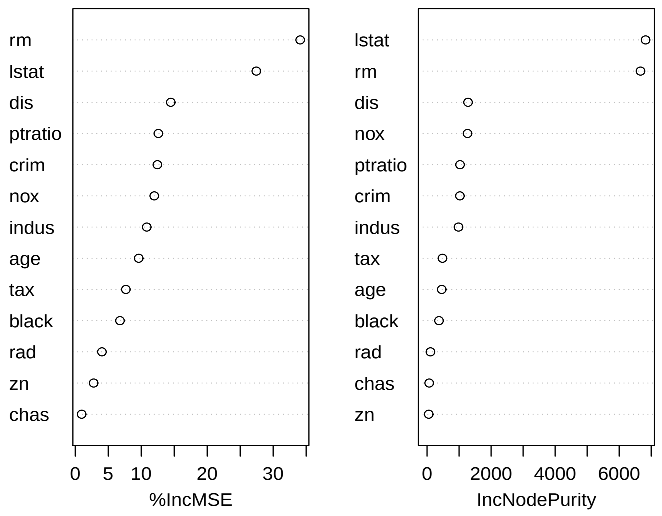

Figure 9: Predictors sorted by importance, rm and lstat, best predictors

```
# boosting.r   p331
library(MASS)      # Boston dataset
library(gbm)       # gbm()

dim(Boston)
# [1] 506  14
# medv is response, p=13 predictors
n = nrow(Boston)
set.seed(1)
train = sample(1:n,n/2)   # 253 train rows

set.seed(1)
boost1=gbm(medv~.,data=Boston[train,],distribution="gaussian",n.trees=5000,interaction.depth=4)
# for categorical response use distribution="bernoulli"
# we limit the depth of each tree to 4 splits
# we want 5000 trees (default is 100 trees)
# interaction.depth is d

names(boost1)
#  [1] "initF"             "fit"               "train.error"       "valid.error"
#  [5] "oobag.improve"     "trees"             "c.splits"          "bag.fraction"
#  [9] "distribution"      "interaction.depth" "n.minobsinnode"    "num.classes"
# [13] "n.trees"           "nTrain"            "train.fraction"    "response.name"
# [17] "shrinkage"         "var.levels"        "var.monotone"      "var.names"
# [21] "var.type"          "verbose"           "data"              "Terms"
# [25] "cv.folds"          "call"              "m"

# lambda -default value
boost1$shrinkage       #[1] 0.001

# importance of predictors
summary(boost1); grid()
#             var     rel.inf
# lstat     lstat 45.96758091
# rm           rm 31.22024973
# dis         dis  6.80540085
# crim       crim  4.06995214
# nox         nox  2.55687077
# ptratio ptratio  2.27547427
# black     black  1.79481300
# age         age  1.65100042
# tax         tax  1.36245454
# indus     indus  1.26933836
# chas       chas  0.80150463
# rad         rad  0.21054835
# zn           zn  0.01481202
# creates a plot
# lstat and rm best predictors

# test MSPE
y.test=Boston[-train,"medv"]       # y values in test set
yhat.boost=predict(boost1,newdata=Boston[-train,],n.trees=5000)
mean((yhat.boost-y.test)^2)        # [1] 11.84445
# similar to Random Forest MSPE

# lambda = 0.2   (shrinkage)
boost2=gbm(medv~.,data=Boston[train,],distribution="gaussian",n.trees=5000,
                                     interaction.depth=4,shrinkage=0.2,verbose=F)

yhat.boost=predict(boost2,newdata=Boston[-train,],n.trees=5000)
mean((yhat.boost-y.test)^2)        # [1] 11.51109

# n.trees in predict function cannot exceed
# n.trees in gbm()
```