# Fitting a regression tree

```
library(MASS)
library(tree)
set.seed(1)

train = sample(1:nrow(Boston), nrow(Boston)/2)
tree.boston=tree(medv~.,Boston,subset=train)
summary(tree.boston)
```

```
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
```
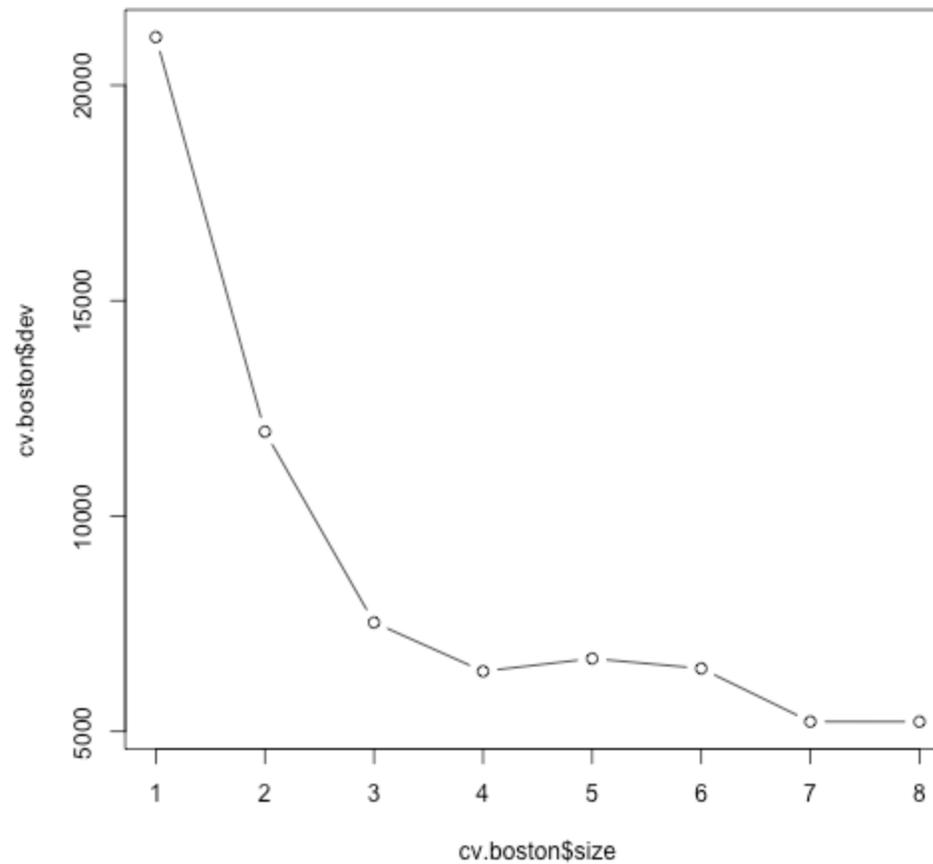
Plotting a regression tree

```
plot(tree.boston)
text(tree.boston,pretty=0)
```
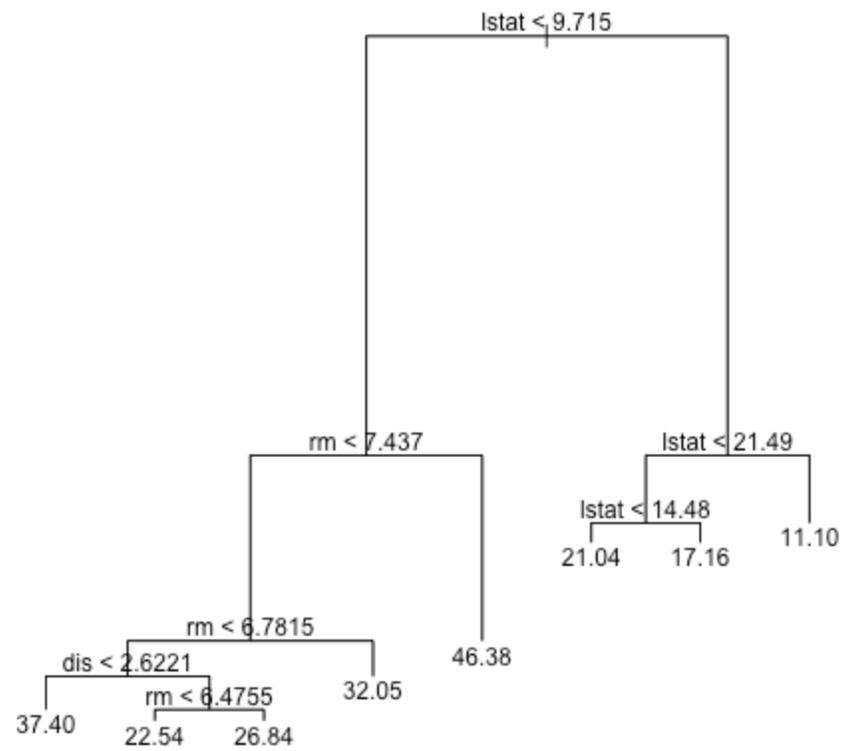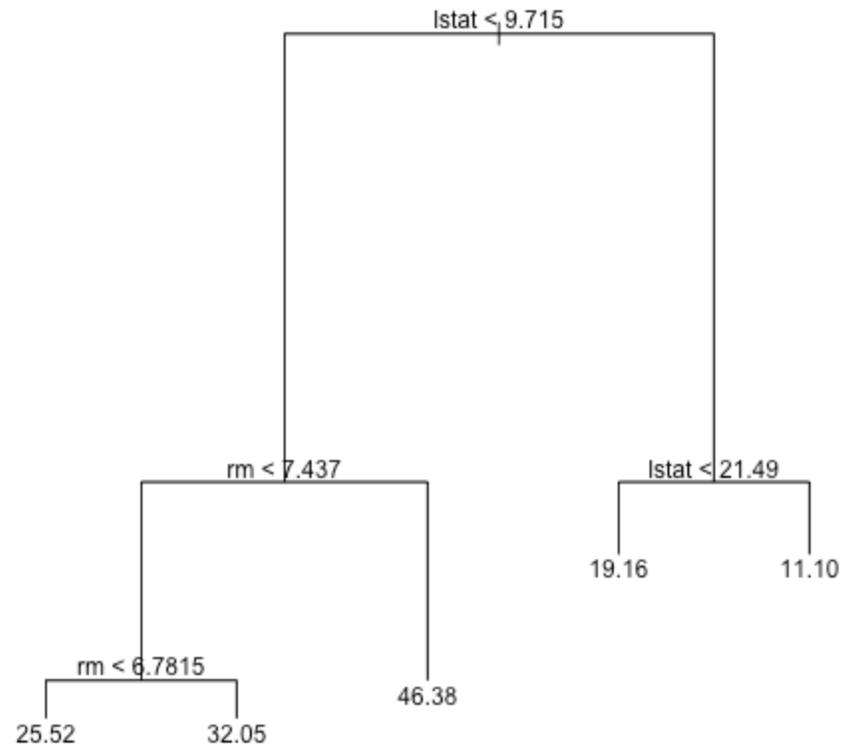
Trees with cross-validation

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

Pruning trees

```
prune.boston=prune.tree(tree.boston,best=5)
plot(prune.boston)
text(prune.boston,pretty=0)
```

Using trees to predict carseat sales, 1

```r
library(tree)
library(ISLR)

High=ifelse(Carseats$Sales<=8,"No","Yes") # Storing the results in object
"High"
Carseats=data.frame(Carseats,High)
head(Carseats)
```

```
  Sales CompPrice Income Advertising Population Price ShelveLoc Age
1  9.50       138     73          11        276   120       Bad  42
2 11.22       111     48          16        260    83      Good  65
3 10.06       113     35          10        269    80    Medium  59
4  7.40       117    100           4        466    97    Medium  55
5  4.15       141     64           3        340   128       Bad  38
6 10.81       124    113          13        501    72       Bad  78
  Education Urban  US High
1        17   Yes Yes  Yes
2        10   Yes Yes  Yes
3        12   Yes Yes  Yes
4        14   Yes Yes   No
5        13   Yes  No   No
6        16    No Yes  Yes
```

```r
tree.carseats=tree(High~.-Sales,Carseats)
summary(tree.carseats)
```

```
Classification tree:
```

```
tree(formula = High ~ . - Sales, data = Carseats)
Variables actually used in tree construction:
[1] "ShelveLoc"    "Price"        "Income"        "CompPrice"     "Population"
[6] "Advertising"  "Age"          "US"
Number of terminal nodes:   27
Residual mean deviance:   0.4575 = 170.7 / 373
Misclassification error rate: 0.09 = 36 / 400
```
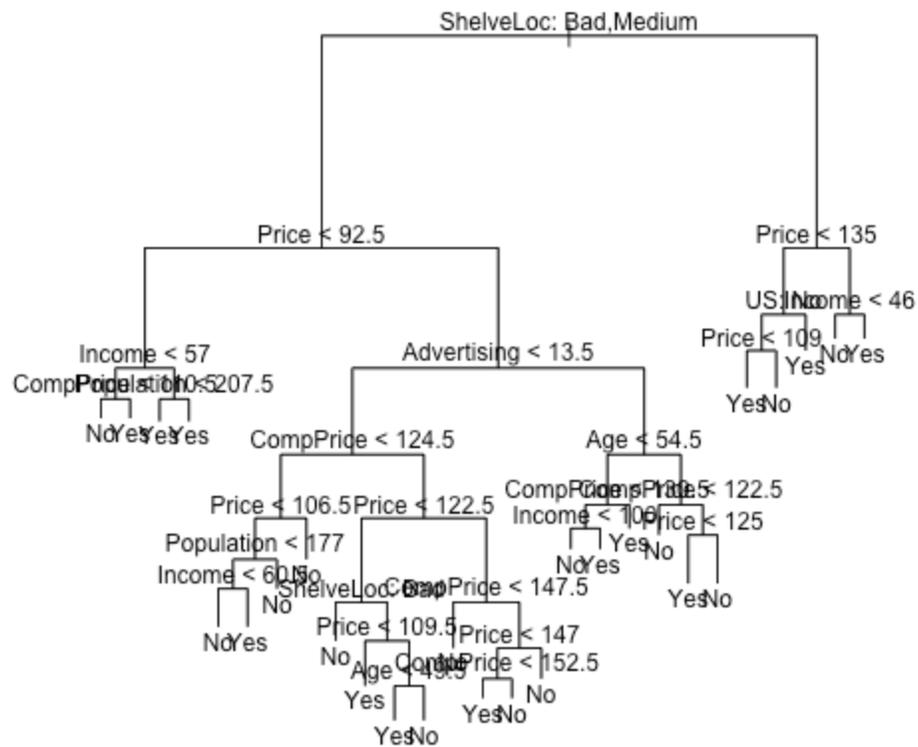
```
plot(tree.carseats)
text(tree.carseats, pretty=0)
```

ShelveLoc: Bad,Medium

Price < 92.5

Price < 135

Income < 57

Advertising < 13.5

US:No Income < 46

CompPrice < 207.5
Population < 207.5

CompPrice < 124.5

Age < 54.5

Price < 109.5
Yes

No Yes

No Yes Yes

Price < 106.5 Price < 122.5

CompPrice < 106.5
Income < 106

Price < 122.5
Price < 125

Yes No

Population < 177

No

No Yes

Income < 60.5
No

ShelveLoc: Bad
Price < 109.5

Comp:Bad
Price < 147.5

No Yes

Yes No

No Yes

Yes No

No Yes

No
Age < 49.5
Comp

Price < 147
No

Price < 152.5
No

Yes

Yes No

Yes No

Using trees to predict carseat sales, 2

```
# Training/testing, with confusion matrix
set.seed(2)
train=sample(1:nrow(Carseats), 200)
Carseats.train=Carseats[train,]
Carseats.train[1:10,]
```

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age
74	12.61	118	90	10	54	104	Good	31
281	2.86	121	86	10	496	145	Bad	51
229	5.40	149	73	13	381	163	Bad	26
67	8.85	127	92	0	508	91	Medium	56
374	5.58	137	71	0	402	116	Medium	78
373	7.80	121	50	0	508	98	Medium	65
51	1.42	99	32	18	341	108	Bad	80
328	6.23	112	38	17	316	104	Medium	80
184	5.32	118	74	6	426	102	Medium	80
216	2.34	116	83	15	170	144	Bad	71

	Education	Urban	US	High
74	11	No	Yes	Yes
281	10	Yes	Yes	No
229	11	No	Yes	No
67	18	Yes	No	Yes
374	17	Yes	No	No
373	11	No	No	No
51	16	Yes	Yes	No
328	16	Yes	Yes	No
184	18	Yes	Yes	No
216	11	Yes	Yes	No

```
Carseats.test=Carseats[-train,]
Carseats.test[1:10,]
```

```
    Sales CompPrice Income Advertising Population Price ShelveLoc Age
1    9.50       138     73          11        276   120       Bad  42
2   11.22       111     48          16        260    83      Good  65
5    4.15       141     64           3        340   128       Bad  38
9    6.54       132    110           0        108   124    Medium  76
10   4.69       132    113           0        131   124    Medium  76
12  11.96       117     94           4        503    94      Good  50
13   3.98       122     35           2        393   136    Medium  62
15  11.17       107    117          11        148   118      Good  52
16   8.71       149     95           5        400   144    Medium  76
18  12.29       147     74          13        251   131      Good  52
    Education Urban  US High
1          17   Yes Yes  Yes
2          10   Yes Yes  Yes
5          13   Yes  No   No
9          10    No  No   No
10         17    No Yes   No
12         13   Yes Yes  Yes
13         18   Yes  No   No
15         18   Yes Yes  Yes
16         18    No  No  Yes
18         10   Yes Yes  Yes
```

```
High.test=High[-train]
tree.carseats=tree(High~.-Sales,Carseats,subset=train)
tree.pred=predict(tree.carseats,Carseats.test,type="class")
table(tree.pred,High.test) # Confusion matrix
```

```
         High.test
tree.pred No Yes
      No  86  27
      Yes 30  57
```

Using trees to predict carseat sales, 3

```
# Cross-validation with error
cv.carseats=cv.tree(tree.carseats,FUN=prune.misclass)
names(cv.carseats)
```

```
[1] "size"    "dev"     "k"        "method"
```

```
cv.carseats
```

```
$size
[1] 19 17 14 13  9  7  3  2  1

$dev
[1] 53 53 50 50 48 51 67 66 80

$k
[1]        -Inf  0.0000000  0.6666667  1.0000000  1.7500000  2.0000000
[7]   4.2500000  5.0000000 23.0000000

$method
[1] "misclass"

attr(,"class")
[1] "prune"          "tree.sequence"
```
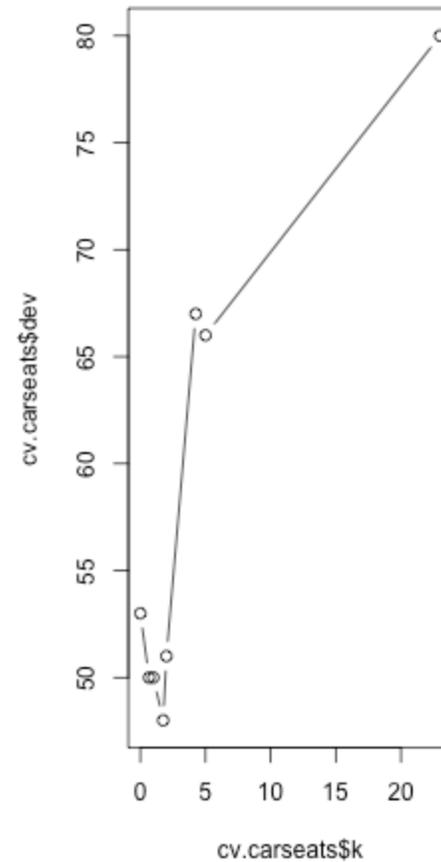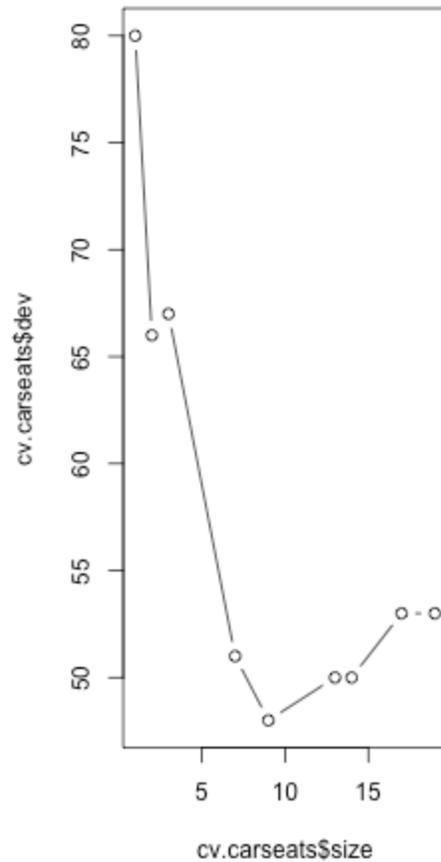
```
par(mfrow=c(1,2)) # Let's split the graphics screen
```

```
plot(cv.carseats$size,cv.carseats$dev,type="b")
plot(cv.carseats$k,cv.carseats$dev,type="b")
```



```
######
# Now prune tree back to 9 nodes
prune.carseats=prune.misclass(tree.carseats,best=9)
plot(prune.carseats)
text(prune.carseats,pretty=0)
```

```
tree.pred=predict(prune.carseats,Carseats.test,type="class")
table(tree.pred,High.test)
```

```
          High.test
tree.pred No  Yes
      No   94   24
      Yes  22   60
```
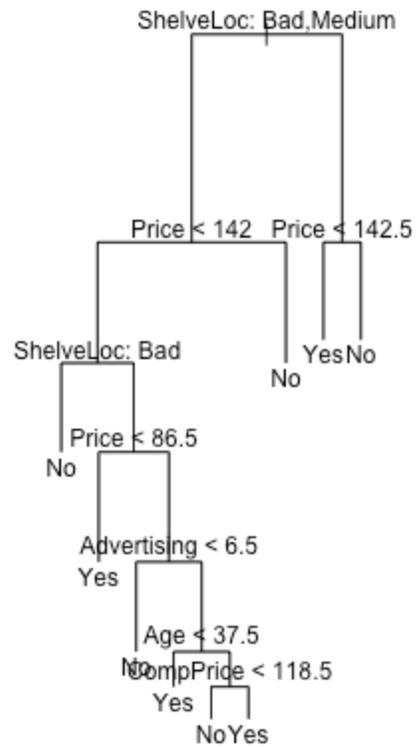
Bagging

```r
library(randomForest)
library(MASS)
set.seed(1)

train = sample(1:nrow(Boston), nrow(Boston)/2)
boston.test=Boston[-train,"medv"]

bag.boston=randomForest(medv~.,data=Boston,subset=train,
mtry=13,importance=TRUE)
bag.boston
```

```
Call:
 randomForest(formula = medv ~ ., data = Boston, mtry = 13, importance =
TRUE,        subset = train)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 13

        Mean of squared residuals: 11.08966
                  % Var explained: 86.57
```
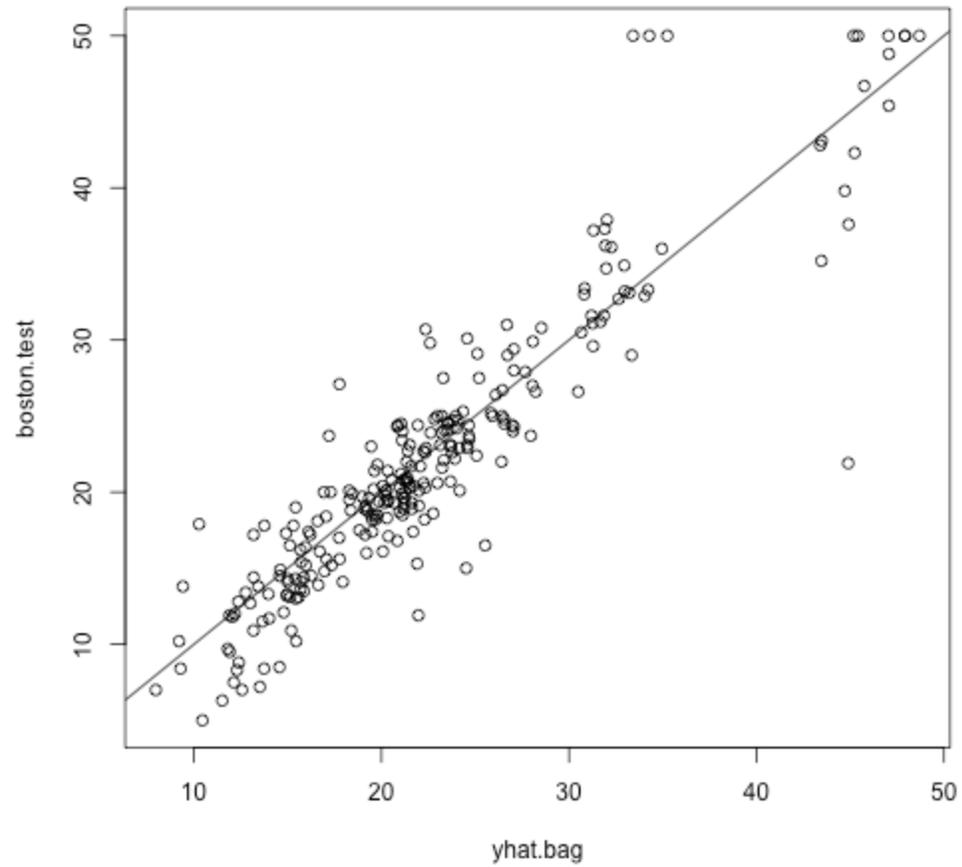
Predicting with a bagging model

```
yhat.bag = predict(bag.boston,newdata=Boston[-train,])
plot(yhat.bag, boston.test)
abline(0,1)
```

```
mean((yhat.bag-boston.test)^2)
```

```
[1] 13.33831
```

Bagging with a different number of trees

```
bag.boston=randomForest(medv~.,data=Boston,subset=train,
                        mtry=13,ntree=25)
yhat.bag = predict(bag.boston,newdata=Boston[-train,])
mean((yhat.bag-boston.test)^2)
```

```
[1] 15.96786
```

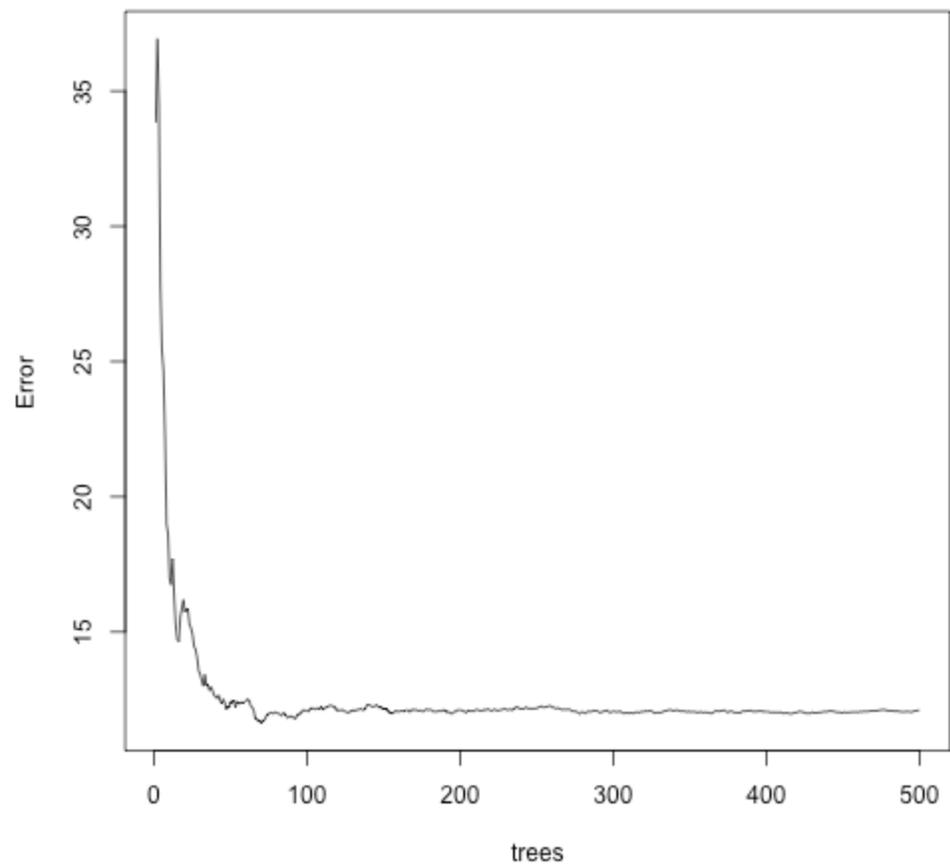# Random forest 1

```
set.seed(1)
rf.boston=randomForest(medv~.,data=Boston,
                       subset=train,mtry=6,importance=TRUE)
rf.boston
```

```
Call:
 randomForest(formula = medv ~ ., data = Boston, mtry = 6, importance =
TRUE,        subset = train)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 6

          Mean of squared residuals: 12.09928
                    % Var explained: 85.35
```

```
plot(rf.boston)
```

rf.boston

Random forest 2

```
yhat.rf = predict(rf.boston,newdata=Boston[-train,])
mean((yhat.rf-boston.test)^2)
```

```
[1] 11.48022
```

Random forest 3

```
importance(rf.boston)
```

```
          %IncMSE  IncNodePurity
crim     12.547772     1094.65382
zn        1.375489       64.40060
indus     9.304258     1086.09103
chas      2.518766       76.36804
nox      12.835614     1008.73703
rm       31.646147     6705.02638
age       9.970243      575.13702
dis      12.774430     1351.01978
rad       3.911852       93.78200
tax       7.624043      453.19472
ptratio  12.008194      919.06760
black     7.376024      358.96935
lstat    27.666896     6927.98475
```

```
varImpPlot(rf.boston)
```