STAT 441: Lecture 17
Classification trees

Classification à la old botanic identification

Venables and Ripley, Chapter 9

Classification trees

Use the original variables rather than their linear combinations or other transformations. A tree-like decision scheme is constructed, upside down; the training set is split acoording to the values of classifiers. Usually, only binary splits are allowed, for simplicity. Splits are chosen to optimize some splitting criterion; as a rule, full search over all possibilities is not possible, so the method looks ahead only few steps, usually one (this is similar to the way how, for instance, programs playing chess are implemented).

The splitting is continued until the minimal prescribed size of the leaf is reached; or the leafs are homogeneous enough (the deviance is small compared to the initial one). The trees are then often adjusted subjectively - *pruning*; cross-validation error rate may be useful in this.

Properties:
- very interpretable
- very good predictive power
- somewhat subjective
- not good for handling linear combinations
- very good for missing data

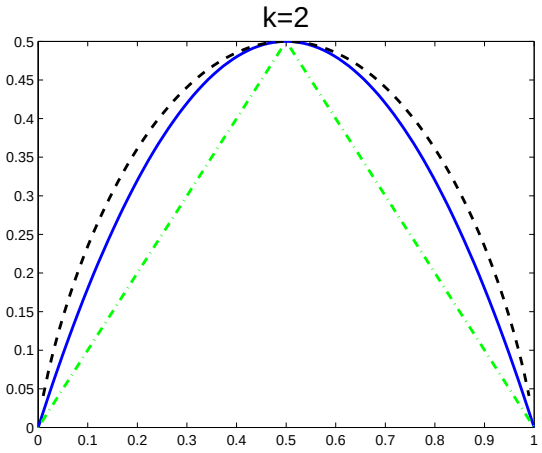
  (capable of producing so-called surrogate splits)

Criteria for splitting I: impurity

Apparent error rate is again not suitable. An i-th node of tree can be seen as giving (unknown) conditional probabilities p_{ik}, estimated by $n_{ik}/n_i = n_{ik}/\sum_k n_{ik}$; the majority of implementations choose splits to maximize the average impurity decrease: the impurity of a node is quantified by some "impurity index", the most common ones are

$$\begin{array}{ll} \text{Entropy} & -\sum_k p_{ik} \log p_{ik} \\ \text{Gini index} & 1 - \sum_k p_{ik}^2 \end{array}$$

- a good measure of this kind is supposed to be minimal (0), if one of the p_{ik}'s is 1 (and others 0), and maximal, if the distribution is uniform (all p_{ik} are the same).



A graph of various impurity criteria for $p_1 = p$, $p_2 = 1 - p$

Impurity decrease

"Impurity decrease" means the impurities of two nodes after the split are subtracted from the impurity of the node before the split; "average" means that the impurities of the subtracted notes are before that each multiplied by the relative proportion of all the objects in the node to all objects in the parent node.

That is, if node s, comprising n_s objects in total, is split into nodes t and u, with n_t and n_u objects in total respectively (that is, $n_s = n_t + n_u$, $n_t = \sum_k n_{tk}$, $n_u = \sum_k n_{uk}$, $n_s = \sum_k n_{sk}$) then we are maximizing

$$i_s - \frac{n_t}{n_s} i_t - \frac{n_u}{n_s} i_u$$

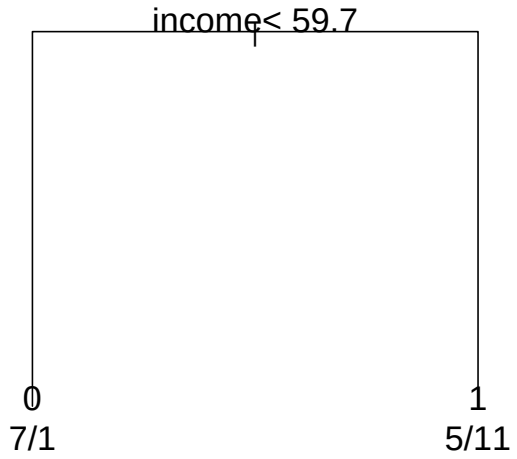
For instance, with entropy as impurity measure this is equal to

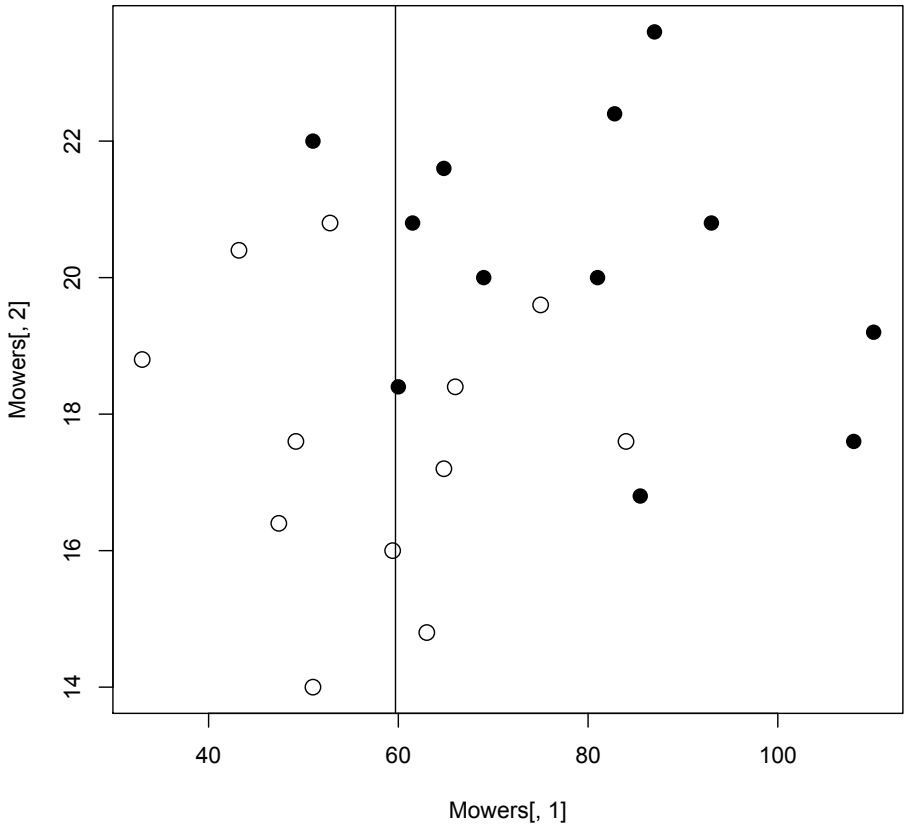
$$-\sum_k \left(\frac{n_{sk}}{n_s} \log \frac{n_{sk}}{n_s} \right) + \frac{n_t}{n_s} \sum_k \left(\frac{n_{tk}}{n_t} \log \frac{n_{tk}}{n_t} \right) + \frac{n_u}{n_s} \sum_k \left(\frac{n_{uk}}{n_u} \log \frac{n_{uk}}{n_u} \right)$$

Owners of riding mowers: a coarse tree

...also called a *stump*.

```
> library(rpart)
> mowtree=rpart(factor(riding)~income+lot,data=Mowers)
> plot(mowtree,margin=0.1)
> text(mowtree,use.n=T,cex=1.3)
```





Which impurity was used?

Average impurity decrease

Entropy:

```
> ent = function(x) sum(-x*log(x))
> ent(c(1/2,1/2))
[1] 0.6931472
> 8/24*ent(c(7/8,1/8))+16/24*ent(c(5/16,11/16))
[1] 0.5396476
```

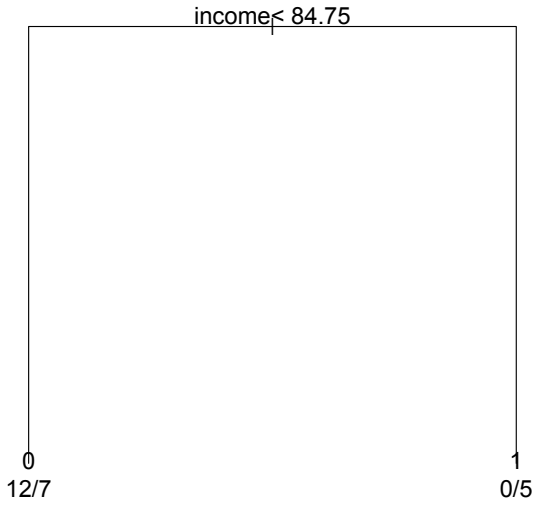
Gini:

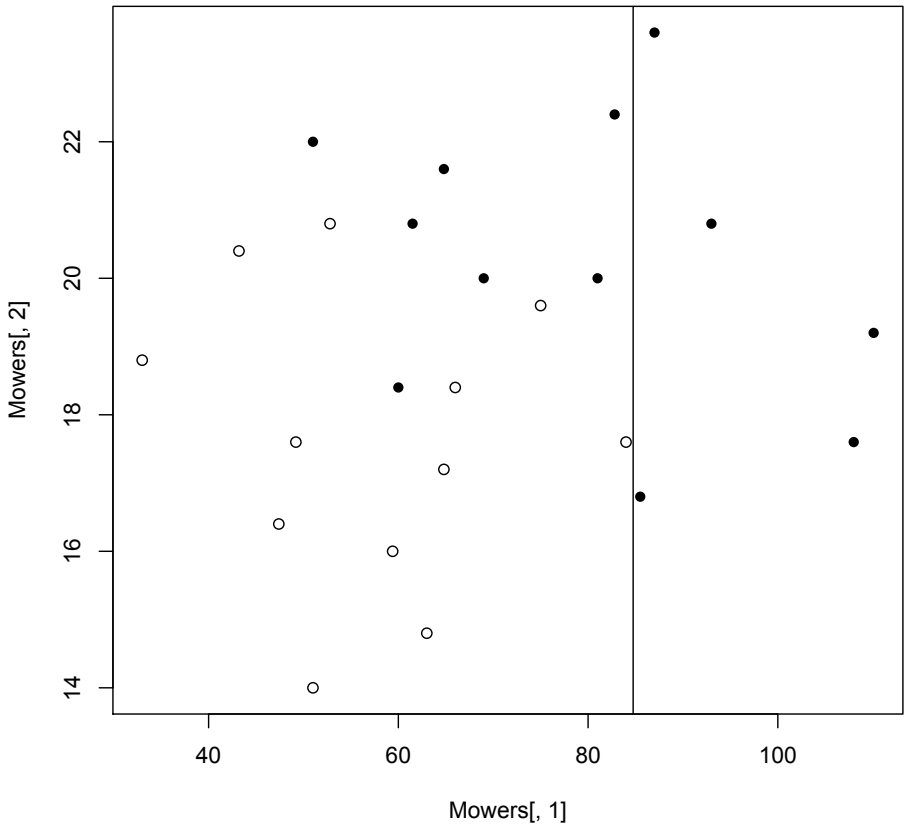
```
> gini = function(x) 1-sum(x*x)
> gini(c(1/2,1/2))
[1] 0.5
> 8/24*gini(c(7/8,1/8))+16/24*gini(c(5/16,11/16))
[1] 0.359375
```

Was some of these used in the picture above?

Owners of riding mowers: another coarse tree

```
> library(tree)
> mowtree=tree(factor(riding)~income+lot,data=Mowers
+ parms=list(split="information"),minsplit=1,maxdepth=1)
> plot(mowtree,margin=0.1)
> text(mowtree,use.n=T,cex=1.3)
```





Criteria for splitting II: likelihood and deviance

Thinking in likelihood terms, the likelihood of the tree is

$$\prod_{i}\prod_{k} p_{ik}^{n_{ik}}$$

where i runs over *leaves*, the terminal nodes, and k over classes

This leads to the deviance

$$D = \sum_{i} D_i = \sum_{i}\left(-2\sum_{k} n_{ik}\log p_{ik}\right)$$

which is the sum, over the leaves, of deviances

$$D_i = -2\sum_{k} n_{ik}\log p_{ik}$$

Likelihood is still maximized after taking logs, "-" makes it a minimization, and the factor 2 is there just for traditional reasons - it does not change anything.

The reduction in deviance

If node s is split into nodes t and u, we observe the reduction in the deviance

$$
\begin{aligned}
&D_s - D_t - D_u \\
&\quad = -2\sum_k n_{sk} \log p_{sk} + 2\sum_k n_{tk} \log p_{tk} + 2\sum_k n_{uk} \log p_{uk} \\
&\quad = -2\sum_k (n_{tk}+n_{uk}) \log p_{sk} + 2\sum_k n_{tk} \log p_{tk} + 2\sum_k n_{uk} \log p_{uk} \\
&\quad = 2\sum_k \left(n_{tk} \log \frac{p_{tk}}{p_{sk}} + n_{uk} \log \frac{p_{uk}}{p_{sk}} \right)
\end{aligned}
$$

It is equivalent to entropy as impurity

After replacing the probabilities by their estimates

$$\hat{p}_{tk} = \frac{n_{tk}}{n_t}; \quad \hat{p}_{uk} = \frac{n_{uk}}{n_u}; \quad \hat{p}_{sk} = \frac{n_{tk} + n_{uk}}{n_t + n_u} = \frac{n_{sk}}{n_s};$$

where $n_t = \sum_k n_{tk}$, $n_u = \sum_k n_{uk}$, $n_s = \sum_k n_{sk}$,

we obtain that the reduction in deviance is equal to

$$2 \sum_k \big(n_{tk} \log n_{tk} + n_{uk} \log n_{uk} - n_{sk} \log n_{sk}$$

$$+ n_s \log n_s - n_u \log n_u - n_t \log n_t \big)$$

$$= 2 \sum_k \left(n_{tk} \log \frac{n_{tk}}{n_t} + n_{uk} \log \frac{n_{uk}}{n_u} - n_{sk} \log \frac{n_{sk}}{n_s} \right)$$

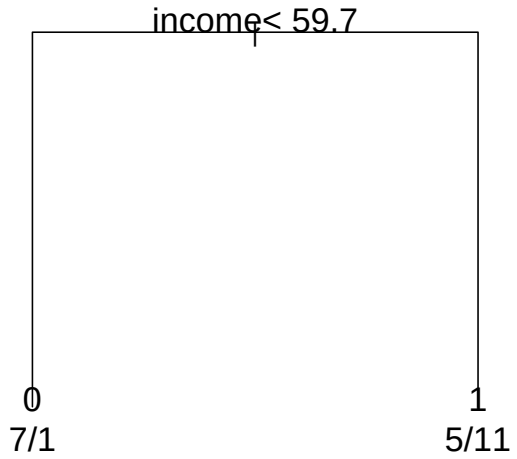
$$= 2 n_s \left(\frac{n_t}{n_s} \sum_k \left(\frac{n_{tk}}{n_t} \log \frac{n_{tk}}{n_t} \right) + \frac{n_u}{n_s} \sum_k \left(\frac{n_{uk}}{n_u} \log \frac{n_{uk}}{n_u} \right) - \sum_k \left(\frac{n_{sk}}{n_s} \log \frac{n_{sk}}{n_s} \right) \right)$$

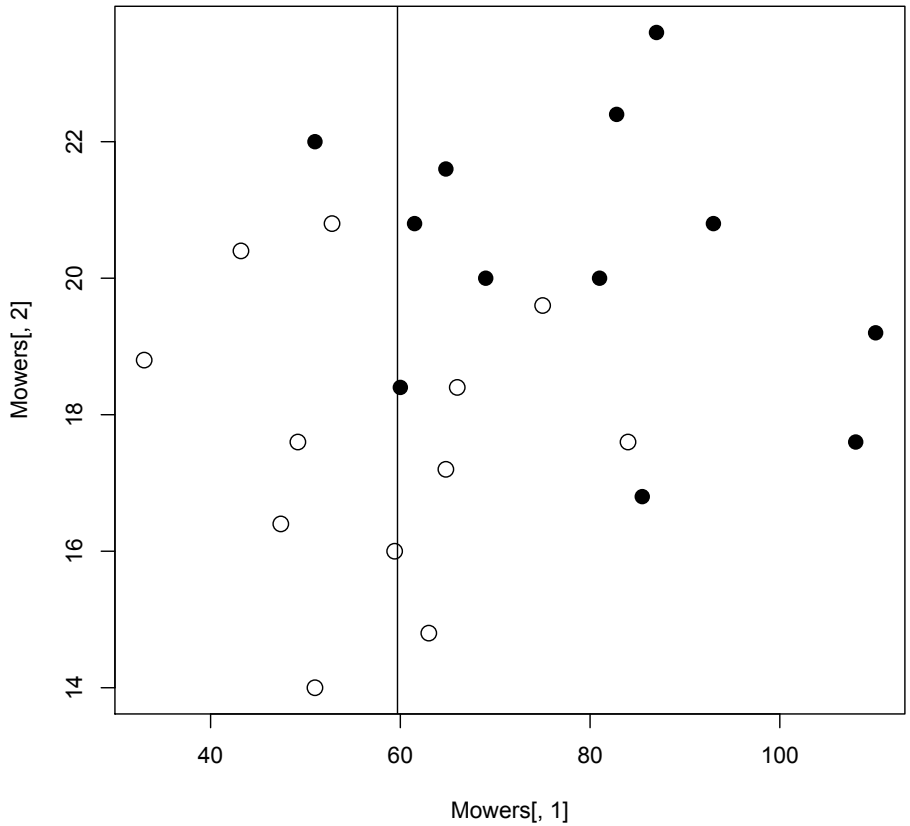
which shows its maximization is equivalent to the maximization of the average impurity decrease for the entropy impurity; it prefers nodes with large size.

Owners of riding mowers: coarse tree

...also called a *stump*.

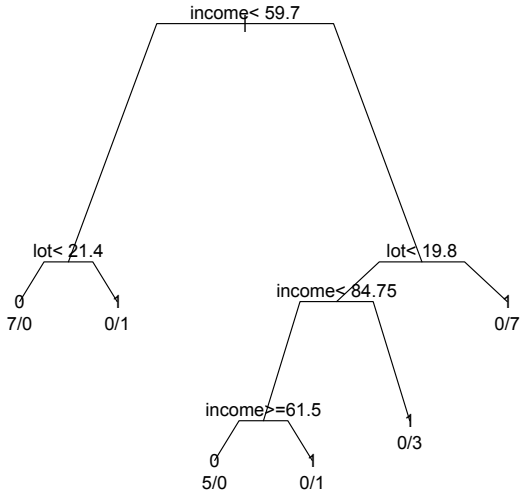
```
> library(rpart)
> mowtree=rpart(factor(riding)~income+lot,data=Mowers)
> plot(mowtree,margin=0.1)
> text(mowtree,use.n=T,cex=1.3)
```

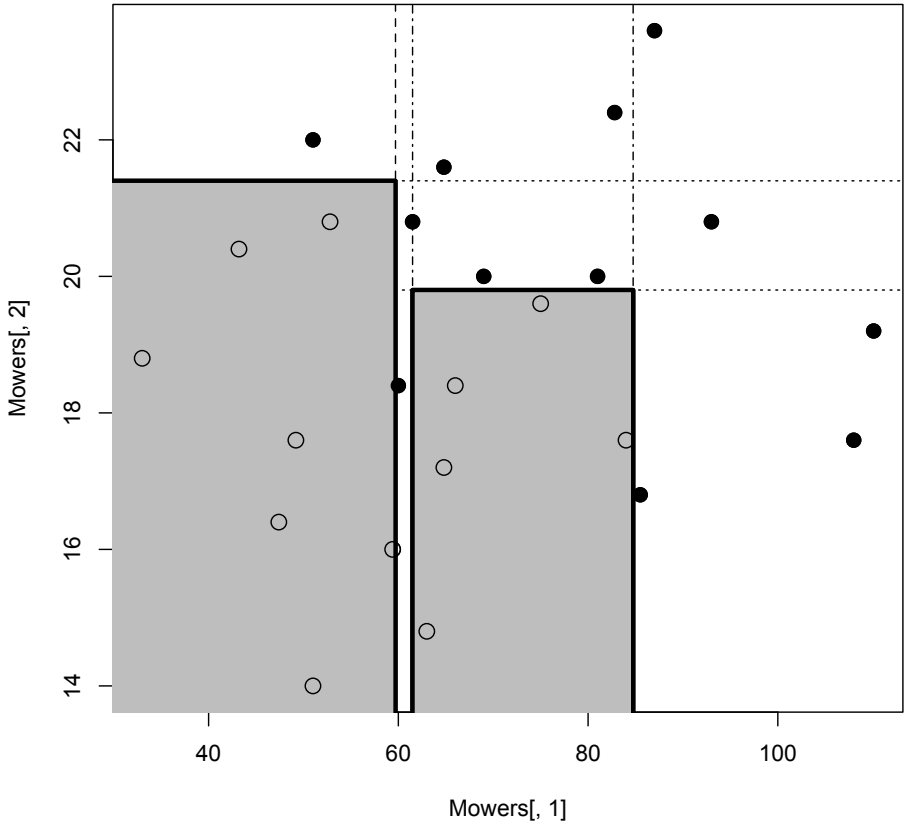




Owners of riding mowers: fine tree

```
> mowtr=rpart(factor(riding)~income+lot,data=Mowers,minsplit=1)
> plot(mowtr,branch=0.5)
> text(mowtr,use.n=T)
```





Criteria for pruning

The criterion for splitting, R, often leads to a maximal tree, the tree with the maximal number of splits. This is not always optimal, and in particular, it can overfit. Thus we usually reduce the tree by cutting of branches - *pruning*.

Given $\alpha \geqslant 0$, there is always a tree (with minimal size) minimizing

$R_\alpha = R + \alpha$ size

The trees minimizing this for various α are nested, so pruning can indeed lead us to one of those - if we know what α we would like.

The selection of α

One possibility is to try some version of cross-validation: 10-fold cross-validation divides data randomly to 10 equally-sized groups, evaluates the quality of prediction on each group (the rule determined from the remaining 9 groups), and then computes the aggregate error rate, which we can plot against α and see.

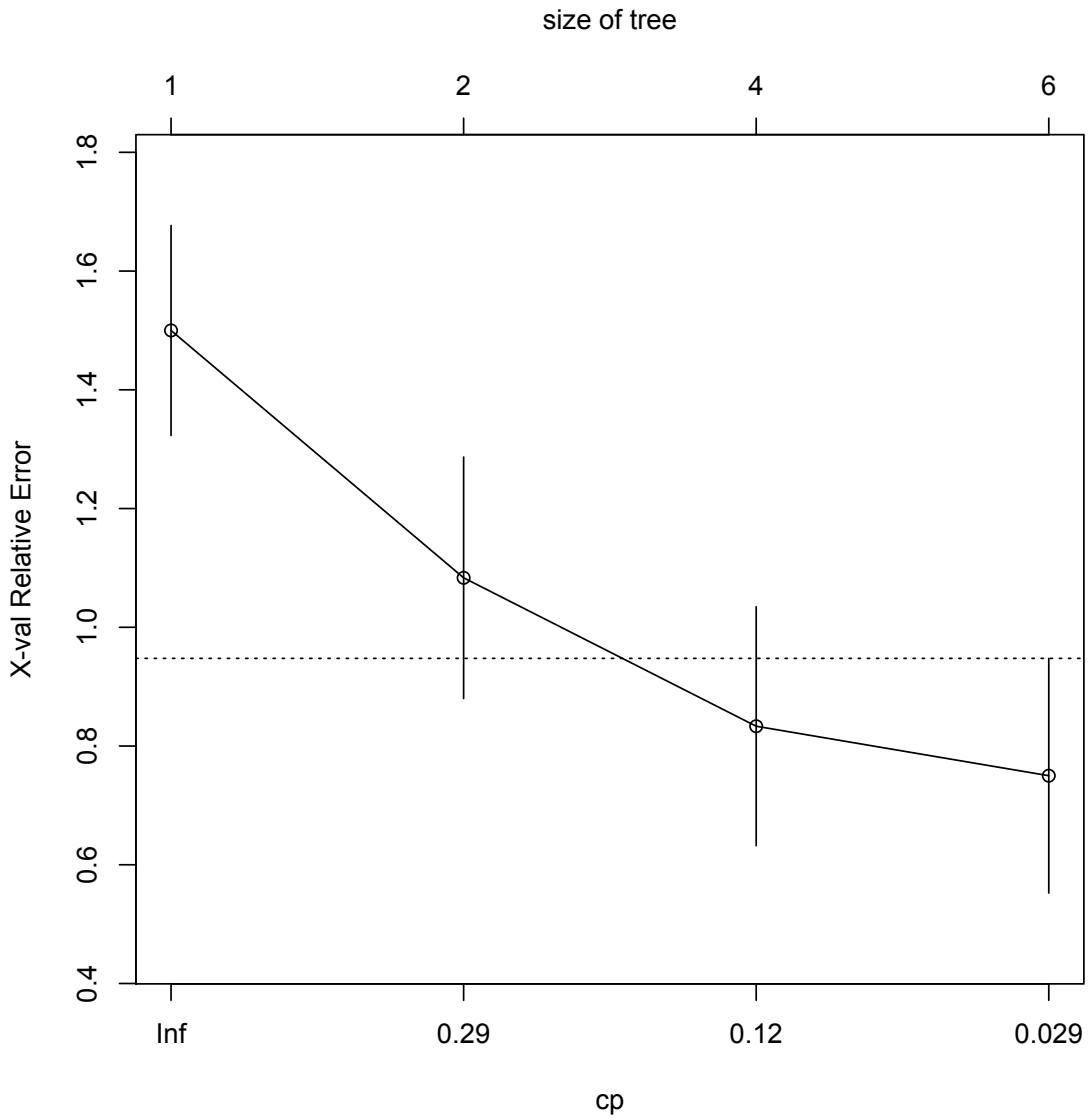
We would like to select α that minimizes the cross-validated error, but this often leads to the maximal tree. Then, another rule may be handy, the 1-SE rule: it selects the α which yields the largest error within 1 standard deviation of the minimal one.

In the `library(rpart)` implementation, `cp` is α divided by R_0. The dashed line on the complexity plot indicates the distance of one standard deviation ftom the minimal error.

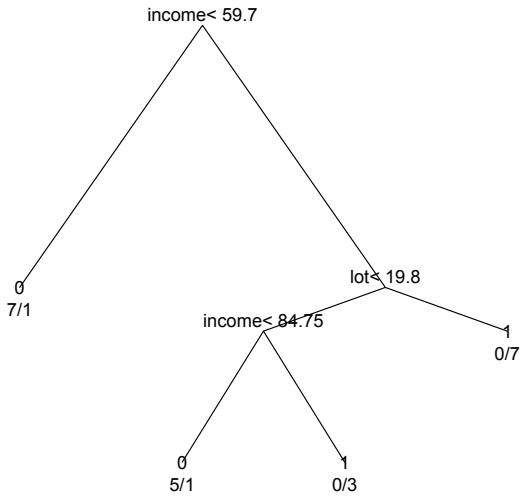
Complexity plot

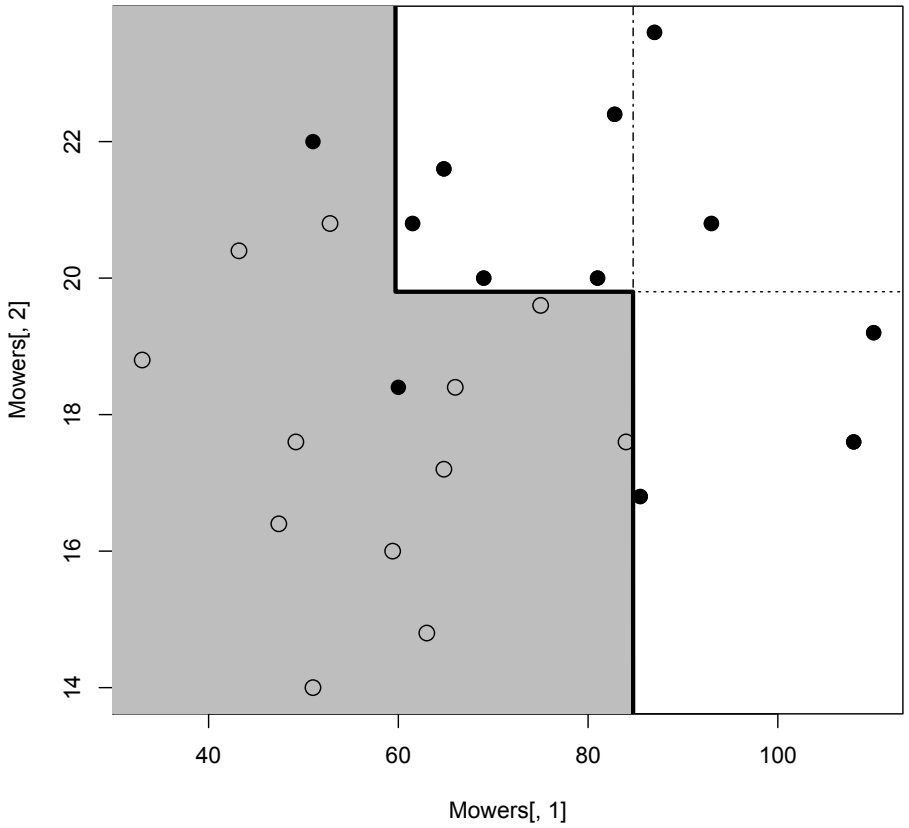
```
> plotcp(mowtr)
> printcp(mowtr)
...
        CP nsplit rel error  xerror    xstd
1 0.500000      0   1.00000 1.50000 0.17678
2 0.166667      1   0.50000 1.00000 0.20412
3 0.083333      3   0.16667 0.75000 0.19764
4 0.010000      5   0.00000 0.66667 0.19245
> mowtree=rpart(factor(riding)~income+lot,
+ data=Mowers,minsplit=1,cp=0.12)
> plot(mowtree,branch=0,margin=0.1)
> text(mowtree)
```

The plot


size of tree
1
2
4
6
X-val Relative Error
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
Inf
0.29
0.12
0.029
cp

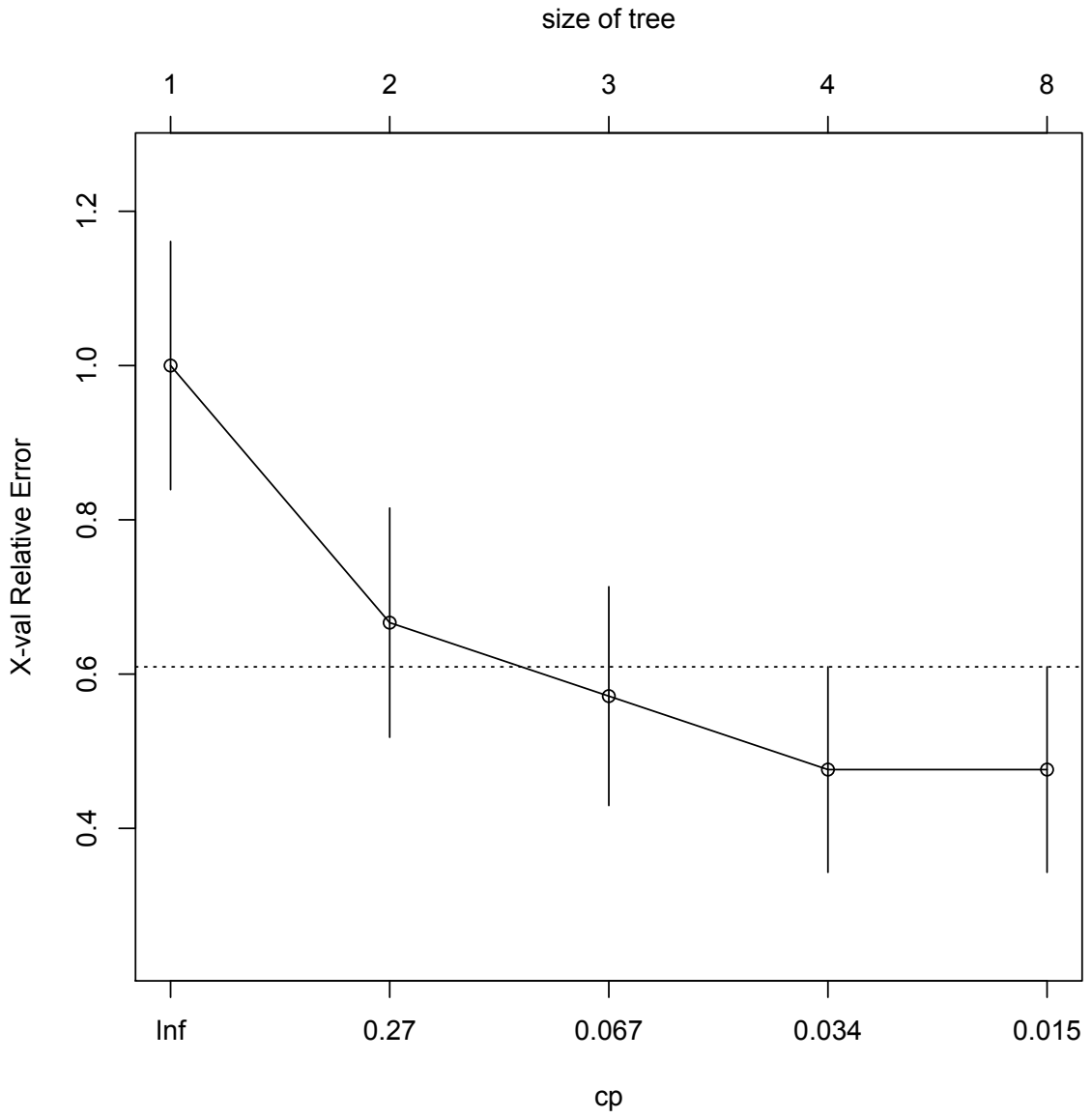
Owners of riding mowers: pruned tree





Bank data: complexity plot...

```
> bankfull=rpart(factor(k)~.,data=Bank,minsplit=1)
> plotcp(bankfull)
```

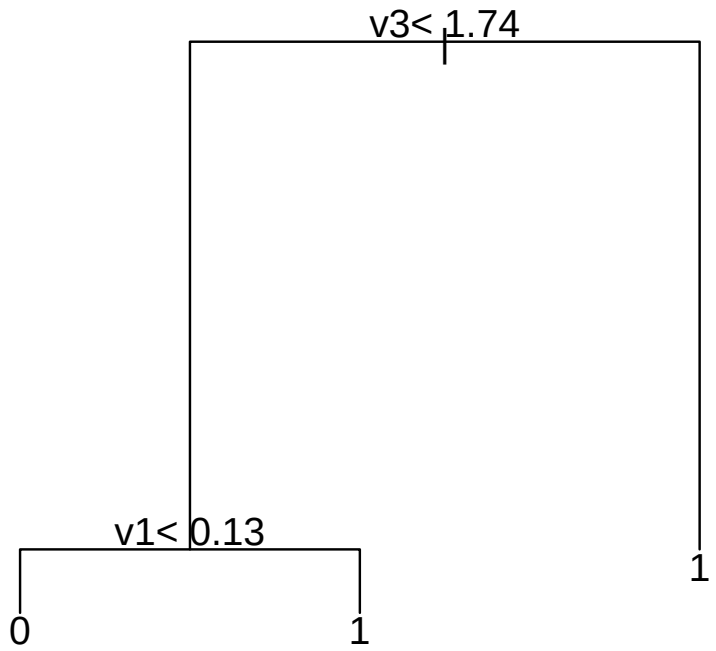


...and printout

```
> printcp(bankfull)

Classification tree:
rpart(formula = factor(k) ~ ., data = Bank, minsplit = 1)

Variables actually used in tree construction:
[1] v1 v3 v4

Root node error: 21/46 = 0.45652

n= 46

        CP nsplit rel error  xerror    xstd
1 0.761905      0  1.000000 1.00000 0.16087
2 0.095238      1  0.238095 0.57143 0.14182
3 0.047619      2  0.142857 0.47619 0.13321
4 0.023810      3  0.095238 0.42857 0.12812
5 0.010000      7  0.000000 0.38095 0.12242
```

Bank data: the resulting tree

```
> banktree=prune(bankfull,cp=0.067)
> plot(banktree,margin=0.1)
> text(banktree)
```

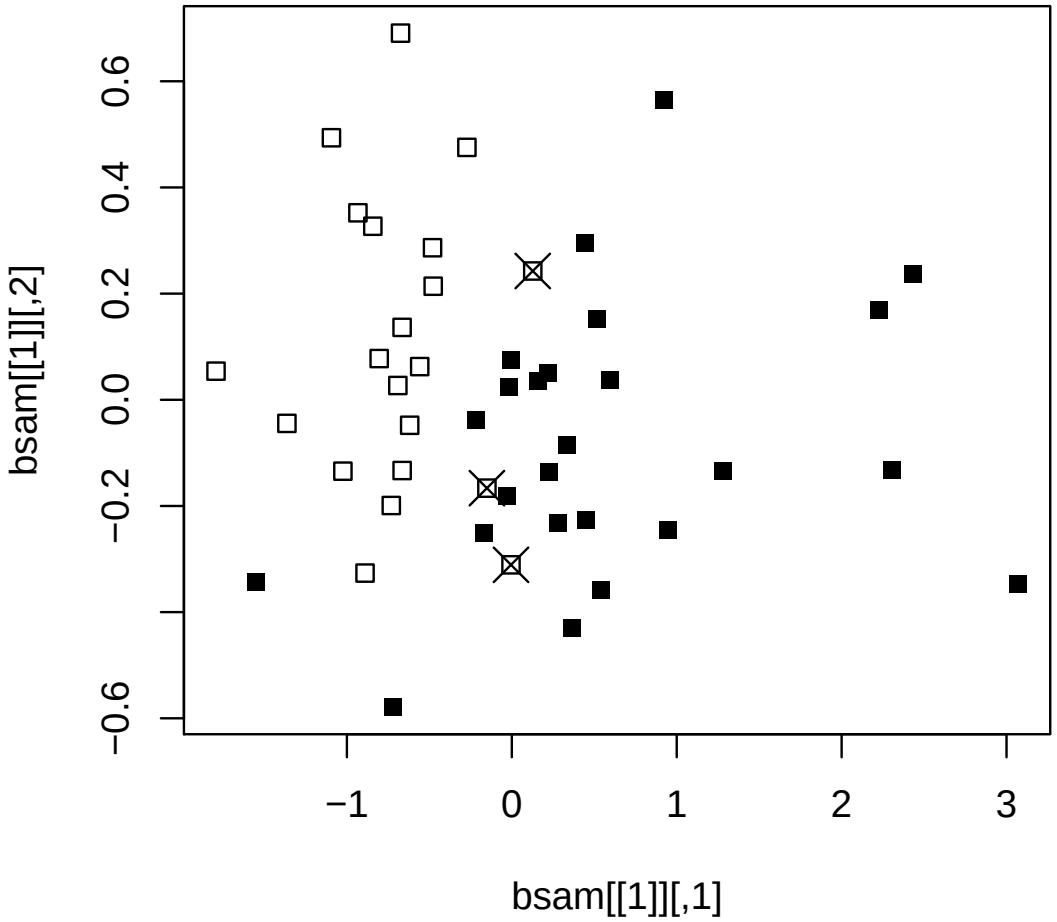


Note

For both datasets, we did not choose the pruned tree with minimal `xerror` based on 10-fold cross-validation, but the one chosen according to 1-SE rule: the largest within 1 standard deviation of the minimum (the largest under the dashed line): $0.38095 + 0.12242 = 0.50337$ for the bank data.

Bank data: Sammon map and tree predictor

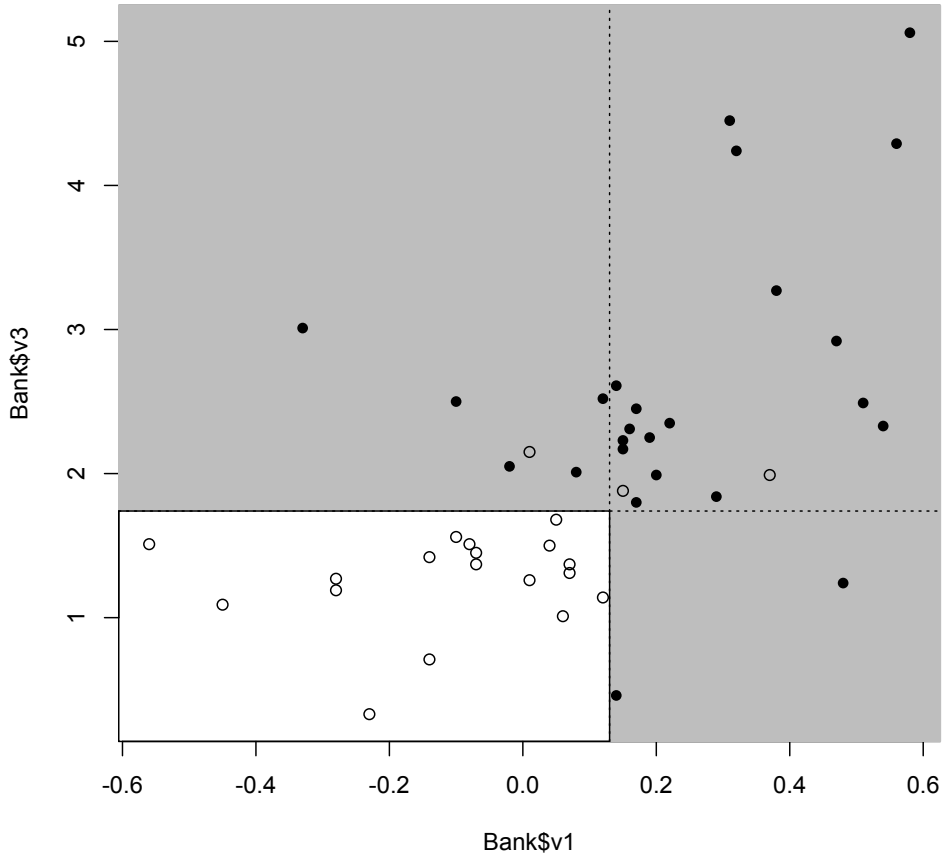
```
> plot(bsam[[1]],pch=15*Bank$k)
> wrong=predict(banktree,type='class') != Bank$k
> points(bsam[[1]][wrong,],pch=4,cex=2)
```



Bank data: better plot here

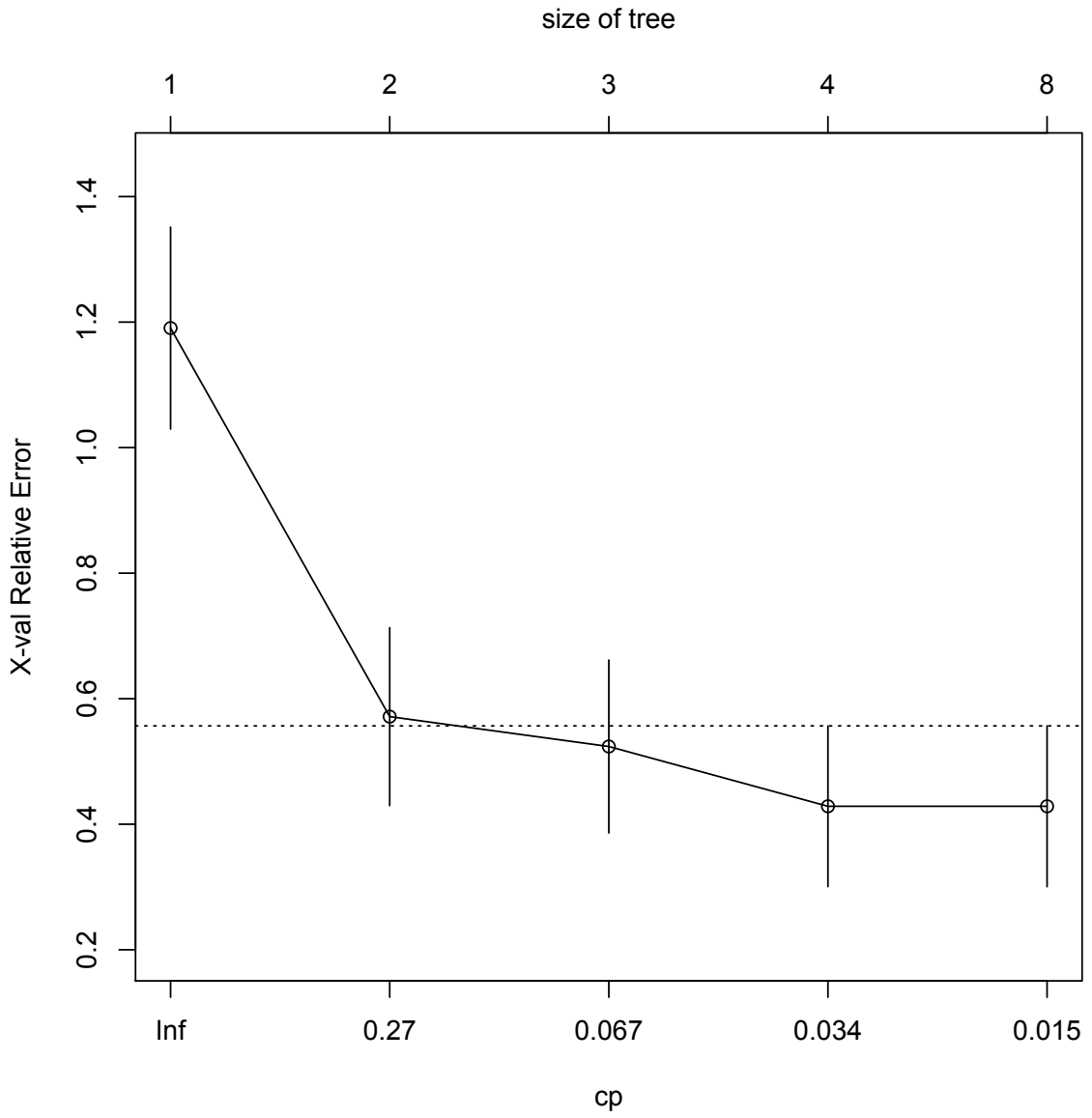
...valid in this particular situation; also some beautifications.

```
> plot(Bank$v1,Bank$v3,pch=15*Bank$k+1,type='n')
> polygon(c(-0.7,0.13,0.13,0.7,0.7,-0.7),c(1.74,1.74,0,0,6,6),col='gray')
> points(Bank$v1,Bank$v3,pch=15*Bank$k+1)
> abline(h=1.74,lty=3)
> abline(v=0.13,lty=3)
```



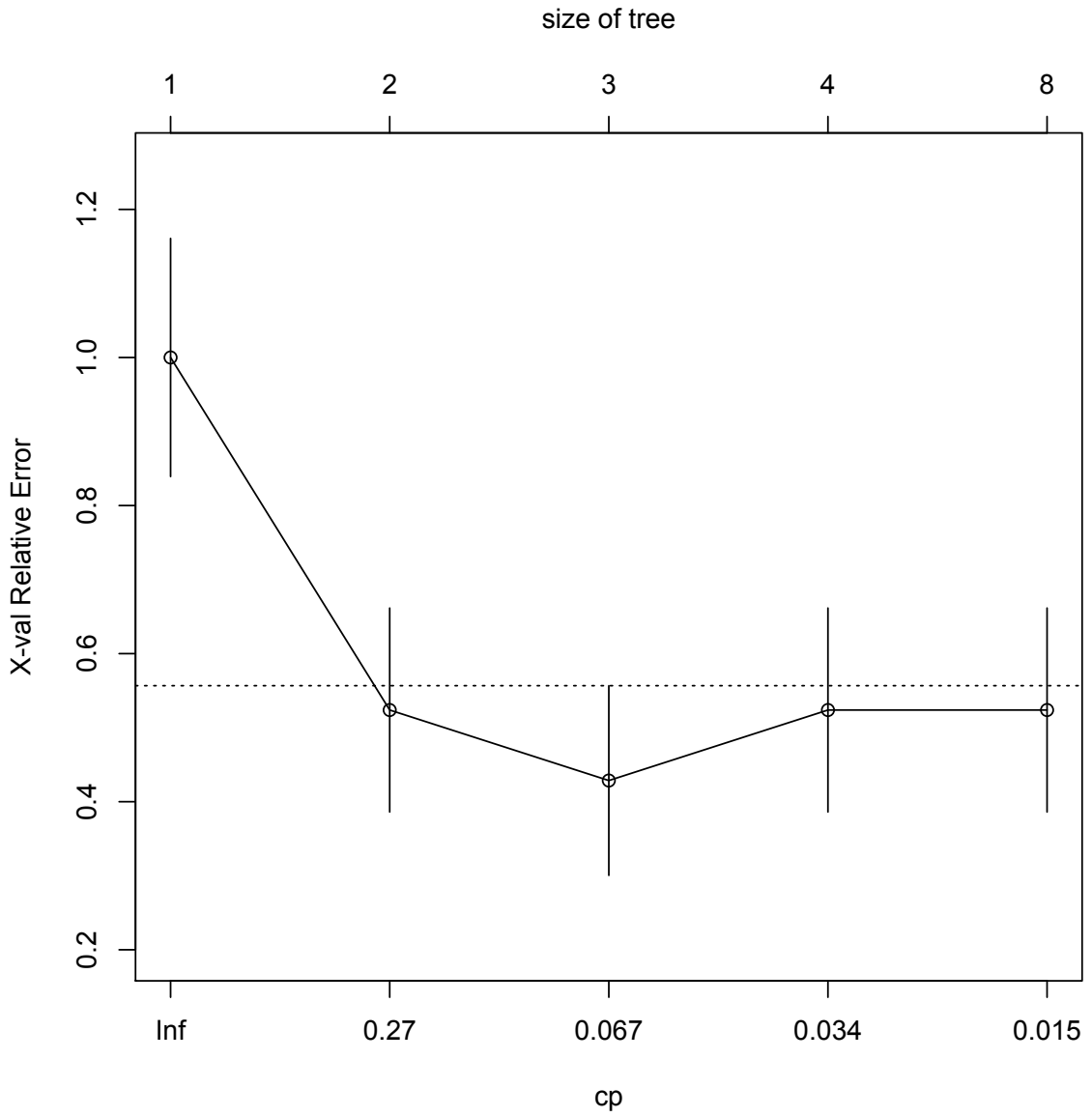
Beware, however

```
> bankfull=rpart(factor(k)~.,data=Bank,minsplit=1)
> plotcp(bankfull)
```



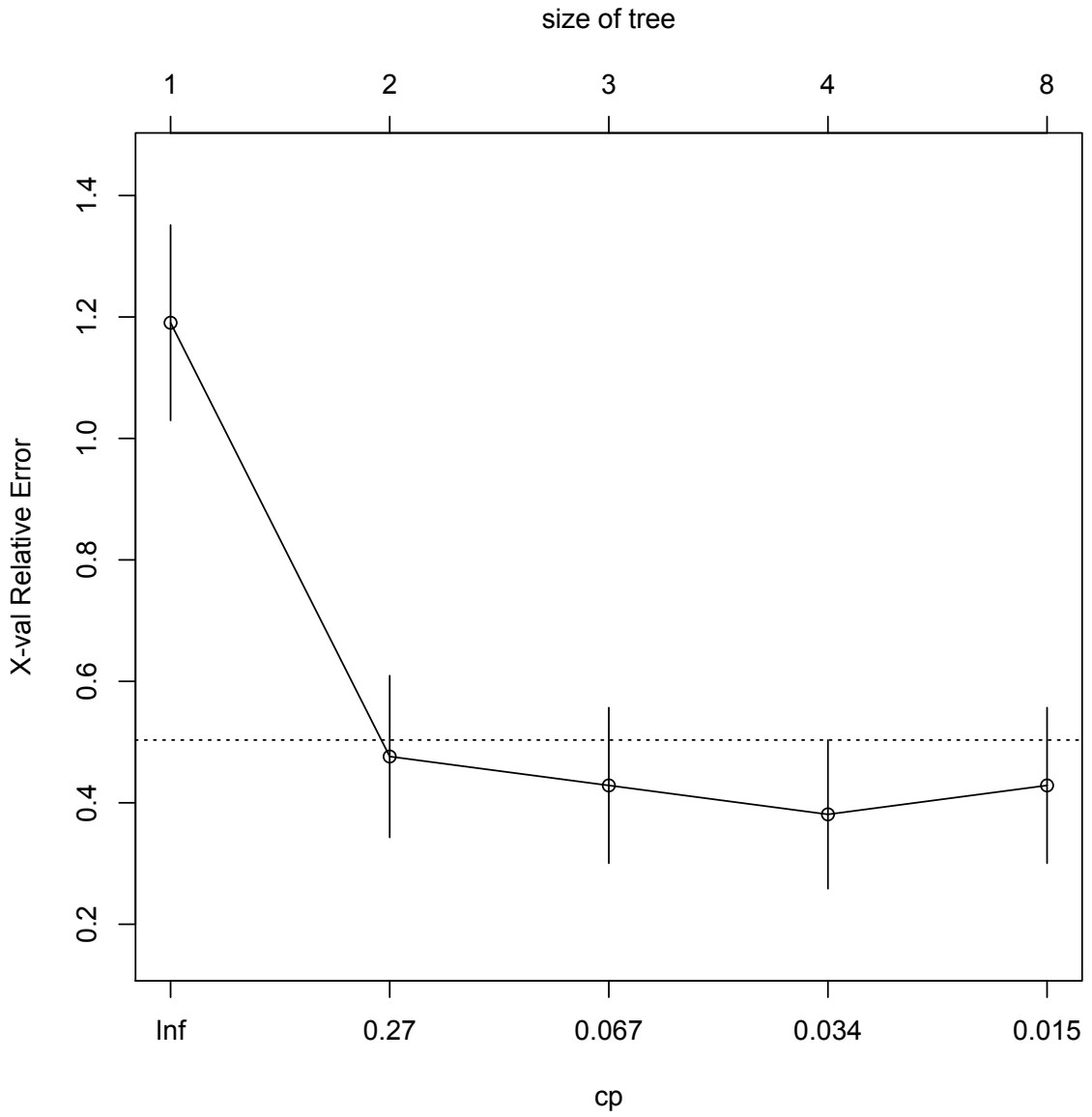
At least the final tree seems to remain...

```
> bankfull=rpart(factor(k)~.,data=Bank,minsplit=1)
> plotcp(bankfull)
```



...most of the time...

```
> bankfull=rpart(factor(k)~.,data=Bank,minsplit=1)
> plotcp(bankfull)
```



...maybe...

```
> bankfull=rpart(factor(k)~.,data=Bank,minsplit=1)
> plotcp(bankfull)
```