

Klemelä, Jussi; Klinke, Sigbert; Sofyan, Hizir

Working Paper

Classification and regression trees

SFB 373 Discussion Paper, No. 2000,62

Provided in Cooperation with:

Collaborative Research Center 373: Quantification and Simulation of Economic Processes,
Humboldt University Berlin

Suggested Citation: Klemelä, Jussi; Klinke, Sigbert; Sofyan, Hizir (2000) : Classification and regression trees, SFB 373 Discussion Paper, No. 2000,62, Humboldt University of Berlin, Interdisciplinary Research Project 373: Quantification and Simulation of Economic Processes, Berlin,
<https://nbn-resolving.de/urn:nbn:de:kobv:11-10047883>

This Version is available at:

<https://hdl.handle.net/10419/62172>

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.

Classification and Regression Trees

Jussi Klemelä, Sigbert Klinke, and Hizir Sofyan

We will call an estimator for the regression function defined by the CART methodology a **regression tree**. The word CART means **classification and regression tree**. This chapter will focus only on the regression trees.

Regression trees are regression function estimators that are constant in rectangles. The rectangles need not have equal size, as in the case of the (standard) histogram estimators. Regression trees have the special property that they are representable as binary trees. This makes the graphical presentation of estimates possible, even in the case of many regression variables. Indeed, the regression tree is especially useful in the multidimensional cases. Furthermore, the regression tree has the advantage that it works also when the regression variables are a mixture of categorical and continuous variables. The response variable is assumed to be a continuous variable. Regression tree is well suited for the random design as well as for the fixed design. The theory of regression trees was developed by Breiman et al. (1984).

CART methodology consists of three parts. First, we grow a regression tree which overfits the data. Secondly we prune from the overfitting tree a sequence of subtrees and lastly we try to select from the sequence of subtrees a subtree which estimates the true regression function as best as possible.

1 Growing the Tree

```
cs = cartsplit(x, y, type{, opt})  
    grows the tree
```

```

opt = cartsplitopt(s1{, s2, s3,...})
      sets the parameters for growing the tree

```

Growing the tree proceeds sequentially. As a first step we take the regression estimator to be just a constant over the sample space. The constant in question is the mean value of the response variable. Thus, when the observed values of the response variable are Y_1, \dots, Y_n , the regression estimator is given by

$$\hat{f}(x) = \left(\frac{1}{n} \sum_{i=1}^n Y_i \right) I_R(x)$$

where R is the sample space and I_R is the indicator function of R . We assume that the sample space R , that is, the space of the values of the regression variables, is a rectangle.

Secondly the sample space is divided into two parts. Some regression variable X_j is chosen, and if X_j is a continuous random variable, then some real number a is chosen, and we define

$$R_1 = \{x \in R : x_j \leq a\}, \quad R_2 = \{x \in R : x_j > a\}.$$

If X_j is categorical random variable with values A_1, \dots, A_q , then some subset $I \subset \{A_1, \dots, A_q\}$ is chosen, and we define

$$R_1 = \{x \in R : x_j \in I\}, \quad R_2 = \{x \in R : x_j \in \{A_1, \dots, A_q\} \setminus I\}.$$

The regression estimator in the second step is

$$\hat{f}(x) = \left(\frac{1}{|I_1|} \sum_{I_1} Y_i \right) I_{R_1}(x) + \left(\frac{1}{|I_2|} \sum_{I_2} Y_i \right) I_{R_2}(x)$$

where $I_1 = \{i : X_i \in R_1\}$ and $|I_1|$ is the number of elements in I_1 .

The splitting of R to R_1 and R_2 is chosen in such a way that the sum of squared residuals of the estimator \hat{f} is minimized. The sum of squared residuals is defined as

$$\sum_{i=1}^n \left(Y_i - \hat{f}(X_i) \right)^2.$$

Now we proceed to split R_1 and R_2 separately. Splitting is continued in this way until the number of observations in every rectangle is small or the sum of squared residuals is small. The rectangle R corresponds to the root node of the binary tree. The rectangle R_1 is the left child node and the rectangle R_2 is the right child node. The end result is a binary tree.

2 Pruning the Tree

```
subcs = prune(cs, alfa)
    prunes a subtree from the given tree, given the complexity pa-
    rameter alfa

subcs = prunetot(cs, lnum)
    prunes a subtree from a tree, given desired number of leaves

subcs = maketr(cs, node)
    forms a subtree, cutting at a given node

resu = prunecv(tr, alfaseq, x, y, type)
    for a given tree, this quantlet calculates the MSE of squared resid-
    uals for the subtrees of a given tree

seq = pruneseq(cs)
    for a given tree, this quantlet gives the sequence of  $\alpha$  values and
    numbers of leaves of the pruned subtrees
```

One might think that the optimal way of choosing a regression tree is to stop growing the tree before it gets too large. For example, one could stop growing when the sum of the mean squared residuals of the regression estimator does not decrease substantially anymore. However, it might happen that the decrease in the sum of the mean squared residuals is momentarily slow, but some further splits result again in considerable decrease in this sum.

Let us denote

$$R(\hat{f}) = \frac{1}{n} \sum_{i=1}^n \left(Y_i - \hat{f}(X_i) \right)^2$$

and for $0 \leq \alpha < \infty$,

$$R_\alpha(\hat{f}) = R(\hat{f}) + \alpha \text{Leaves}(\hat{f})$$

where $\text{Leaves}(\hat{f})$ is the number of the leaves of the regression tree \hat{f} , which could be viewed as the number of rectangulars on which \hat{f} is constant. The criterion $R_\alpha(\hat{f})$ takes into account not only the sum of the squared residuals but it also penalizes with respect to the complexity of the tree, measured by number of leaves in the tree. The number α is like smoothing parameter in kernel estimation.

Let \hat{f} be a large, overfitting tree and let \hat{f}_α be a subtree of \hat{f} for which $R_\alpha(\cdot)$ is minimal. Note that \hat{f}_0 is \hat{f} , and when α is sufficiently large, \hat{f}_α is the constant estimator, that is, it consists of the root node only. When α increases from 0 to infinity, there are only finite number of values of α at which \hat{f}_α is different. Let us call those values $0 = \alpha_1 < \dots < \alpha_k$. The number k is less or equal to the number of leaves in \hat{f} . For $\alpha_k \leq \alpha < \alpha_{k+1}$, \hat{f}_α is the smallest subtree of \hat{f} minimizing $R_\alpha(\cdot)$. Now the sequence $\hat{f}_{\alpha_1}, \dots, \hat{f}_{\alpha_k}$ forms a decreasing sequence of trees, \hat{f}_{α_1} is the original tree \hat{f} , and \hat{f}_{α_k} consists of the root node only.

3 Selecting the Final Tree

```
cross = cartcv(x, y, type, opt, wv)
    cross-validation is done by this quantlet, which calculates the se-
    quence of  $\alpha$  values, number of leaves of the corresponding pruned
    subtrees, estimates of the expected values of the mean squared
    residuals and their standard errors

ln = leafnum(cs, node)
    gives the number of leaves for a given tree

res = ssr(cs, node)
    calculates the sum of squared residuals

enn = pred (tr, x, type)
    calculates the prediction of the regression tree for a point x
```

```
mssr = prederr(tr, x, y, type)
      calculates the sum of prediction errors for given tree and number
      of x and y values
```

Now we have to choose the best tree from the sequence $\hat{f}_{\alpha_1}, \dots, \hat{f}_{\alpha_k}$. In other words, we have to choose the smoothing parameter α . We will try to estimate the expectation of the mean of squared residuals $R(\hat{f}_{\alpha_i})$, and then choose the regression estimate for which this estimate is minimal. This can be done by way of cross-validation.

For example, in the ten fold cross-validation we take 90% of the sample, grow the tree using this part of the sample, prune a sequence of subtrees and calculate the mean of squared residuals for every subtree in the sequence using the rest 10% of the sample as a test set. This is repeated 10 times, every time using different part of the sample as an estimation set and as a test set.

There is a problem that because we have used every time different data to grow and prune, we get every time different α -sequences. The approach proposed by Breiman, Friedman, Olshen, and Stone (1984, Section 8.5.2, page 234) is to first grow and prune using all of the data, which gives us the sequence $\{\alpha_i\}$, then form a new sequence $\beta_i = \sqrt{\alpha_i \alpha_{i+1}}$. The number β_i is the geometric mean of α_i and α_{i+1} . When pruning trees grown with 90% of the sample, we choose subtrees which minimize $R_{\beta_i}()$.

Finally, the estimate for the expectation of $R(\hat{f}_{\alpha_i})$ is the mean of $R(\hat{f}_{\beta_i}^v)$. Mean is over 10 cross-validation estimates $\hat{f}_{\beta_i}^v$, $v = 1, \dots, 10$. In practice, the estimates for the expectation of $R()$ do not have clear minimum, and it is reasonable to choose the smallest tree such that the estimate for the expectation of $R()$ is reasonably close to the minimum.

4 Plotting the Result of CART

```
plotcarttree(carttree{, outval})
      plots the CART tree
```

```

dispcarttree(ctdisp, xn, yn, carttree{, outval})
    plots the CART tree in an user given display ctdisp

{tree, treelabel} = grcarttree(carttree{, outval})
    generates two graphical objects which contain the plot of the
    CART tree and the labels in the CART tree

plotcart2(x, tree{, xname})
    plots the cuts of CART in a two-dimensional projection of the
    dataset x

dispcart2(ctdisp, xn, yn, x, carttree, ix, iy, depth, ssr)
    plots the cuts of CART in a twodimensional projection of the
    dataset x in an user given display ctdisp

cut = grcart2(x, carttree, ix, iy, depth, ssr)
    generates the cuts which contain a two-dimensional projection of
    the data x

```

For visualizing the results two methods are provided:

1. plotting the tree via `plotcarttree`, `dispcarttree` or `grcarttree`
2. `plotcart2`, `dispcart2` or `grcart2` show how CART tessellates a two-dimensional projection of the data

As an output of `cartsplit` we receive the CART tree (`carttree`). The first method plots the CART tree. Depending on the value given in `outval` we get as labels at the nodes in the tree the splitting rules (default), the numbers of observations in a node ("`nelem`"), the mean value of the observations in a node ("`mean`") or the sum of squared residuals ("`ssr`") in a node.

To get an overview how CART tessellates the space we can plot the tessellations in two dimensional projections of the data. The quantlet `plotcart2` allows the user to interactively change the projections. Also interactively we can choose if we want to see all cuts to a specified level (see `depth`) or all cuts where the sum of squared residuals (see `ssr`) is above a specified limit.

Note: if you view the cuts at the projection then be aware that cuts could appear in the tree which are not visible. Thus the plot has to be interpreted with

care! Additionally the plot can only be applied to projections of continuous variables.

5 Examples

5.1 Simulated Example

Let us generate the observations

$$Y_i = f(X_i) + \epsilon_i, \quad i = 1, \dots, 200,$$

where


$$\begin{aligned} f(x_1, x_2) = & 100 I(0 \leq x_1 \leq 0.5, 0.5 < x_2 \leq 1) \\ & + 120 I(0.5 < x_1 \leq 1, 0.5 < x_2 \leq 1), \end{aligned}$$

X_i are independently uniformly distributed on $[0, 1] \times [0, 1]$, and ϵ_i are independently standardly normally distributed.

Figure 1 shows the data simulated from the function f . The quantlet for generating the observations is

```
proc(y)=tuto(seed,n)
  randomize(seed)
  xdat=uniform(n,2)
  index=(xdat[,2]<=0.5)+(xdat[,2]>0.5).*(xdat[,1]<=0.5)*2
  layout=3*(index==1)+4.*(index==0)+5.*(index==2)
  ydat=100.*(index==2)+120.*(index==0)
  y=list(xdat,ydat,layout)
endp

library("xclust")
d=createdisplay(1,1)
data=tuto(1,100)
x=data.xdat
setmaskp(x, data.layout, data.layout, 8)
show(d,1,1,x)
```

 cart01.xpl

Let us grow such a tree that the number of observations in a leaf nodes is less

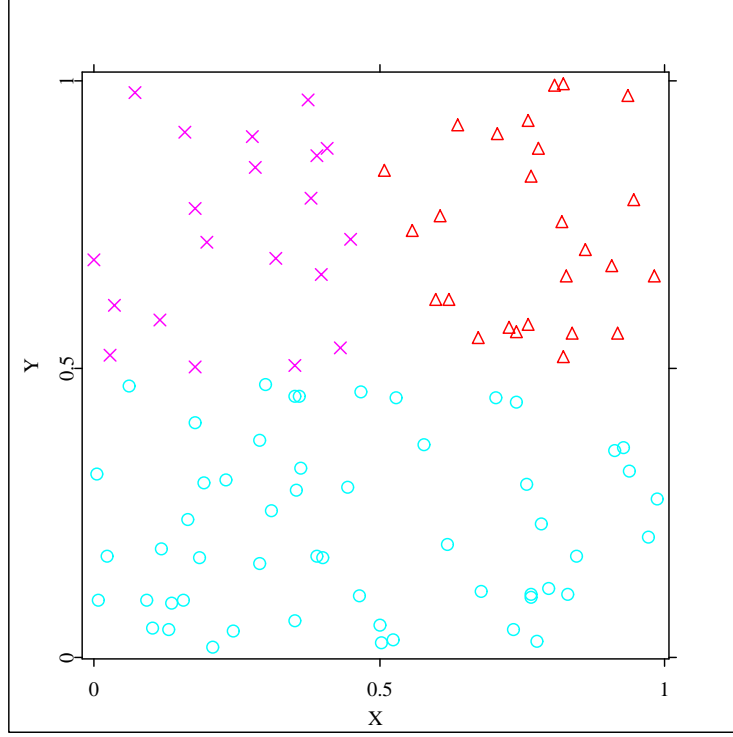


Figure 1: Plot of 100 simulated data from function $f(x_1, x_2)$. The datapoints in the upper left (marked with crosses) are in the area of $f(x_1, x_2) = 100$, the datapoints in the upper right (marked with triangles) are in the area of $f(x_1, x_2) = 120$ and the datapoints in the lower part (marked with circles) are in the area of $f(x_1, x_2) = 0$.

or equal to 5 (`mincut`), the deviation in a leaf node is larger or equal 0 (`mindev`) and cut will be only done if the number of the resulting nodes is larger as 1 (`minsize`). The type of variable is continuous.

```
library("xclust")
data=tuto(1,100)
type=#(1,1)
opt=cartsplitopt("minsize",1,"mindev",0,"mincut",5)
tr=cartsplit(data.xdat,data.ydat,type,opt)
totleaves=leafnum(tr,1)
totleaves
plotcarttree(tr)
```


 cart02.xpl

Figure 2 shows the regression tree `tr` with 41 leaves. From this figure, we prefer to choose the tree with 3 leaves because it is easier to see that in general it has three groups.

Let us choose the tree with 3 leaves with the following command.

```
trfin=prunetot(tr,3)
plotcarttree(trfin)
```


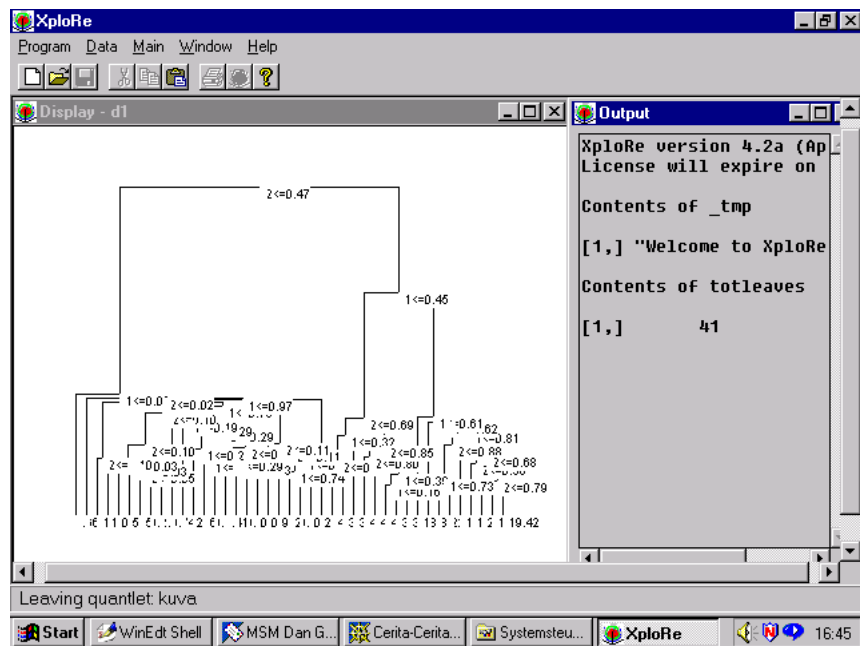
 cart03.xpl

Figure 3 shows the final tree for simulated data.

5.2 Boston Housing Data

The Boston housing data set `bostonh` was collected by Harrison and Rubinfeld (1978). The following variables are in the data:

1. crime rate
2. percent of land zoned for large lots
3. percent non retail business
4. Charles river indicator, 1 if on Charles river, 0 otherwise
5. nitrogen oxide concentration



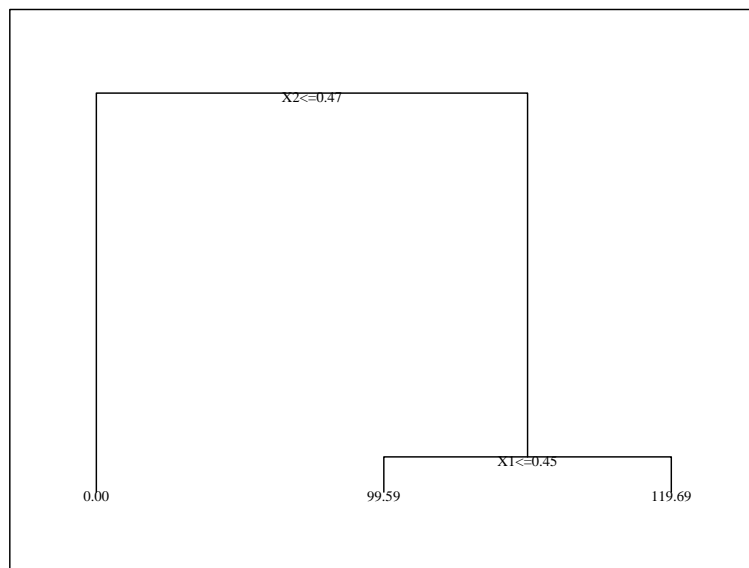



Figure 3: Final regression tree for 100 simulated data from function $f(x_1, x_2)$ after pruning. The final tree consists of three leaves which separate the x_1, x_2 -plane into three parts.

6. average number of rooms
7. percent built before 1980
8. weighted distance to employment centers
9. accessibility to radial highways
10. tax rate
11. pupil-teacher ratio
12. percent black
13. percent lower status
14. median value of owner-occupied homes in thousands of dollars.

The variable 14 is the response variable. The variables 1–13 are predictor variables. The 4-th and 9-th are categorical variables, the other are continuous. There are 506 observations. Let us generate such a tree that the number of observations in leaf nodes is less or equal to 8.

```
library("xclust")
randomize(10)
boston=read("bostonh")
boston=paf(boston,uniform(rows(boston))<0.20)
yvar=boston[,14]
xvar=boston[,1:13]
type=matrix(13)
type[4]=0
type[9]=0
opt=cartsplitopt("minsize",1,"mindev",0,"mincut",8)
tr=cartsplit(xvar,yvar,type,opt)
totleaves=leafnum(tr,1)
totleaves
plotcarttree(tr)
```

 cart04.xpl

We can observe that the tree `tr` with 29 leaves is large.

It is not so easy to read Figure 4. We can look at the optimal subtree consisting of 10 leaves by using these commands:

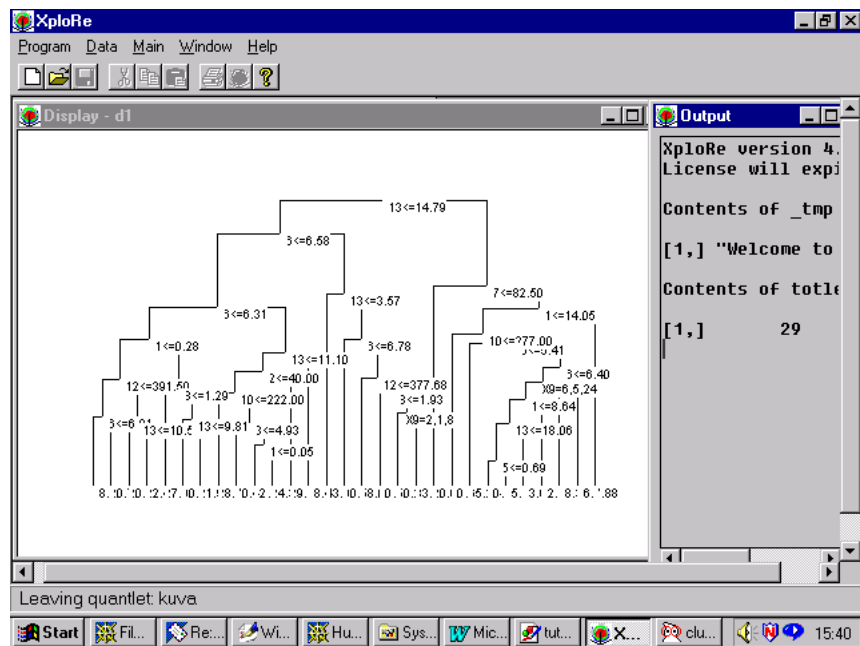



Figure 4: Initial regression tree for Boston housing data. The total number of leaves (29) is shown at the right.


```
prtr=prunetot(tr,10)
plotcarttree(prtr)
```

 cart05.xpl

The Figure 5 shows pruning tree for Boston housing data. Let us try to choose the optimal number of leaves with 10 fold cross validation.

```
cval=cartcv(xvar,yvar,type,opt,10)
res=cval.lnumber~cval.alfa~cval.cv~cval.cvstd
res=sort(res,1)
res=res[1:12,]
title=" no    alfa    cv    cvstd"
restxt=title|string("%3.0f %6.2f %6.2f %6.2f",
res[,1], res[,2], res[,3], res[,4])

dd=createdisplay(2,2)
show(dd, 1, 1, cval.lnumber~cval.alfa)
setgopt(dd, 1, 1, "title","number obs. vs alpha")
show(dd, 1, 2, cval.lnumber~cval.cv)
setgopt(dd, 1, 2, "title","number obs. vs cv")
show(dd, 2, 1, cval.lnumber~cval.cvstd)
setgopt(dd, 2, 1, "title","number obs. vs cvstd")
show(dd, 2, 2, restxt)
```

 cart06.xpl

We get the result shown in Figure 6.

The first column gives the numbers of leaves in the sequence of pruned subtrees and the second column gives the sequence α_i . The estimates for the expectation of the mean of squared residuals, $ER(\hat{f}_{\alpha_i})$, are in the third column of the above matrix. The fourth column gives the estimates of the standard error of the corresponding estimators.

We can see that there is a clear minimum for the estimates for the expectation of the mean of squared residuals.

Therefore, it seems reasonable to choose as final estimate the tree with 7 leaves. Let us choose $\alpha = 0.9$ and form the corresponding tree.

```
fin=prune(tr,0.9)
plotcarttree(fin)
```

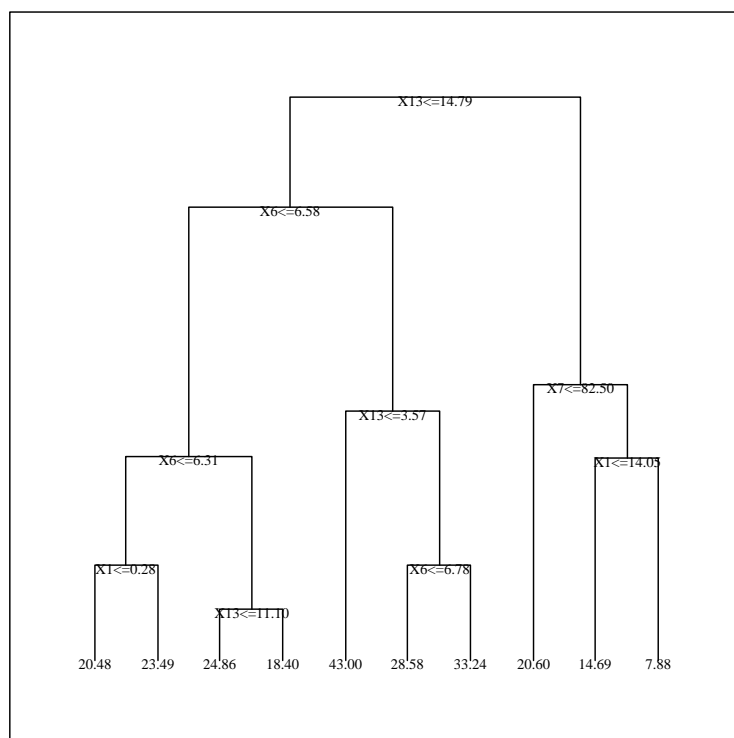


Figure 5: Sub-Tree consisting of 10 leaves for 20% sample of the Boston housing data

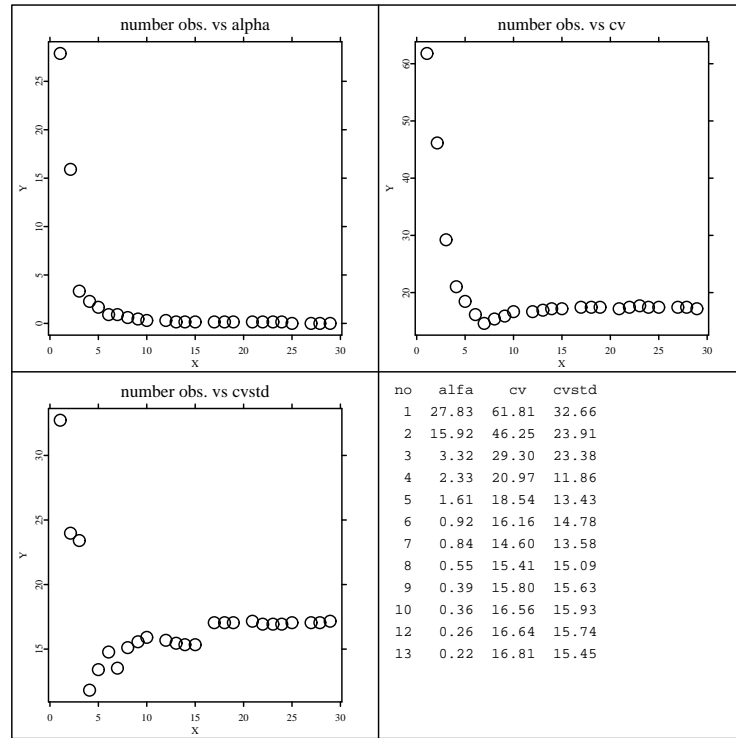


Figure 6: Cross-validation for 20% sample of Boston housing data.

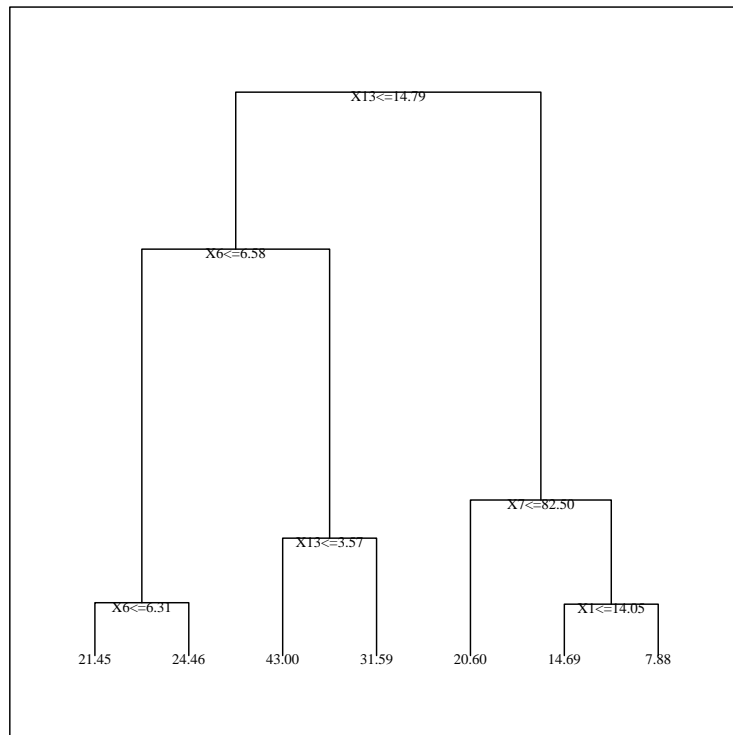




Figure 7: Final tree for 20% sample of Boston housing data

 cart07.xpl

The final estimate is in the Figure 7. Let us look at the numbers of observations and the mean value in each node with command

```
plotcarttree(fin,"nelem")
plotcarttree(fin,"mean")
```

 cart08.xpl

The result is displayed in the Figure 8 and Figure 9 respectively.

5.3 Density Estimation

```
regdat = dentoreg(dendat, binlkm)
          transforms density data to regression data using variance stabi-
          lizing transform
```

Instead of writing separate procedures for the estimation of density functions, we will transform density data to the regression data and use regression tree to estimate density functions.

The basic idea is to divide the sample space into bins, calculate the number of observations in every bin, and consider these frequencies as a dependent regression variable. The independent regression variables are the midpoints of the bins. To be more precise, after we have calculated the frequencies of the bins Z_i , we will transform these to

$$Y_i = \sqrt{Z_i + 3/8}.$$

This was suggested by Anscombe (1948) and Donoho, Johnstone, Kerkyacharian, and Picard (1995, page 327).

Use the procedure first to make a histogram estimator for the density. This estimator will have a large number of equal size bins and so it will not be a good density estimator, but we will then combine some of these bins together in an optimal way using CART. The new regression data will have dimension equal to the number of bins to the power of the number of variables. Given moment computing capability, probably 9 is the maximum number of variables for this method.

As an example we will analyze data which consists of 200 measurements on Swiss bank notes. These data are taken from Flury and Riedwyl (1988). One half of these bank notes are genuine, the other half are forged bank notes. The following variables are in the data.

1. length of the note (width)
2. height of the note (left)
3. height of the note (right)

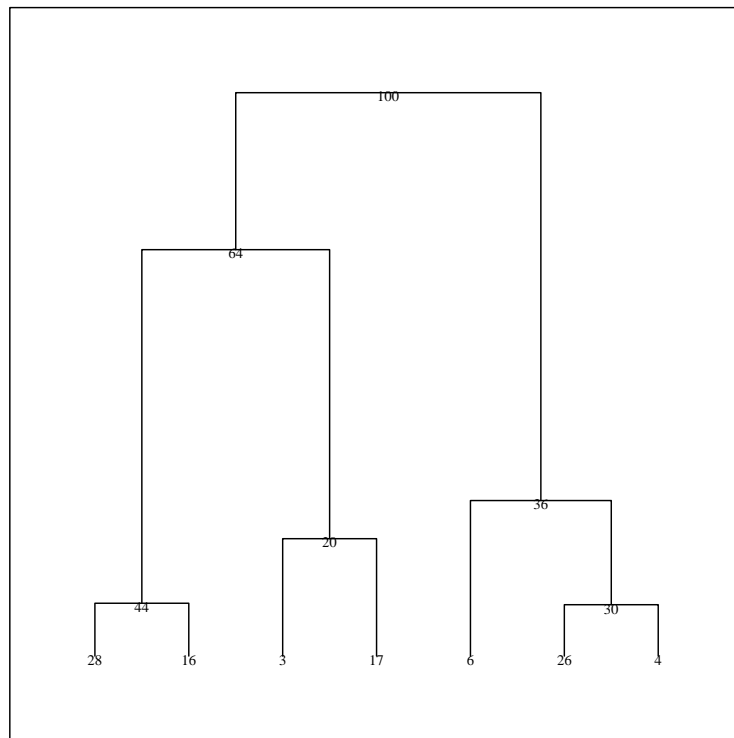


Figure 8: Final tree for 20% sample of Boston housing data with numbers of observations

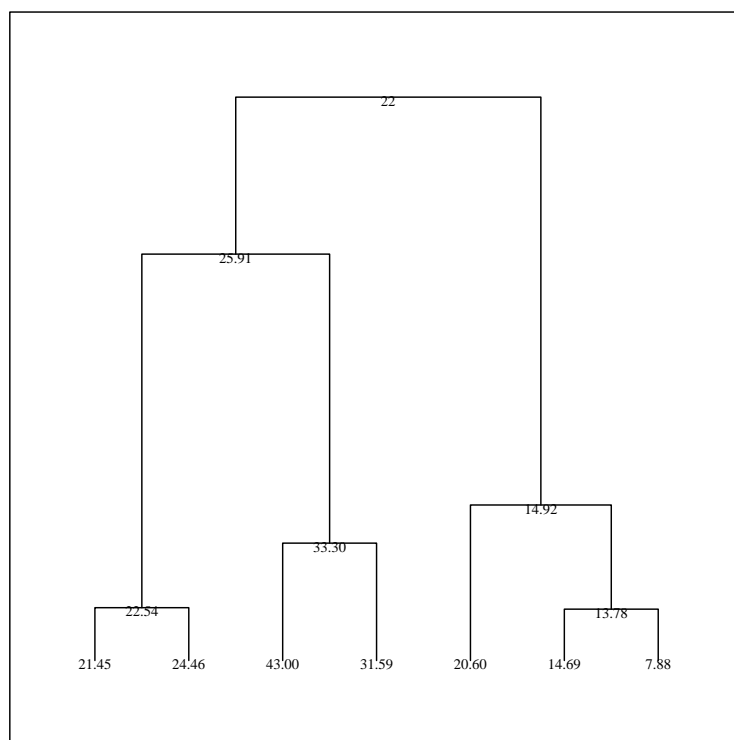


Figure 9: Final tree for 20% sample of Boston housing data with mean values

4. distance of the inner frame to the lower border (bottom)
5. distance of the inner frame to the upper border (top)
6. length of the diagonal of the central picture (diagonal)

The macro `dentoreg` transforms density data to regression data. Let us choose 9 bins for every coordinate axes because we for the last 3 variables in the data.

```
; load library xclust and plot
library("xclust")
library("plot")

; set random seed
randomize(1)
; read swiss banknote data
dendat=read("bank2")
; select the last three variables
dendat=dendat[,4:6]
; choose 9 bins in each dimension
binlkm=9

; compute density estimate
regdat=dentoreg(dendat,binlkm)

; compute CART and tree
type=matrix(cols(dendat))
opt=cartsplitopt("minsize",50,"mindev",0,"mincut",1)
tr=cartsplit(regdat.ind,regdat.dep,type,opt)
; color datapoints after node the fall in
g=cartregr(tr, dendat, "node")
{gcat,gind}=groupcol(g, rows(g))
; compute cuts up level 2 for (X1,X2)
xdat=regdat.ind
gr12=grcart2(xdat, tr, 1, 2, 10, 0)
xdat12=dendat[,1|2]
setmaskp(xdat12, gind)
; compute cuts up level 2 for (X1,X3)
gr13=grcart2(xdat, tr, 1, 3, 10, 0)
xdat13=dendat[,1|3]
```


```

setmaskp(xdat13, gind)
; compute cuts up level 2 for (X2,X3)
gr23=grcart2(xdat, tr, 2, 3, 10, 0)
xdat23=dendat[,2|3]
setmaskp(xdat23, gind)

; compute tree and its labels
{tree, treelabel}=grcarttree(tr)
; show all projections and the tree in a display
setsize(640, 480)
d=createdisplay(2,2)
show(d, 1,1, xdat12, gr12)
setgopt(d,1,1, "xlabel", "top (X1)", "ylabel", "bottom (X2)")


show(d,2,1, xdat13, gr13)
setgopt(d,2,1, "xlabel", "top (X1)")
setgopt(d,2,1, "ylabel", "diagonal (X3)")
show(d, 2,2, xdat23, gr23)
setgopt(d,2,2, "xlabel", "bottom (X2)")
setgopt(d,2,2, "ylabel", "diagonal (X3)")
axesoff()
show(d, 1,2, tree, treelabel)
axeson()


```

 cart09.xpl

The result is shown in Figure 10. The upper left plot gives the cuts in the bottom-top plane, the lower left plot the cuts in the bottom-diagonal plane and the lower right plot the cuts in the top-diagonal plane. The CART tree is shown in the upper right window.

All splits are done in the bottom-diagonal plane. The lower right plot shows that CART algorithm just cuts from the main bulk of the data. Note the different colors in the left plots which shows that we have some cuts which are not visible in the top-bottom or top-diagonal projection.

Since we have chosen to stop splitting if the number of the observations is less than 75 (see the parameters `cartsplitopt` in  cart09.xpl) we may choose a smaller number.

In  cart10.xpl we have chosen a smaller number (20), do not color of the

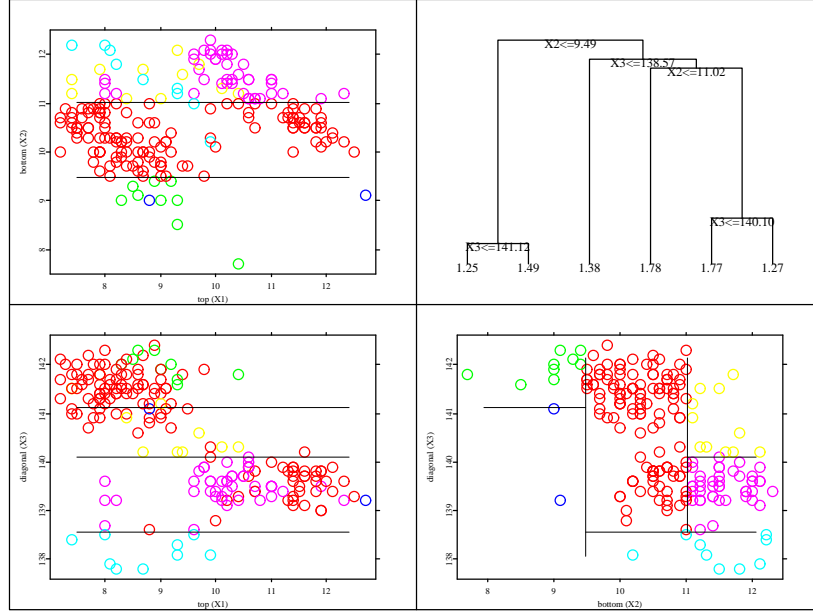


Figure 10: The upper left plot gives the cuts in the bottom-top plane, the lower left plot the cuts in the bottom-diagonal plane and the lower right plot the cuts in the top-diagonal plane. The CART tree is shown in the upper right window.

datapoints and omit the tree labels. The main result is here again that the CART algorithm cuts away the tails of the distribution and generates at least 4 different group of nodes.

```

; load library xclust and plot
library("xclust")
library("plot")
; set random seed
randomize(1)
; read swiss banknote data
dendat=read("bank2")
; select the last three variables
dendat=dendat[,4:6]
; choose 9 bins in each dimension
binlkm=9
; compute density estimate
regdat=dentoreg(dendat,binlkm)
; compute CART and tree
type=matrix(cols(dendat))
opt=cartsplitopt("minsize",20,"mindev",0,"mincut",1)
tr=cartsplit(regdat.ind,regdat.dep,type,opt)
; compute cuts up level 2 for (X1,X2)
xdat=regdat.ind
gr12=grcart2(xdat, tr, 1, 2, 10, 0)
xdat12=dendat[,1|2]
; compute cuts up level 2 for (X1,X3)
gr13=grcart2(xdat, tr, 1, 3, 10, 0)
xdat13=dendat[,1|3]
; compute cuts up level 2 for (X2,X3)
gr23=grcart2(xdat, tr, 2, 3, 10, 0)
xdat23=dendat[,2|3]
; compute tree and its labels
{tree, treelabel}=grcarttree(tr)
; show all projections and the tree in a display
setsize(640, 480)
d=createdisplay(2,2)
show(d, 1,1, xdat12, gr12)
setgopt(d,1,1, "xlabel", "top (X1)", "ylabel", "bottom (X2)")
show(d, 2,1, xdat13, gr13)
setgopt(d,2,1, "xlabel", "top (X1)", "ylabel", "diagonal (X3)")

```

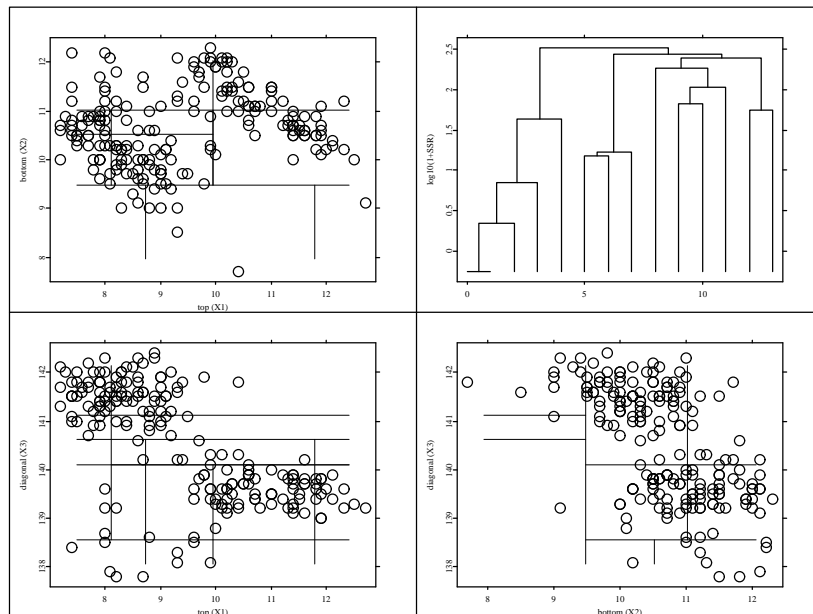



Figure 11: The upper left plot gives the cuts in the bottom-top plane, the lower left plot the cuts in the bottom-diagonal plane and the lower right plot the cuts in the top-diagonal plane. The CART tree is shown in the upper right window.

```
show(d, 2,2, xdat23, gr23)
setgopt(d,2,2, "xlabel", "bottom (X2)", "ylabel", "diagonal (X3)")
show(d, 1,2, tree)
setgopt(d,1,2, "xlabel", " ", "ylabel", "log10(1+SSR)")
```

 cart10.xpl

References

- Anscombe, F. (1948). The transformation of Poisson, binomial and negative-binomial data, *Biometrika* **35**: 246–254.
- Breiman, L., Friedman, J., Olshen, R., and Stone, C. J. (1984). *Classification and Regression Trees*, Chapman and Hall, New York.
- Donoho, D. L., Johnstone, I. M., Kerkycharian, G., and Picard, D. (1995). Wavelet shrinkage: asymptotia? (with discussion), *J. Roy. Statist. Soc. B* **57**: 301–369.
- Flury, B. and Riedwyl, H. (1981). Graphical representation of multivariate data by means of asymmetrical faces, *J. Amer. Statist. Assoc.* **76**: 757–765.
- Harrison, D. and Rubinfeld, D. L. (1978). Hedonic prices and the demand for clean air, *J. Envir. Econ. and Management* **5**: 81–102.