Data Analysis

Christophe Ambroise

Clustering

Introduction

A widespread confusion exists between the terms classification and clustering:

- Classification presupposes the existence of classes with certain objects are known, while
- Clustering tries to discover a class structure which is “natural” to the data.

In the literature related to pattern recognition, the distinction between the two approaches is often referred to by the terms “supervised” and “unsupervised” learning.

Objectives

Clustering can have different motivations:

- compress information, to describe in a simplified way large masses of data,
- structure a set of knowledge,
- reveal structures, hidden causes,
- make a diagnosis . . .

Similarity

the fist step to any clustering is to define a measure resemblance between objects (shapes vectors). Traditionally two approaches are envisaged:

- *monothetic* : we can say that two objects are similar if they share a certain feature. Basis of the Aristotelian approach [@Sutcliffe1994]. All objects in the same class share a number of characteristics (e.g. "All men are mortal");
- *polythetic* : we can also measure the similarity by using a measure of proximity (distance, dissimilarity). In this case the notion of resemblance is measured more fuzzy and two objects of the same class will have "close" characteristics within the meaning of the measure used.

Context of this lecture

polythetic classification

Classes or groups

A classification leads to the distribution of the set of vectors forms in different *homogeneous classes*. The definition of a class and the relations between classes can be very varied. In this chapter we will focus on both main classification structures:

- partition,
- hierarchy.

Partitions

Définition

Ω being a finite set, a set $P = (\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_K)$ non-empty parts of Ω is a partition if:

1. $\forall i \neq j$, $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$,
2. $\cup_i \mathcal{C}_i = \Omega$.

In a set $\Omega = (\boldsymbol{x}_1, ..., \boldsymbol{x}_N)$ partitioned into K classes, each element of the set belongs to a class and only one. A practical way of describing this P partition consists of using a matrix notation.
Let $\boldsymbol{C}(P)$ the characteristic matrix of the partition $P = (\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_K)$ (ou matrice de

$$\boldsymbol{C}(P) = \boldsymbol{C} = \begin{pmatrix} c_{11} & \cdots & c_{1K} \\ \vdots & \ddots & \vdots \\ c_{N1} & \cdots & c_{NK} \end{pmatrix}$$

where $c_{ik} = 1$ iff $\boldsymbol{x}_i \in \mathcal{C}_k$, and $c_{ik} = 0$ otherwise.
Note that the sum of the i th row is equal to 1 (an element belongs to a single class) and the sum of the values of the k th column is worth n_k the number of elements of the class $\mathcal{C}_k$. On a donc $\sum_{k=1}^{K} n_k = N$.

Hard and Fuzzy partition

The notion of hard partition is based on a set design classic. Considering the work of [@Zadeh1965] on the sets fuzzy, a definition of the concept of fuzzy partition seems "Natural". The fuzzy classification, developed at the beginning of 1970s [@Ruspini1969], generalizes an approach classical classification by broadening the concept belonging to a class.

Fuzzy sets

As part of the classic set design, an individual $\boldsymbol{x}_i$ belongs to or does not belong to a given set $\mathcal{C}_k$. In the theory of fuzzy subsets, an individual can belong to several classes with different degrees of membership.

Fuzzy clustering

In classification this amounts to authorize the vectors forms to belong to all classes, which results in the releasing the binarity constraint on the coefficients belonging to c_{ik}. A fuzzy partition is defined by a fuzzy classification matrix $\boldsymbol{C} = \{c_{ik}\}$ checking the following conditions:

1. $\forall k = 1..K$, $\forall \boldsymbol{x}_i \in \Omega$, $c_{ik} \in [0, 1]$.
2. $\forall k = 1..K$, $0 < \sum_{i=1}^{N} c_{ik} < N$,
3. $\forall \boldsymbol{x}_i \in \Omega$, $\sum_{k=1}^{K} c_{ik}$.

The second condition reflects the fact that no class should not be empty and the third expresses the concept of total membership.

Indexed Hierarchy

Ω is a finite set. H a set of non empty subsets of Ω is a hiearchy iff

- $\Omega \in H$
- $\forall \boldsymbol{x} \in \Omega, \{\boldsymbol{x}\} \in H$
- $\forall h,\ h' \in H,\ h \cap h' = \emptyset \ or\ h' \subset h \ or\ h \subset h'$

Index

The index of a hierarchy is a function i from H to $\mathbb{R}^+$ having the following properties

- $h \subset h' \Rightarrow i(h) < i(h')$
- $\forall \boldsymbol{x} \in \Omega,\ i(\{\boldsymbol{x}\}) = 0$

(h, i) is then an indexed hierarchy

Partition and Hierarchy

- each level of a indexed hierarchy is a partition
- $\{\Omega, P1, P2, \ldots, P_K, x_1, .., x_n\}$ is a hierarchy

Clustering ideal

Ideal

2 objects of a class should be `closer` than 2 objects of 2 different classes
$\Rightarrow$ most of the time impossible to achieve

- numerical approach: definition of a criterion (with or without a unique extremum)

The Bell numbers

But the number of partitions possible, even for a problem of reasonable size, is huge. Indeed if we consider a set of N objects to partition into K classes, the number of possible partitions is:

$$NP(N, K) = \frac{1}{K!} \sum_{k=0}^{K} (-1)^{k-1} \cdot C_k^K \cdot k^N.$$

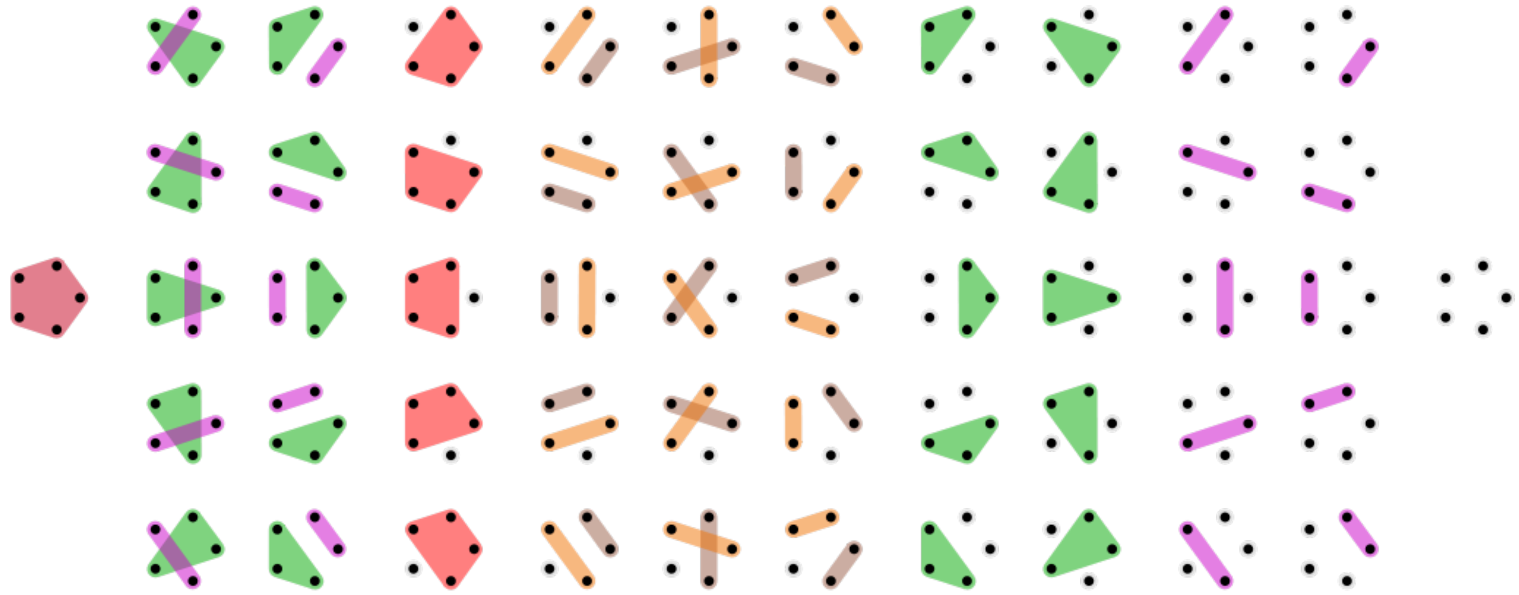


Figure 1: The 52 partitions of a set with 5 elements

The bell number

The bell number reccurence

Show that the number of partition of n objects verifies

$$B_{n+1} = \sum_{k=0}^{n} C_k^n B_k$$

Compute the number of partition of 5 objects.

The bell number

- fix an element x in a set with $n+1$ elements,
- sort the partitions according to the number k of elements outside the part containing x,
- For each value of k from 0 to n, it is necessary to choose k elements among the n different elements of x, then to give a partition.

Criteria and algorithms I

The concepts of partition and polythetic classification being specified, the following question emerges: how to find an optimal partition of a dataset, when the resemblance between two individuals is assessed by a measure of proximity?

The first thing to do is to formally clarify the meaning of the word optimal.

The solution generally adopted is to choose a digital measure of the quality of a partition.

This measure is sometimes called a criterion, functional, or still function of energy. The purpose of a classification procedure so is to find the partition or partitions that give the best value (the smallest or the largest) for a given criterion.

Rather than looking for the best score, the one that gives the optimum value of the criterion, more methods are used which converge towards “local” optima “of the criterion. thus found scores are often satisfying.

Intra-classe Inertia and partition I

Many criteria exist [@Gordon1980]. Some may be related, as we will see in the following, at the choice of a model for all the data. One of most used functions is the sum of intra-class variances:

$$\begin{aligned} I_W &= \sum_{k=1}^{K}\sum_{i=1}^{N} c_{ik}\left\|\boldsymbol{x}_i - \boldsymbol{\mu}_k\right\|^2 \\ &= trace(\boldsymbol{S}_W) \end{aligned}$$

where the $\boldsymbol{\mu}_k$ are the prototypes (centers) of classes and the c_{ik} are the elements of a hard partition matrix. The problem is then a problem of optimization under constraints (related to c_{ik}) :

$$(\hat{\boldsymbol{c}}, \hat{\boldsymbol{\mu}}) = arg \min_{(\boldsymbol{c},\boldsymbol{\mu})} I_W((\boldsymbol{c}, \boldsymbol{\mu})) \tag{1}$$

where $\boldsymbol{\mu}$ represents all the centers of gravities.

K-means algorithm I

A common algorithm for solving this problem is the one **k-means**. Historically, this algorithm dates sixties. It has been proposed by several researchers in different areas at close dates [@ Edwards1965, @ Lloyd1957]. This algorithm based on considerations Geometric certainly owes its success to its simplicity and efficiency:

Algorithm

1. Initialization of centers: a common method is to initialize centers with coordinates of K randomly selected points.
2. Then the iterations have the following alternate form:
    1. given $\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_K$, chose c_{ik} minimizing I_W,
    2. given $\boldsymbol{C} = \{c_{ik}\}$, minimise I_W with respect to $\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_K$.

The first step

assigns each $\boldsymbol{x}_i$ to the nearest prototype,

the second step

recalculates the position of the prototypes considering that the prototype of the class i becomes its mean vector.

K-means algorithm II

Convervegence of kmeans algorithm

It is possible to show that each iteration decreases the criterion but no guarantee of convergence towards a global maximum does not exist in general.

Link with fuzzy clustering

If the k-means criterion is considered from the point of view of research of a fuzzy partition, that is, if the constraints on the c_{ik} are released and become $c_{ik} \in [0, 1]$ instead of $c_{ik} \in \{0, 1\}$, the optimal partition in the sense of the new criterion is the optimal one for the classical criterion [@Selim1984]. In other words, there is no interest in considering fuzzy scores when working with the criterion of k-means.

Kmeans algorithm's

Nuées dynamiques (dynamic swarm)

This form of alternate algorithm where a certain criterion is optimized, alternatively with respect to the class membership variables, then compared to the parameters defining these classes has been extensively exploited. Let's mention among others *the dynamic clouds* of Diday [@Diday1971] and the *fuzzy c-means* algorithm [@Bezdeck1974].
Note that Webster Fisher [@Fisher1958] (not to be confused with Ronald Fisher) had proposed an algorithm finding the optimal partition, within the intra-class variance, of a set of N data one-dimensional in $O(N \cdot K^2)$ operations using methods from dynamic programming.

In R, the function `kmeans {stats}` I

effectively implements the algorithm.

```r
data(iris)
kmeans.res <- iris %>%
              select(-Species,-Sepal.Length,-Sepal.Width) %>%
              kmeans(3,nstart = 10)

cluster<-as.factor(kmeans.res$cluster)
centers <-as.tibble(kmeans.res$centers)
```

```r
ggplot(iris, aes(x=Petal.Length, y=Petal.Width, color=cluster)) +
  geom_point() +
  geom_point(data=centers, color='coral',size=4,pch=21)+
  geom_point(data=centers, color='coral',size=50,alpha=0.2)
```

In R, the function `kmeans {stats}` II

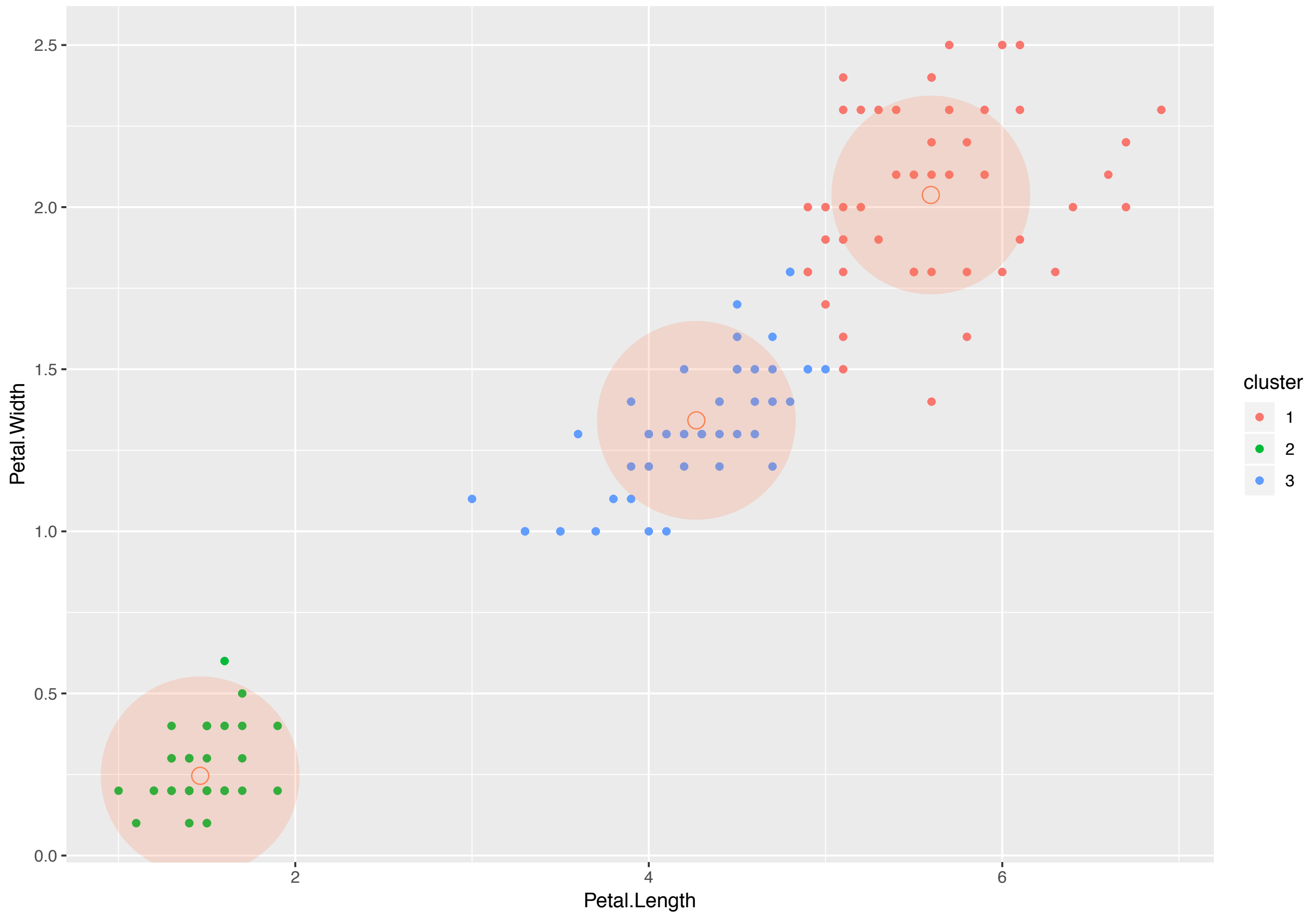


Figure 2: Automatic classification into three classes of iris by the k-means algorithm

In R, the function `kmeans {stats}` III

```r
knitr::kable(table(iris$Species, kmeans.res$cluster),
             caption = "Table de contingence croisant
             réalité et estimation de la structure par k-means")
```

Table 1: Table de contingence croisant réalité et estimation de la structure par k-means

	1	2	3
setosa	0	50	0
versicolor	2	0	48
virginica	46	0	4

Voronoi tiling I

By definition, `kmeans` partition the space by a Voronoi tiling defined by the centers.

```r
library(deldir)
```

```
## Warning: package 'deldir' was built under R version 3.4.4
```

```
## deldir 0.1-15
```

```r
#This creates the voronoi line segments
voronoi <- deldir(centers$Petal.Length, centers$Petal.Width)
```

```
##
##      PLEASE NOTE:  The components "delsgs" and "summary" of the
##  object returned by deldir() are now DATA FRAMES rather than
##  matrices (as they were prior to release 0.0-18).
##  See help("deldir").
##
##      PLEASE NOTE: The process that deldir() uses for determining
##  duplicated points has changed from that used in version
##  0.0-9 of this package (and previously). See help("deldir").
```

Voronoi tiling II

```r
 #Now we can make a plot
ggplot(data=iris, aes(x=Petal.Length, y=Petal.Width, color=cluster)) +
  geom_point()+
  #Plot the voronoi lines
  geom_segment(
    aes(x = x1, y = y1, xend = x2, yend = y2),
    size = 2,
    data = voronoi$dirsgs,
    linetype = 1,
    color= "coral") +
  #Plot the points
  geom_point(data=centers,
    fill='coral',
    pch=21,
    size = 4,
    color="#333333")
```

Voronoi tiling III

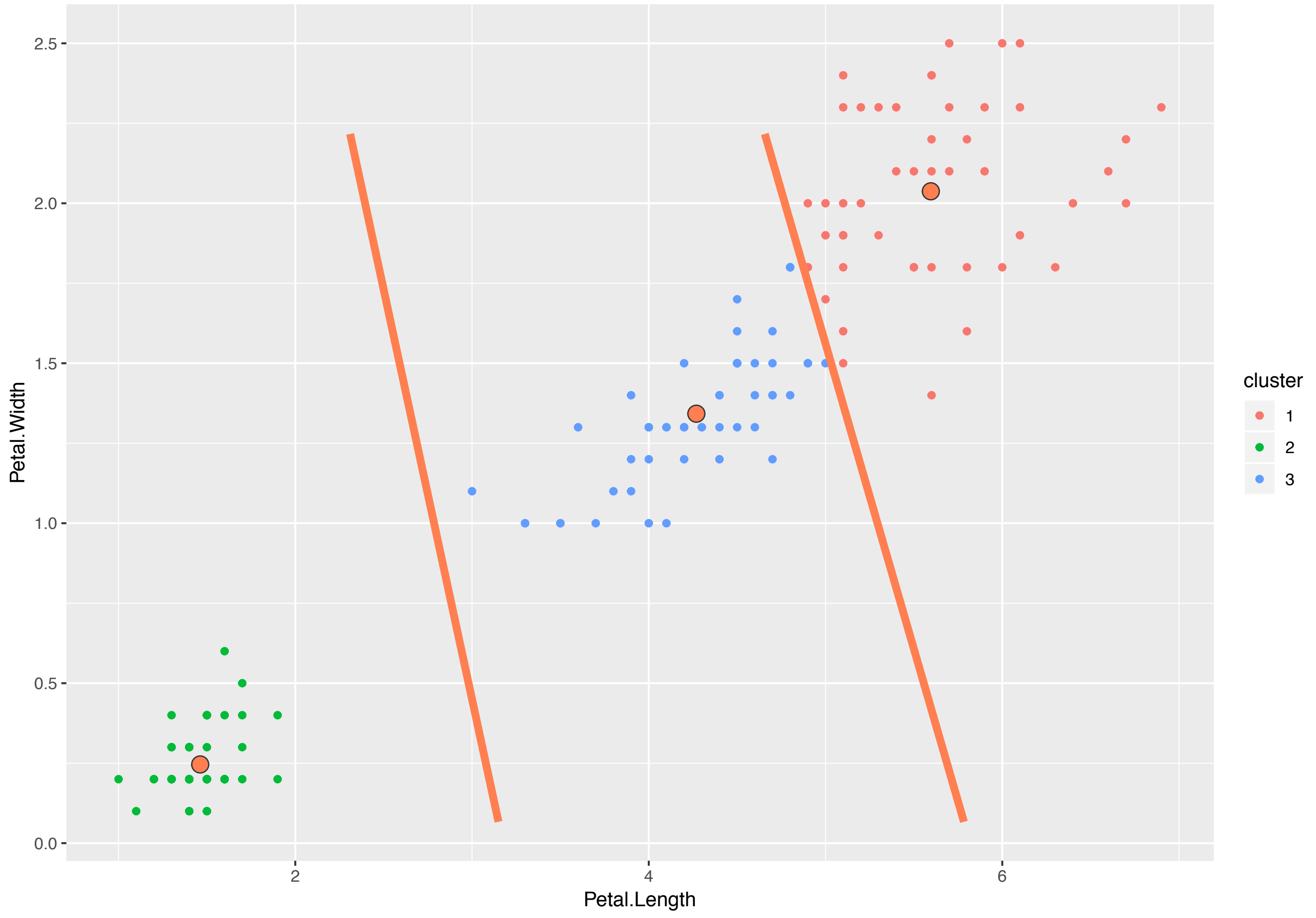


Figure 3: Classification automatique en trois classes des iris par l'algorithme des centres mobiles et pavage de Voronoï

Spectral Clustering

Hierachical Clustering

There are two main approaches

- Agglomerative
- Divisive

Hierarchical Agglomerative Clusering

The principle of Hierarchical Agglomerative Clusering is fairly easy

1. **Initialisation** each Ω element constitutes a class. A dissimilarity D is computed between all classes.
2. **While** number of cluster > 1

   1. group the two closest classes in the sense of the dissimilarity D,
   2. calculation of “distances” between the new class and the others.

Cluster dissimilarity

The dissimilarity D between two parts h and h' of Ω, can be defined in many ways from a measure of dissimilarity d on Ω.

- single link (critère du lien minimum):

$$D(h,h') = \min\left[d(\boldsymbol{x},\boldsymbol{y})/\boldsymbol{x} \in h \textit{ and } \boldsymbol{y} \in h'\right],$$

- complete link (critère du lien maximum)

$$D(h,h') = \max\left[d(\boldsymbol{x},\boldsymbol{y})/\boldsymbol{x} \in h \textit{ and } \boldsymbol{y} \in h'\right],$$

* group average

$$D(h,h') = \frac{\sum_{i=1}^{n_h}\sum_{j=1}^{n_{h'}} d(\boldsymbol{x}_i,\boldsymbol{x}_j)}{n_h \cdot n_{h'}},$$

- Ward criterion

$$D(h,h') = \frac{n_h \cdot n_{h'}}{n_h + n_{h'}}\|\boldsymbol{m}_h - \boldsymbol{m}_{h'}\|^2.$$

Intra class criterion and Ward method I

When we have a partition in K classes, the criterion of intra-class inertia measures its homogeneity:

$$\begin{aligned} I_W &= trace(\boldsymbol{S}_W), \\ &= \sum_{k=1}^{K} \sum_{i=1}^{n_k} (\boldsymbol{x}_{ik} - \boldsymbol{m}_k)^t (\boldsymbol{x}_{ik} - \boldsymbol{m}_k). \end{aligned}$$

Let us consider two partitions

- $P = (P_1, \cdots, P_K)$,
- and P', the partition obtained by merging the classes $\mathcal{C}_k$ et $\mathcal{C}_\ell$.

Intra class criterion and Ward method II

We can show that the difference between the inertia of the two partitions is equal to Ward's aggregation criteria:

$$I_{W'} - I_W = \frac{n_k \cdot n_\ell}{n_k + n_\ell} \|\boldsymbol{m}_k - \boldsymbol{m}_\ell\|^2.$$

Thus, each stage of Ward's algorithm chooses a new partition that limits the increase of intra-class inertia. Note that this property does not guarantee overall optimization of the criteria.

Ultrametric distance

An ultrametric distance δ checks all the properties that define a classical distance and satisfies in addition the inequality

$$\delta(\boldsymbol{x}, \boldsymbol{z}) \leq \max\left(\delta(\boldsymbol{x}, \boldsymbol{z}), \delta(\boldsymbol{z}, \boldsymbol{y})\right),$$

stronger than the triangular inequality.

- it is possible to interpret the minimum number of nestings required for two form vectors to belong to one class, as a dissimilarity.
- this dissimilarity is an ultrametric distance.
- it is possible to understand the problem of hierarchical clustering as the search for an ultrametric δ close to d, the dissimilarity used on Ω

Divisive hierarchical clustering

- the divisive clustering approach is much less popular
- In theory, the first step of a downward method must compare \$2ˆ{N-1}-1 \$ possible partitions from N vectors, in two classes.
- To avoid those impossible calculations, one solution is to apply a method of partitioning to get both classes. Repeating this process recursively on each class obtained allows to have fast algorithms

Hierarchical agglomerative clustering with R I

agnes

Agglomerative Nesting Description: Computes agglomerative hierarchical clustering of the dataset. Usage: agnes(x, diss = inherits(x, "dist"), metric = "euclidean", stand = FALSE, method = "average", keep.diss = n < 100, keep.data = !diss)

Example of use I

```
## Warning: package 'MASS' was built under R version 3.4.4
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(cluster)
```

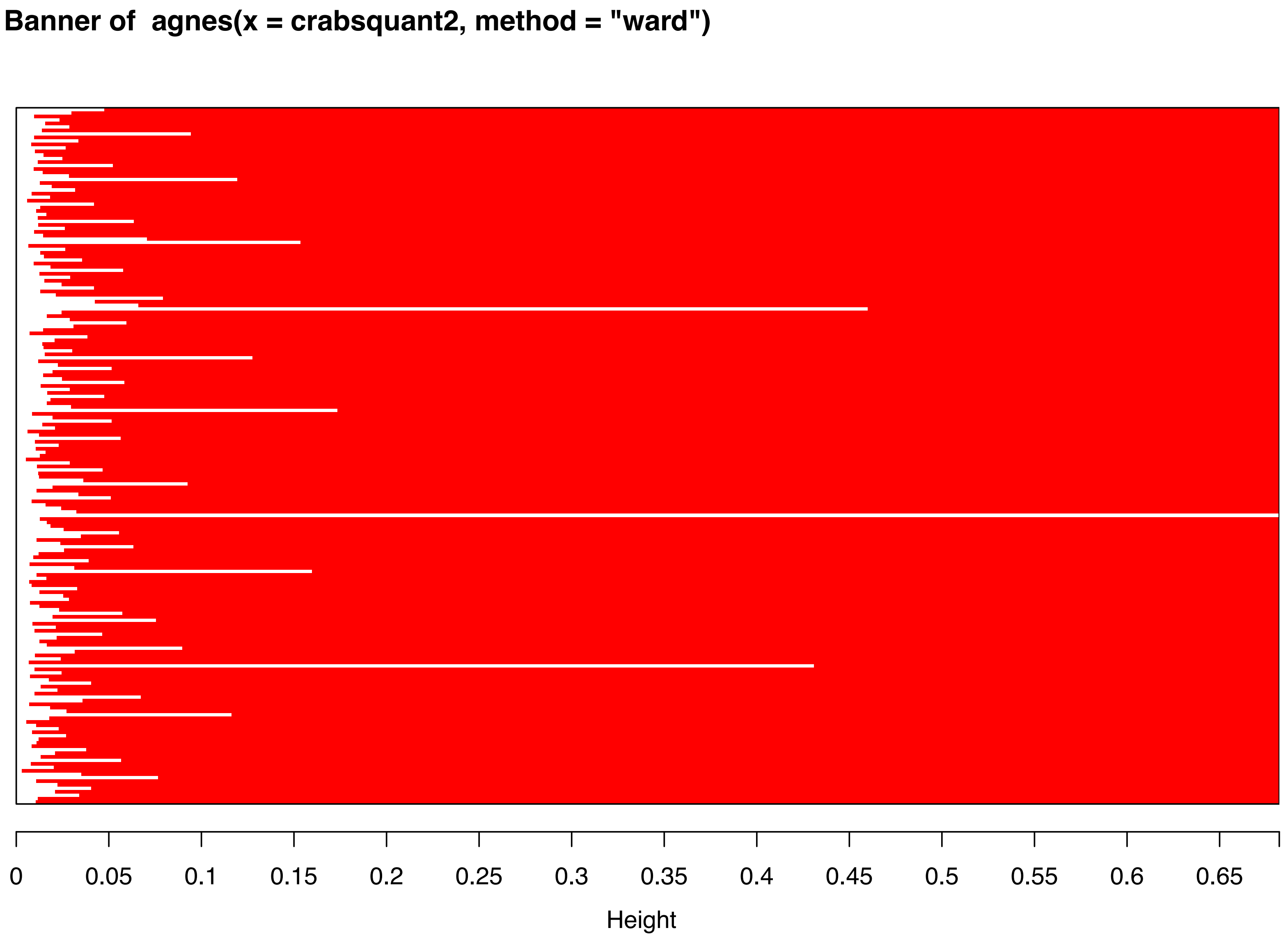
```
## Warning: package 'cluster' was built under R version 3.4.4
```

Example of use II

```r
res<-agnes(crabsquant2,method="ward")
```

```r
plot(res,which.plots=1)
```

Example of use III

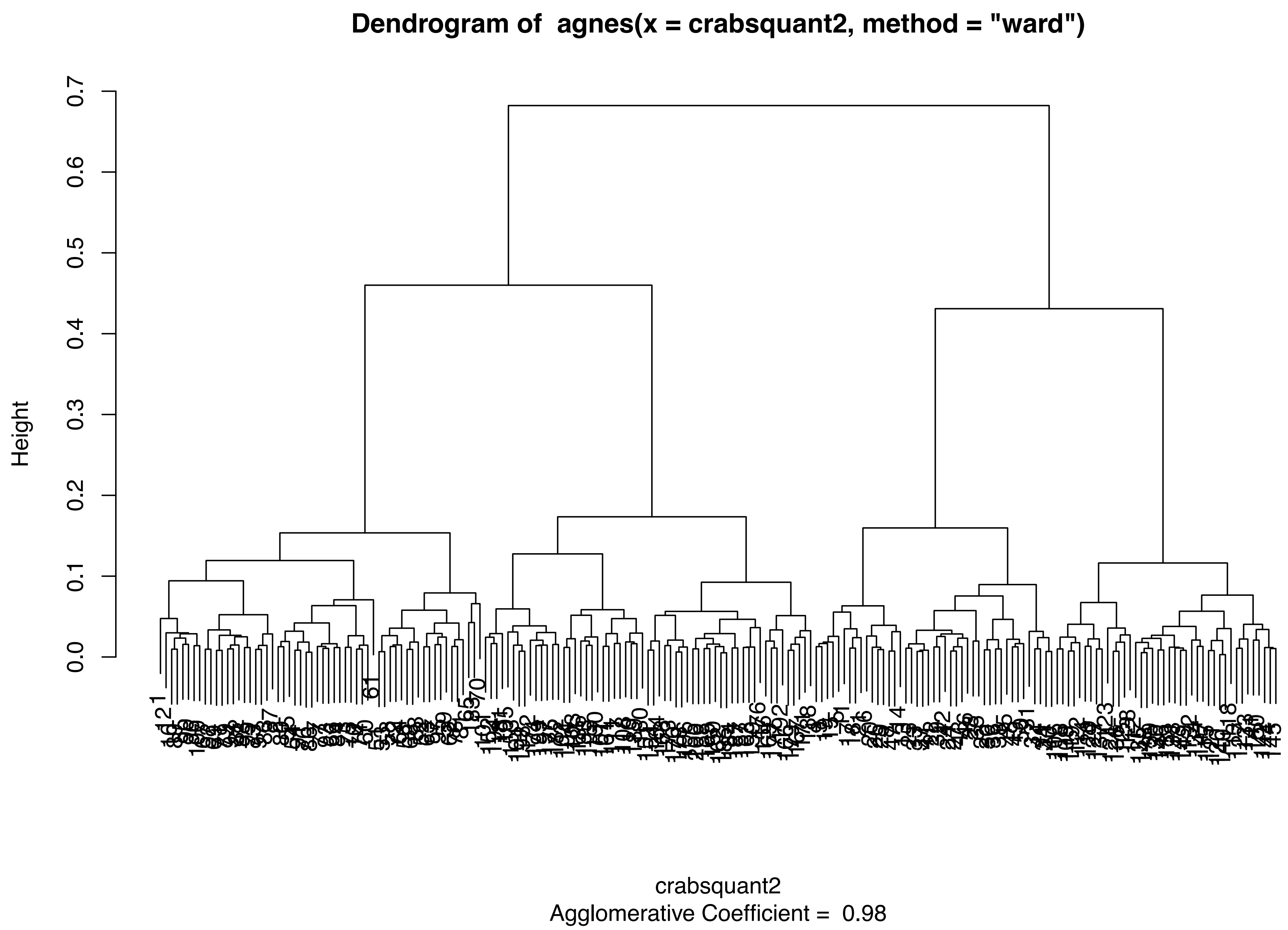


Agglomerative Coefficient = 0.98

Example of use IV

```
plot(res,which.plots=2)
```

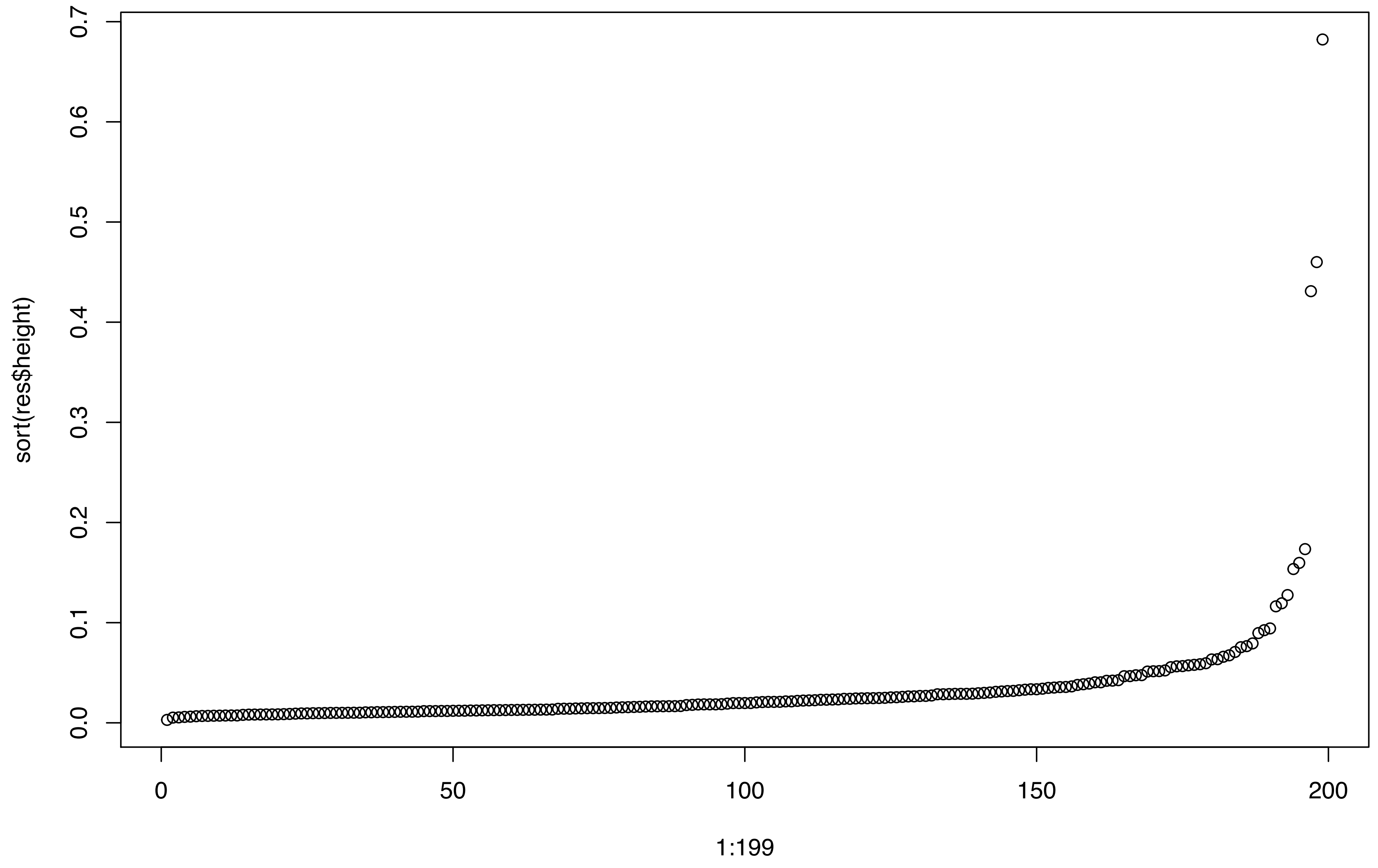
Example of use V



Example of use VI

```r
plot(1:199,sort(res$height))
```

Example of use VII



Divisive Hiearchical Clustering with R

diana

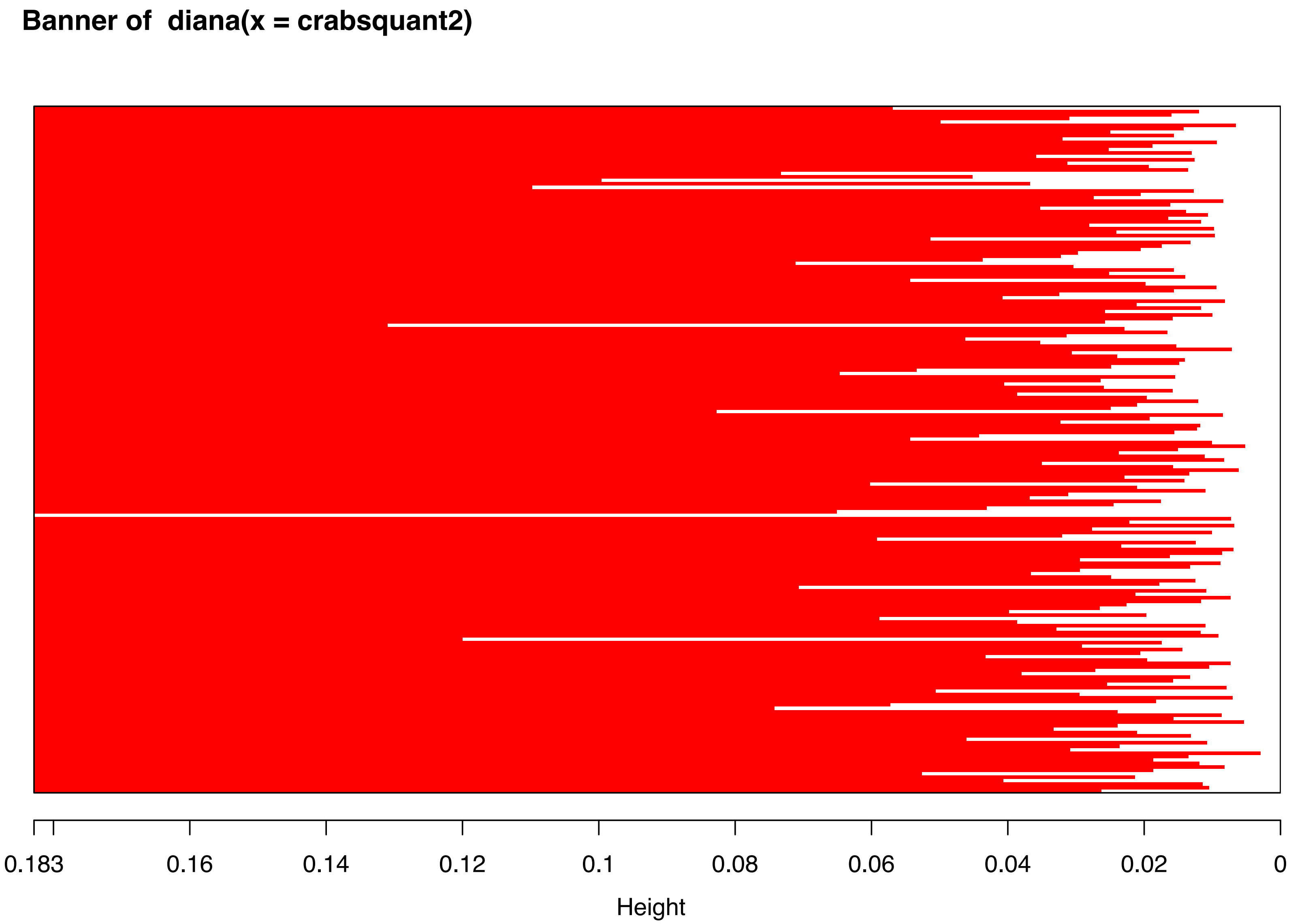
DIvisive ANAlysis Clustering Description: Computes a divisive hierarchical clustering of the dataset returning an object of class 'diana'. Usage: diana(x, diss = inherits(x, "dist"), metric = "euclidean", stand = >FALSE, keep.diss = n < 100, keep.data = !diss)

Example of use I

```r
res<-diana(crabsquant2)
```

```r
plot(res,which.plots=1)
```

Example of use II

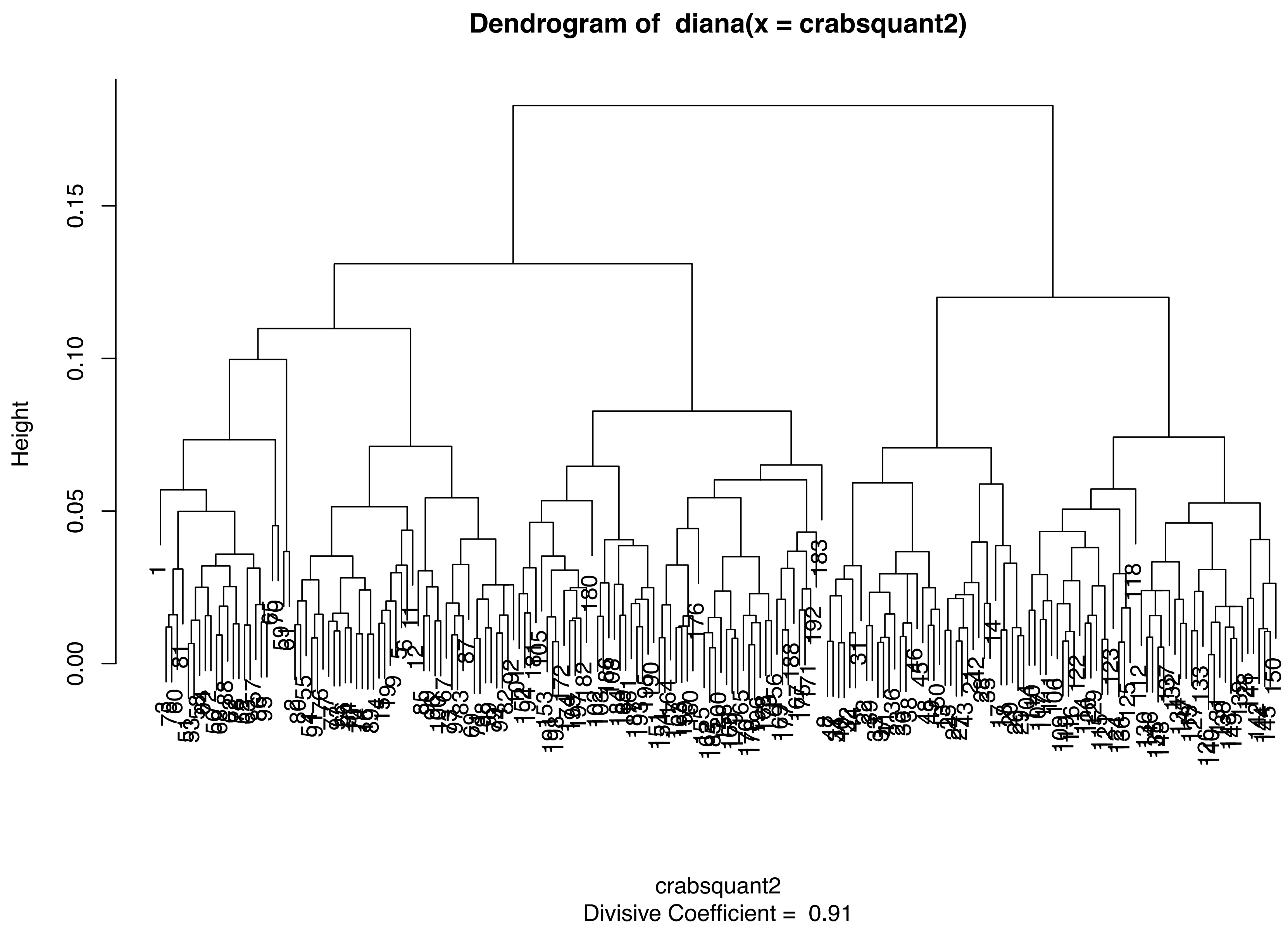


Divisive Coefficient = 0.91

Example of use III

```
plot(res,which.plots=2)
```

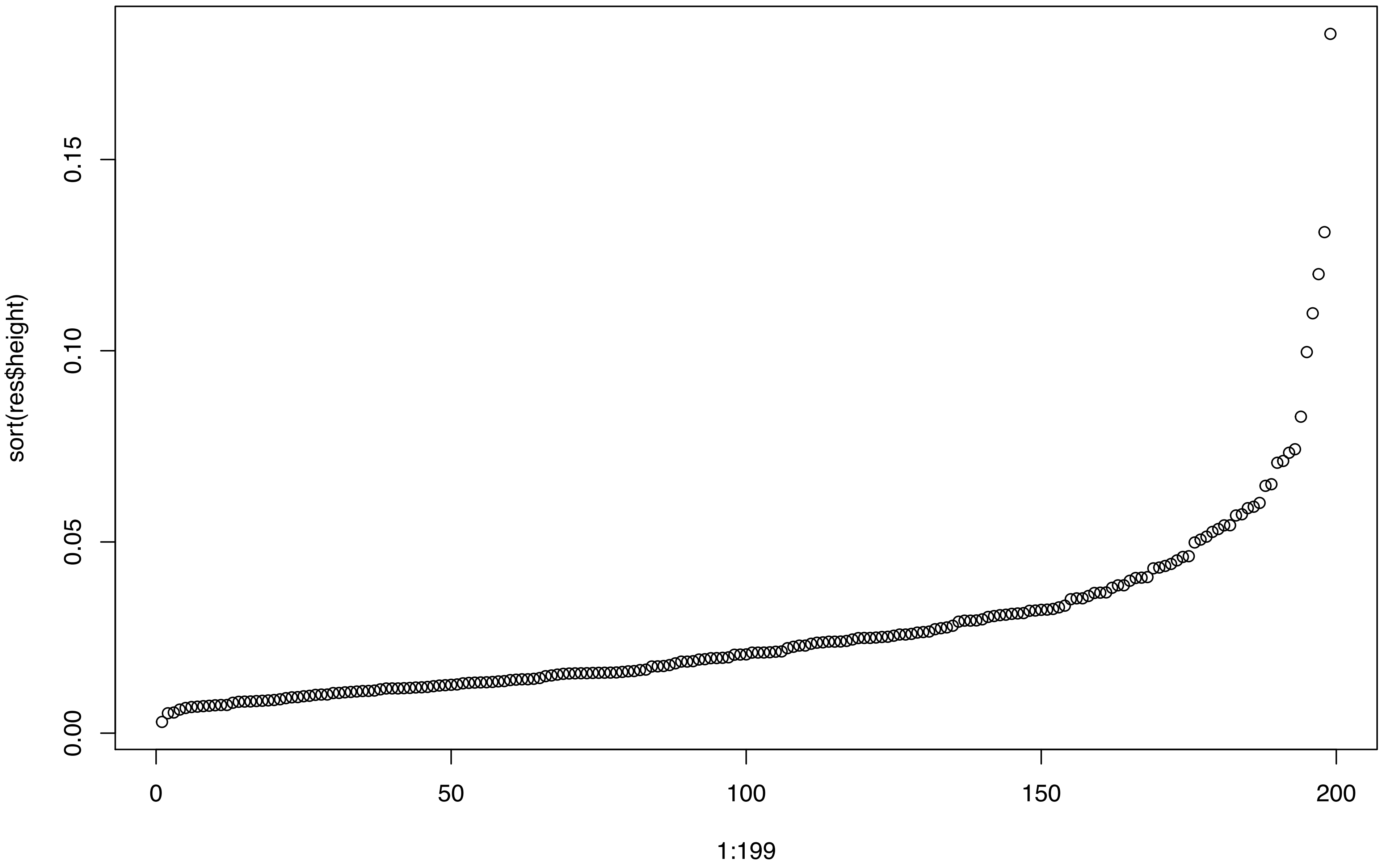
Example of use IV



Example of use V

```
plot(1:199,sort(res$height))
```

Example of use VI


sort(res$height)
1:199