Cutting the dendrogram through permutation tests

Dario Bruzzese
dbruzzes@unina.it

Department of
Preventive Medical Sciences

UNIVERSITY OF NAPLES ITALY

Domenico Vistocco
vistocco@unicas.it

Department of
Economics

UNIVERSITY OF CASSINO ITALY

La Carte

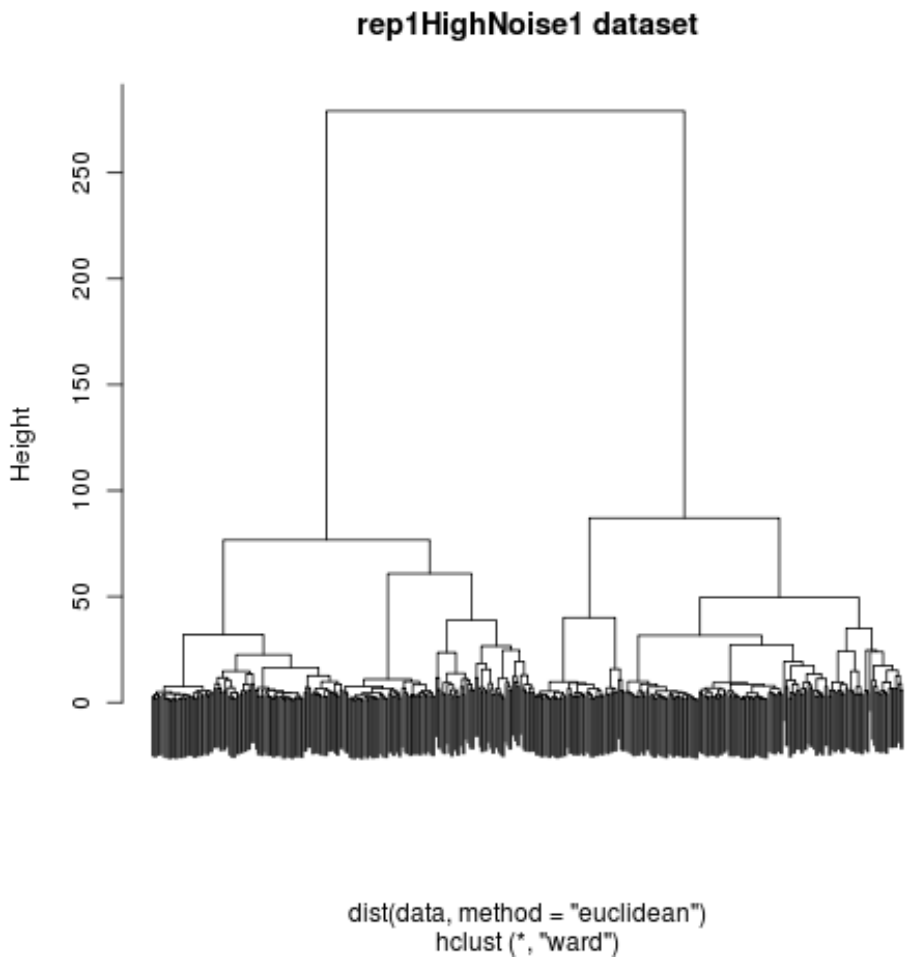
La Carte

Motivation

- Automatically determine the *optimal* cut-off level of a dendrogram
- Explore partitions different from those allowed by an horizontal cut

Motivation

- Automatically determine the *optimal* cut-off level of a dendrogram
- Explore partitions different from those allowed by an horizontal cut



The rep1HighNoise dataset

Yeung KY, Medvedovic M, Bumgarner KY: Clustering gene-expression data with repeated measurements.

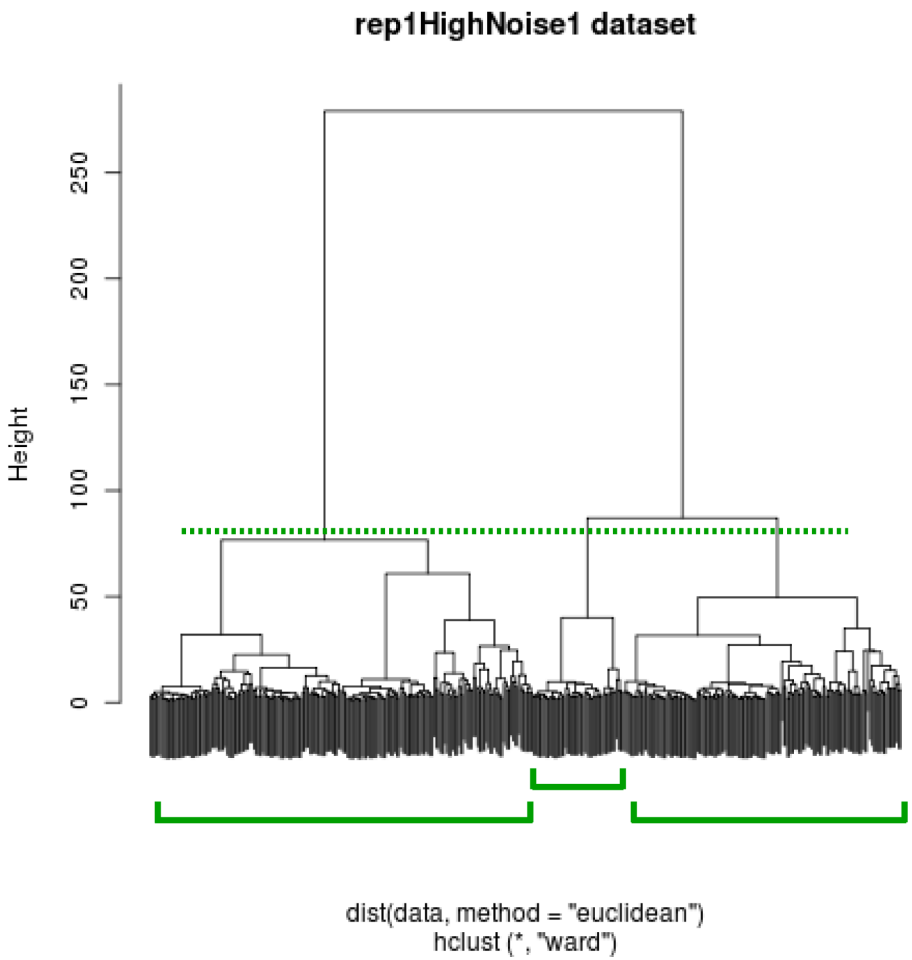
Genome Biology, 2003, 4:R34

$n = 200$

$p = 20$

Motivation

- Automatically determine the *optimal* cut-off level of a dendrogram
- Explore partitions different from those allowed by an horizontal cut

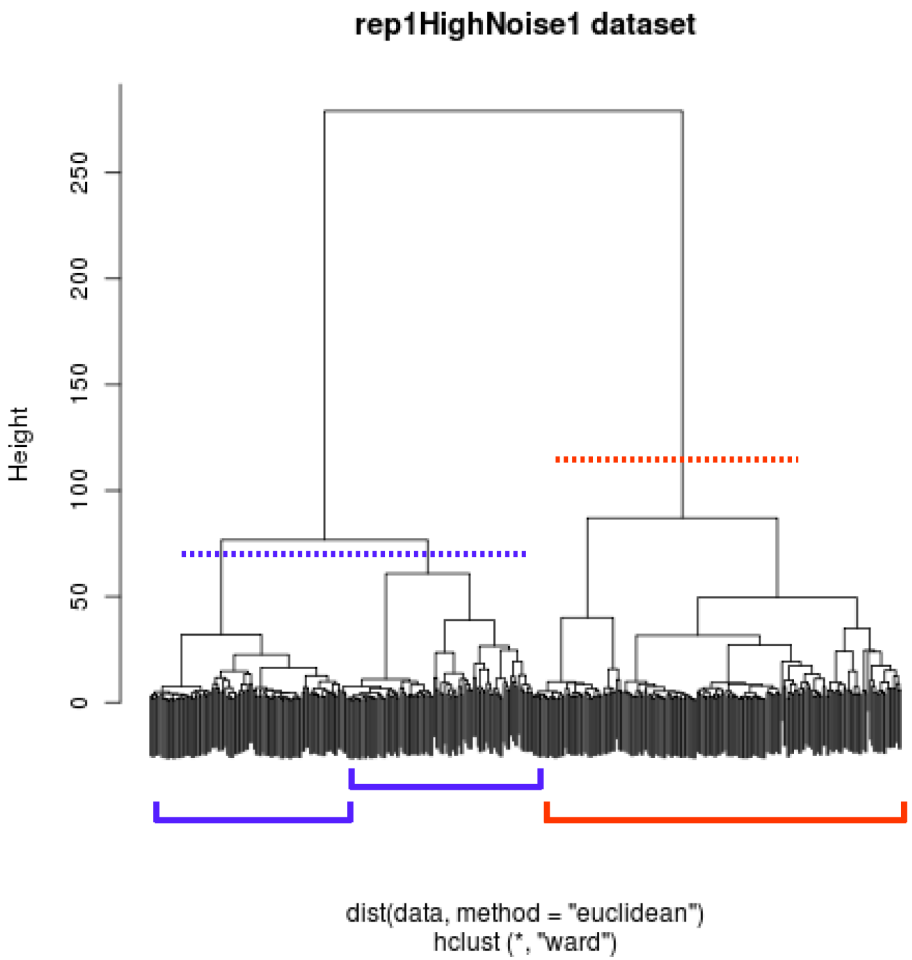


Horizontal cut

$k = 3$

Motivation

- Automatically determine the *optimal* cut-off level of a dendrogram
- Explore partitions different from those allowed by an horizontal cut

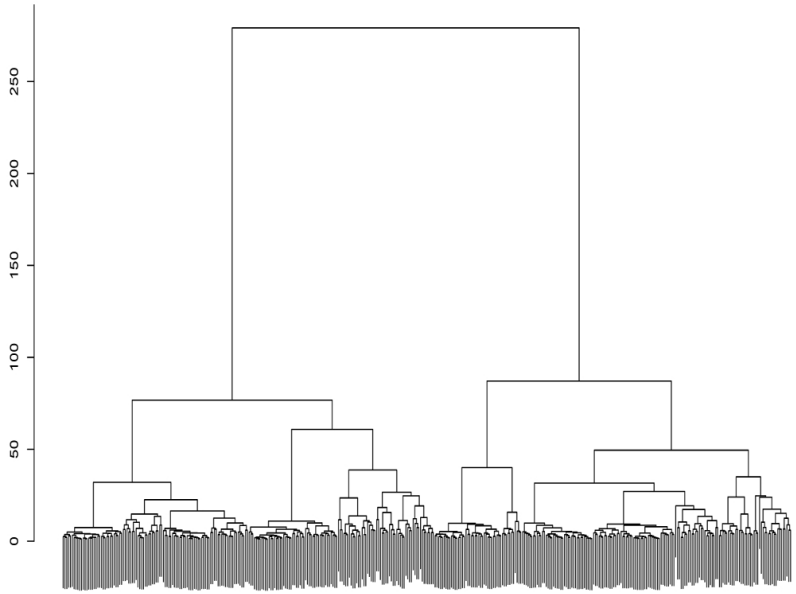


An alternative cut

$k = 3$

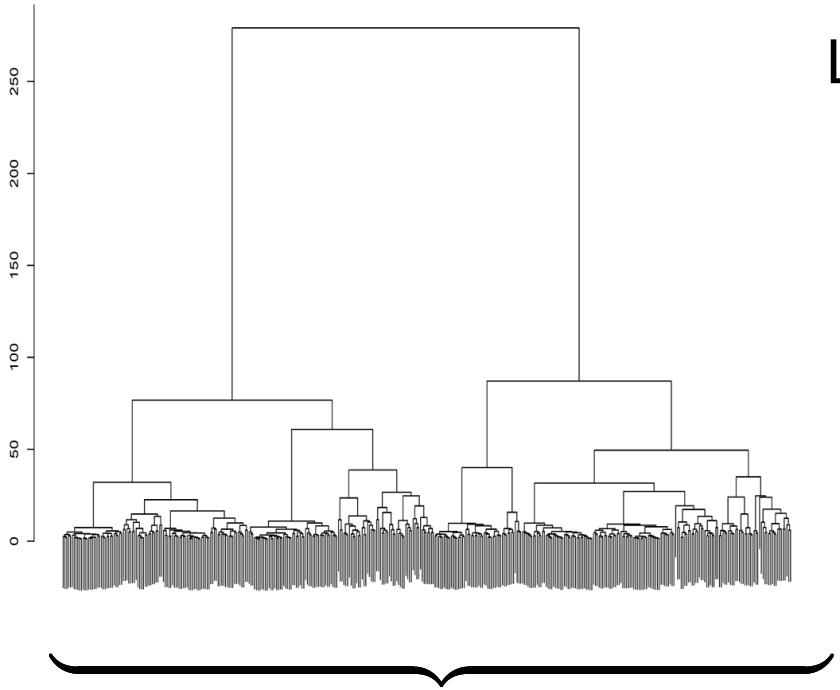
La Carte

Notation



Let:

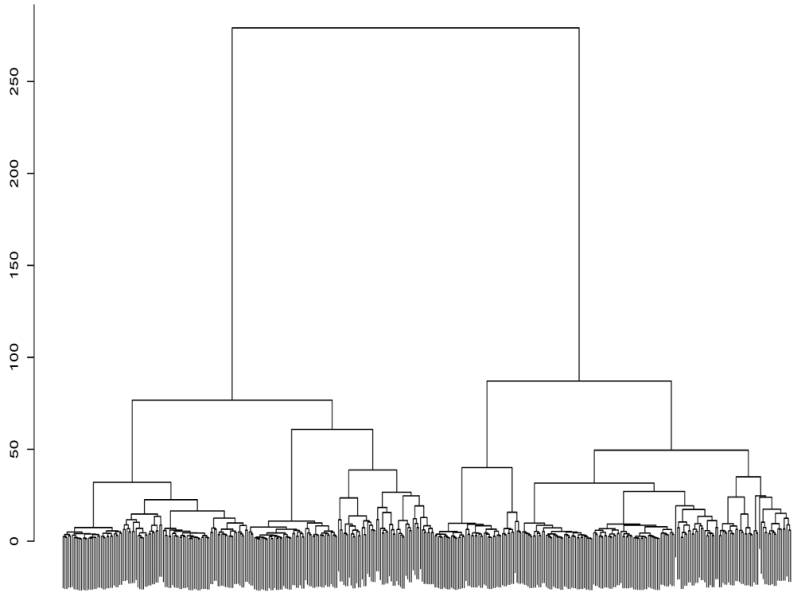
Notation



Let:

- n the number of objects to classify;

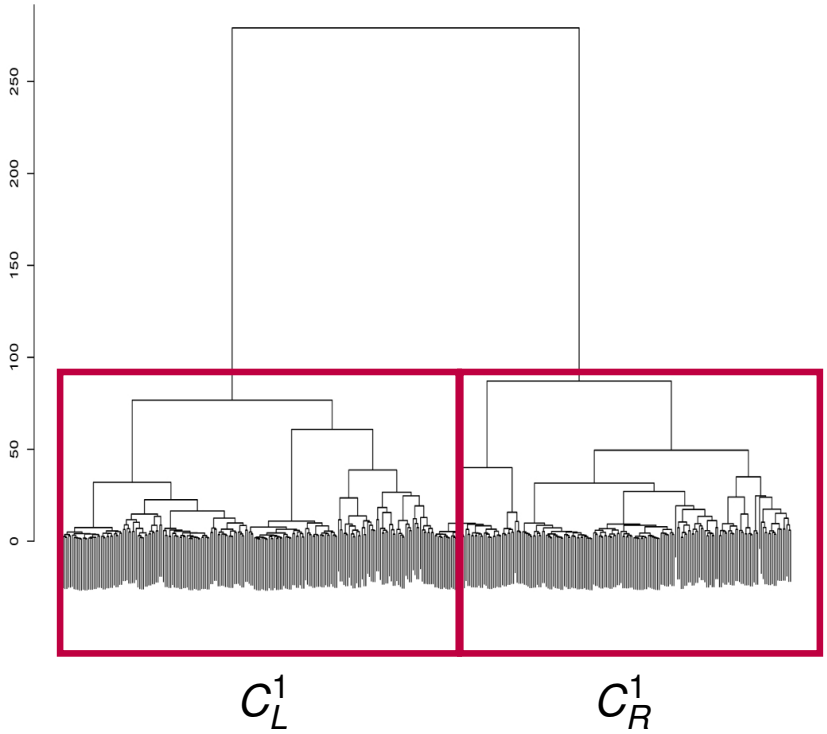
Notation



Let:

- n the number of objects to classify;
- C_L^k and C_R^k the two classes merged at level k (k=1,...,n-1)

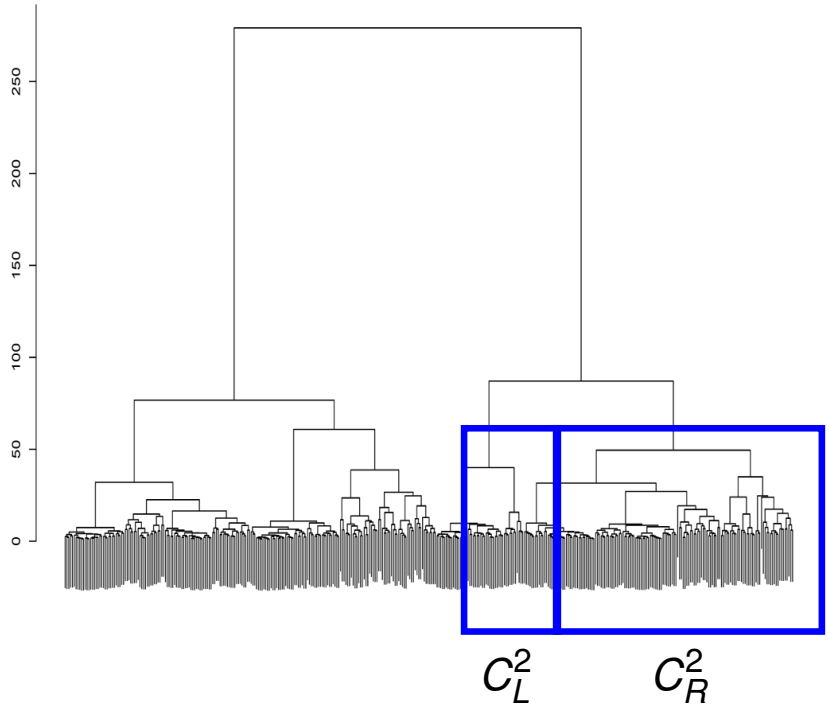
Notation



Let:

- n the number of objects to classify;
- C_L^k and C_R^k the two classes merged at level k (k=1,...,n-1)

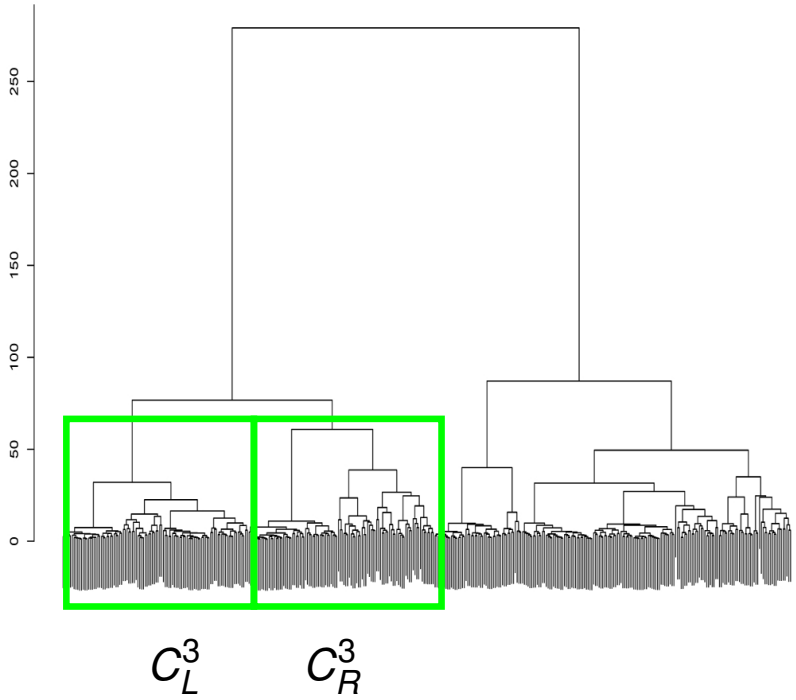
Notation



Let:

- n the number of objects to classify;
- C_L^k and C_R^k the two classes merged at level k (k=1,...,n-1)

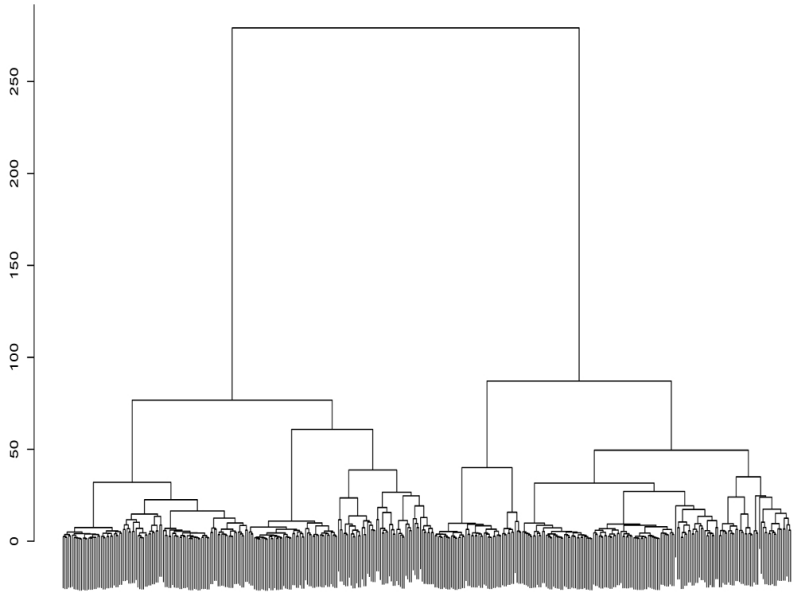
Notation



Let:

- n the number of objects to classify;
- C_L^k and C_R^k the two classes merged at level k (k=1,...,n-1)

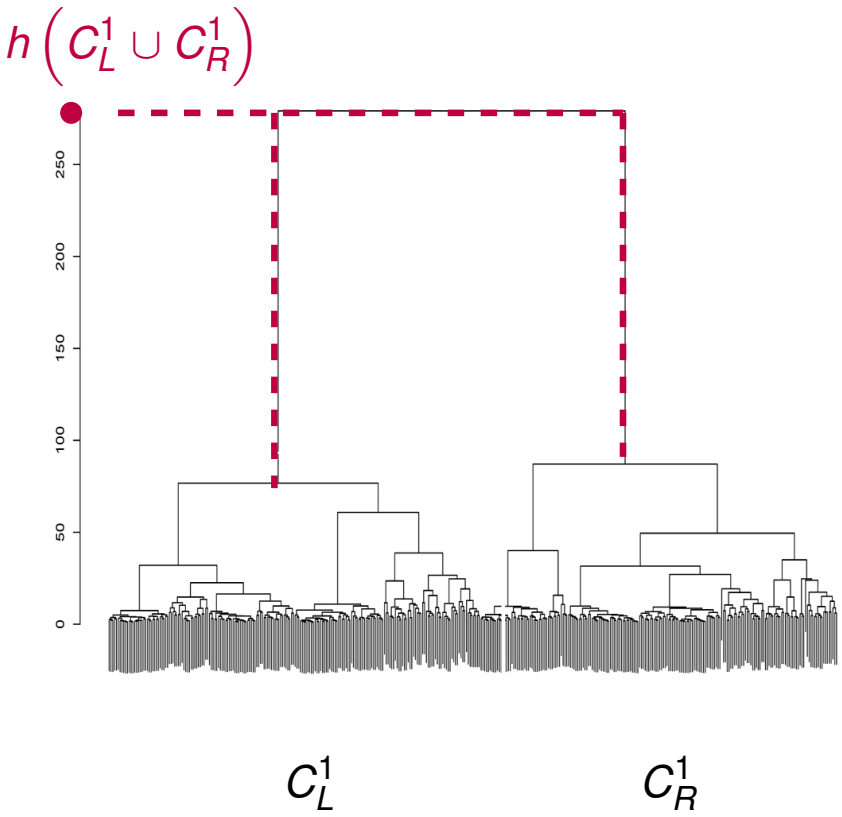
Notation



Let:

- n the number of objects to classify;
- C_L^k and C_R^k the two classes merged at level k (k=1,...,n-1)
- $h\left(C_L^k \cup C_R^k\right)$ the height necessary to merge C_L^k and C_R^k

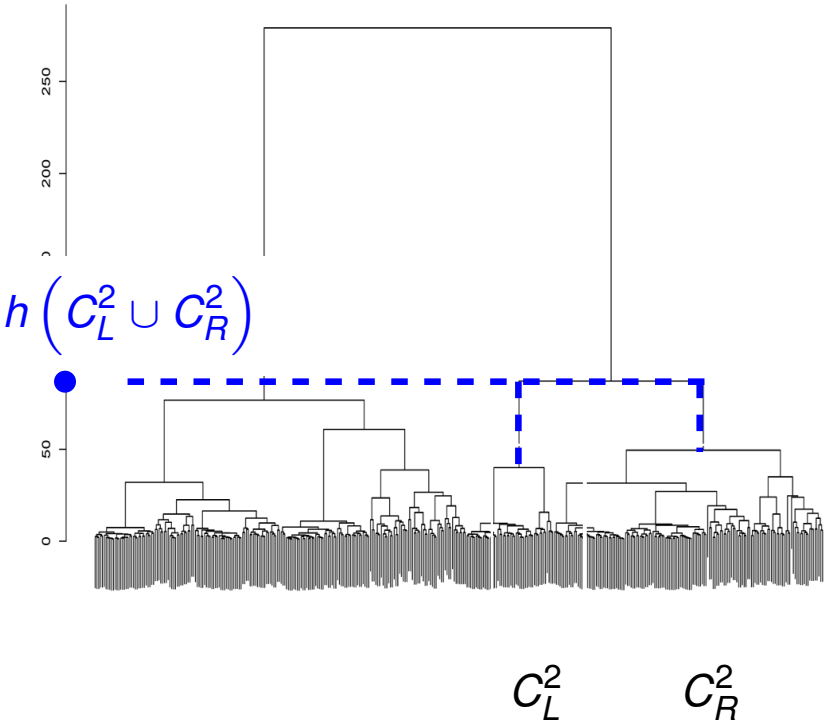
Notation



Let:

- n the number of objects to classify;
- C_L^k and C_R^k the two classes merged at level k (k=1,...,n-1)
- $h\left(C_L^k \cup C_R^k\right)$ the height necessary to merge C_L^k and C_R^k

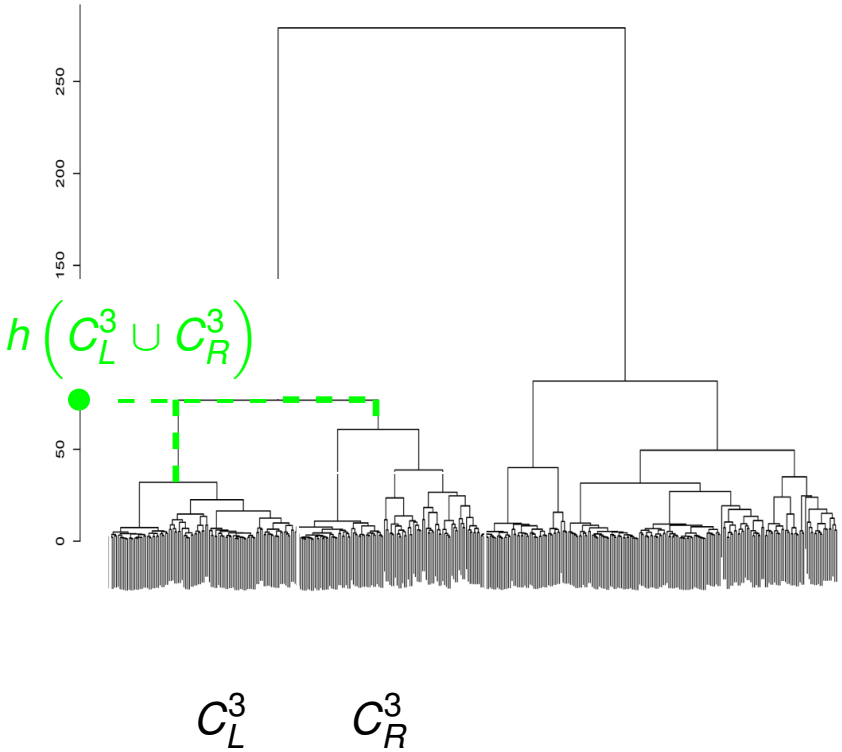
Notation



Let:

- n the number of objects to classify;
- C_L^k and C_R^k the two classes merged at level k (k=1,...,n-1)
- $h\left(C_L^k \cup C_R^k\right)$ the height necessary to merge C_L^k and C_R^k

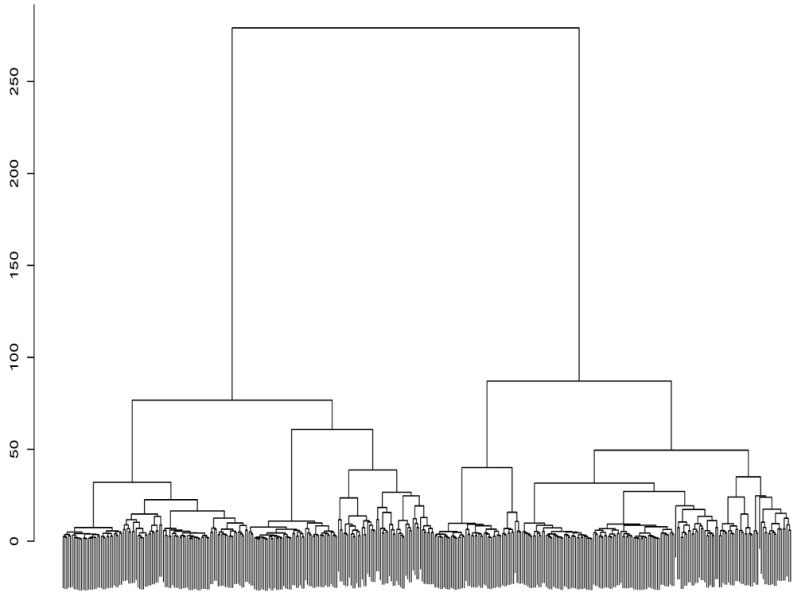
Notation



Let:

- n the number of objects to classify;
- C_L^k and C_R^k the two classes merged at level k (k=1,...,n-1)
- $h\left(C_L^k \cup C_R^k\right)$ the height necessary to merge C_L^k and C_R^k

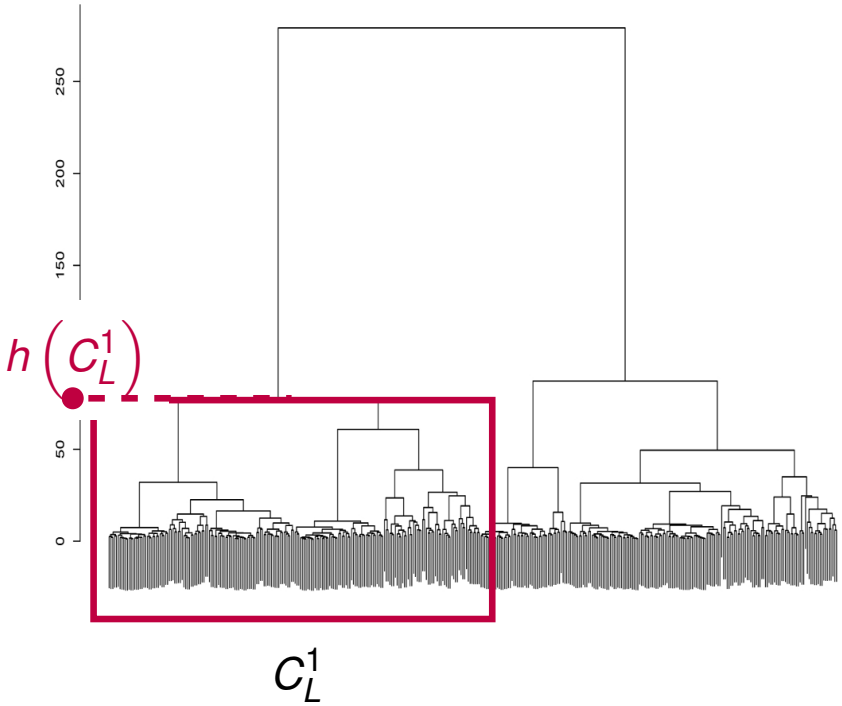
Notation



Let:

- n the number of objects to classify;
- C_L^k and C_R^k the two classes merged at level k (k=1,...,n-1)
- $h\left(C_L^k \cup C_R^k\right)$ the height necessary to merge C_L^k and C_R^k
- $h\left(C_j^k\right)$ the height at which C_j^k has been obtained ($j \in \{ L, R \}$)

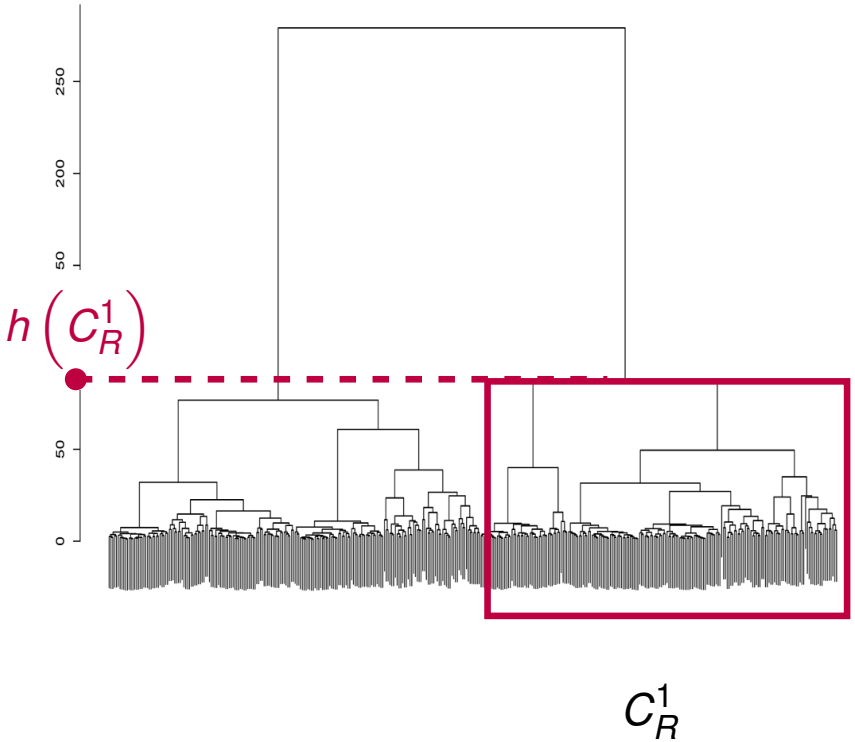
Notation



Let:

- n the number of objects to classify;
- C_L^k and C_R^k the two classes merged at level k (k=1,...,n-1)
- $h\left(C_L^k \cup C_R^k\right)$ the height necessary to merge C_L^k and C_R^k
- $h\left(C_j^k\right)$ the height at which C_j^k has been obtained ($j \in \{ L, R \}$)

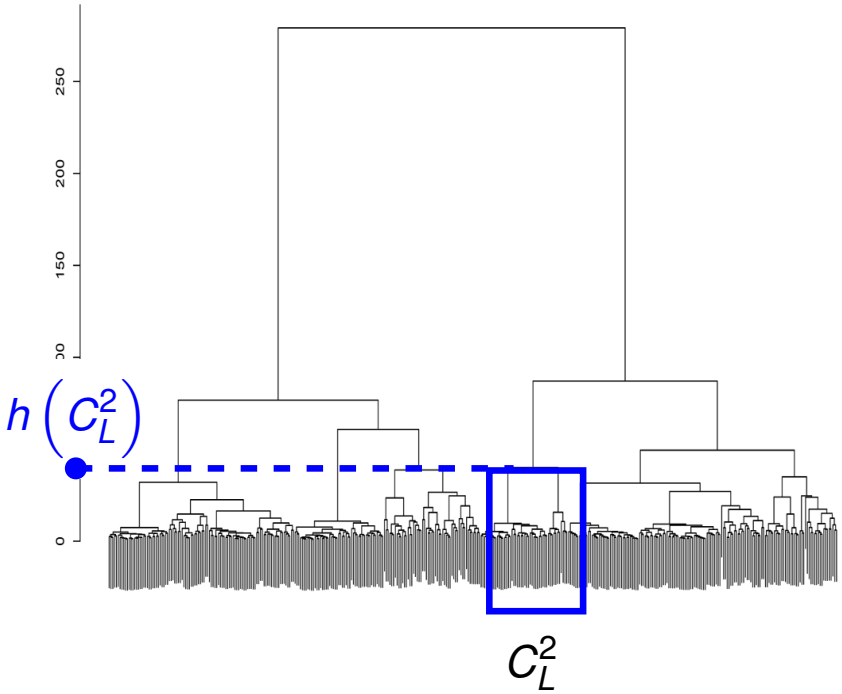
Notation



Let:

- n the number of objects to classify;
- C_L^k and C_R^k the two classes merged at level k (k=1,...,n-1)
- $h\left(C_L^k \cup C_R^k\right)$ the height necessary to merge C_L^k and C_R^k
- $h\left(C_j^k\right)$ the height at which C_j^k has been obtained ($j \in \{ L, R \}$)

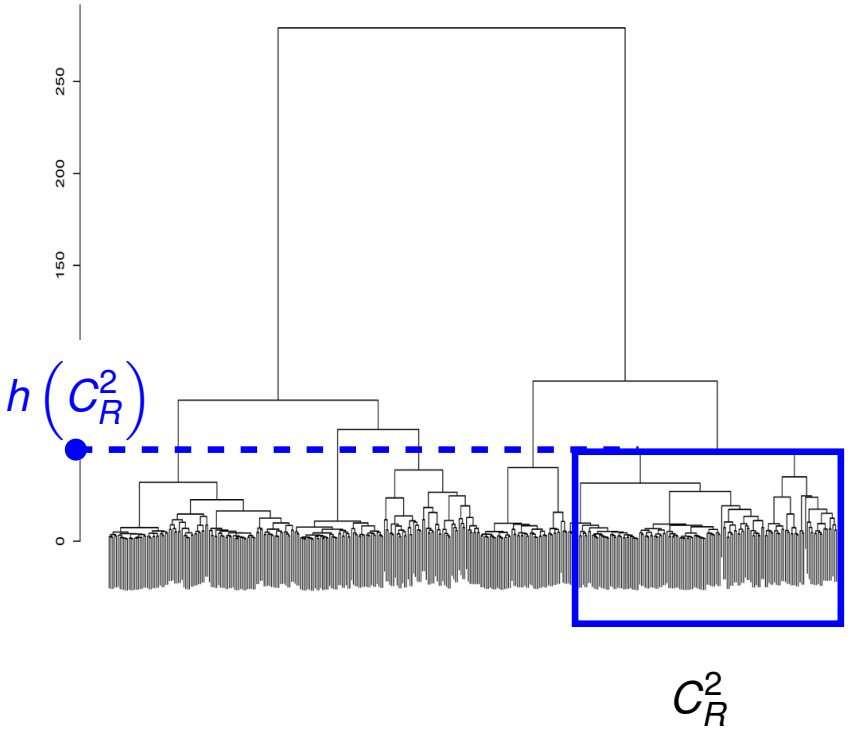
Notation



Let:

- n the number of objects to classify;
- C_L^k and C_R^k the two classes merged at level k (k=1,...,n-1)
- $h\left(C_L^k \cup C_R^k\right)$ the height necessary to merge C_L^k and C_R^k
- $h\left(C_j^k\right)$ the height at which C_j^k has been obtained ($j \in \{ L, R \}$)

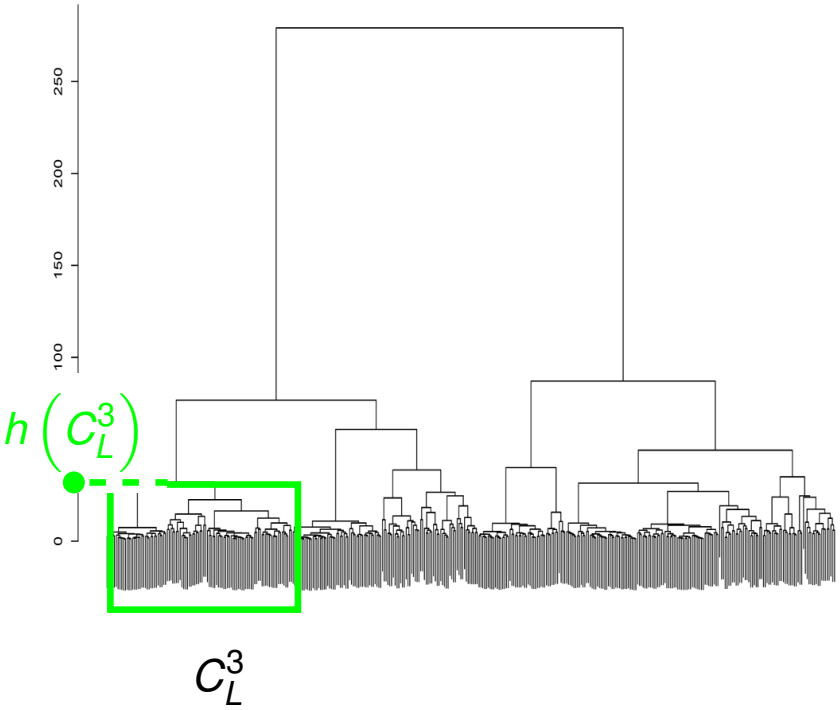
Notation



Let:

- n the number of objects to classify;
- C_L^k and C_R^k the two classes merged at level k (k=1,...,n-1)
- $h\left(C_L^k \cup C_R^k\right)$ the height necessary to merge C_L^k and C_R^k
- $h\left(C_j^k\right)$ the height at which C_j^k has been obtained ($j \in \{ L, R \}$)

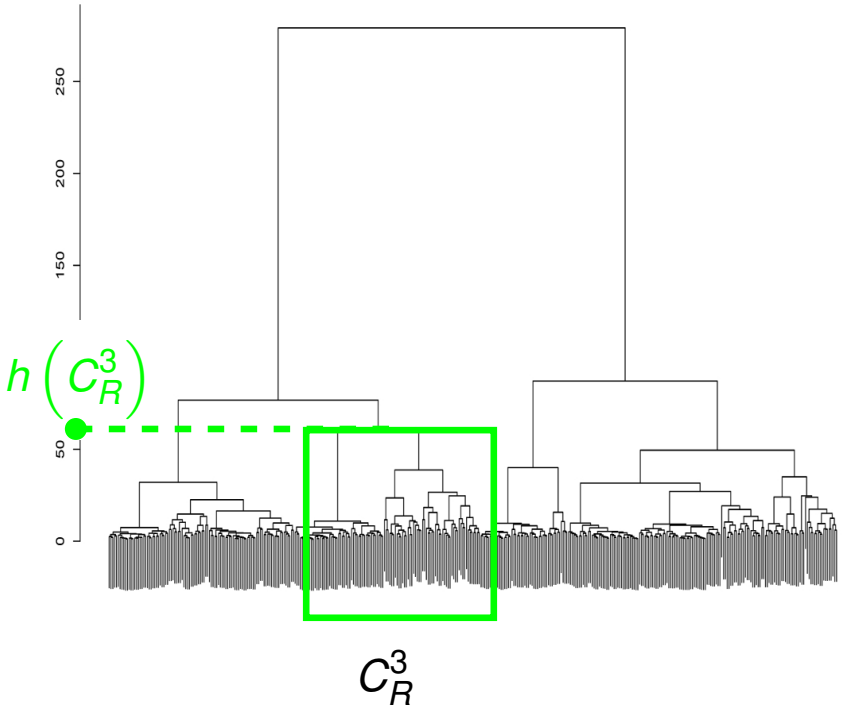
Notation



Let:

- n the number of objects to classify;
- C_L^k and C_R^k the two classes merged at level k (k=1,...,n-1)
- $h\left(C_L^k \cup C_R^k\right)$ the height necessary to merge C_L^k and C_R^k
- $h\left(C_j^k\right)$ the height at which C_j^k has been obtained ($j \in \{ L, R \}$)

Notation



Let:

- n the number of objects to classify;
- C_L^k and C_R^k the two classes merged at level k (k=1,...,n-1)
- $h\left(C_L^k \cup C_R^k\right)$ the height necessary to merge C_L^k and C_R^k
- $h\left(C_j^k\right)$ the height at which C_j^k has been obtained ($j \in \{ L, R \}$)

The algorithm - Pseudo Code

Input: A dataset and its related dendrogram
Output: A partition of the dataset

The algorithm - Pseudo Code

Input: A dataset and its related dendrogram
Output: A partition of the dataset

initialization:
aggregationLevelsToVisit $\leftarrow h(C_L^1 \cup C_R^1)$
permClusters $\leftarrow$ [ ]
i $\leftarrow$ 1

The algorithm - Pseudo Code

Input: A dataset and its related dendrogram
Output: A partition of the dataset

initialization:
aggregationLevelsToVisit $\leftarrow h(C_L^1 \cup C_R^1)$
permClusters $\leftarrow$ [ ]
i $\leftarrow$ 1
repeat
  if $C_L^i \equiv C_R^i$ **then**
    add $C_L^i \cup C_R^i$ to permClusters
  else
    add $h(C_L^i)$ and $h(C_R^i)$ to aggregationLevelsToVisit
    sort aggregationLevelsToVisit in descending order
  end

The algorithm - Pseudo Code

Input: A dataset and its related dendrogram
Output: A partition of the dataset

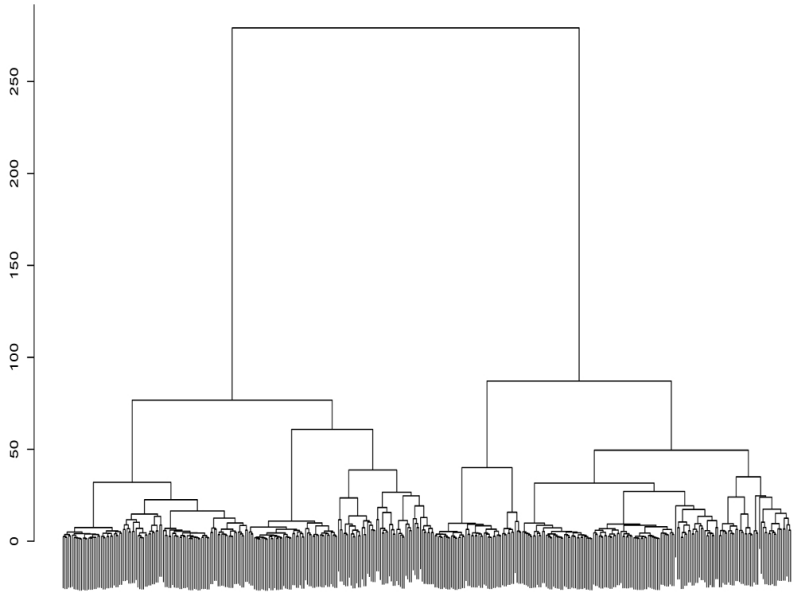
initialization:
aggregationLevelsToVisit $\leftarrow h(C_L^1 \cup C_R^1)$
permClusters $\leftarrow$ [ ]
i $\leftarrow$ 1
repeat
  if $C_L^i \equiv C_R^i$ **then**
    add $C_L^i \cup C_R^i$ to permClusters
  else
    add $h(C_L^i)$ and $h(C_R^i)$ to aggregationLevelsToVisit
    sort aggregationLevelsToVisit in descending order
  end
  remove the first element from aggregationLevelsToVisit
  i $\leftarrow$ i+1

The algorithm - Pseudo Code

Input: A dataset and its related dendrogram
Output: A partition of the dataset

initialization:
aggregationLevelsToVisit $\leftarrow h(C_L^1 \cup C_R^1)$
permClusters $\leftarrow$ [ ]
i $\leftarrow$ 1
repeat
  if $C_L^i \equiv C_R^i$ **then**
    add $C_L^i \cup C_R^i$ to permClusters
  else
    add $h(C_L^i)$ and $h(C_R^i)$ to aggregationLevelsToVisit
    sort aggregationLevelsToVisit in descending order
  end
  remove the first element from aggregationLevelsToVisit
  i $\leftarrow$ i+1
until *aggregationLevelsToVisit is empty*

The algorithm - The outline



Initialization

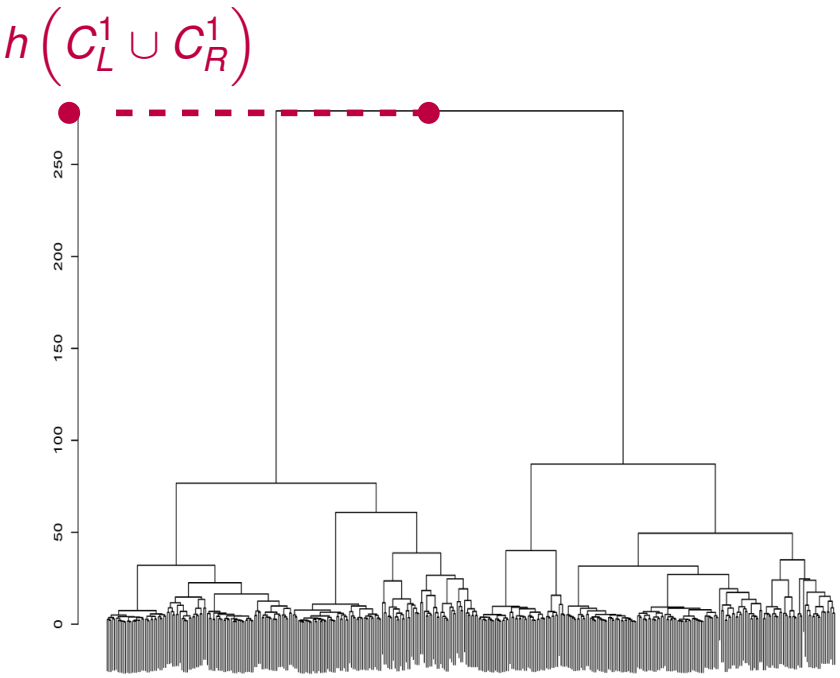
$i \leftarrow 0$

aggregationLevelsToVisit

$h(C_L^1 \cup C_R^1)$

permClusters

The algorithm - The outline



Iteration

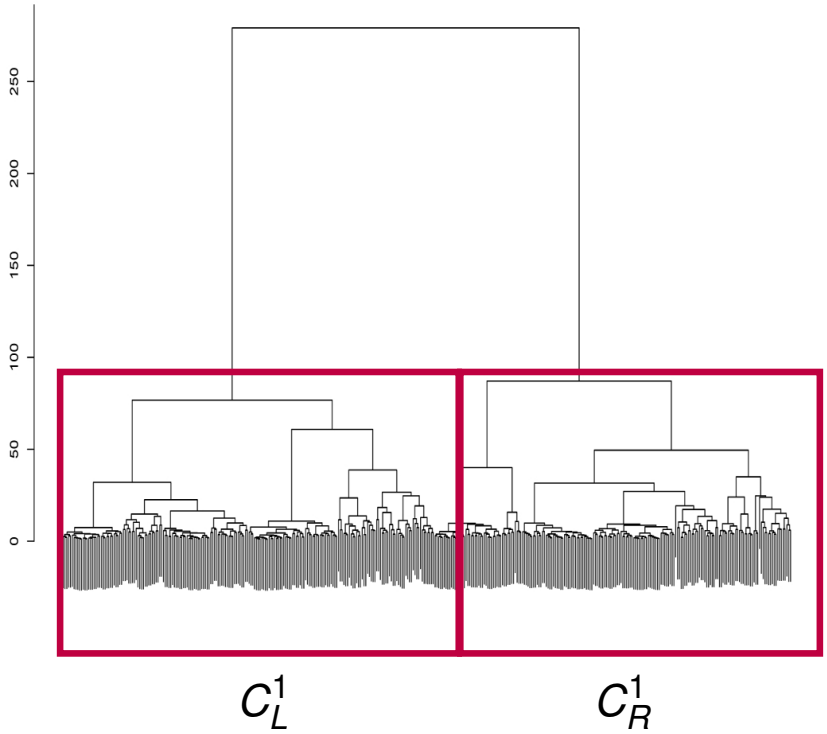
$i \leftarrow 1$

aggregationLevelsToVisit

$h(C_L^1 \cup C_R^1)$

permClusters

The algorithm - The outline



Iteration

$i \leftarrow 1$

aggregationLevelsToVisit

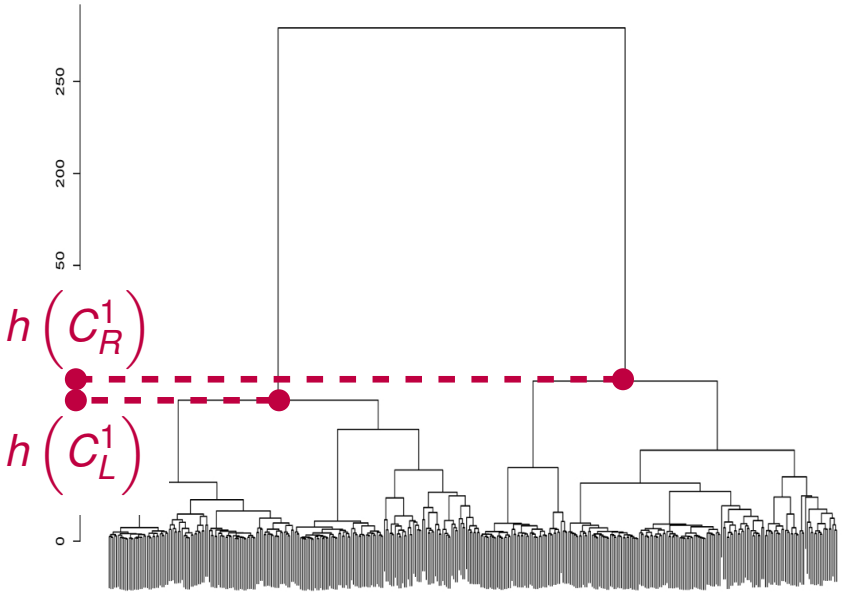
$h(C_L^1 \cup C_R^1)$

permClusters

clusters to compare

$H_0 : C_L^1 \equiv C_R^1 \mapsto$ reject

The algorithm - The outline



Iteration

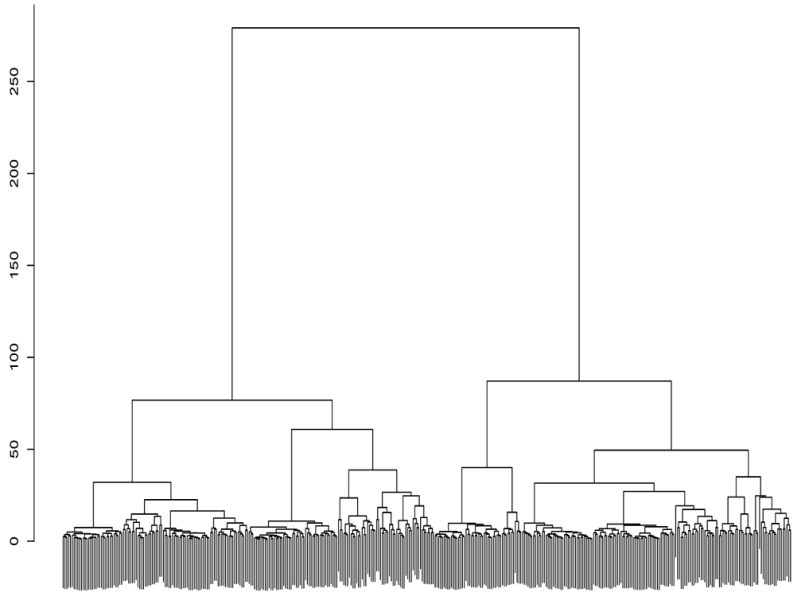
$i \leftarrow 1$

aggregationLevelsToVisit

$h(C_L^1 \cup C_R^1), h(C_R^1), h(C_L^1)$

permClusters

The algorithm - The outline



Iteration

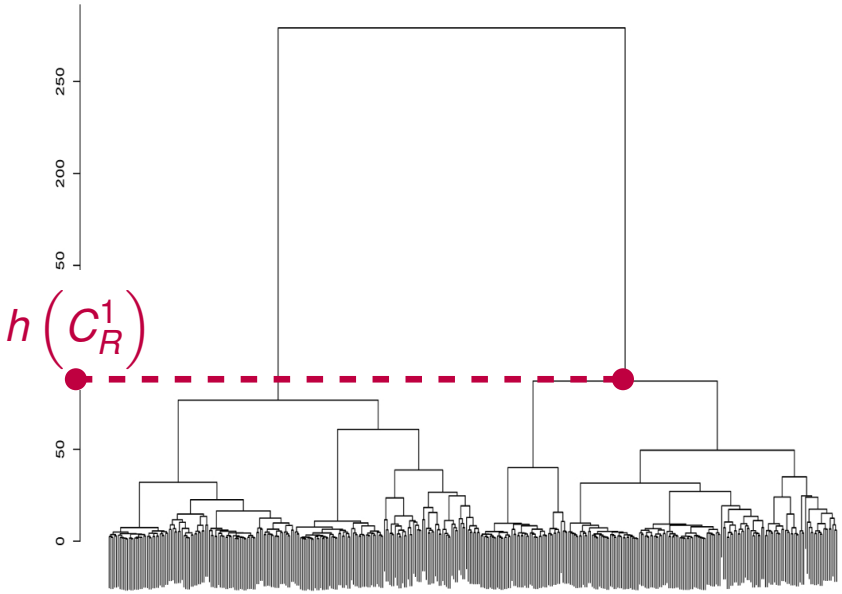
$i \leftarrow 1$

aggregationLevelsToVisit

$h(C_R^1), h(C_L^1)$

permClusters

The algorithm - The outline



Iteration

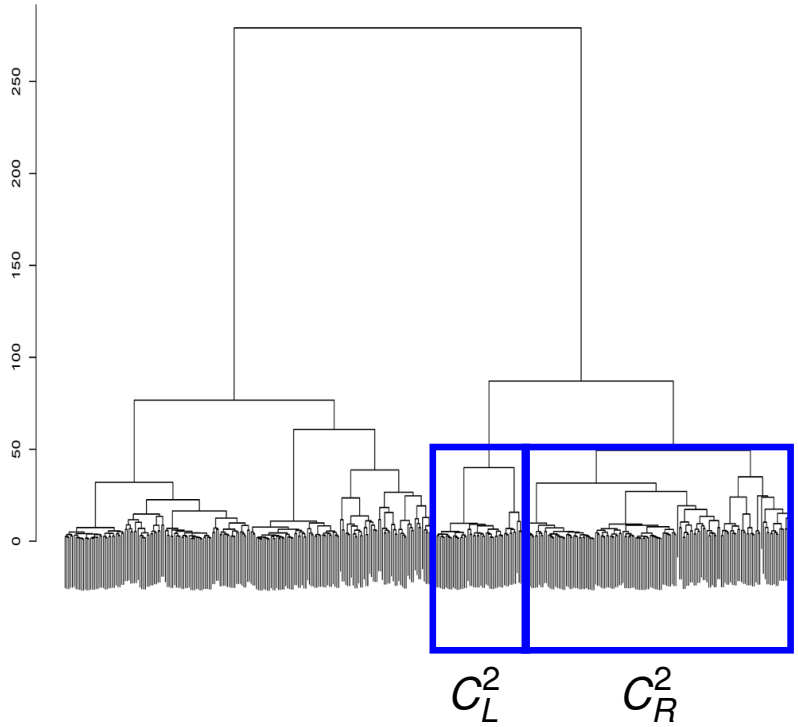
$i \leftarrow 2$

aggregationLevelsToVisit

$h(C_R^1), h(C_L^1)$

permClusters

The algorithm - The outline



Iteration

$i \leftarrow 2$

aggregationLevelsToVisit

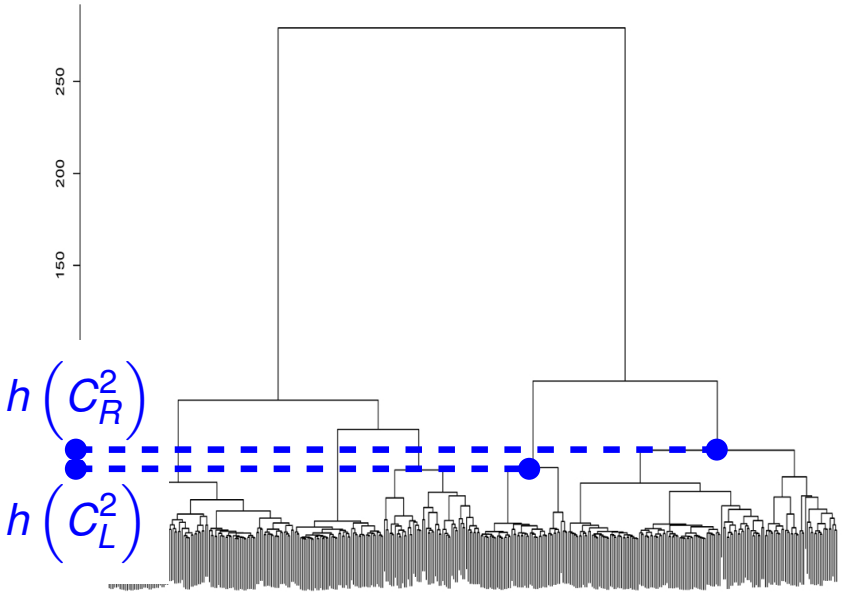
$h(C_R^1), h(C_L^1)$

permClusters

clusters to compare

$H_0 : C_L^2 \equiv C_R^2 \mapsto$ reject

The algorithm - The outline



Iteration

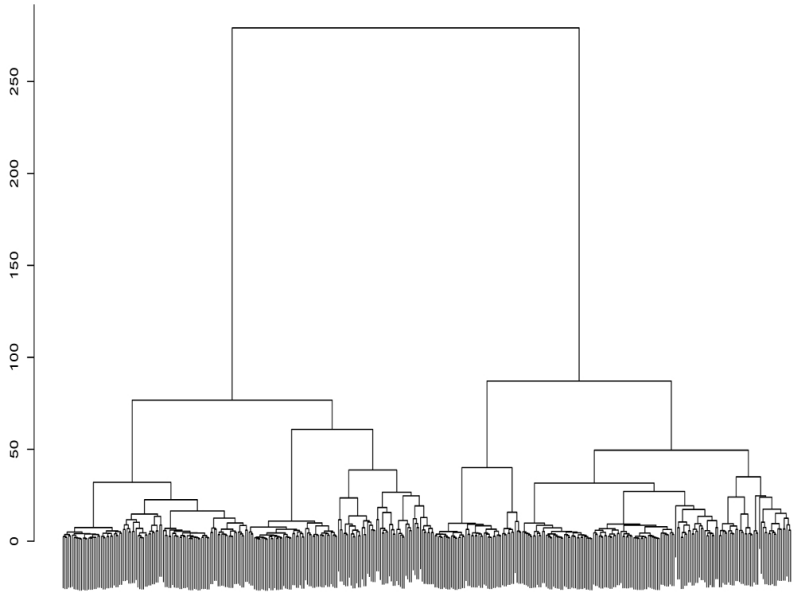
$i \leftarrow 2$

aggregationLevelsToVisit

$h(C_R^1), h(C_L^1), h(C_R^2), h(C_L^2)$

permClusters

The algorithm - The outline



Iteration

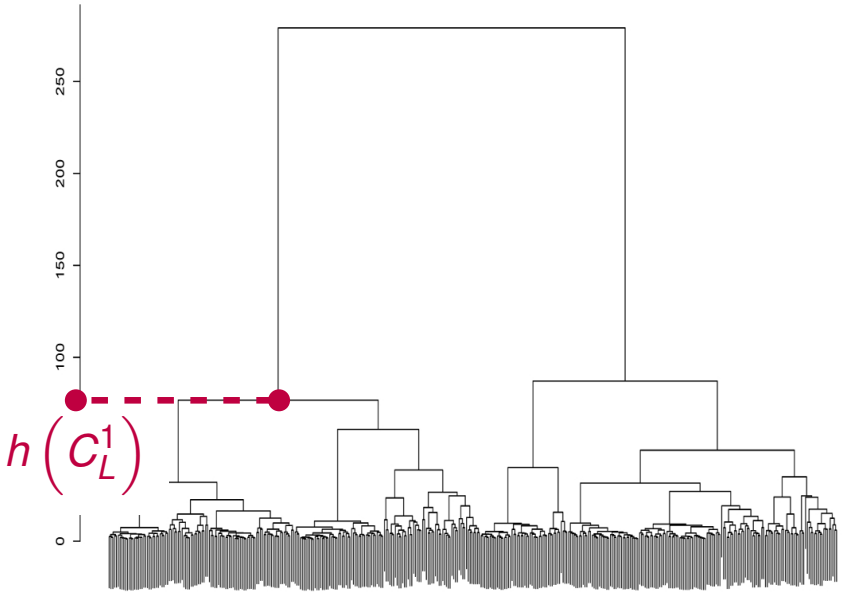
$i \leftarrow 2$

aggregationLevelsToVisit

$h(C_L^1), h(C_R^2), h(C_L^2)$

permClusters

The algorithm - The outline



Iteration

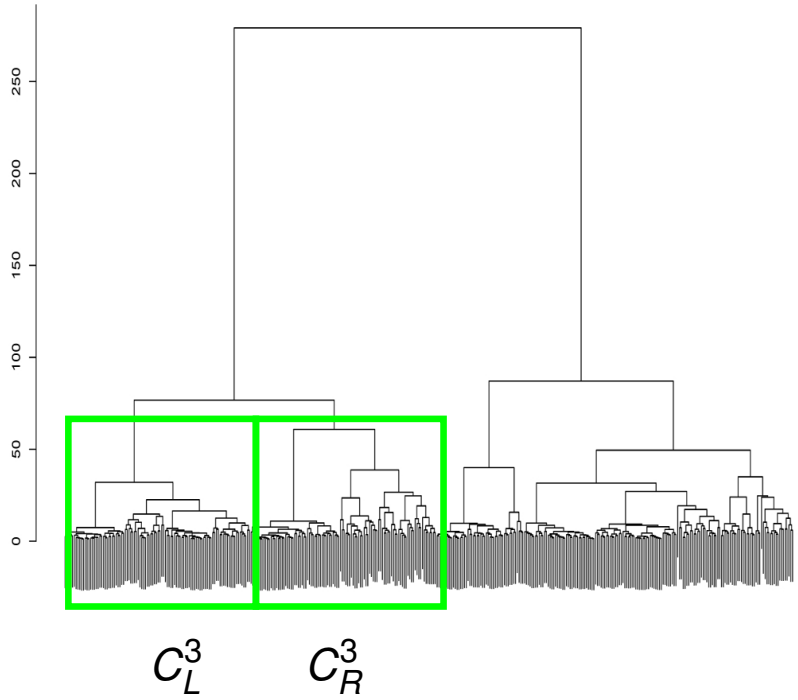
$i \leftarrow 3$

aggregationLevelsToVisit

$h(C_L^1), h(C_R^2), h(C_L^2)$

permClusters

The algorithm - The outline



Iteration

$i \leftarrow 3$

aggregationLevelsToVisit

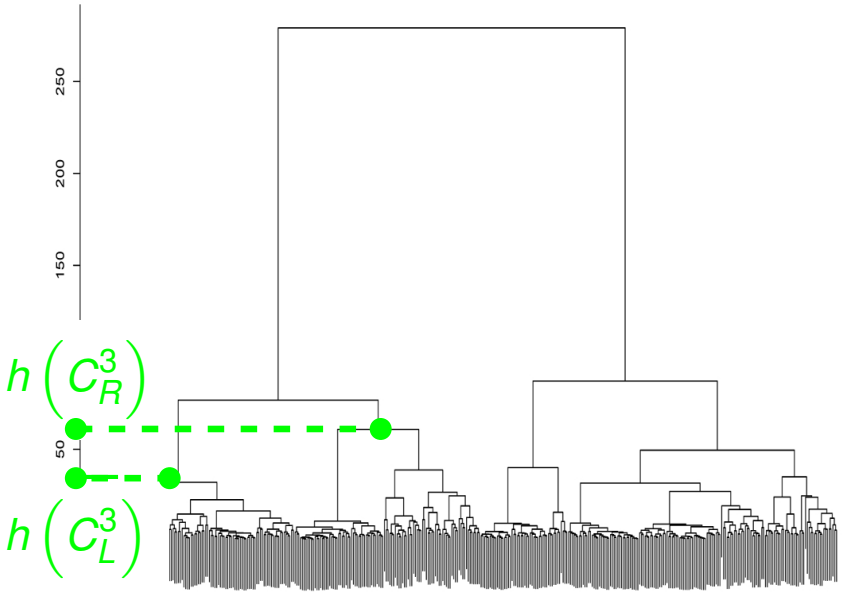
$h(C_L^1), h(C_R^2), h(C_L^2)$

permClusters

clusters to compare

$H_0 : C_L^3 \equiv C_R^3 \mapsto$ reject

The algorithm - The outline



Iteration

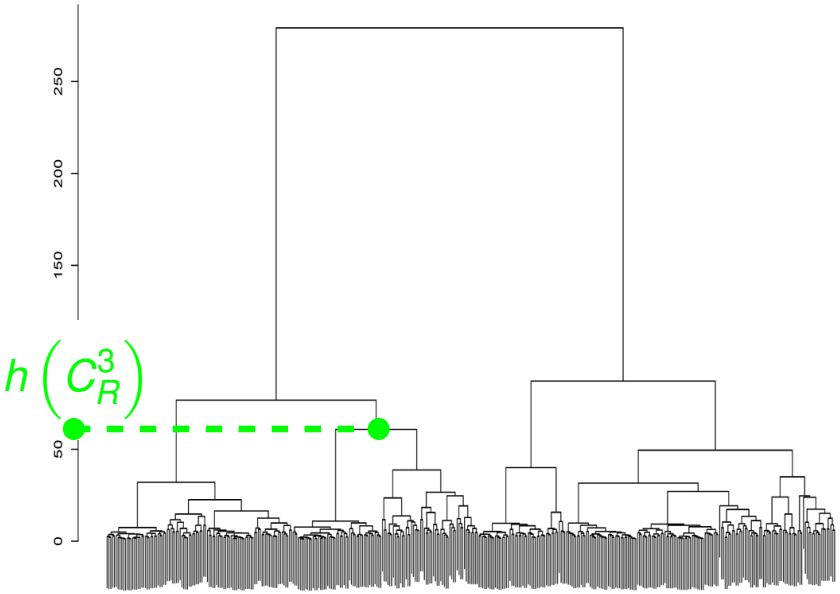
$i \leftarrow 3$

aggregationLevelsToVisit

$h(C_R^3), h(C_R^2), h(C_L^2), h(C_L^3)$

permClusters

The algorithm - The outline



Iteration

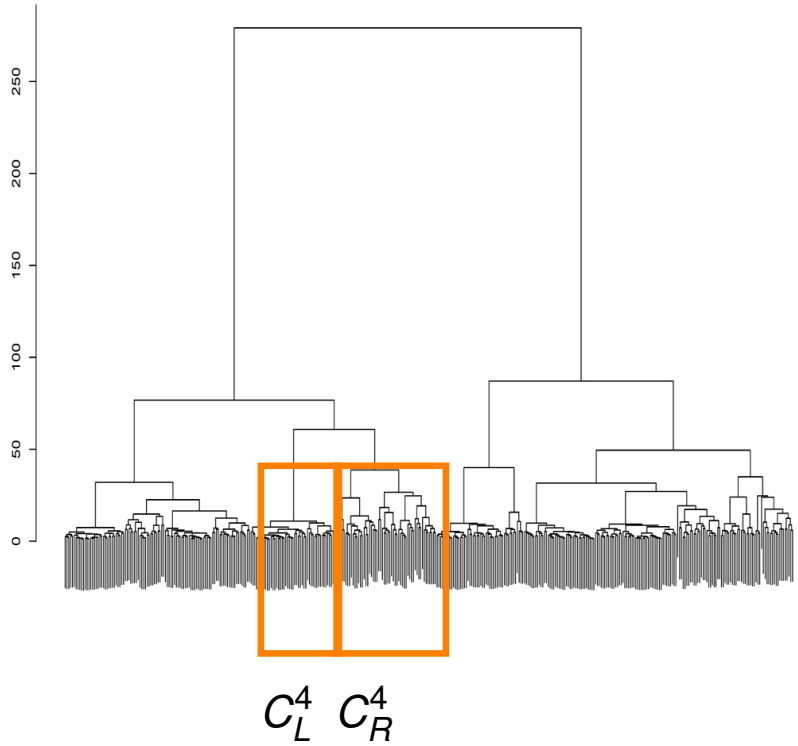
$i \leftarrow 4$

aggregationLevelsToVisit

$h(C_R^3), h(C_R^2), h(C_L^2), h(C_L^3)$

permClusters

The algorithm - The outline



Iteration

$i \leftarrow 4$

aggregationLevelsToVisit

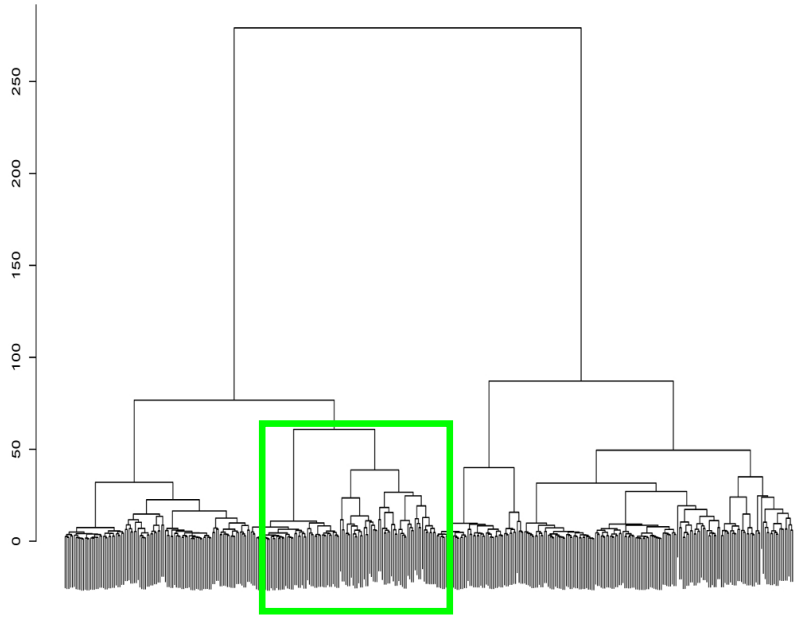
$h(C_R^3), h(C_R^2), h(C_L^2), h(C_L^3)$

permClusters

clusters to compare

$H_0 : C_L^4 \equiv C_R^4 \mapsto$ accept

The algorithm - The outline



C_R^3

Iteration

$i \leftarrow 4$

aggregationLevelsToVisit

$h(C_R^3), h(C_R^2), h(C_L^2), h(C_L^3)$

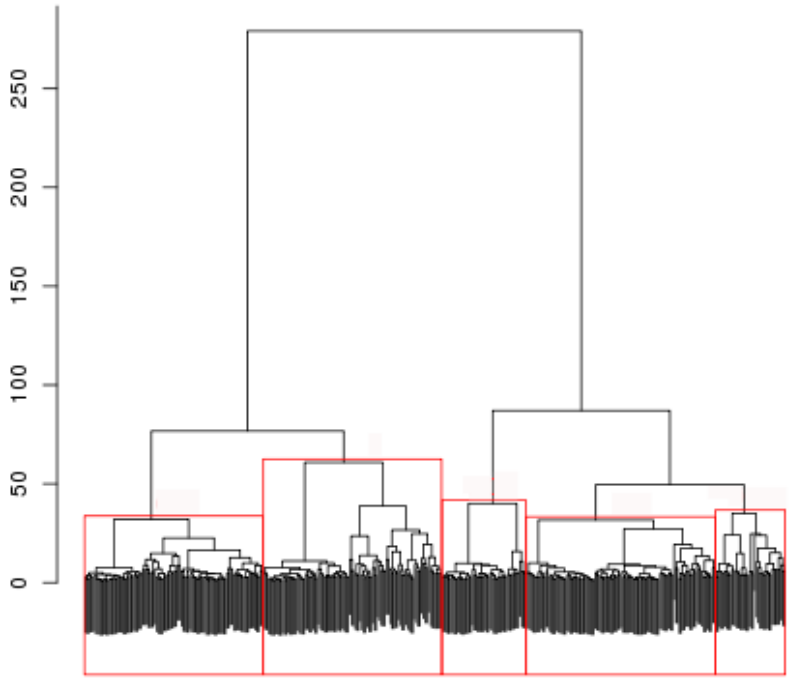
permClusters

$C_L^4 \cup C_R^4 \Leftrightarrow C_R^3$

clusters to compare

$H_0 : C_L^4 \equiv C_R^4 \mapsto$ accept

The algorithm - The outline



Iteration

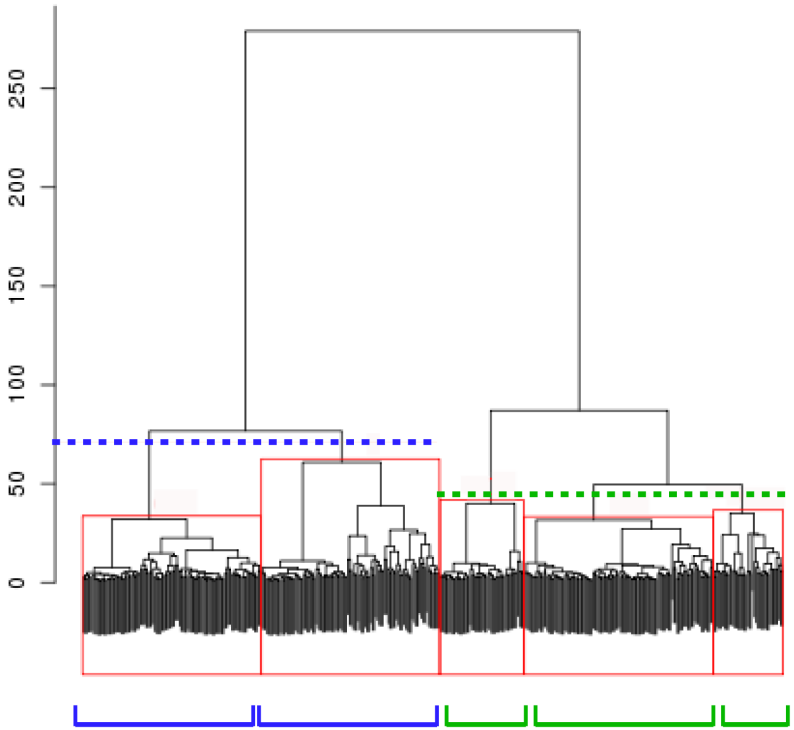
$i \leftarrow 9$

aggregationLevelsToVisit

permClusters

$C_L^3, C_R^3, C_L^2, C_L^4, C_R^4$

The algorithm - The outline



Iteration

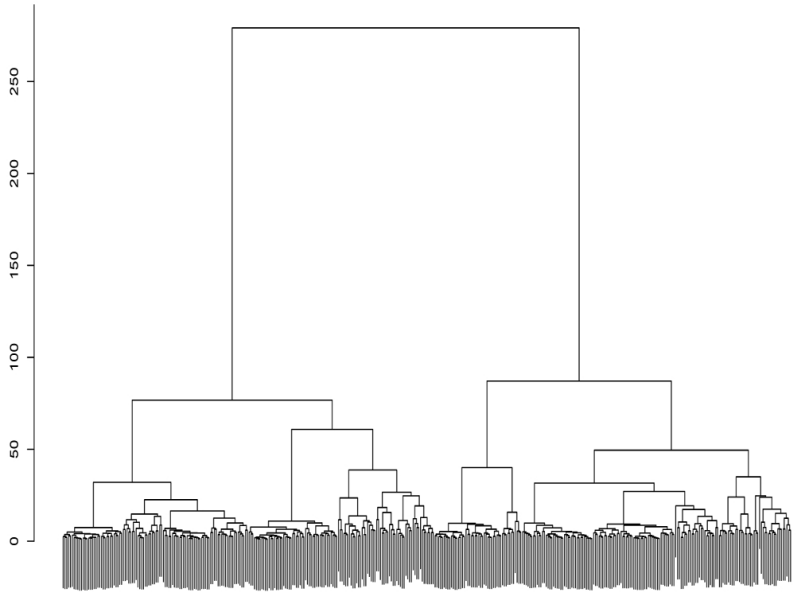
$i \leftarrow 9$

aggregationLevelsToVisit

permClusters

$C_L^3, C_R^3, C_L^2, C_L^4, C_R^4$

The algorithm - The permutation Test

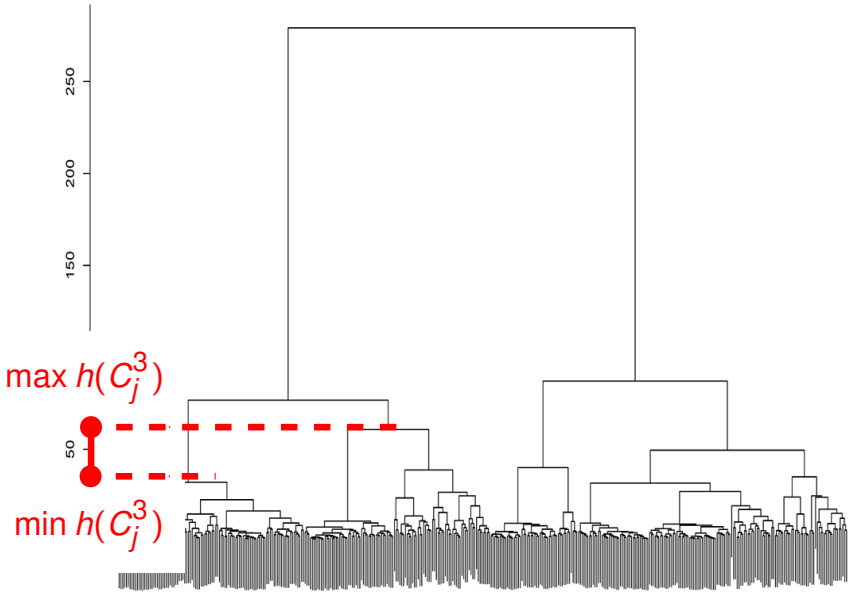


For each aggregation level k a *permutation test* is designed to test the *Null Hypothesis* that the two groups C_L^k and C_R^k really belong to the same cluster, i.e. :

$$H_0 : \quad C_L^k \equiv C_R^k$$

Under this *null*, mixing up (*permuting*) the statistical units of C_L^k and C_R^k should not alter the aggregation process resulting in their merging in.

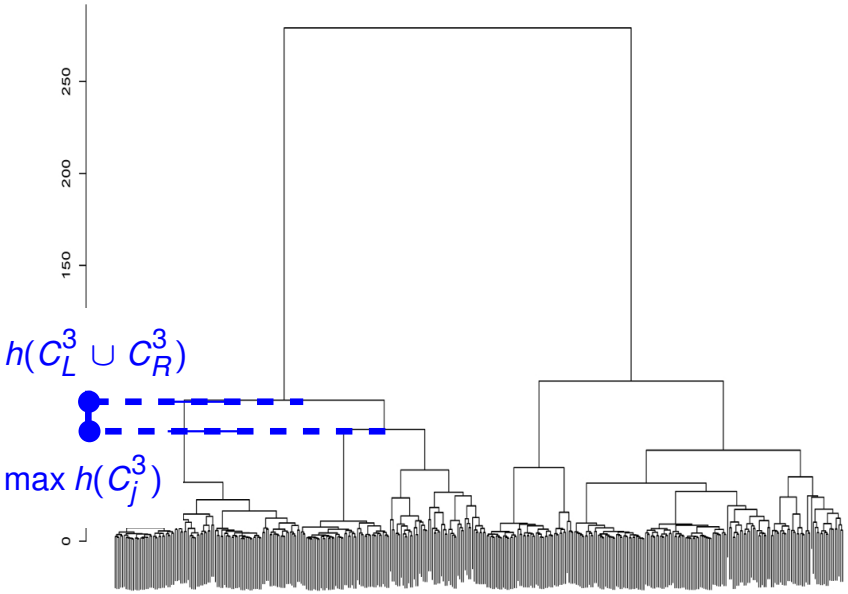
The algorithm - The permutation Test



For each k, the difference between $\max_{j \in \{L,R\}} h\left(C_j^k\right)$ and $\min_{j \in \{L,R\}} h\left(C_j^k\right)$ can be considered as the *minimum cost* necessary to merge the two classes.

.

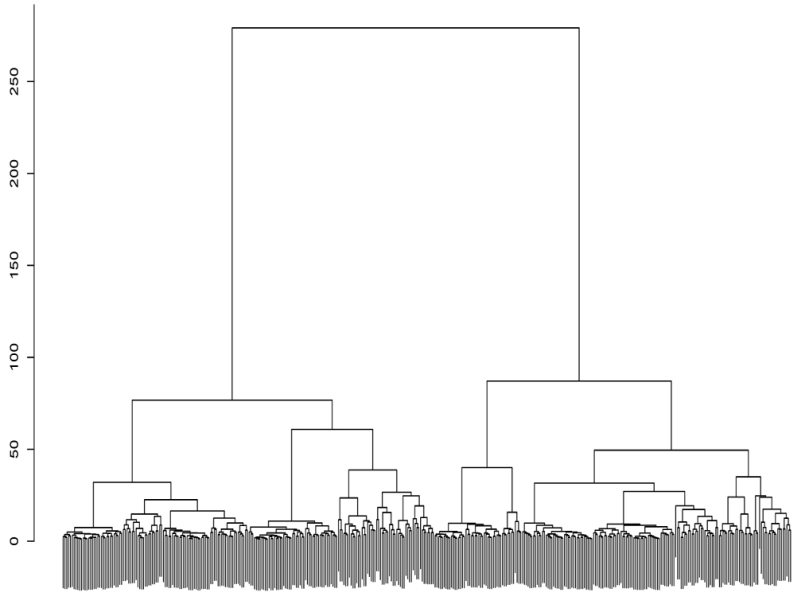
The algorithm - The permutation Test



For each k, the difference between $\max_{j \in \{L,R\}} h\left(C_j^k\right)$ and $\min_{j \in \{L,R\}} h\left(C_j^k\right)$ can be considered as the *minimum cost* necessary to merge the two classes.

The difference between $h\left(C_L^k \cup C_R^k\right)$ and $\max_{j \in \{L,R\}} h\left(C_j^k\right)$ can be, instead, considered as the *cost* actually incurred for merging C_L^k and C_R^k.

The algorithm - The permutation Test



For each k, the difference between $\max_{j \in \{L,R\}} h\left(C_j^k\right)$ and $\min_{j \in \{L,R\}} h\left(C_j^k\right)$ can be considered as the *minimum cost* necessary to merge the two classes.

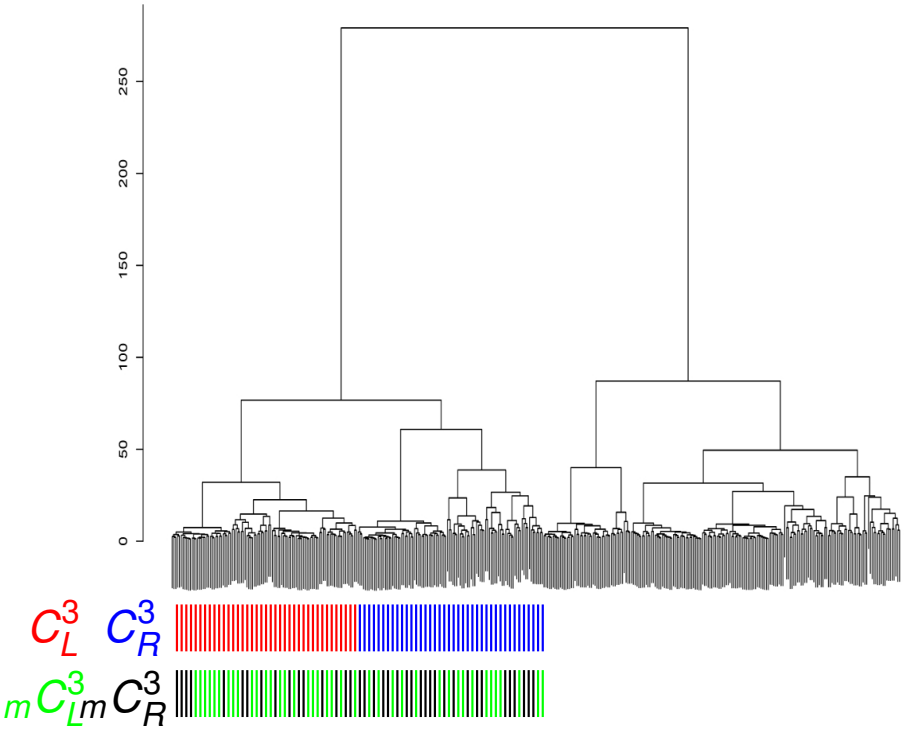
The difference between $h\left(C_L^k \cup C_R^k\right)$ and $\max_{j \in \{L,R\}} h\left(C_j^k\right)$ can be, instead, considered as the *cost* actually incurred for merging C_L^k and C_R^k.

The ratio between these two differences:

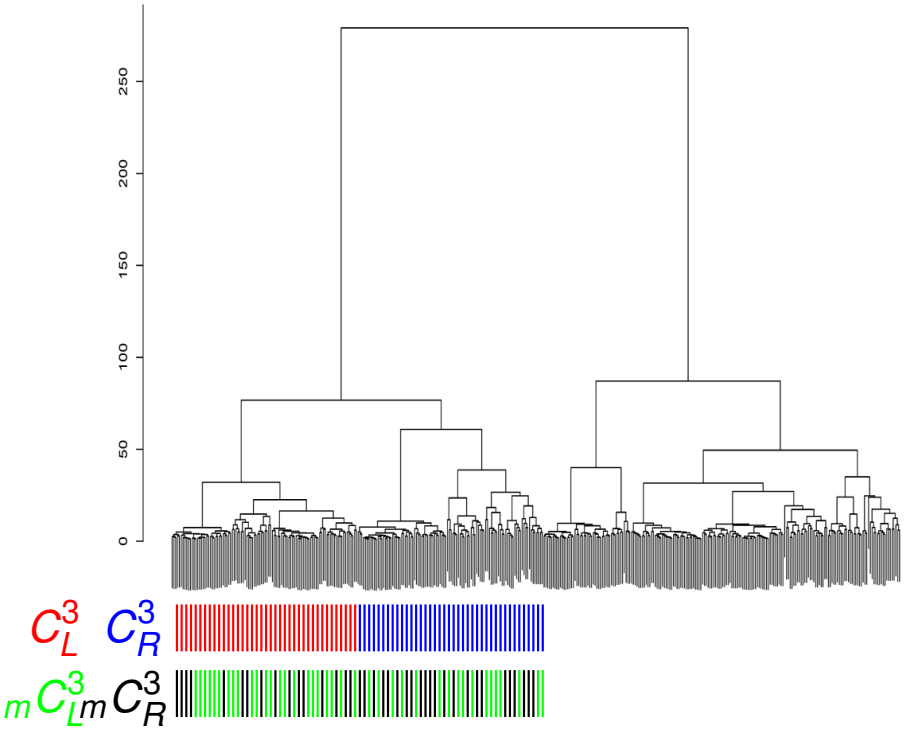
$$cost\left(C_L^k \cup C_R^k\right) = \frac{\max_{j \in \{L,R\}} h\left(C_j^k\right) - \min_{j \in \{L,R\}} h\left(C_j^k\right)}{h\left(C_L^k \cup C_R^k\right) - \max_{j \in \{L,R\}} h\left(C_j^k\right)}$$

is thus a measure that characterizes the aggregation process resulting in the new class $C_L^k \cup C_R^k$

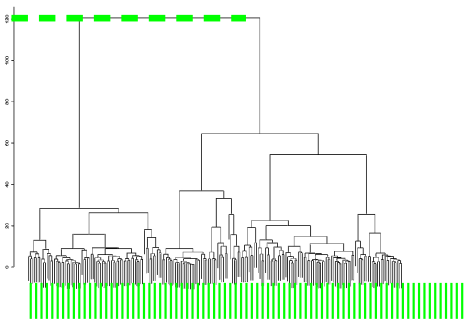
The algorithm - The permutation Test



The algorithm - The permutation Test

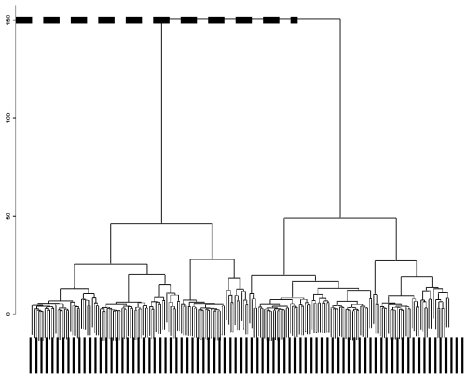


$h(_mC_L^3)$



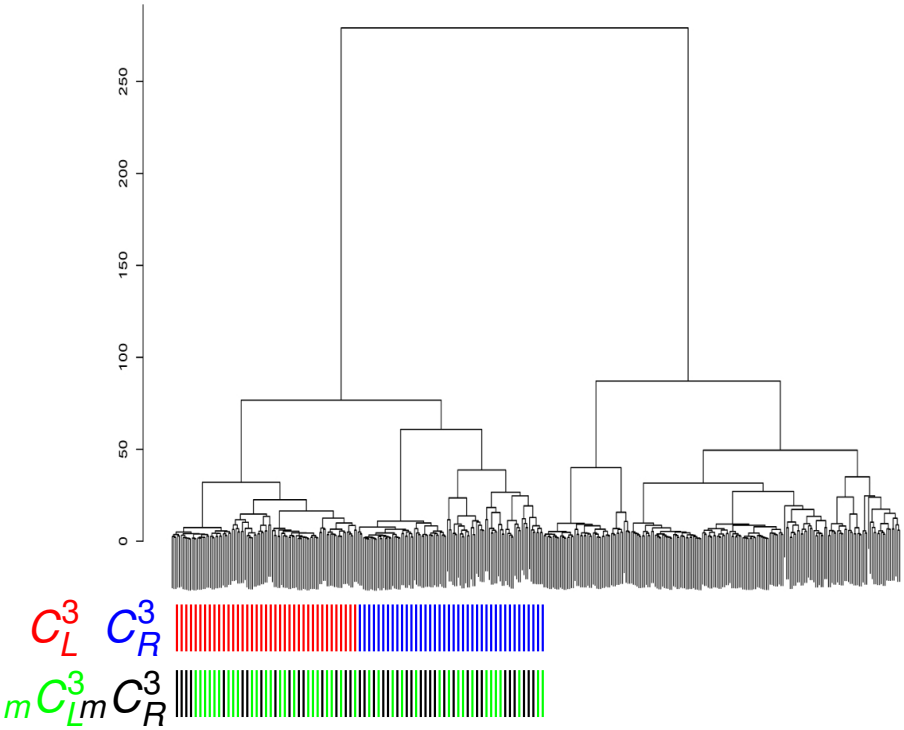
$_mC_L^3$

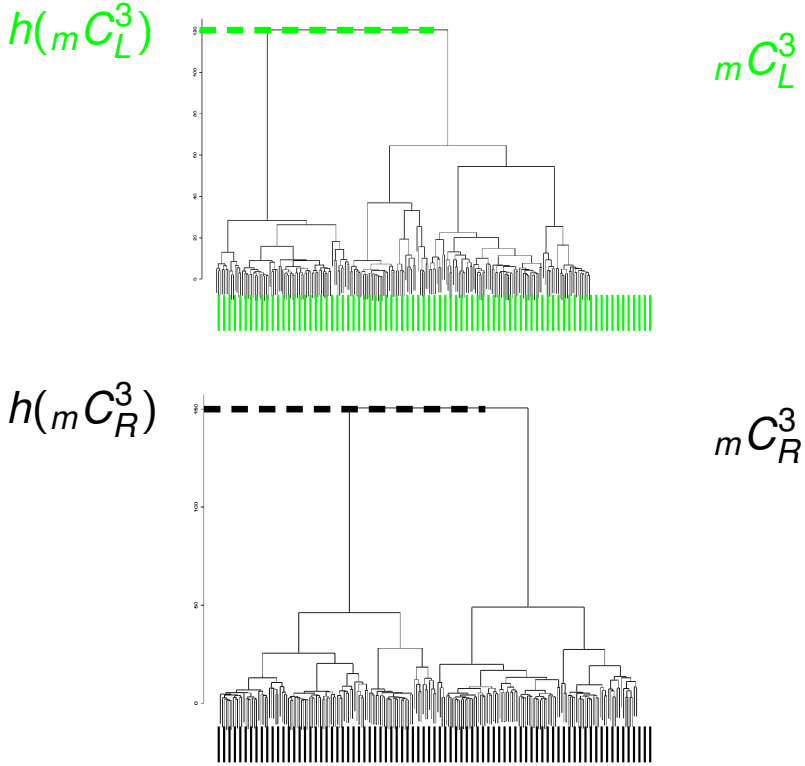
$h(_mC_R^3)$



$_mC_R^3$

The algorithm - The permutation Test



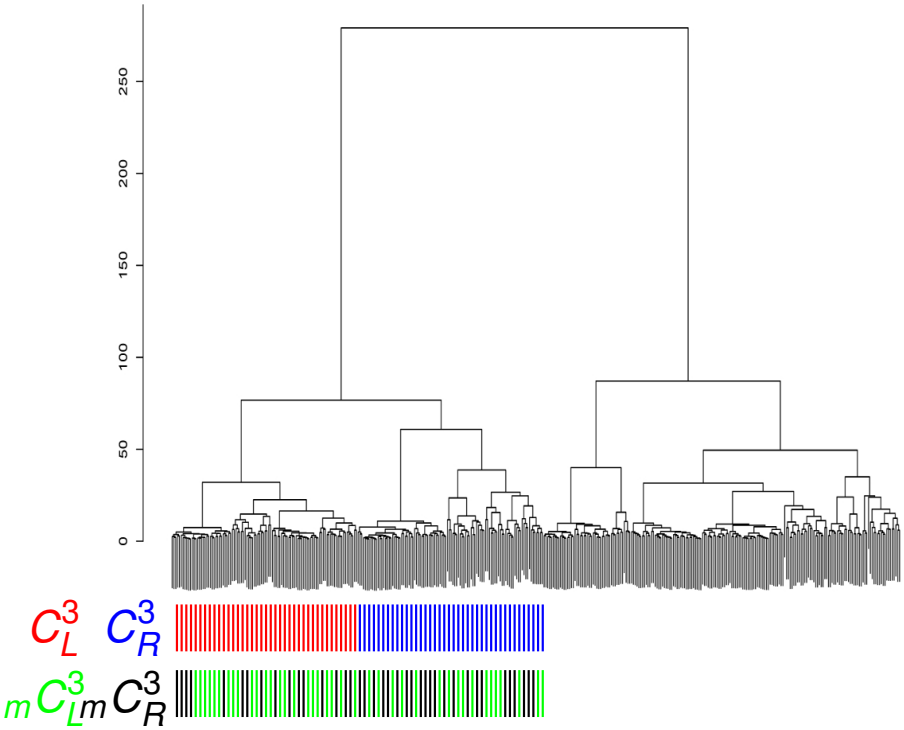


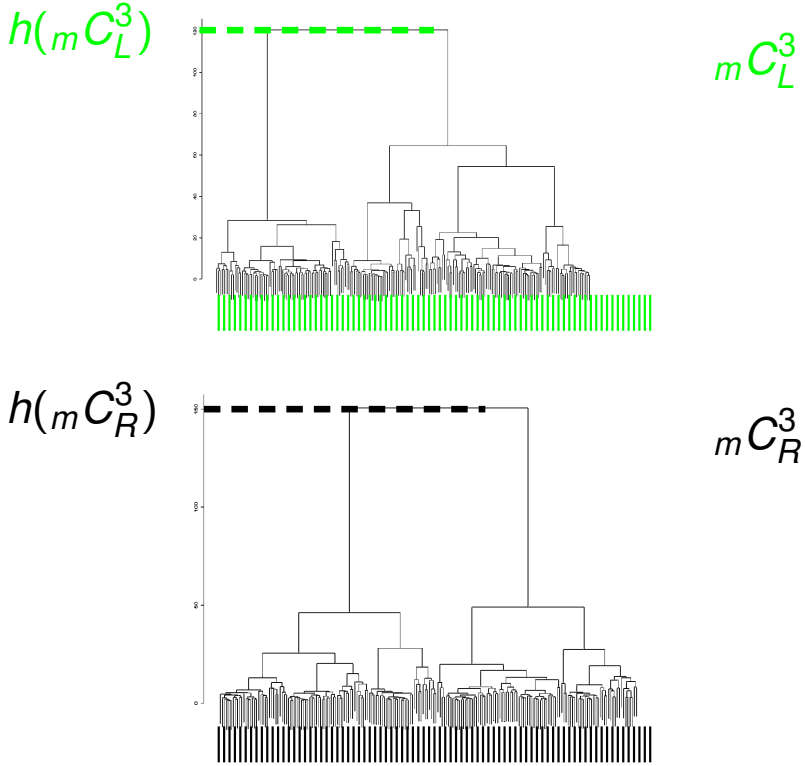
The ratio:

$$cost\left({}_mC_L^k \cup {}_mC_R^k\right) = \frac{\max\limits_{j\in\{L,R\}} h\left({}_mC_j^k\right) - \min\limits_{j\in\{L,R\}} h\left({}_mC_j^k\right)}{h\left(C_L^k \cup C_R^k\right) - \max\limits_{j\in\{L,R\}} h\left({}_mC_j^k\right)}$$

is thus a measure that characterizes the aggregation process resulting in the new (*potential*) class ${}_mC_L^k \cup {}_mC_R^k$

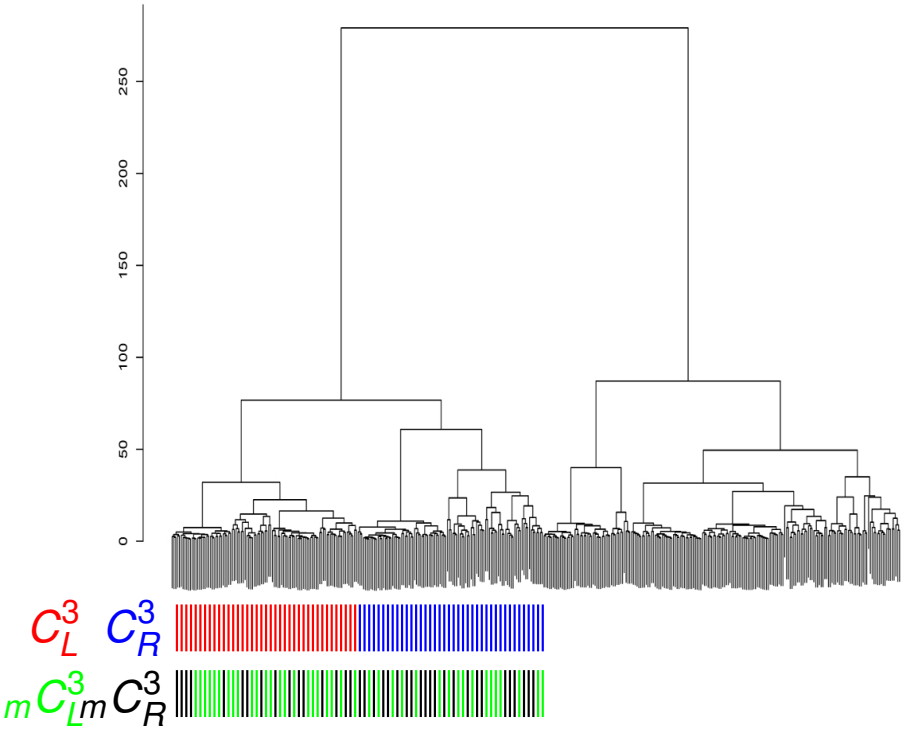
The algorithm - The permutation Test

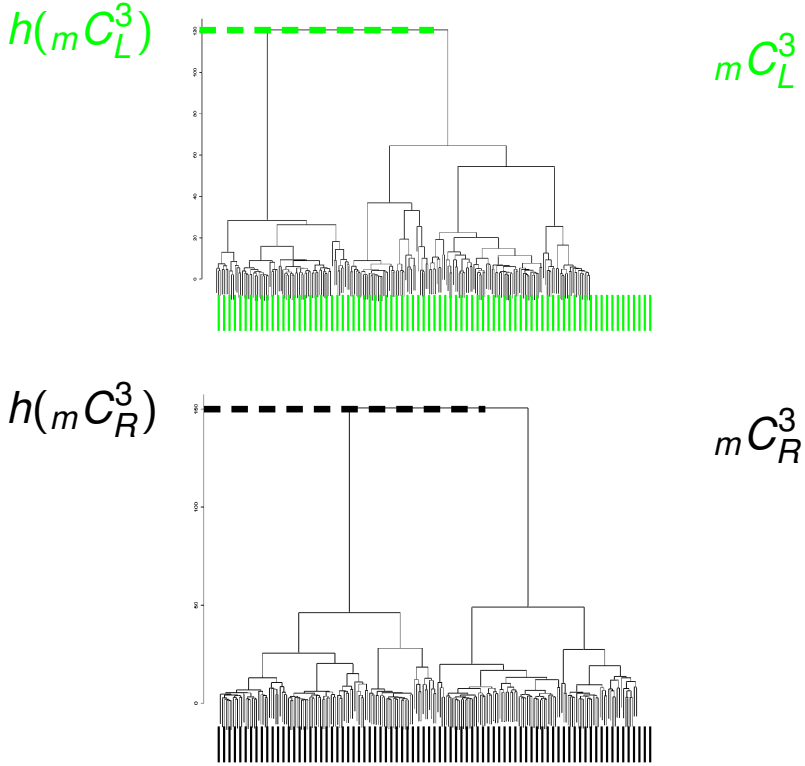




Under H_0 the aggregation process resulting in the new cluster $C_L^k \cup C_R^k$ should be very similar to the one that *potentially* produces ${}_mC_L^k \cup {}_mC_R^k$; thus the two values $cost\left({}_mC_L^k \cup {}_mC_R^k\right)$ and $cost\left(C_L^k \cup C_R^k\right)$ should be close enough.

The algorithm - The permutation Test



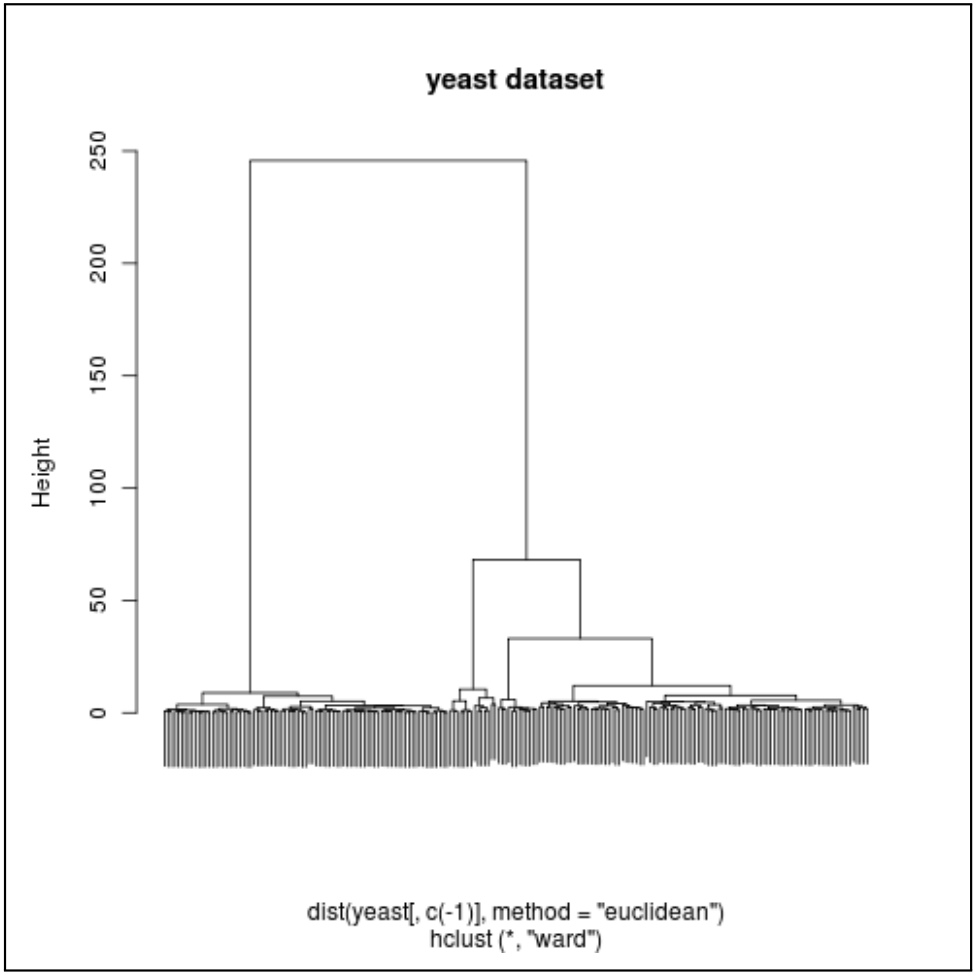


The permutation procedure is repeated M times and each time a new couple ${}_mC_L^k$, ${}_mC_R^k$ is obtained. The pvalue Montecarlo is thus computed as:

$$p = \frac{\#\left\{cost\left({}_mC_L^k \cup {}_mC_R^k\right) \leq cost\left(C_L^k \cup C_R^k\right)\right\} + 1}{M + 1}$$

La Carte

Some results - Real datasets



The yeast galactose dataset

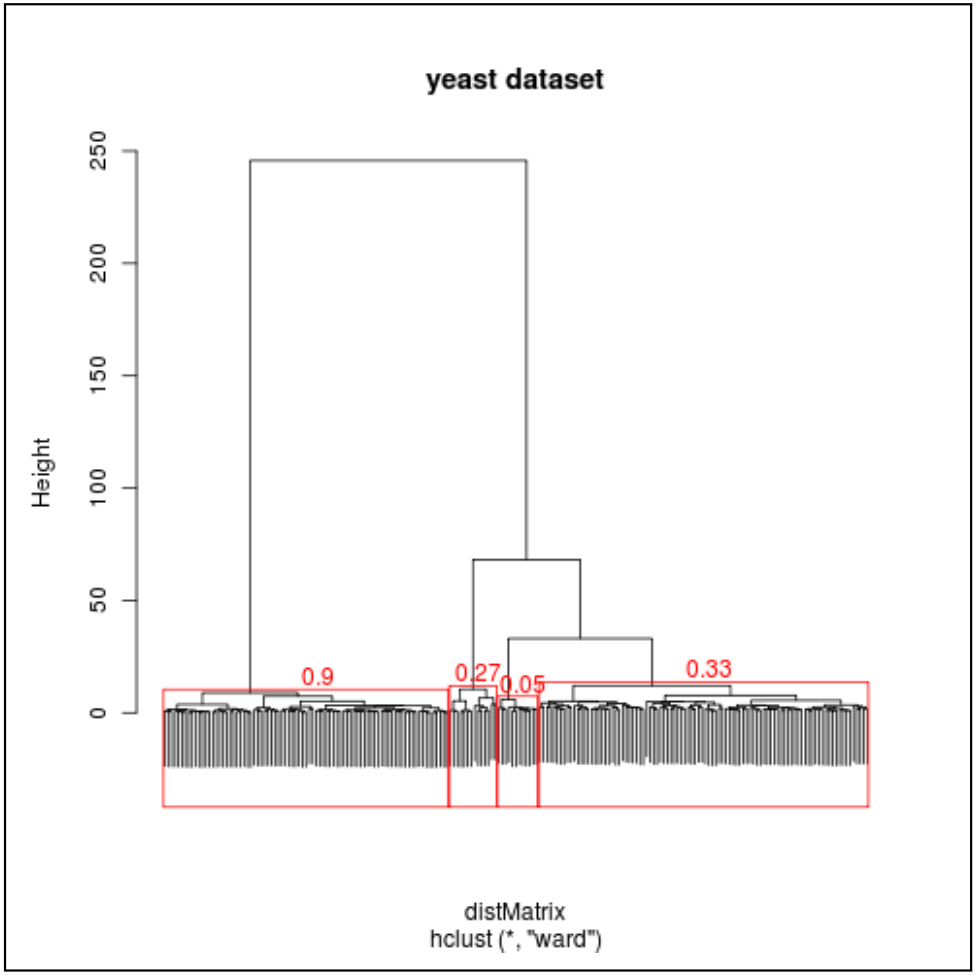
Ideker T, Thorsson V, Ranish JA, Christmas R, Buhler J, Eng JK, Bumgarner RE, Goodlett DR, Aebersold R, Hood L
Integrated genomic and proteomic analyses of a systemically perturbed metabolic network.

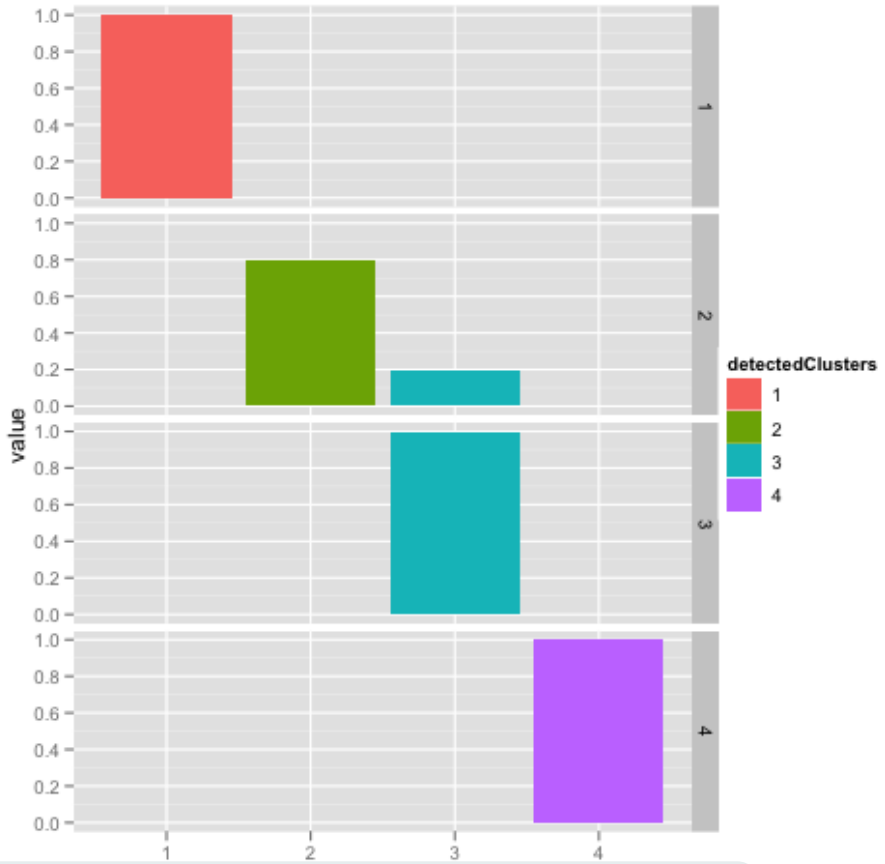
Science 2001, 292:929-934.

$n = 205$

$p = 80$

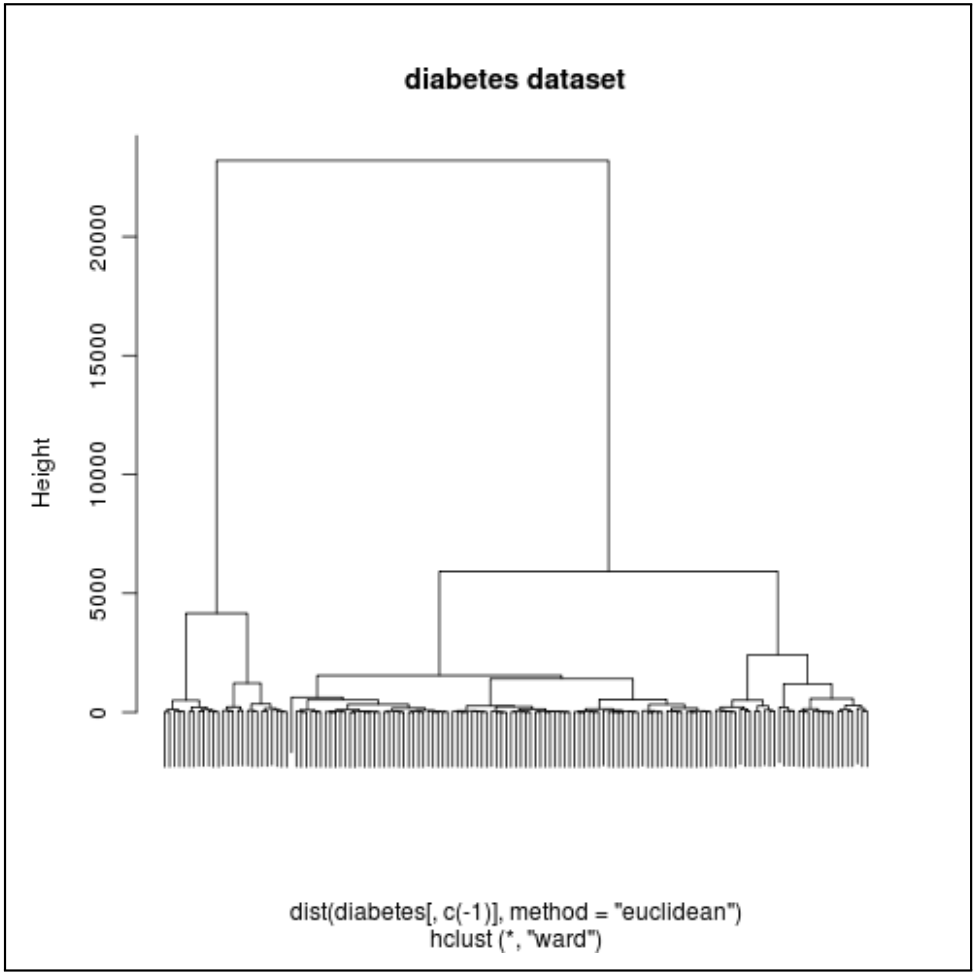
Some results - Real datasets





% of misclassification = 1.5

Some results - Real datasets



The diabetes dataset

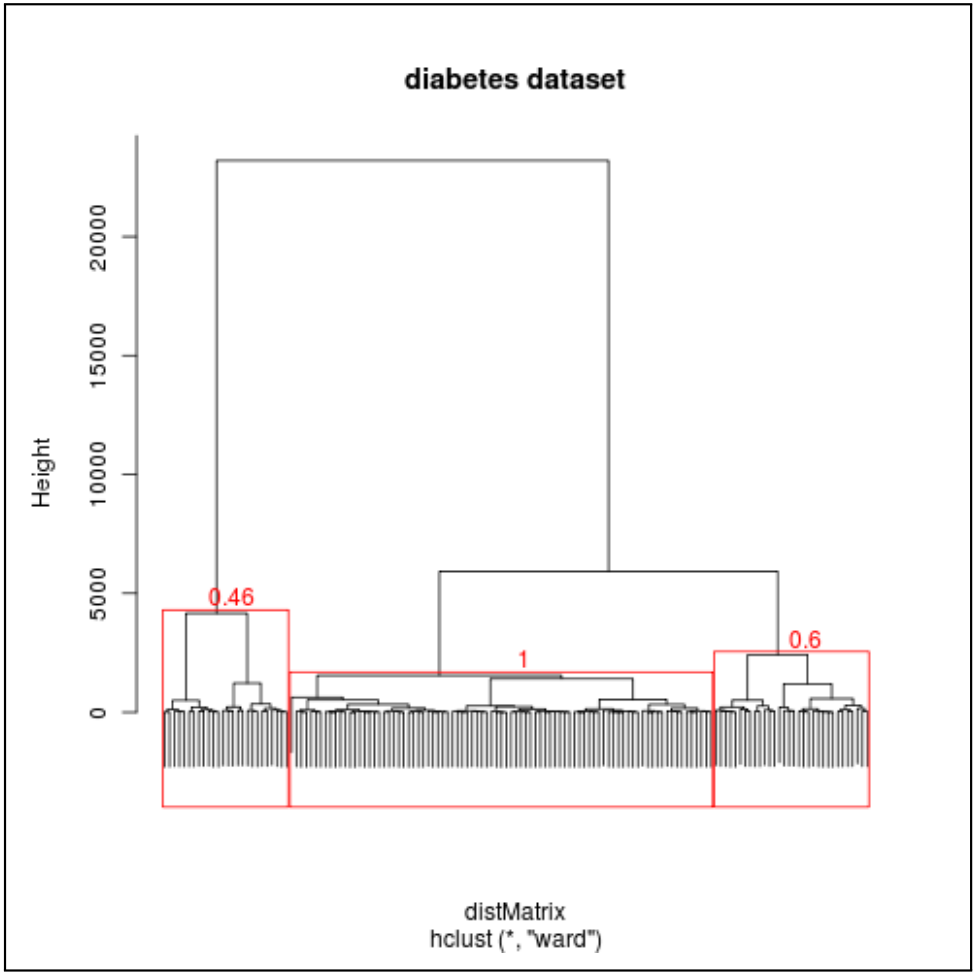
Banfield JD, Raftery AE
Model–based Gaussian and Non–Gaussian Clustering.

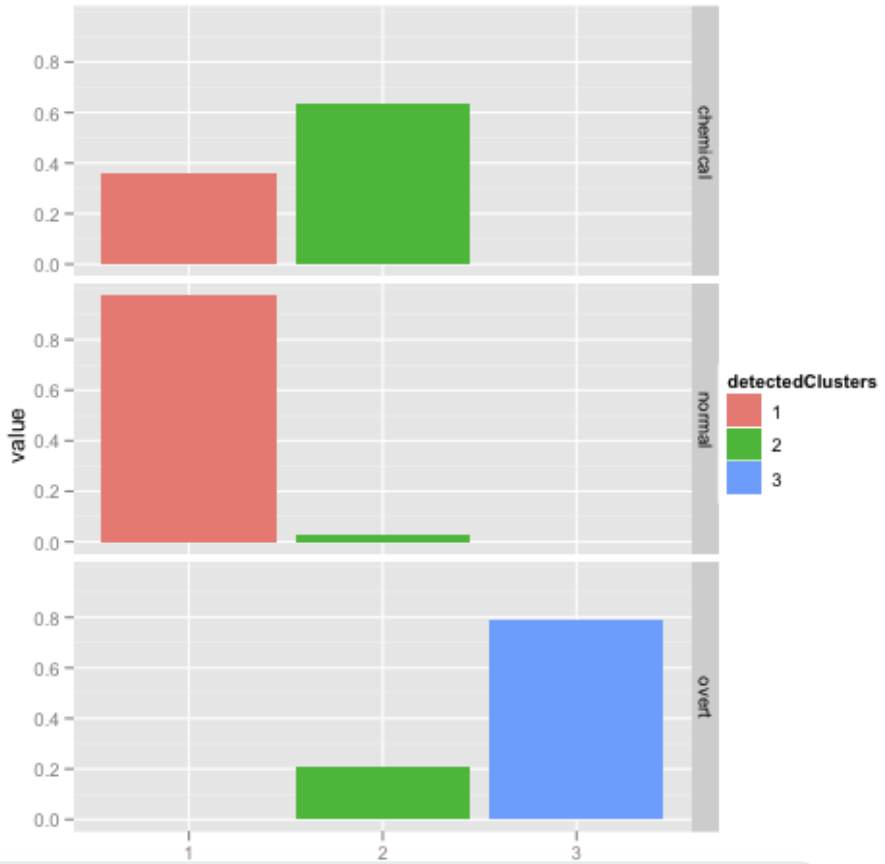
Biometrics, 1993, 49, 803-821.

$n = 145$

$p = 3$

Some results - Real datasets





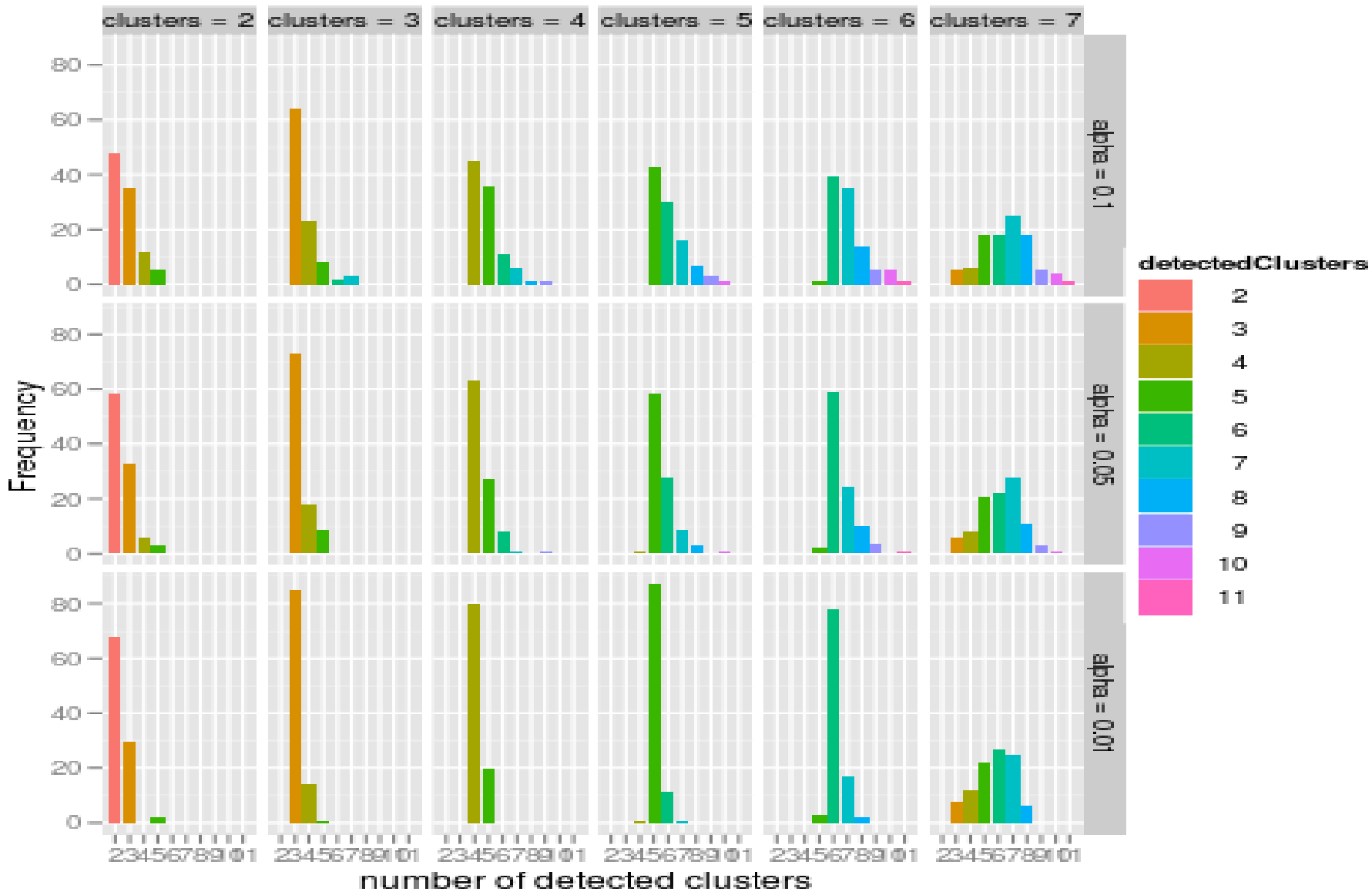
% of misclassification = 15.2

Some results - Synthetic dataset

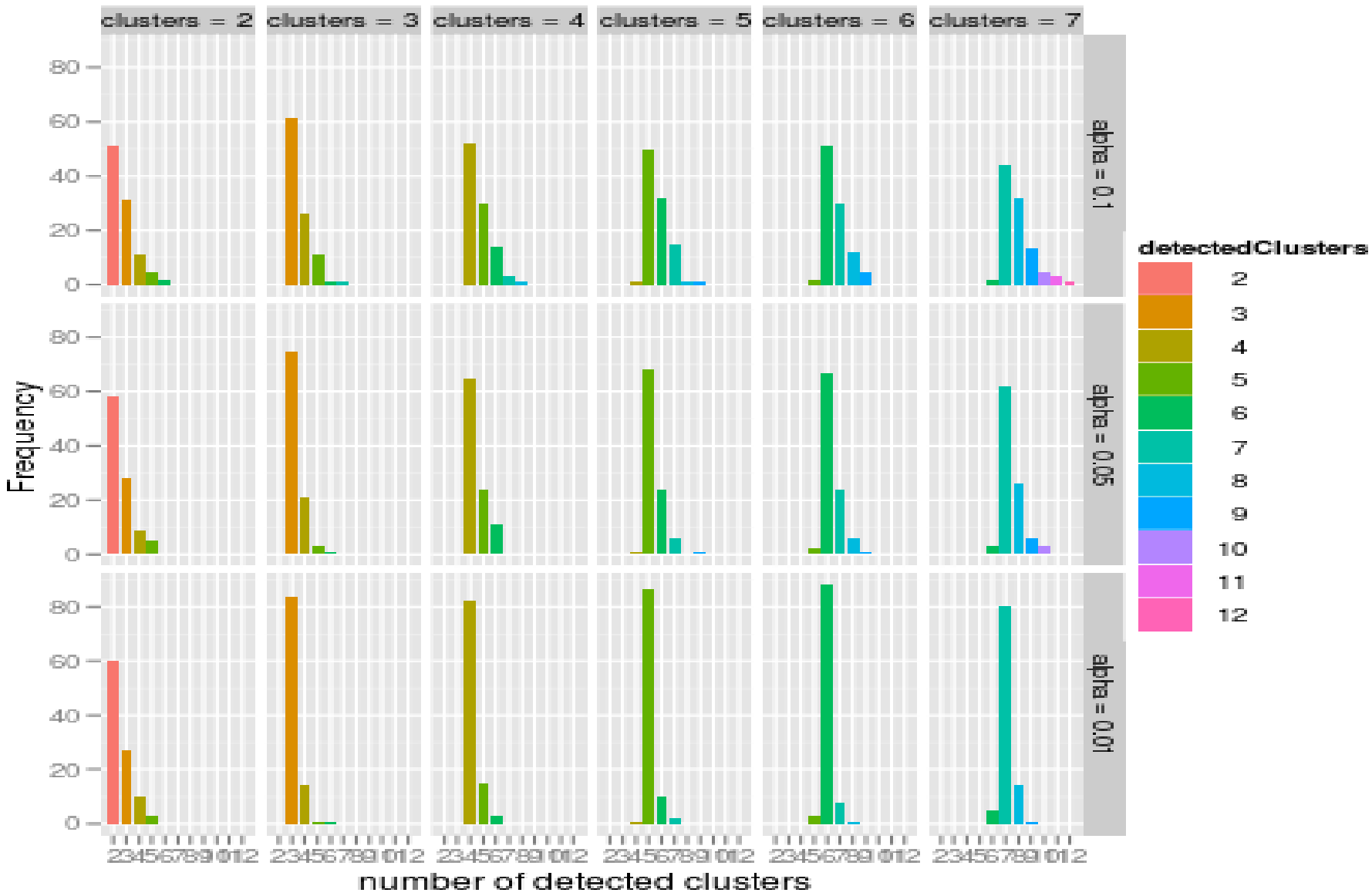
QIU W.-L, JOE H. (2009). clusterGeneration: random cluster generation (with specified degree of separation). R package version 1.2.7.

- different number of clusters (k = 2; 3; 4; 5; 6; 7)
- separation index = 0.01
- different number of variables (p = 5; 10; 15)
- 100 replications for each combination of k and p

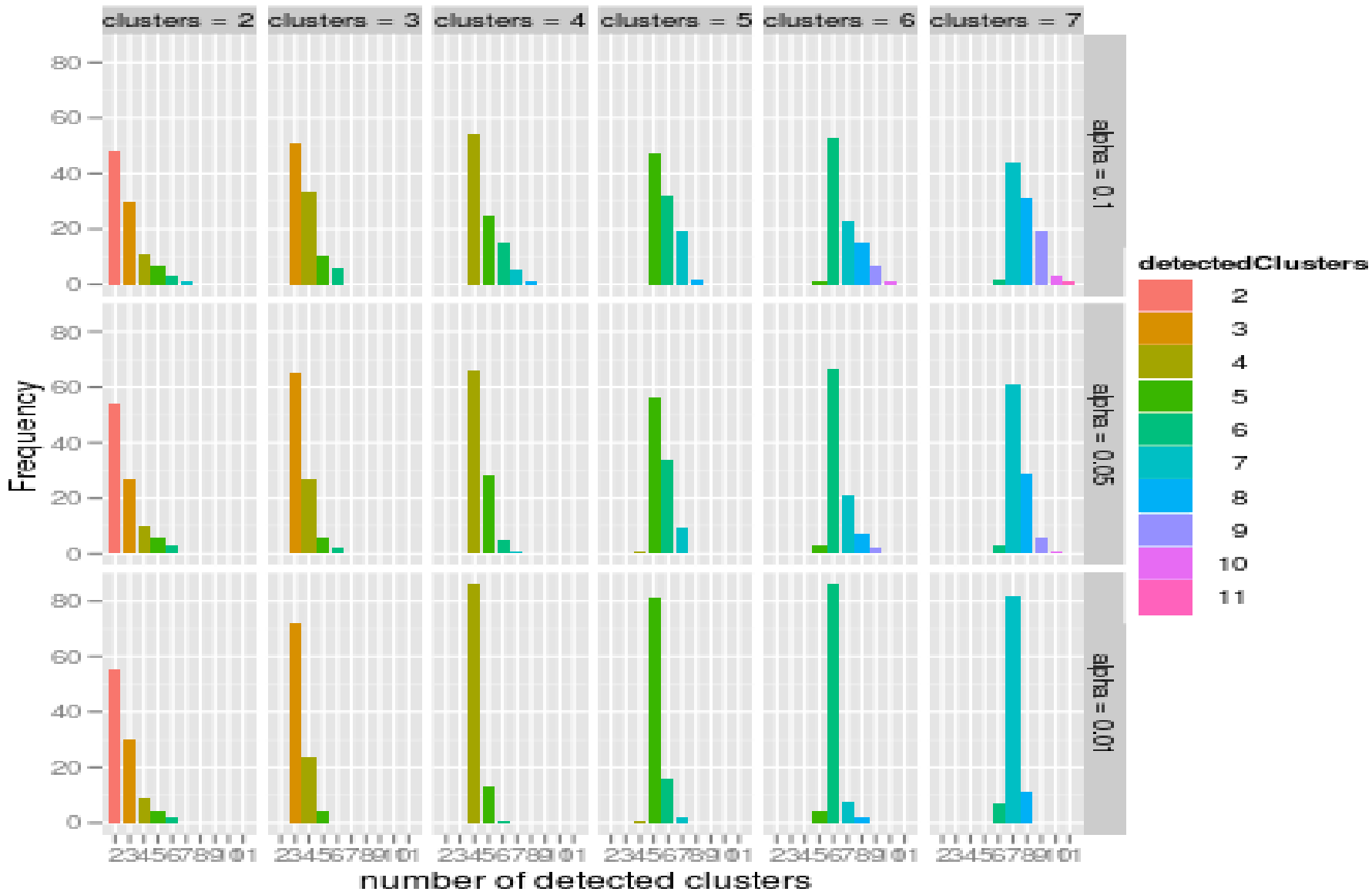
Some results - Synthetic dataset (p=5)



Some results - Synthetic dataset (p=10)



Some results - Synthetic dataset (p=15)



La Carte

ToDo List

Statistical issues

- Introducing a *penalty* term in the permutation test step
- Quality measures of the obtained partition
- Multiple Testing Problem (???)

Computational issues

- profiling and optimizing the R code
  - use of compiled code
  - deploying a package