Stairstep-like dendrogram cut: a permutation test approach

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

All computations and graphics were done using the R system (packages: cluster, clusterGeneration, ggplot2)

Slides has been composed using LaTeX (*beamer* class) and the Sweave tool

Stairstep-like dendrogram cut: a permutation test approach

(a not necessarily regular cut for a dendrogram)

Dario Bruzzese
dbruzzes@unina.it

Domenico Vistocco
vistocco@unicas.it

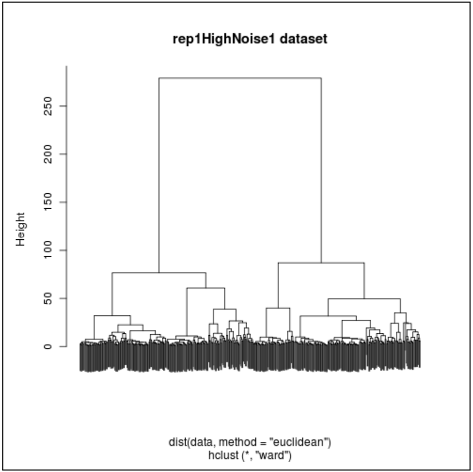
Department of
Preventive Medical Sciences
UNIVERSITY OF NAPLES
ITALY

Department of
Economics
UNIVERSITY OF CASSINO
ITALY

All computations and graphics were done using the R system (packages: cluster, clusterGeneration, ggplot2)

Slides has been composed using LaTeX (*beamer* class) and the Sweave tool

Motivation



The rep1HighNoise dataset

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

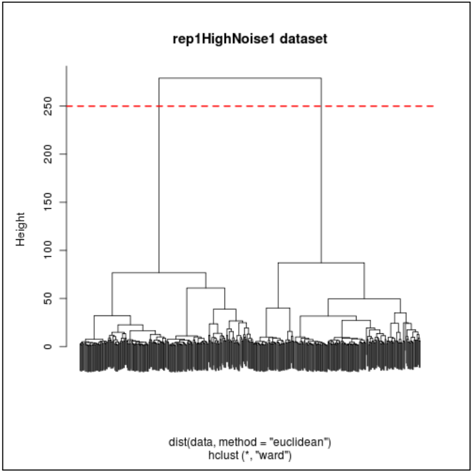
Genome Biology, 2003, 4:R34

$n = 200$

$p = 20$

It is a synthetic data set with error distributions derived from real array data.

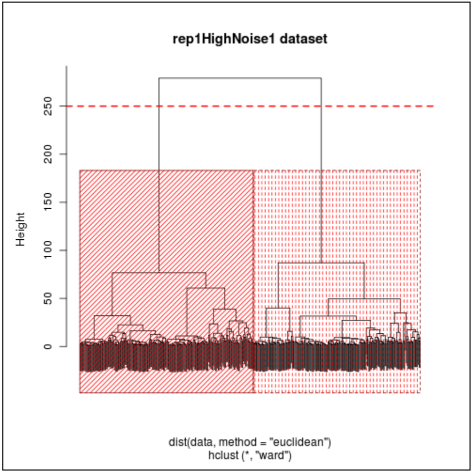
Motivation



Horizontal cut

$k = 2$

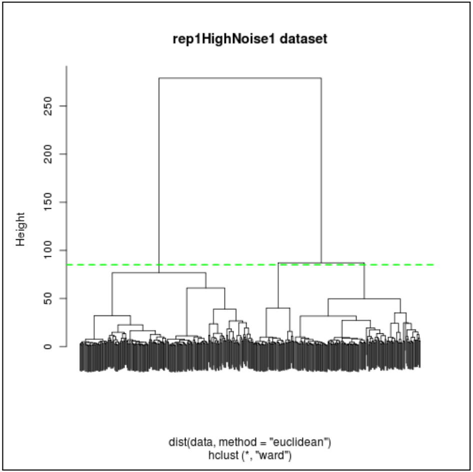
Motivation



Horizontal cut

$k = 2$ (red clusters)

Motivation

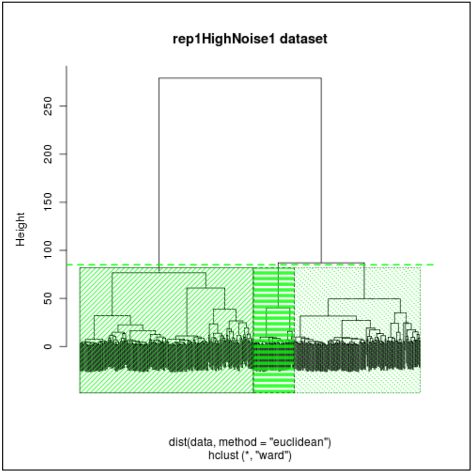


Horizontal cut

$k = 2$ (red clusters)

$k = 3$

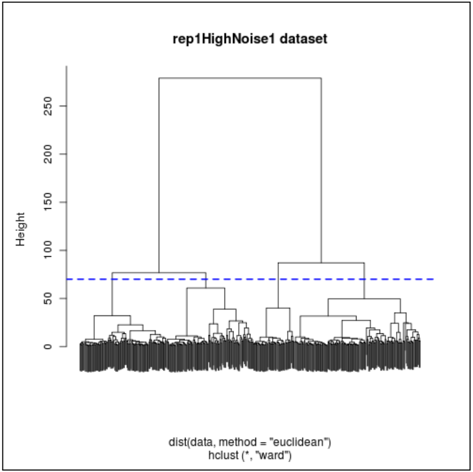
Motivation



Horizontal cut

$k = 2$ (red clusters)
$k = 3$ (green clusters)

Motivation



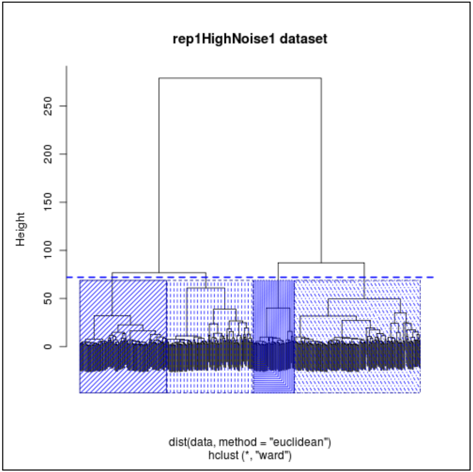
Horizontal cut

$k = 2$ (red clusters)

$k = 3$ (green clusters)

$k = 4$

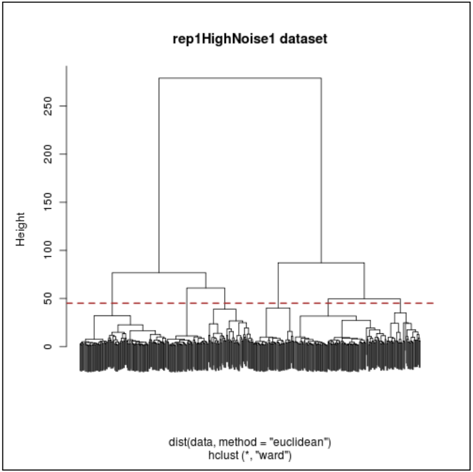
Motivation



Horizontal cut

$k = 2$ (red clusters)
$k = 3$ (green clusters)
$k = 4$ (blue clusters)

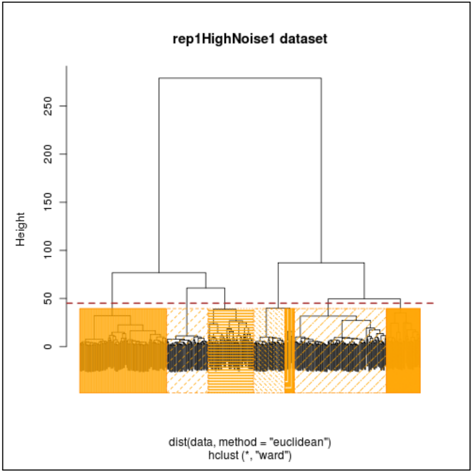
Motivation



Horizontal cut

$k = 2$ (red clusters)
$k = 3$ (green clusters)
$k = 4$ (blue clusters)
$\ldots$
$k = 7$

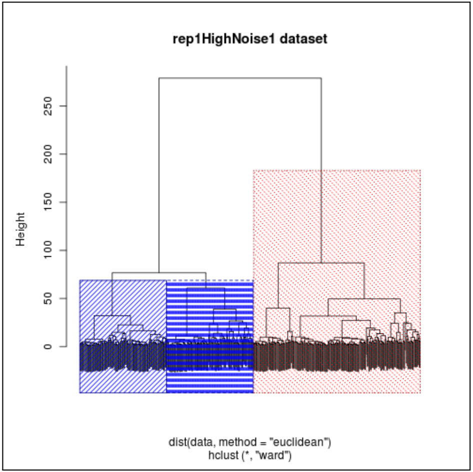
Motivation



Horizontal cut

$k = 2$ (red clusters)
$k = 3$ (green clusters)
$k = 4$ (blue clusters)
$\dots$
$k = 7$ (brown clusters)

Motivation



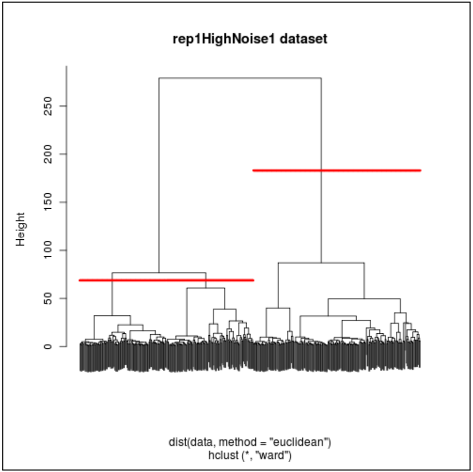
Horizontal cut

$k = 2$ (red clusters)
$k = 3$ (green clusters)
$k = 4$ (blue clusters)
$\dots$
$k = 7$ (brown clusters)

An alternative cut

$k = 3$ (rainbow clusters)

Motivation



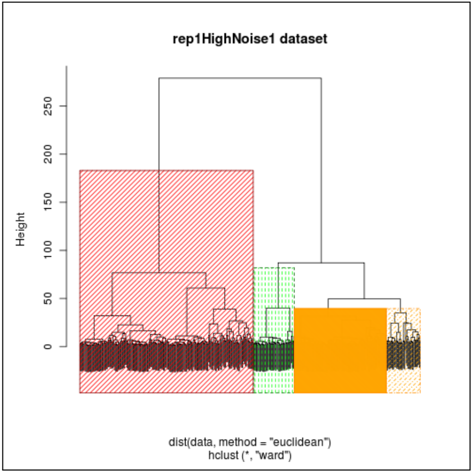
Horizontal cut

$k = 2$ (red clusters)
$k = 3$ (green clusters)
$k = 4$ (blue clusters)
$\ldots$
$k = 7$ (brown clusters)

An alternative cut

$k = 3$ (rainbow clusters)

Motivation



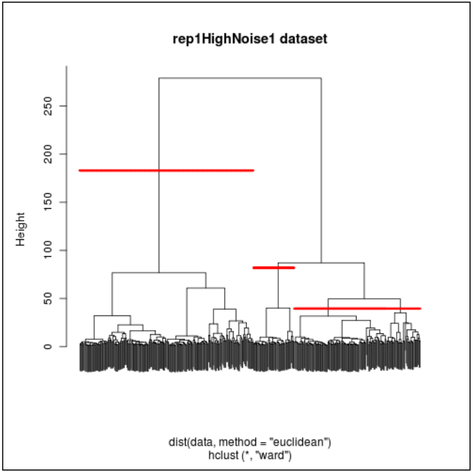
Horizontal cut

$k = 2$ (red clusters)
$k = 3$ (green clusters)
$k = 4$ (blue clusters)
$\dots$
$k = 7$ (brown clusters)

An alternative cut

$k = 4$ (rainbow clusters)

Motivation



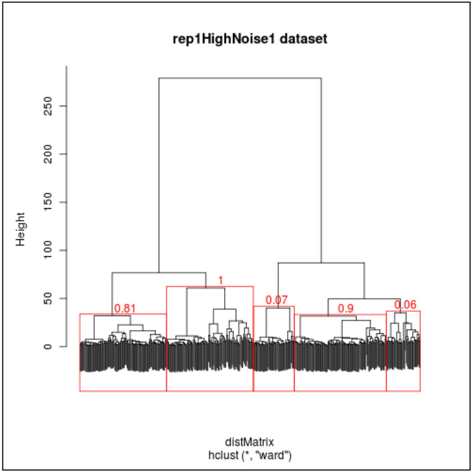
Horizontal cut

$k = 2$ (red clusters)
$k = 3$ (green clusters)
$k = 4$ (blue clusters)
$\ldots$
$k = 7$ (brown clusters)

An alternative cut

$k = 4$ (rainbow clusters)

Motivation



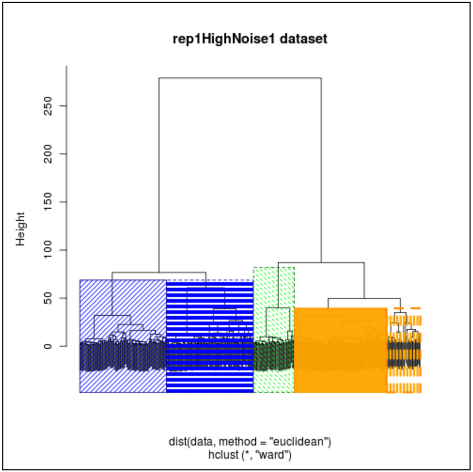
Horizontal cut

$k = 2$ (red clusters)
$k = 3$ (green clusters)
$k = 4$ (blue clusters)
$\dots$
$k = 7$ (brown clusters)

$\alpha = 0.01$

5 clusters

Motivation



Horizontal cut

$k = 2$ (red clusters)
$k = 3$ (green clusters)
$k = 4$ (blue clusters)
...
$k = 7$ (brown clusters)

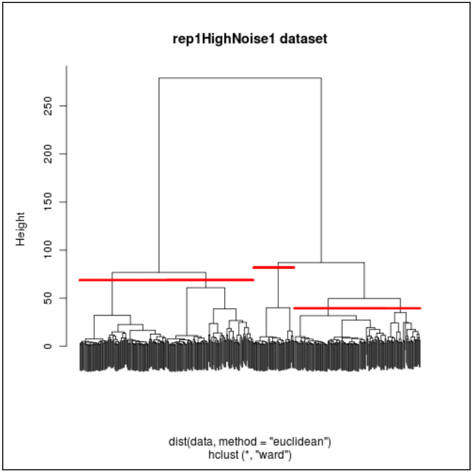
$\alpha = 0.01$

5 clusters

An alternative cut

$k = 5$ (rainbow clusters)

Motivation



Horizontal cut

$k = 2$ (red clusters)
$k = 3$ (green clusters)
$k = 4$ (blue clusters)
$\ldots$
$k = 7$ (brown clusters)

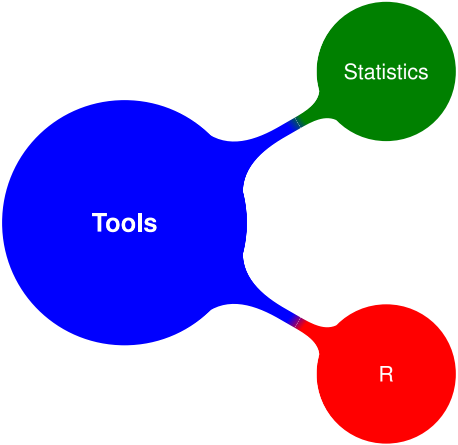
$\alpha = 0.01$

5 clusters

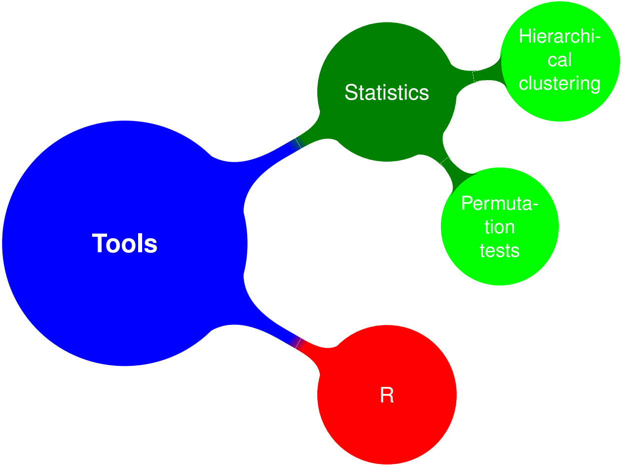
An alternative cut

$k = 5$ (rainbow clusters)

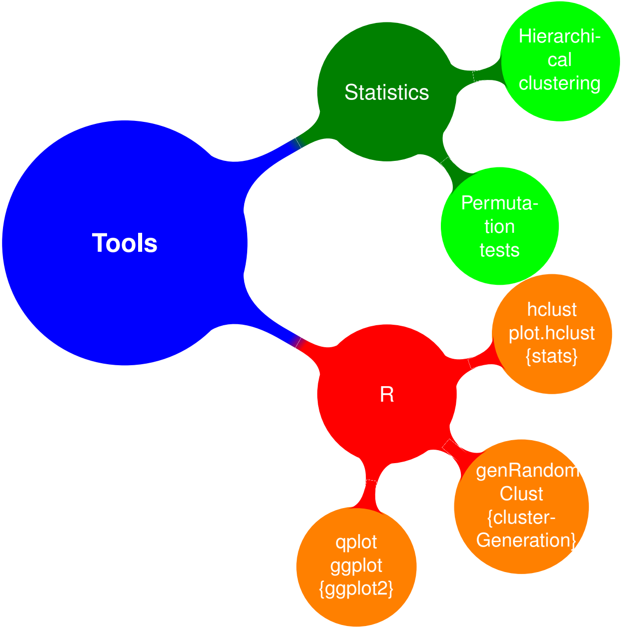
The reference framework



The reference framework



The reference framework



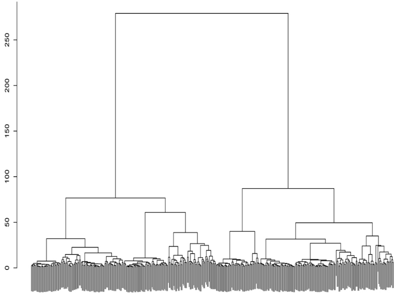
La Carte

1. A (? simple ?) idea
2. A (? not so ?) simple procedure
3. Some results
4. The Wishlist

La Carte

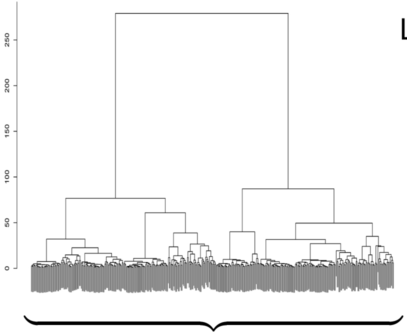
1. A (? simple ?) idea
2. A (? not so ?) simple procedure
3. Some results
4. The Wishlist

The (? not so ?) simple idea - notation



Let:

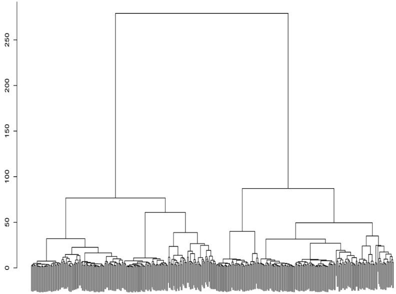
The (? not so ?) simple idea - notation



Let:

- *n* the number of objects to classify;

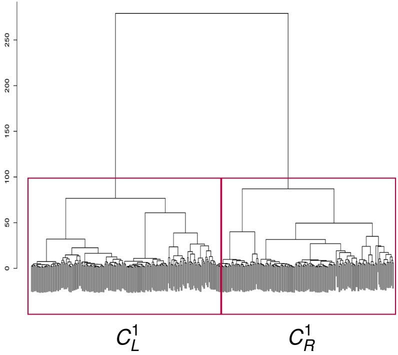
The (? not so ?) simple idea - notation



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)

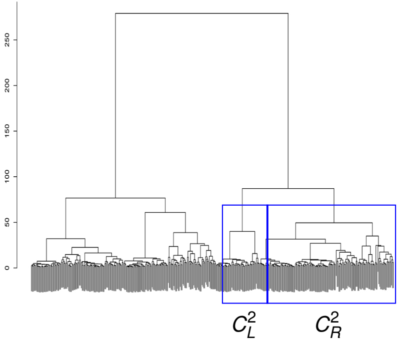
The (? not so ?) simple idea - notation



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)

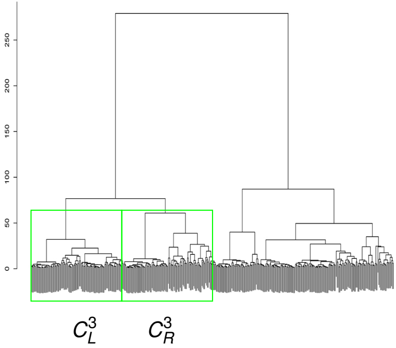
The (? not so ?) simple idea - notation



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)

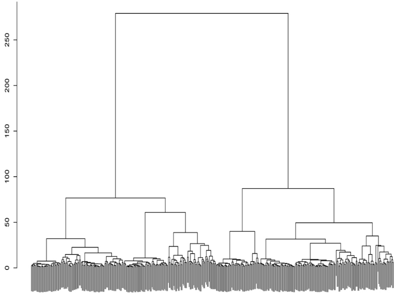
The (? not so ?) simple idea - notation



Let:

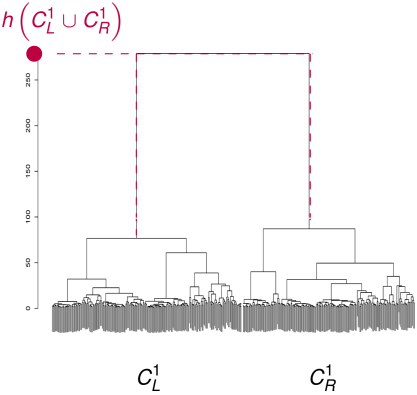
- n the number of objects to classify;
- C_L^k and C_R^k the two classes merged at level k (k=1,...,n-1)

The (? not so ?) simple idea - notation



Let:

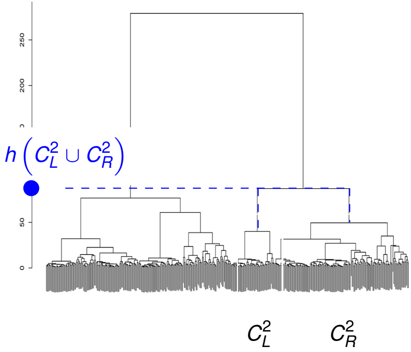
- n the number of objects to classify;
- C_L^k and C_R^k the two classes merged at level k (k=1,...,n-1)
- $h\left(C_L^k \cup C_R^k\right)$ the height necessary to merge C_L^k and C_R^k



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

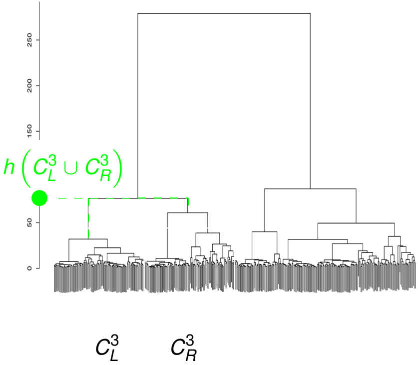
The (? not so ?) simple idea - notation



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

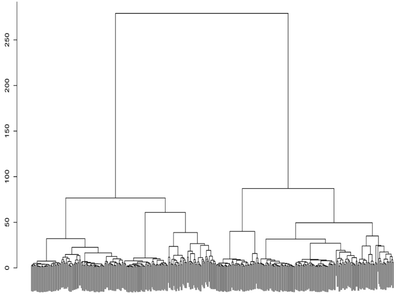
The (? not so ?) simple idea - notation



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

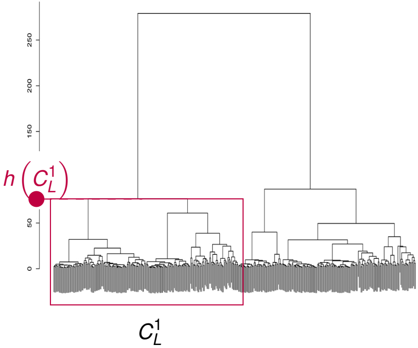
The (? not so ?) simple idea - notation



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 ($j \in \{ L, R \}$)

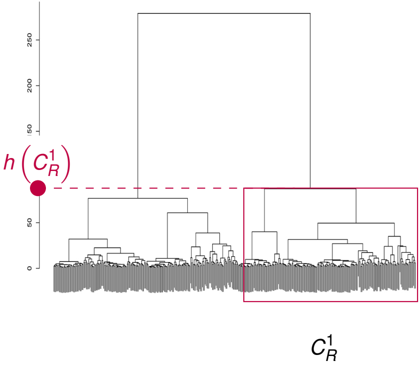
The (? not so ?) simple idea - notation



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 ($j \in \{ L, R \}$)

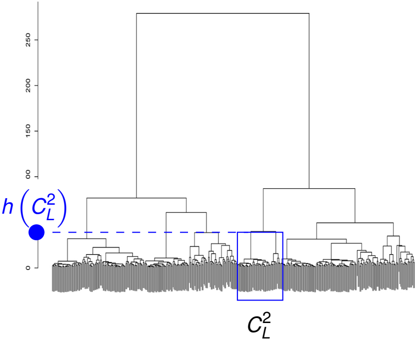
The (? not so ?) simple idea - notation



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 $\}$)

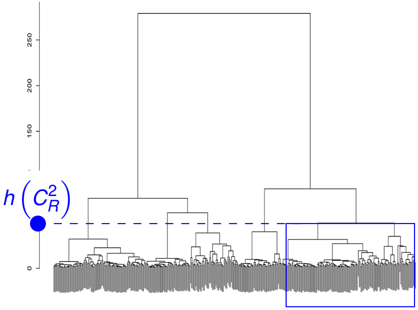
The (? not so ?) simple idea - notation



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 (? not so ?) simple idea - notation

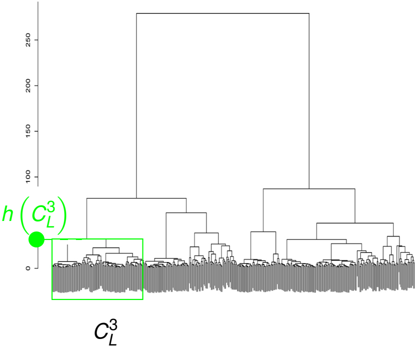


C_R^2

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 ($j \in \{ L, R \}$)

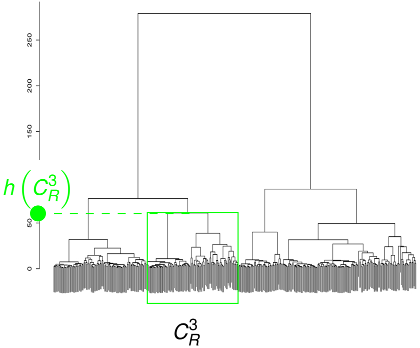
The (? not so ?) simple idea - notation



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 (? not so ?) simple idea - 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 (? simple ?) idea

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

The (? simple ?) idea

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 (? simple ?) idea

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 (? simple ?) idea

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

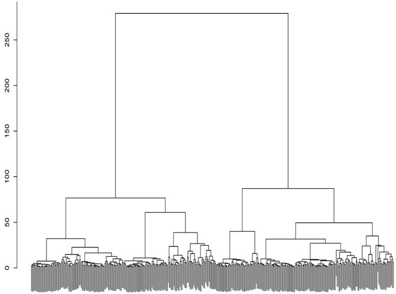
initialization:
aggregationLevelsToVisit $\leftarrow h(C_L^1 \cup C_R^1)$
permClusters $\leftarrow$ [ ]
i $\leftarrow$ 1
repeat
    if $C_L^i \equiv C_R^i$ **then**
        add $C_L^i \cup C_R^i$ to permClusters
    else
        add $h(C_L^i)$ and $h(C_R^i)$ to aggregationLevelsToVisit
        sort aggregationLevelsToVisit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

The (? simple ?) idea

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*

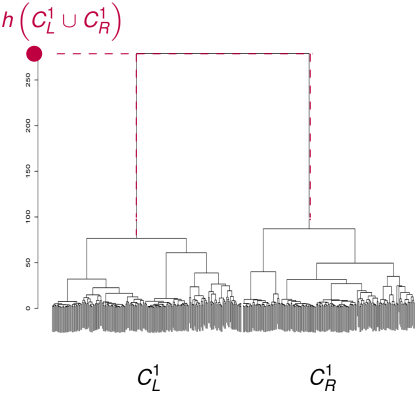
The (? not so ?) simple idea in action



Iteration

$i \leftarrow 1$

The (? not so ?) simple idea in action



Iteration

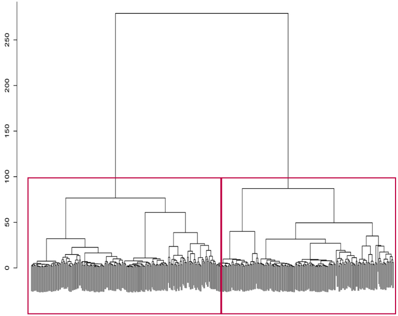
$i \leftarrow 1$

aggregationLevelsToVisit

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

permClusters

The (? not so ?) simple idea in action



C_L^1 C_R^1

Iteration

$i \leftarrow 1$

aggregationLevelsToVisit

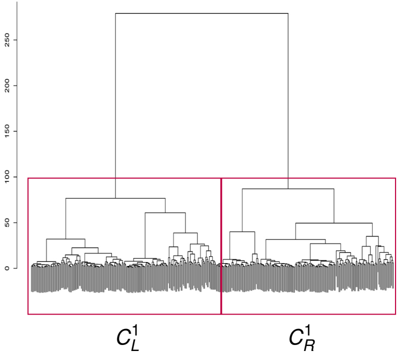
$h(C_L^1 \cup C_R^1)$

permClusters

clusters to compare

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

The (? not so ?) simple idea in action



Iteration

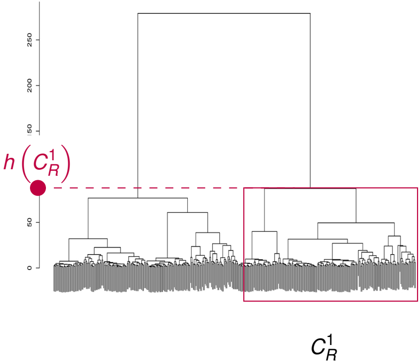
$i \leftarrow 2$

aggregationLevelsToVisit

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

permClusters

The (? not so ?) simple idea in action



Iteration

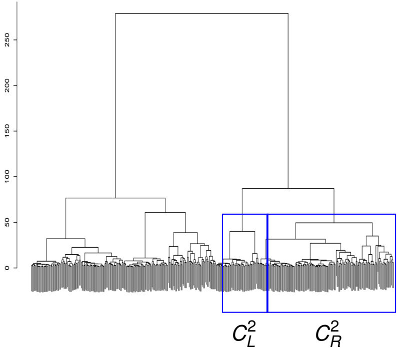
$i \leftarrow 2$

aggregationLevelsToVisit

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

permClusters

The (? not so ?) simple idea in action



Iteration

$i \leftarrow 2$

aggregationLevelsToVisit

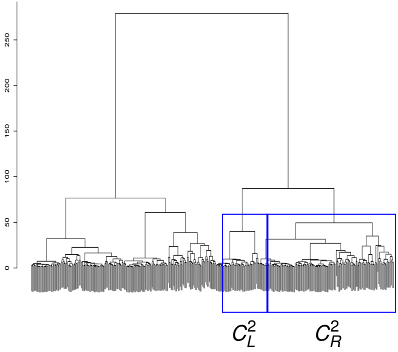
$h(C_R^1), h(C_L^1)$

permClusters

clusters to compare

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

The (? not so ?) simple idea in action



Iteration

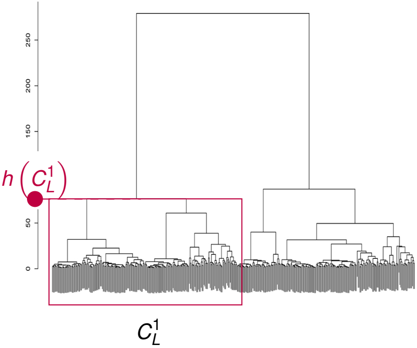
$i \leftarrow 3$

aggregationLevelsToVisit

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

permClusters

The (? not so ?) simple idea in action



Iteration

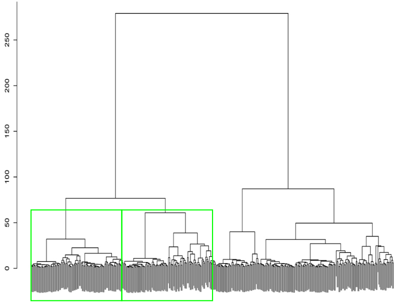
$i \leftarrow 3$

aggregationLevelsToVisit

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

permClusters

The (? not so ?) simple idea in action



C_L^3 C_R^3

Iteration

$i \leftarrow 3$

aggregationLevelsToVisit

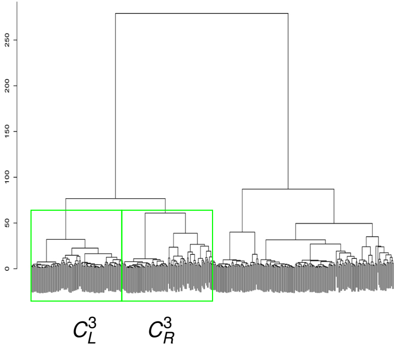
$h(C_L^1), h(C_R^2), h(C_L^2)$

permClusters

clusters to compare

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

The (? not so ?) simple idea in action



Iteration

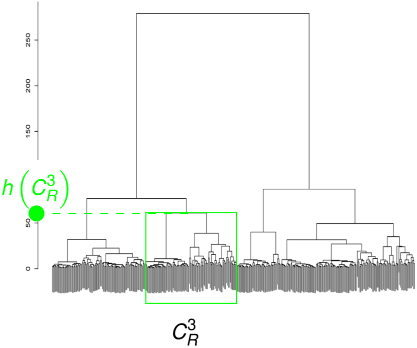
$i \leftarrow 4$

aggregationLevelsToVisit

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

permClusters

The (? not so ?) simple idea in action



Iteration

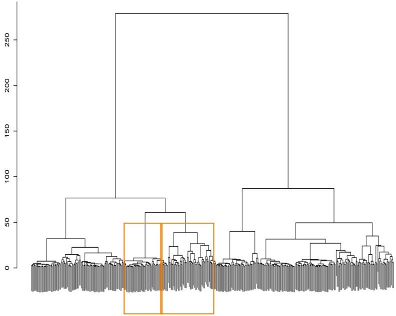
$i \leftarrow 4$

aggregationLevelsToVisit

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

permClusters

The (? not so ?) simple idea in action



C_L^4 C_R^4

Iteration

$i \leftarrow 4$

aggregationLevelsToVisit

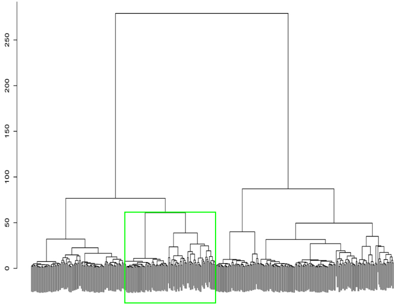
$h(C_R^3), h(C_R^2), h(C_L^2), h(C_L^3)$

permClusters

$C_L^4 \cup C_R^4$

clusters to compare

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



C_R^3

Iteration

$i \leftarrow 4$

aggregationLevelsToVisit

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

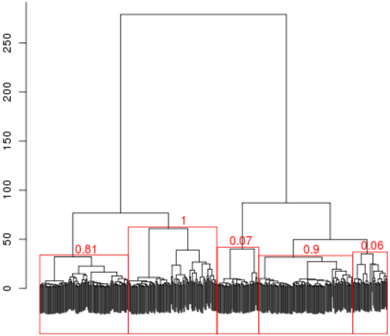
permClusters

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

clusters to compare

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

The (? not so ?) simple idea in action



Iteration

$i \leftarrow 9$

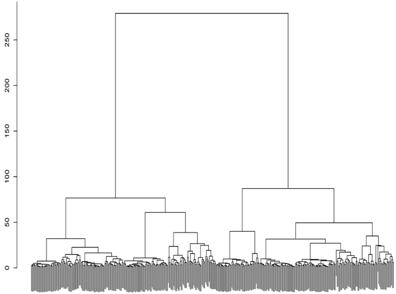
aggregationLevelsToVisit

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

permClusters

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

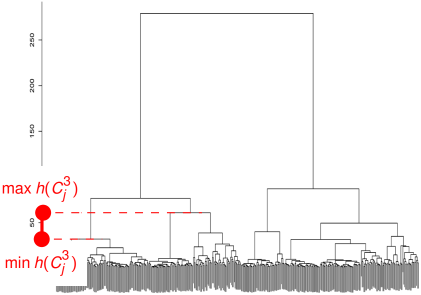
La Carte



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 (? not so ?) simple procedure



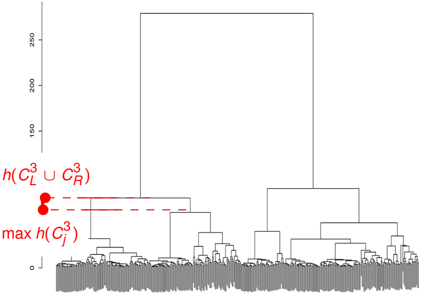
Let:

- 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 })

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

.

The (? not so ?) simple procedure



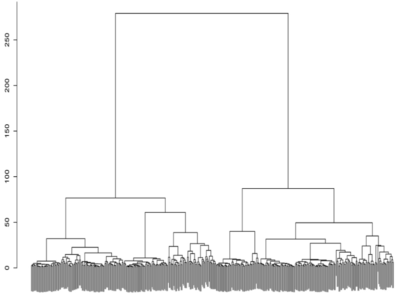
Let:

- 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 ∈ { L, R })

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 (? not so ?) simple procedure



Let:

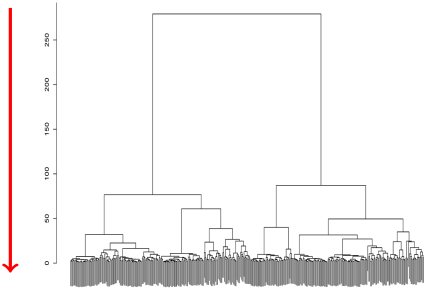
- n the number of objects to classify;
- C_L^k and C_R^k the two classes merged at level k (k=1,...,n-1)
- $h\left(C_L^k \cup C_R^k\right)$ the height necessary to merge C_L^k and C_R^k
- $h\left(C_j^k\right)$ the height at which C_j^k has been obtained (j $\in$ { L, R })

The ratio between these two costs:

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

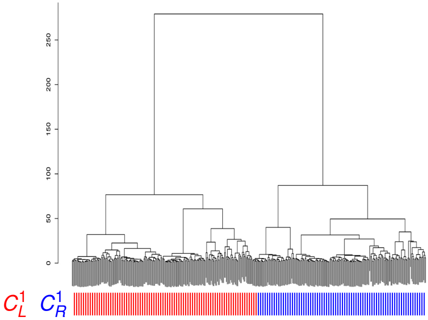
is thus a measure that characterizes the aggregation process resulting in the new class $C_L^k \cup C_R^k$

The (? not so ?) simple procedure: detail



The algorithm retraces down-ward the tree, starting from the root of the dendrogram where all objects are classified in a unique cluster.

The (? not so ?) simple procedure: detail

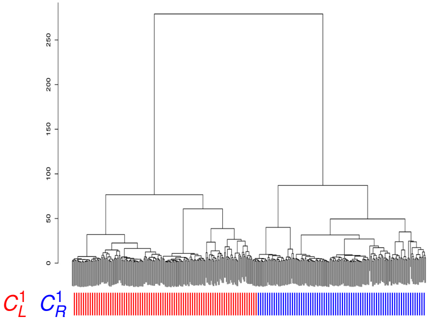


The algorithm retraces down-ward the tree, starting from the root of the dendrogram where all objects are classified in a unique cluster.

$\forall$ k a *permutation test* is designed to test the *Null Hypothesis* that the two classes C_L^k and C_R^k really belong to the same cluster, i.e. :

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

The (? not so ?) simple procedure: detail



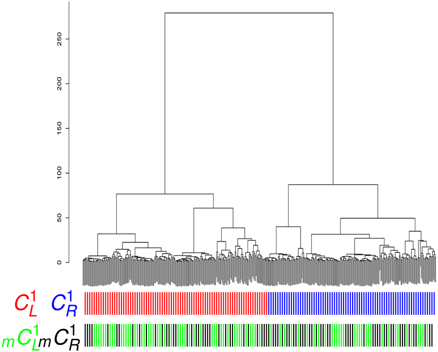
The algorithm retraces down-ward the tree, starting from the root of the dendrogram where all objects are classified in a unique cluster.

$\forall$ k a *permutation test* is designed to test the *Null Hypothesis* that the two classes 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 H_0, 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 (? not so ?) simple procedure: detail



The algorithm retraces down-ward the tree, starting from the root of the dendrogram where all objects are classified in a unique cluster.

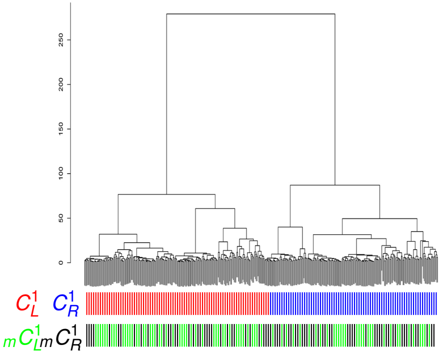
$\forall$ k a *permutation test* is designed to test the *Null Hypothesis* that the two classes C_L^k and C_R^k really belong to the same cluster, i.e. :

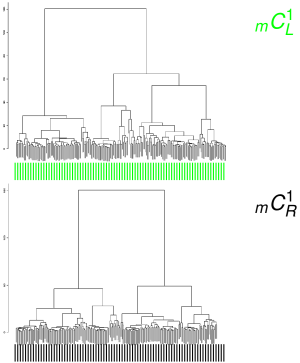
$$H_0: \quad C_L^k \equiv C_R^k$$

Under H_0, 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.

Let $_mC_L^k$ and $_mC_R^k$ be the two new classes obtained by permuting the elements in C_L^k and C_R^k

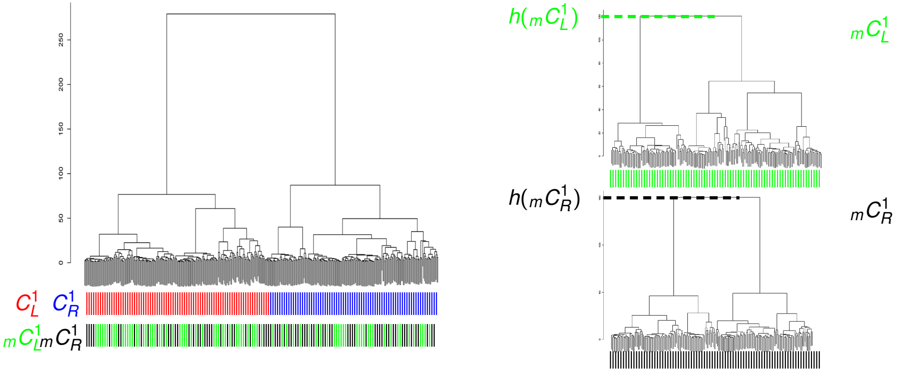
The (? not so ?) simple procedure: detail





Let $_mC_L^k$ and $_mC_R^k$ be the two new classes obtained by permuting the elements in C_L^k and C_R^k

For each of them a new dendrogram is generated.

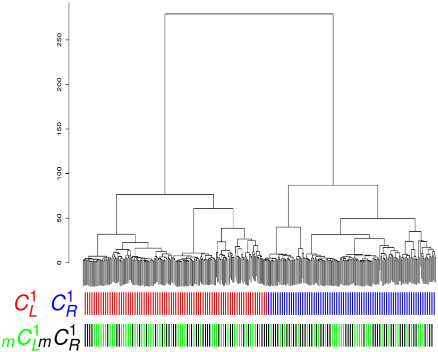


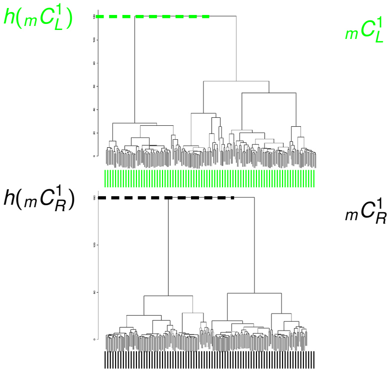
Let $_mC_L^k$ and $_mC_R^k$ be the two new classes obtained by permuting the elements in C_L^k and C_R^k

For each of them a new dendrogram is generated.

The heights at which each of the two classes are buit up again, clearly correspond to the heights of the root nodes of the corresponding dendrograms.

The (? not so ?) simple procedure: detail



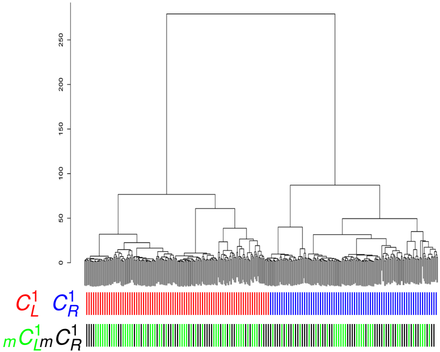


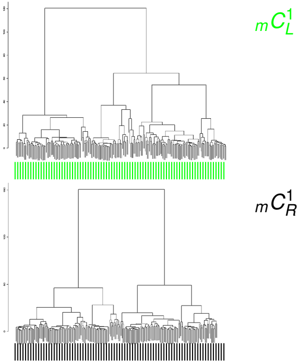
The ratio:

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

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

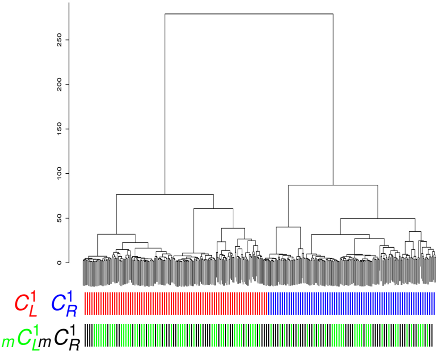
The (? not so ?) simple procedure: detail

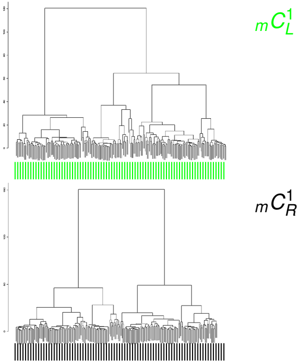




Under H_0 the aggregation process resulting in the new cluster $C_L^k \cup C_R^k$ should be very similar to the one that *potentially* produces ${}_mC_L^k \cup {}_mC_R^k$; thus the two values $cost\left({}_mC_L^k \cup {}_mC_R^k\right)$ and $cost\left(C_L^k \cup C_R^k\right)$ should be close enough.

The (? not so ?) simple procedure: detail





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

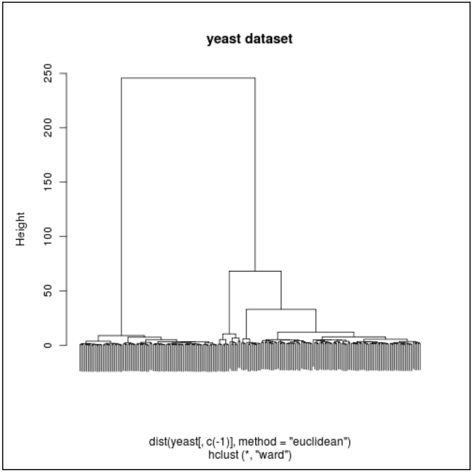
1 A (? simple ?) idea

2 A (? not so ?) simple procedure

3 Some results

4 The Wishlist

Some results



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.

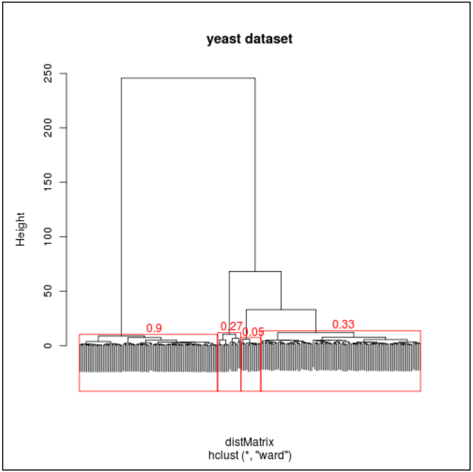
Science 2001, 292:929-934.

$n = 205$

$p = 80$

It is a subset of 205 genes that reflect four functional categories in the Gene Ontology listings.

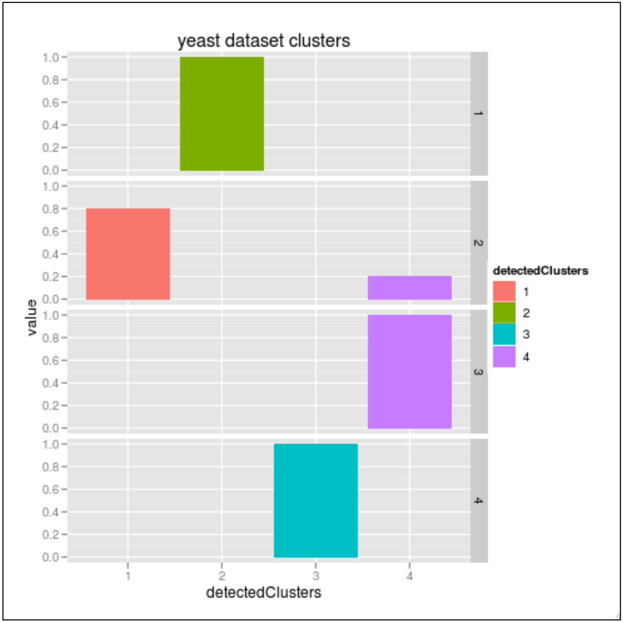
Some results



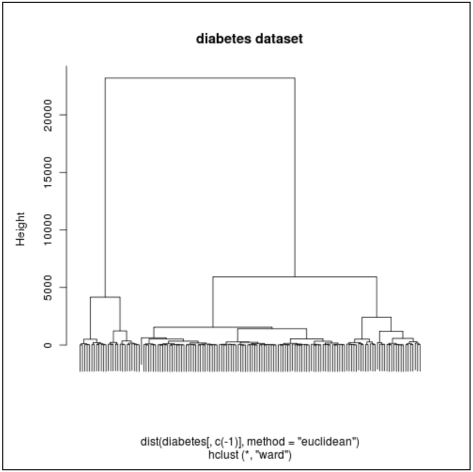
Settings

distanceMethod = euclidean
aggregationMethod = Ward
$\alpha = 0.05$
$M = 999$

Some results



Some results



The diabetes dataset

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

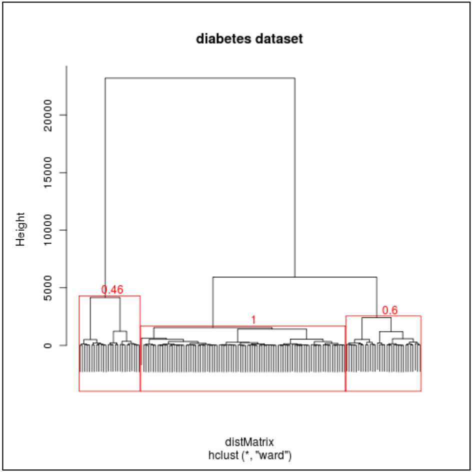
Biometrics, 1993, 49, 803-821.

$n = 145$

$p = 3$

It contains 145 subjects divided into three groups (normal, chemical diabetes, overt diabetes) on the basis of their oral glucose tolerance descripted by three variables

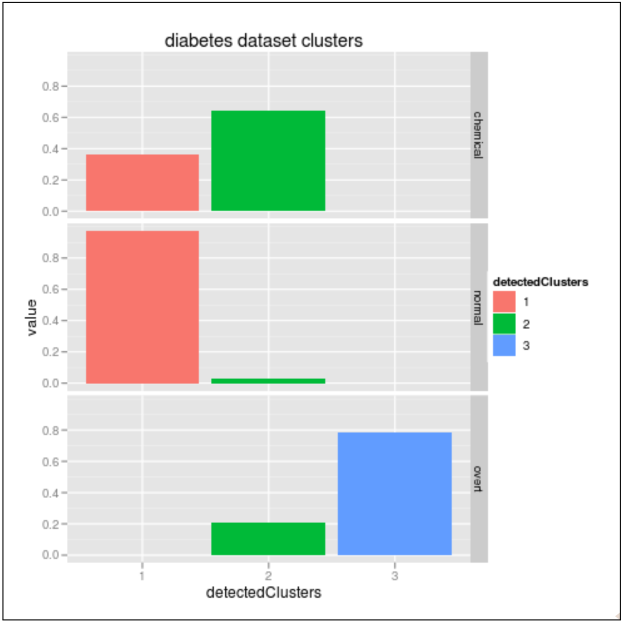
Some results



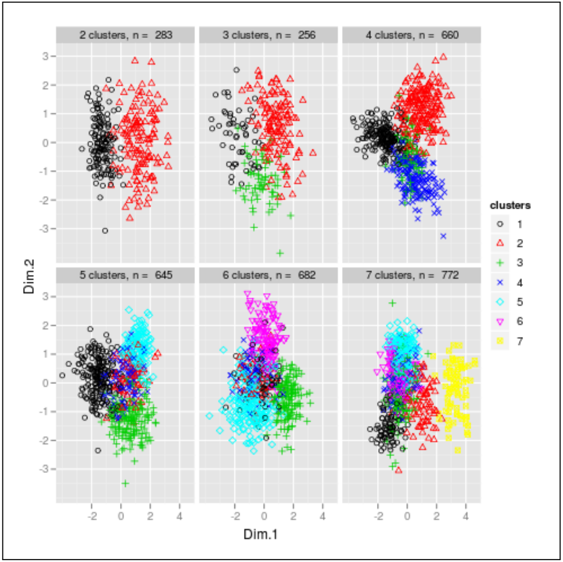
Settings

distanceMethod = euclidean
aggregationMethod = Ward
$\alpha = 0.05$
$M = 999$

Some results



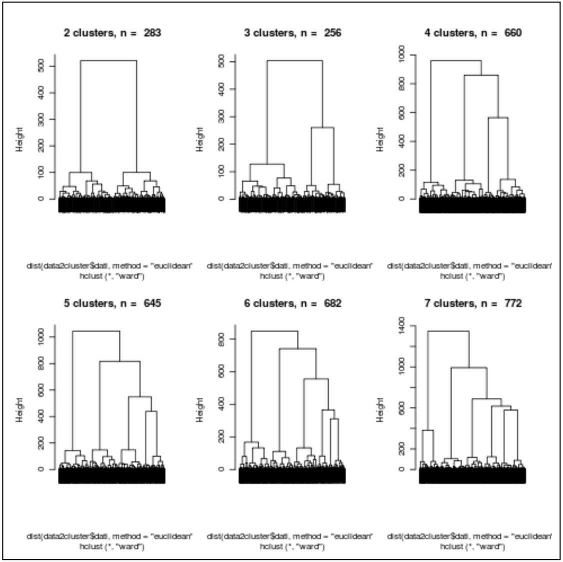
Some results... for 5 variables



genRandomCluster

numClust = 2:7
numNonNoisy = 5
sepVal = 0.01

Some results... for 5 variables



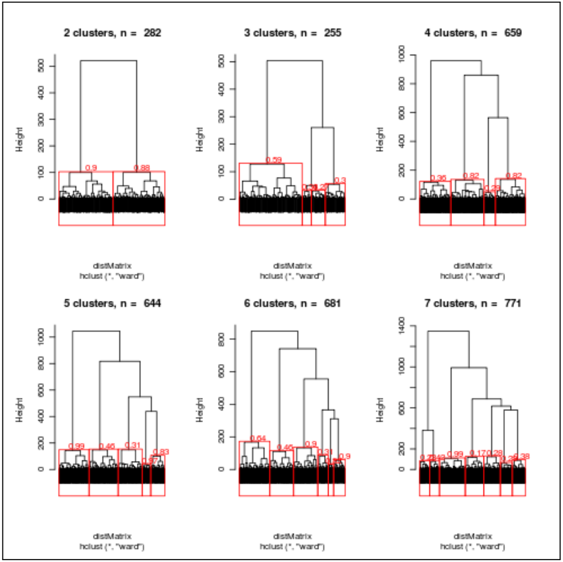
genRandomCluster

numClust = 2:7
numNonNoisy = 5
sepVal = 0.01

Settings

distanceMethod = euclidean
aggregationMethod = Ward

Some results... for 5 variables



genRandomCluster

numClust = 2:7

numNonNoisy = 5

sepVal = 0.01

Settings

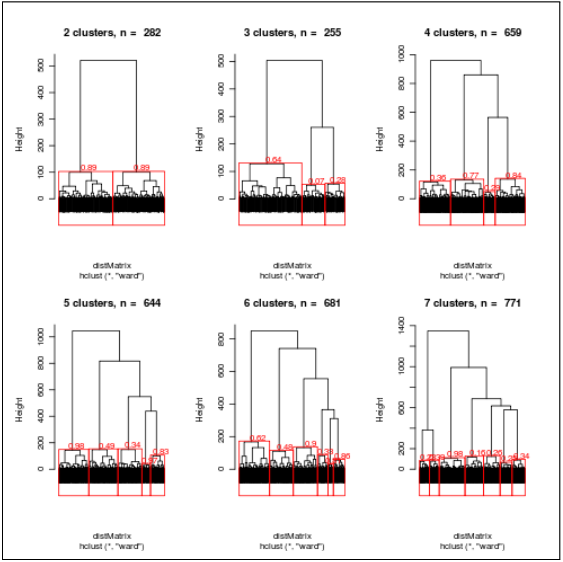
distanceMethod = euclidean

aggregationMethod = Ward

$M = 999$

$\alpha = 0.1$

Some results... for 5 variables



genRandomCluster

numClust = 2:7

numNonNoisy = 5

sepVal = 0.01

Settings

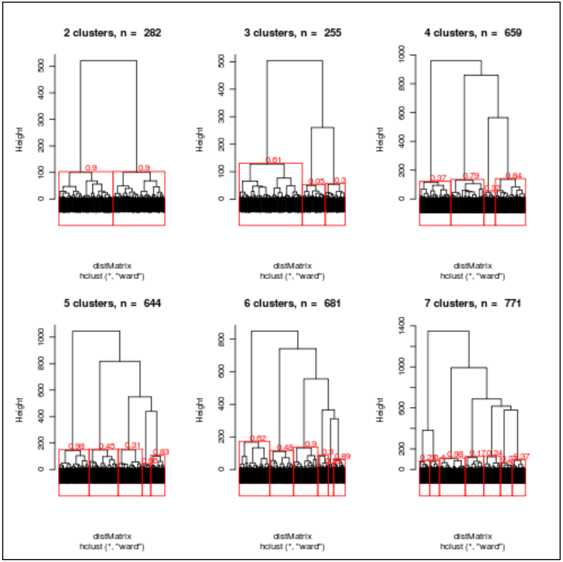
distanceMethod = euclidean

aggregationMethod = Ward

$M = 999$

$\alpha = 0.05$

Some results... for 5 variables



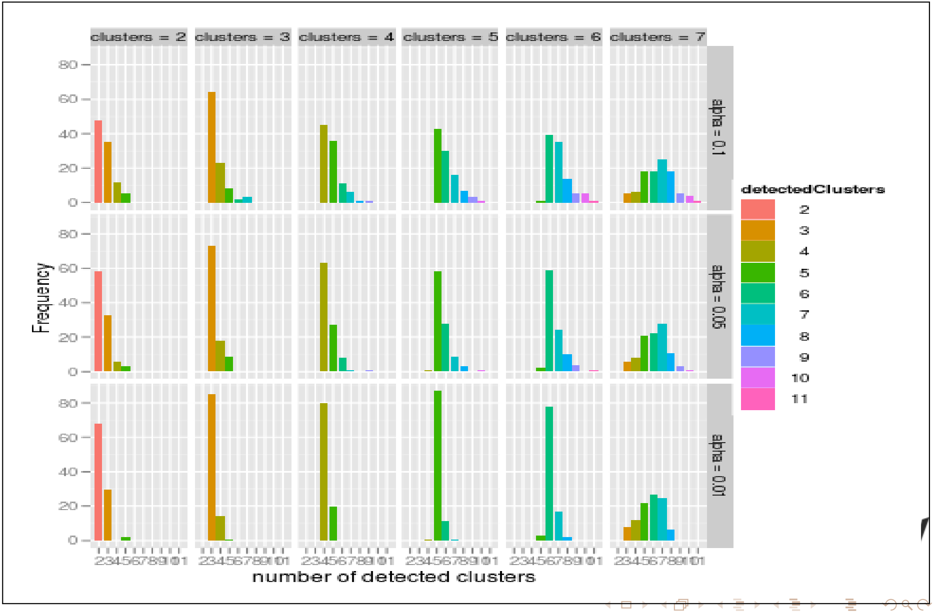
genRandomCluster

numClust = 2:7
numNonNoisy = 5
sepVal = 0.01

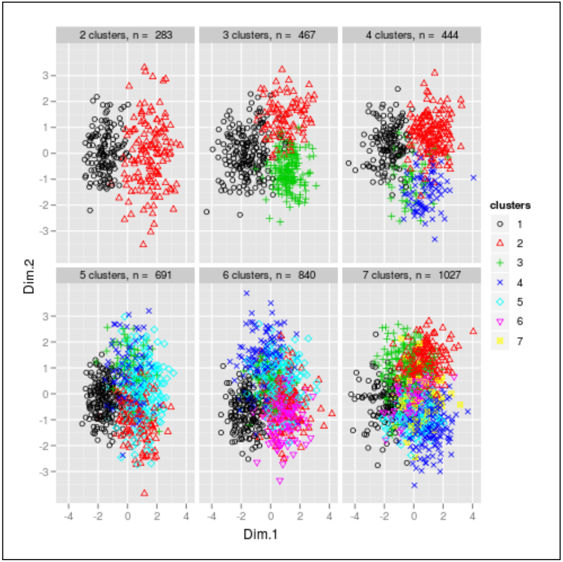
Settings

distanceMethod = euclidean
aggregationMethod = Ward
$M = 999$
$\alpha = 0.01$

Some results... for 5 variables (100 replications)



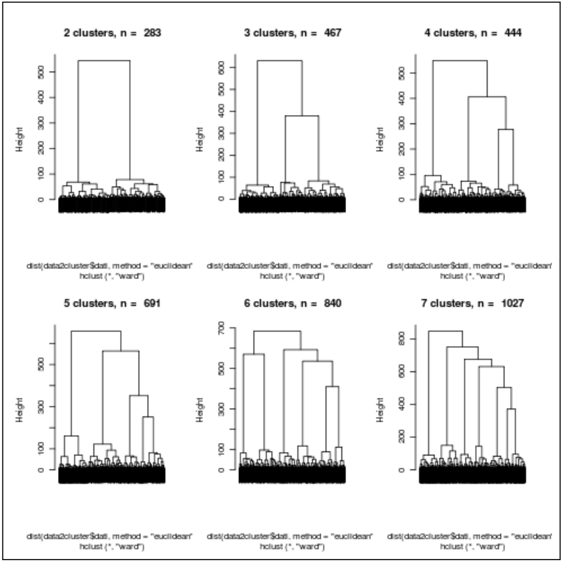
Some results... for 10 variables



genRandomCluster

numClust = 2:7
numNonNoisy = 10
sepVal = 0.01

Some results... for 10 variables



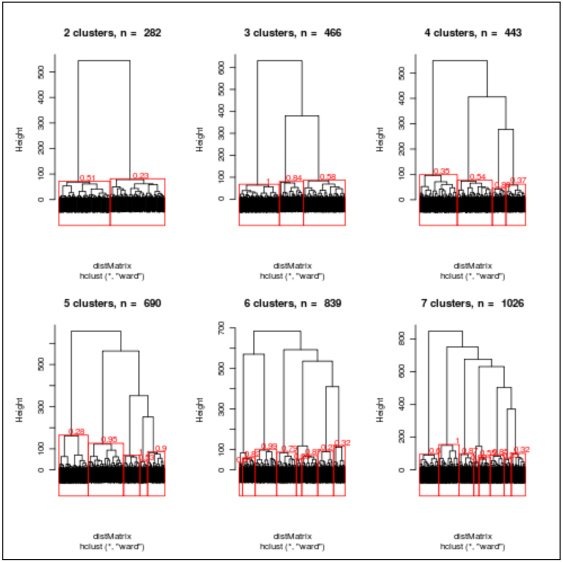
genRandomCluster

numClust = 2:7
numNonNoisy = 10
sepVal = 0.01

Settings

distanceMethod = euclidean
aggregationMethod = Ward

Some results... for 10 variables

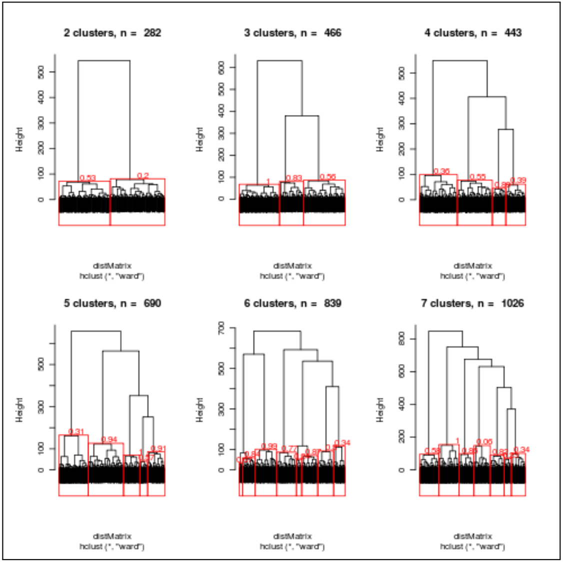


genRandomCluster

numClust = 2:7
numNonNoisy = 10
sepVal = 0.01

Settings

distanceMethod = euclidean
aggregationMethod = Ward
$M = 999$
$\alpha = 0.1$



genRandomCluster

numClust = 2:7

numNonNoisy = 10

sepVal = 0.01

Settings

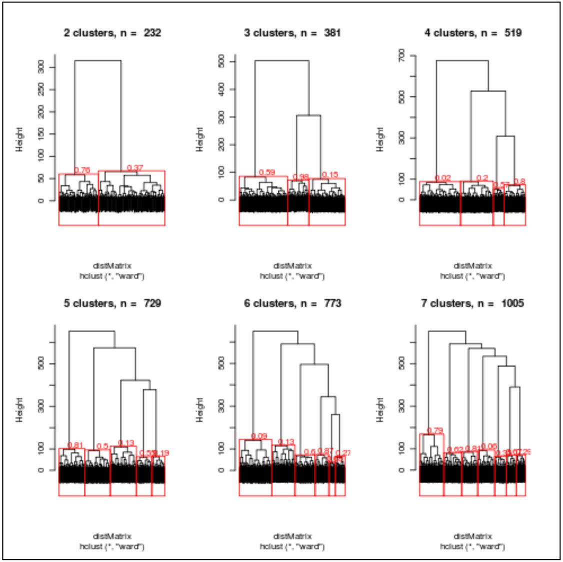
distanceMethod = euclidean

aggregationMethod = Ward

$M = 999$

$\alpha = 0.05$

Some results... for 10 variables



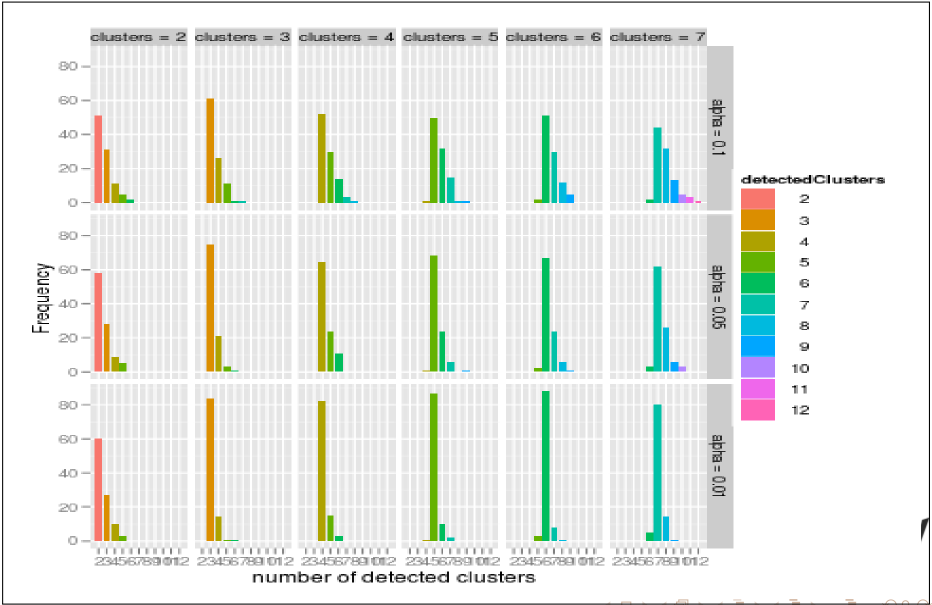
genRandomCluster

numClust = 2:7
numNonNoisy = 10
sepVal = 0.01

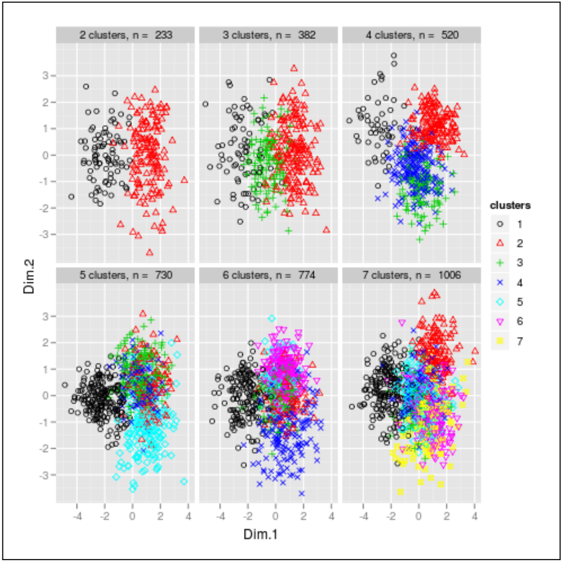
Settings

distanceMethod = euclidean
aggregationMethod = Ward
$M = 999$
$\alpha = 0.01$

Some results... for 10 variables (100 replications)



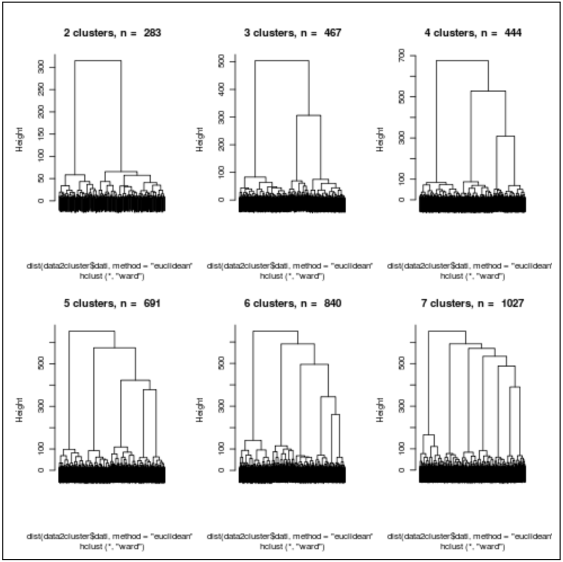
Some results... for 15 variables



genRandomCluster

numClust = 2:7
numNonNoisy = 15
sepVal = 0.01

Some results... for 15 variables

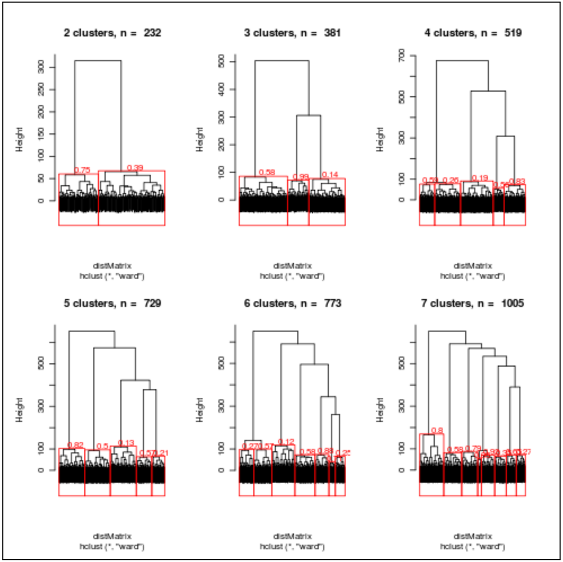


genRandomCluster

numClust = 2:7
numNonNoisy = 15
sepVal = 0.01

Settings

distanceMethod = euclidean
aggregationMethod = Ward



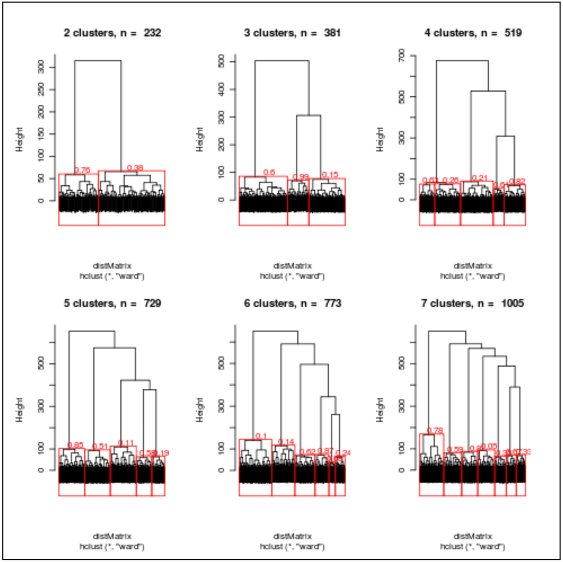
genRandomCluster

numClust = 2:7
numNonNoisy = 15
sepVal = 0.01

Settings

distanceMethod = euclidean
aggregationMethod = Ward
$M = 999$
$\alpha = 0.1$

Some results... for 15 variables



genRandomCluster

numClust = 2:7

numNonNoisy = 15

sepVal = 0.01

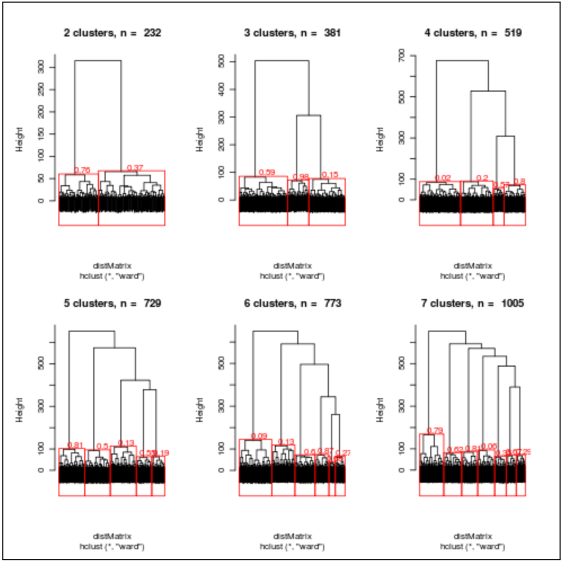
Settings

distanceMethod = euclidean

aggregationMethod = Ward

$M = 999$

$\alpha = 0.05$



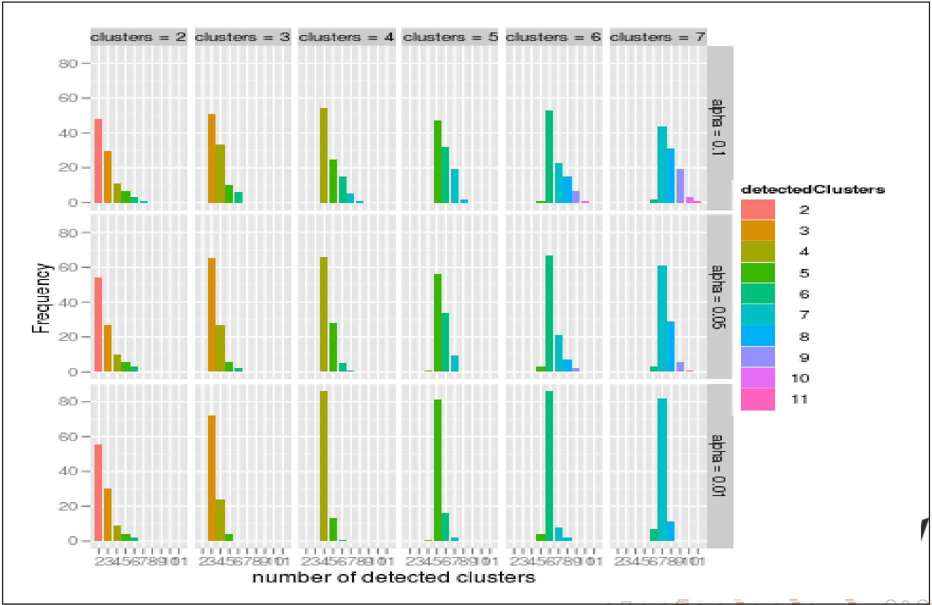
genRandomCluster

numClust = 2:7
numNonNoisy = 15
sepVal = 0.01

Settings

distanceMethod = euclidean
aggregationMethod = Ward
$M = 999$
$\alpha = 0.01$

Some results... for 15 variables (100 replications)



La Carte

1. A (? simple ?) idea
2. A (? not so ?) simple procedure
3. Some results
4. The Wishlist

The wishlist

Statistical issues

R issues

The wishlist

Statistical issues

- Quality measures of the obtained partition
- Use of different types of clusters
  - different cardinality of clusters
  - different type of cluster generation
- Study on the stability of the number of Montecarlo replications
- Computational complexity

R issues

The wishlist

Statistical issues

- Quality measures of the obtained partition
- Use of different types of clusters
  - different cardinality of clusters
  - different type of cluster generation
- Study on the stability of the number of Montecarlo replications
- Computational complexity

R issues

- profiling and optimizing the R code
- use of compiled code
- use of S3–S4 methods
- deploying a package