MASTER EN SCIENCES MATHÉMATIQUES

NBCLUST: un programme de détermination du nombre de classes pour des données symboliques

Dereppe, Stéphane

Award date:
2005

FUNDP
Faculté des Sciences
Département de Mathématique

Rempart de la Vierge, 8
B-5000 Namur Belgique

NBCLUST : Un programme de détermination du nombre de classes pour des données symboliques.

Mémoire présenté pour l'obtention
du grade de
Licencié en Sciences Mathématiques
par

Stéphane DEREPPE

Promoteur : Prof. A. Hardy

Année Académique 2004-2005

Je tiens à remercier tout particulièrement mon promoteur, Monsieur A. Hardy, qui m'a guidé, conseillé et aidé dans l'élaboration de ce mémoire. Je tiens également à remercier Madame A. Debaenst et Monsieur Y. Lechevallier qui m'ont aidé à résoudre un certain nombre de problèmes de type informatique.

Je remercie aussi l'ensemble des professeurs qui ont contribué à ma formation.
Enfin, je remercie ma famille ainsi que mes amis proches qui m'ont soutenu pendant ces quatre dernières années.

Résumé

Dans ce mémoire, nous nous intéressons à la détermination du nombre de classes pour des objets symboliques décrits par des combinaisons de variables quantitatives classiques, qualitatives classiques, intervalles, multivaluées et modales. Nous adaptons les meilleures méthodes de détermination du nombre de classes issues de l'étude Milligan et Cooper au programme de classification symbolique Sclust. Nous testons la méthode sur différents ensembles de données artificiels et réels et comparons les résultats obtenus avec les différentes distances (L_1, L_2, Hausdorff et De Carvalho).

Abstract

In this report, we are interrested in the determination of the number of clusters for symbolic data described by combinations of classical quantitatives, classical qualitatives, intervals, multi-valued and modals variables. We adapt the best methods of determination of the number of clusters stemmed from the study of Milligan and Cooper to the program of symbolic classification Sclust. We test the method on various artificials and reals data sets and compare the results obtained with the differents distances (L_1, L_2, Hausdorff et De Carvalho).

Table des matières

Introduction.

Depuis quelques années, les progrès réalisés dans le domaine de l'informatique permettent de collecter facilement un grand nombre de données.

Le problème devient alors d'analyser et de tirer des informations utiles de ces données. C'est pourquoi un grand nombre de méthodes, statistiques ou autres, ont été développées dans cette optique.

Ce mémoire s'interesse à l'une de ces techniques: **la classification automatique**. Le but de la classification est d'obtenir, à partir d'une population d'individus (ces individus peuvent être des animaux, des pays, ...) sur lesquels un certain nombre de variables ont été mesurées, une partition de la population de départ en un nombre de classes généralement fixé a priori.
Une des difficultés rencontrée lors d'une classification est la détermination du nombre optimal de classes. En effet, comme nous l'avons dit ci-dessus, un grand nombre de méthodes supposent que le nombre de classes est connu.
Il existe, dans la littérature, une multitude d'indices permettant de rechercher ce nombre. C'est pourquoi, en 1988, Milligan et Cooper ont testé une trentaine de ces indices et ont établi un classement.

Les données symboliques sont une extension des données classiques. Elles sont décrites par des variables intervalles, multivaluées catégoriques, multivaluées quantitatives et modales et permettent de caractériser un grand nombre d'individus sans perdre trop d'informations.

Le but de ce mémoire est d'adapter les meilleures méthodes de détermination du nombre de classes selon le classement de Milligan et Cooper à tous les types de données, qu'elles soient classiques, symboliques ou les deux simultanément.

Nous commencerons par présenter de manière formelle les données classiques et symboliques ainsi quelques méthodes de classification automatique. Nous nous attarderons plus particulièrement sur la méthode des nuées dynamiques avant de présenter les cinq meilleures méthodes de détermination du nombre de classes de Milligan et Cooper.
Nous détaillerons ensuite le programme de classification symbolique Sclust avant d'appliquer nos méthodes de détermination du nombre de classes à plusieurs jeux de données.

Chapitre 1

Les données classiques.

1.1 Introduction.

Considérons :
- $\Omega = \{x_1, x_2, \ldots, x_n\}$, un ensemble de n individus.
- $Y_1, \ldots, Y_p$, les p variables qui caractérisent chacun des individus.
- $\mathcal{Y}_j$, l'ensemble des valeurs prises par la variable Y_j ($j \in \{1, \ldots, p\}$), appelé espace ou domaine d'observation de la variable Y_j.

On définit alors **une variable classique** par:

$$\begin{array}{rccl} Y_j : & \Omega & \to & \mathcal{Y}_j \\ & x_k & \rightsquigarrow & Y_j(x_k) = x_{kj} \end{array}$$

où x_{kj} est la valeur observée de la variable j pour l'individu k.
Toutes ces valeurs peuvent être placées dans une matrice de données

$$\tilde{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}$$

1.2 Types de variables.

Nous allons, à présent, distinguer deux types de variables :
- les variables quantitatives et
- les variables qualitatives,

ainsi que les différentes mesures de proximité δ_j entre deux individus $x, y \in \Omega$.
N.B. : par la suite, les indices de variables seront omis.

1.2.1 Les variables quantitatives.

Une variable quantitative est une variable dont l'espace d'observation $\mathcal{Y}$ est tel que $\mathcal{Y} \subseteq \mathbb{R}$.

Variable quantitative continue.

Une variable quantitative est dite continue si elle prend un nombre infini non dénombrable de valeurs dans $\mathbb{R}$.
On distingue alors les 3 situations suivantes:

- $\mathcal{Y} = \mathbb{R}$ ou
- $\mathcal{Y} = \mathbb{R}^+$ ou
- $\mathcal{Y} = [a,b] = \{x \in \mathbb{R} \quad | \quad a \leq x \leq b\}$

Exemple 1.2.1.
La variable Y : "Poids d'un véhicule (en kilogramme)" admet l'espace d'observation $\mathcal{Y} = \mathbb{R}^+$.

Variable quantitative discrète.

Une variable quantitative Y est dite discrète si l'espace d'observation $\mathcal{Y}$ contient un nombre fini ou infini dénombrable de valeurs $\xi_i \in \mathbb{R}$.
Dans ce cas, on distingue deux possibilités :

- $\mathcal{Y} = \{\xi_1, \ldots, \xi_N\} \subset \mathbb{R}$ pour un certain entier N ou
- $\mathcal{Y} = \{\xi_1, \xi_2, \ldots\} \subset \mathbb{R}$

Exemple 1.2.2.
La variable Y : "Nombre de gibiers recensés dans une forêt" admet l'espace d'observation $\mathcal{Y} = \{\xi_1, \xi_2, \ldots\} = \mathbb{N}$

1.2.2 Les variables qualitatives.

Une variable Y est dite qualitative ou catégorique si le nombre de valeurs de l'espace d'observation $\mathcal{Y}$ est fini et si les éléments de $\mathcal{Y}$ n'ont aucune signification numérique.
Les éléments de $\mathcal{Y}$ sont appelés des catégories (ou des modalités).

Variable qualitative nominale.

Une variable qualitative est dite nominale si elle a des modalités distinctes les unes des autres, mais sans structure interne, c'est-à-dire, sans possibilité d'ordre ou de calcul entre elles.
Dans ce cas, $\forall x,y \in \mathcal{Y}$, nous ne pouvons distinguer que $x = y$ ou $x \neq y$.

La mesure de proximité $\delta(x,y)$ entre deux catégories $x,y \in \mathcal{Y}$ est alors définie par

$$\delta(x,y) = \begin{cases} 1 & \text{si} \quad x = y \\ 0 & \text{si} \quad x \neq y \end{cases}$$

Exemple 1.2.3.
La variable Y : "la couleur d'une voiture" a pour espace d'observation $\mathcal{Y}$ = { rouge, noire, blanche, verte, bleue, jaune }.
On a bien que $|\mathcal{Y}| < \infty$.

Dans le cas particulier de variables binaires ou dichotomiques, l'espace d'observation $\mathcal{Y}$ ne comprend que deux alternatives qui sont codées par 0 ou 1.

Exemple 1.2.4.
La variable Y : "le vote à un référendum" a pour espace d'observation $\mathcal{Y}$ = { pour , contre } que nous pouvons coder par $\{0,1\}$.

Si la variable considérée possède plus de deux modalités, on peut coder ces s catégories par $0,1,\ldots,s-1$.
Cependant, aucune opération arithmétique ne peut être définie avec ces codes.

Exemple 1.2.5.
Les couleurs des voitures peuvent être codées $\mathcal{Y} = \{0,1,2,3,4,5\}$ où les codes 0, 1, 2, 3, 4, 5 correspondent respectivement aux modalités rouge, noire, blanche, verte, bleue et jaune.

Variable qualitative ordinale.

Une variable qualitative est ordinale si l'espace des observations $\mathcal{Y}$ est muni d'un ordre total $\prec$ tel que $\forall x,y \in \mathcal{Y}$, $x \neq y$, nous avons soit $x \prec y$, soit $y \prec x$.
C'est-à-dire que les modalités prises par la variable Y peuvent être hiérarchisées entres elles mais qu'aucun calcul ne peut être défini.

Comme pour les variables nominales, les s différentes modalités d'une variable peuvent être codées de sorte que $\mathcal{Y} = \{0,1,\ldots,s-1\}$.
On peut alors se servir de ce codage pour définir une mesure de proximité entre deux individus $x,y \in \Omega$:

$$\delta(x,y) = |x - y| \qquad \forall x,y \in \mathcal{Y}.$$

Cette mesure représente le nombre de catégories strictement comprises entre x et y selon l'ordre total imposé plus un.

Exemple 1.2.6.
La variable Y : "le résultat d'un étudiant en première année" a pour espace d'observation $\mathcal{Y}$ = { ajournement, satisfaction, distinction, grande disctinction, la plus

grande distinction, la plus grande distinction avec félicitation du jury }.
Et on a δ(*ajournement, distinction*) $= |0-2| = 2$.

1.3 Vecteurs et matrices de données.

Considérons de nouveau, l'ensemble de n individus $\Omega = \{x_1,x_2,\ldots,x_n\}$ caractérisés par p variables notées $Y_1,\ldots,Y_p$ avec $\mathcal{Y}_j$ qui représente l'espace d'observation de la variable Y_j où $j = 1,\ldots,p$.

Nous désignons par X le vecteur des p variables $Y_1,\ldots,Y_p$:

$$X = \begin{pmatrix} Y_1 \\ \vdots \\ Y_p \end{pmatrix} \in \chi := \bigotimes_{j=1}^{p} \mathcal{Y}_j.$$

où $\chi := \bigotimes_{j=1}^{p} \mathcal{Y}_j$ est le produit cartésien des p espaces d'observation $\mathcal{Y}_1,\ldots,\mathcal{Y}_p$.
Pour chaque individu $x_k \in \Omega$, nous pouvons représenter les p observations $x_{k1},\ldots,x_{kp}$ dans un vecteur colonne p-dimensionnel.

$$x_k = X(k) = \begin{pmatrix} x_{k1} \\ x_{k2} \\ \vdots \\ x_{kp} \end{pmatrix} \in \chi := \bigotimes_{j=1}^{p} \mathcal{Y}_j$$

En prenant en compte les n individus, nous obtenons la matrice de données classique déjà présentée précédemment:

$$\tilde{X} = (x_{kj})_{n\times p} = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1p} \\ x_{21} & x_{22} & \ldots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{np} \end{pmatrix} = \begin{pmatrix} x_1' \\ x_2' \\ \vdots \\ x_n' \end{pmatrix} = (y_1,\ldots,y_p)$$

tel que la k-ième ligne x_k' contient les données observées pour l'individu x_k et la j-ième colonne y_j représente les valeurs prises par la variable Y_j.

1.4 Exemple.

Considérons $\Omega = \{a,b,c,d\}$ un ensemble de 4 véhicules.
Soient

- Y_1 = la longueur en cm du véhicule (variable quantitative continue),
- Y_2 = le statut utilitaire du véhicule (0= non , 1= oui) (variable qualitative binaire),

- Y_3 = la couleur du véhicule (variable qualitative nominale),
- Y_4 = l'état du véhicule (0= mauvais, 1= moyen, 2=bon, 3= parfait) (variable qualitative ordinale),

On obtient la matrice de données $\tilde{X}$ suivante:

	Y_1	Y_2	Y_3	Y_4
a	276	0	rouge	1
b	375	0	bleue	3
c	532	1	blanche	0
d	321	1	noire	2

Remarque:
Cette matrice de données est un tableau mixte car elle contient des variables qui ne sont pas toutes de la même nature.

Chapitre 2

Les données symboliques.

2.1 Introduction.

Les données symboliques permettent de représenter des objets plus complexes que les données classiques.
Nous allons définir trois types de variables symboliques:

- les variables intervalles,
- les variables multivaluées et
- les variables modales.

L'ensemble des objets E peut être défini de deux manières différentes:

1. un ensemble $E = \Omega = \{x_1, \ldots, x_n\}$ d'individus appelés **objets du premier ordre**;
2. un ensemble $E = \Omega = \{C_1, C_2, \ldots\}$ de classes (telle que $C_i \subseteq \Omega$) d'individus appelés **objets du second ordre**

2.2 Variable à valeurs d'ensemble.

Une variable Y, dont l'espace d'observation est $\mathcal{Y}$, est dite **à valeurs d'ensemble** lorsque, $\forall x_k \in E$,

$$\begin{array}{rccc} Y: & E & \to & \mathcal{B} \\ & x_k & \rightsquigarrow & Y(x_k) \end{array}$$

où $\mathcal{B} = \mathcal{P} = \{U \neq \emptyset \mid U \subseteq \mathcal{Y}\}$.
On a donc que $Y(x_k)$ est un sous-ensemble de $\mathcal{Y}$.
Remarque:
On retrouve la définition d'une variable classique à une seule valeur dans le cas où $|Y(x_k)| = 1, \quad \forall x_k \in E$, avec $|Y(x_k)|$ qui représente le nombre de valeurs que prend la variable Y pour décrire l'individu $x_k \in E$.

2.3 Variable de type intervalle.

Définition.

Une variable à valeurs d'ensemble est de **type intervalle** si $\forall x_k \in E$, l'ensemble $U \equiv Y(x_k)$ est un intervalle de $\mathbb{R}$.
Nous avons donc

$$\begin{array}{rccl} Y: & \Omega & \to & \mathcal{B} \\ & x_k & \rightsquigarrow & Y(x_k) = [\alpha,\beta] \end{array}$$

où $\alpha,\beta \in \mathbb{R}$ sont tels que $\alpha \leq \beta$.
$\mathcal{B}$ devient donc l'ensemble $\mathcal{I}$ des intervalles fermés bornés de $\mathbb{R}$.

Exemple 2.3.1.
Soient

- $E = \{$ *les étudiants des facultés de Namur.* $\}$
- $Y =$ *"le nombre d'heures passées sur internet par semaine" (minimum et maximum sur chaque semaine pendant le premier semestre);*
- $\mathcal{B} = \{[\alpha,\beta] \mid \alpha,\beta \in \mathbb{R}^+, 0 \leq \alpha \leq \beta < \infty\}$;
- $\mathcal{Y} = \mathbb{R}$.

On peut alors obtenir les résultats suivants:

$$\begin{array}{rcl} Y(x_i) & = & [5,15] \\ Y(x_j) & = & [10,22] \end{array}$$

où $x_i,x_j \in E$.

2.4 Variable multivaluée.

Définition

Une variable à valeurs d'ensemble est **multivaluée** si les valeurs de $Y(x_k)$ sont toutes des sous-ensembles finis de $\mathcal{Y}$.
Nous avons donc $|Y(x_k)| < \infty, \quad \forall x_k \in E$.

Une variable à valeurs d'ensemble est multivaluée **catégorique** si *l'espace d'observation* $\mathcal{Y}$ contient un nombre fini de catégories.

Une variable à valeurs d'ensemble est multivaluée **quantitative** si les valeurs $Y(x_k)$ sont des ensembles finis de nombres réels, c'est-à-dire:
$Y(x_k) \subset \mathbb{R}$ et $|Y(x_k)| < \infty, \forall x_k \in E$.

Exemple 2.4.1.
Soient :
- $E =$ { *les pièces d'une maison* };
- $Y =$ *"les différents meubles dans la pièce";*
- $\mathcal{Y} =$ {*lit, chaise, bureau, table, étagère, armoire,...*}

On peut alors obtenir les résultats suivants:

$$\begin{aligned} Y(x_i) &= \{ \textit{chaise, table, armoire} \} \\ Y(x_j) &= \{ \textit{lit, chaise, bureau} \} \end{aligned}$$

où $x_i, x_j \in E$.

2.5 Variable modale.

Définition.

Une variable **modale** Y d'espace d'observation $\mathcal{Y}$ sur $E = \{x_1, \ldots, x_n\}$ est une variable multivaluée pour laquelle :
- $\forall x_k \in E$, $Y(x_k) \subset \mathcal{Y}$
- $\forall y \in Y(x_k)$, nous définissons un poids, une probabilité ou encore une fréquence $w(y)$ qui indique la pertinence de la catégorie y pour l'objet x_k.

Formellement, une variable modale Y sur un ensemble $E = \Omega = \{x_1, \ldots, x_n\}$ d'objets à valeurs dans $\mathcal{Y}$ est définie par:

$$Y(x_k) = (U(x_k), \pi_k), \ \forall x_k \in E$$

où
- π_k est une mesure ou une distribution (poids, probabilité ou fréquence) sur les valeurs possibles de $\mathcal{Y}$, et
- $U(x_k) \subseteq \mathcal{Y}$ est le support de π_k dans le domaine $\mathcal{Y}$.

Exemple 2.5.1.
Soient :
- $E =$ {$x_1, x_2, x_3, \ldots$} *un ensemble d'orchestres;*
- $Y =$ *"le type de musiciens jouant dans l'orchestre";*
- $\mathcal{Y} =$ { *guitariste, violoniste, pianiste, percussionniste, trompettiste,...*}

On pourrait obtenir le résultat sous forme d'histogramme :

$$Y(x_i) = \{(\textit{guitariste}, 0.2), (\textit{pianiste}, 0.1), (\textit{violoniste}, 0.5), (\textit{percussionniste}, 0.2)\}$$

2.6 Variable symbolique par agrégation.

Considérons un ensemble $\Omega = \{x_1,\ldots,x_n\}$ d'individus (aussi appelés objets du premier ordre) et la variable classique $\tilde{Y}$ ayant comme espace d'observation $\mathcal{Y}$.
Soit $E = \{C_1,\ldots,C_m\}$ un ensemble de classe $C_i \subseteq \Omega$ d'individus (aussi appelés objets du second ordre).

Nous allons caractériser le comportement de ces classes en fonction de la variable $\tilde{Y}$.
C'est-à-dire qu'à partir de la variable $\tilde{Y}$, nous allons définir une nouvelle variable "**agrégée**" qui spécifiera les valeurs prises par la variable $\tilde{Y}$ sur les classes C_i.

Exemple de variable symbolique par agrégation

Soient:
- Ω = { élèves d'une école secondaire };
- $E = \{C_1,C_2,\ldots,C_6\}$ = regroupement des élèves selon leurs années respectives;

Exemple 2.6.1. *Variable intervalle par agrégation.*

Soit la variable classique $\tilde{Y}$:"moyenne des résultats obtenus aux examens de Noël en pourcentage".
Nous pouvons décrire la catégorie d'année C_i par $Y(C_i) = [\alpha,\beta]$ où

$$\begin{aligned} \alpha &= \min_{x_k \in C_i} \{\tilde{Y}(x_k)\} \\ \beta &= \max_{x_k \in C_i} \{\tilde{Y}(x_k)\}. \end{aligned}$$

On peut donc obtenir comme résultat pour les élèves de troisième année:

$$Y(C_3) = [42,93].$$

Exemple 2.6.2. *Variable multivaluée par agrégation.*

Soit la variable classique $\tilde{Y}$:" résultat obtenu au cours de mathématique au bulletin de Noël (en pourcentage)".
La description de la classe de cinquième année (C_5) pourrait être

$$Y(C_5) = \{60,80,70,65,83,64,\ldots\}.$$

Exemple 2.6.3. *Variable modale par agrégation.*

Soit la variable classique $\tilde{Y}$:"matière où l'élève a eu le meilleur résultat".
Ici, la description de la classe C_4 (les élèves de quatrième année) pourrait être donnée par

$$Y(C_i) = \{\ (mathématique,0.4),\ (francais,0.1),\ (anglais,0.5)\}.$$

2.7 Résumé sur les données symboliques.

Une **variable symbolique** d'espace d'observation $\mathcal{Y}$ est définie de la manière suivante:

$$\begin{array}{rccll} Y: & E & \to & \mathcal{B} & \forall x_k \in E \\ & x_k & \rightsquigarrow & Y(x_k) & \end{array}$$

où $\mathcal{B} = \mathcal{P}(\mathcal{Y}) = \{U \neq \emptyset \mid U \subseteq \mathcal{Y}\}$.
On a que:

1. si $\mathcal{B} = \mathcal{Y}$, nous sommes dans le cas d'une variable classique univaluée;
2. Y est *à valeur dans un ensemble* $\mathcal{B}$ si $Y(x_k) \subseteq \mathcal{Y}$, $\forall x_k \in E$, ce qui revient à considérer $\mathcal{B} = \mathcal{P}(\mathcal{Y})$;
3. Y est une variable de type intervalle si $\forall x_k \in E$, $Y(x_k) = [\alpha,\beta]$ est un intervalle de $\mathcal{Y}$.
   Et donc $\mathcal{B}$ est l'ensemble $\mathcal{I}$ des intervalles fermés bornés dans $\mathcal{Y}$;
4. Y est une variable multivaluée (catégorique ou quantitative) si $Y(x_k) \subseteq \mathcal{Y}$ et $|Y(x_k)| < \infty$, $\forall x_k \in E$;
5. Y est une variable modale d'espace d'observation $\mathcal{Y}$ si, pour chaque $x_k \in E$, $Y(x_k) = \pi$ est une distribution de fréquence, de probabilité ou un poids sur $\mathcal{Y}$.

Ceci nous permet de considérer les données symboliques comme une extension des données classiques.
On peut donc, comme dans le cas des données classiques, compiler toutes les données dans une matrice de données symboliques.
La cellule x_{ij} pourra contenir soit un intervalle, soit un ensemble ou encore un histogramme.
Et la k-ième ligne de la matrice sera la description symbolique de l'élément $x_k \in E$.

2.8 Dissimilarités entre objets symboliques.

Dans cette section, nous allons nous intéresser aux dissimilarités entre deux objets symboliques c'est-à-dire des objets décrits par des variables intervalles, multivaluées et modales.

2.8.1 Le cas de variables intervalles.

Considérons une matrice de données symboliques X composée de n objets décrits par p variables intervalles.

$$X = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix}$$

où $x_{ij} = Y_j(x_i) = [\alpha_{ij},\beta_{ij}]$ est la valeur de la variable j pour l'objet x_i.

Pour chacune des p variables, on définit une mesure de dissimilarité sur les $\mathcal{B}_j$:

$$\begin{array}{rccc} \delta_j : & \mathcal{B}_j \times \mathcal{B}_j & \to & \mathbb{R} \\ & (x_{ij},x_{kj}) & \rightsquigarrow & \delta_j(x_{ij},x_{kj}) \end{array}$$

A partir de 2 intervalles $x_{ij} = [\alpha_{ij},\beta_{ij}]$ et $x_{kj} = [\alpha_{kj},\beta_{kj}]$, on peut définir trois types de distances:

1. La distance de Haussdorff:

$$\delta_j(x_{ij},x_{kj}) = \max\{|\alpha_{ij} - \alpha_{kj}|,|\beta_{ij} - \beta_{kj}|\}.$$

2. La distance L_1:

$$\delta_j(x_{ij},x_{kj}) = |\alpha_{ij} - \alpha_{kj}| + |\beta_{ij} - \beta_{kj}|.$$

3. La distance L_2:

$$\delta_j(x_{ij},x_{kj}) = (\alpha_{ij} - \alpha_{kj})^2 + (\beta_{ij} - \beta_{kj})^2.$$

Grâce à ces distances, on peut se ramener à une mesure de dissimilarité sur l'ensemble E:

$$\begin{array}{rccc} d : & E \times E & \to & \mathbb{R}^+ \\ & (x_i,x_k) & \rightsquigarrow & d(x_i,x_k) = \left(\sum_{j=1}^{p} \delta_j^2(x_{ij},x_{kj}) \right)^{\frac{1}{2}} \end{array}$$

où δ_j est une des dissimilarités définies précédemment.

La modélisation milieu-longueur.[1]

Une autre manière de calculer la dissimilarité entre deux intervalles $x_{ij} = [\alpha_{ij},\beta_{ij}]$ et $x_{kj} = [\alpha_{kj},\beta_{kj}]$ est d'utiliser la représentation milieu-longueur des intervalles.

Considérons n individus décrits par p variables de type intervalles.
Le principe de cette modélisation est de transformer chacune des p variables symboliques de type intervalles en deux variables classiques; l'une calculant le centre de l'intervalle et l'autre calculant sa longueur.
Nous obtenons ainsi p nuages de n points dans $\mathbb{R}^2$.

Soient donc deux objets symboliques x et y décrits par p variables de type intervalles.
Notons:

$$\begin{aligned} x_j &= (M_j(x),L_j(x)), \qquad x_j \in \mathbb{R}^2 \quad (j = 1,\ldots,p) \\ y_j &= (M_j(y),L_j(y)), \qquad y_j \in \mathbb{R}^2 \quad (j = 1,\ldots,p) \end{aligned}$$

Alors la dissimilarité entre les deux individus x et y sera:

$$\delta(x,y) = \sum_{j=1}^{p} d(x_j,y_j)$$

où $d(x_j,y_j)$ est la distance euclidienne dans $\mathbb{R}^2$, c'est-à-dire:

$$d(x_j,y_j) = \sqrt{(M_j(x) - M_j(y))^2 + (L_j(x) - L_j(y))^2}$$

2.8.2 Le cas de variables multivaluées.

Considérons $E = \{x_1,\ldots,x_n\}$ un ensemble d'individus décrits par p variables multivaluées catégoriques $Y_1,\ldots,Y_p$ respectivement d'espaces d'observation $\mathcal{Y}_1,\ldots,\mathcal{Y}_p$.

Nous allons transformer la matrice de données originale $X = (Y_j(x_k))_{n\times p}$ en une matrice de fréquences $\tilde{X}$.

Par définition, $Y_j(x_k)$ est l'ensemble des catégories que prend la variable Y_j pour décrire l'objet $x_k \in E$.
Notons

- m_j le nombre de modalités que peut prendre la variable Y_j $(m_j = |\mathcal{Y}_j|)$;
- $c_s (s = 1,\ldots,m_j)$ les différentes modalités de la variable Y_j;
- $q_{j,x_k}(c_s)$ la distribution de Y_j associée à la modalité c_s pour l'individu x_k.

Cette distribution sera donnée par:

$$q_{j,x_k}(c_s) = \begin{cases} \frac{1}{|Y_j(x_k)|} & \text{si } c_s \in Y_j(x_k) \\ 0 & \text{sinon .} \end{cases}$$

Ainsi chaque objet symbolique peut être représenté par un vecteur de dimension $m_1 + \ldots + m_p$, c'est-à-dire:

$$x_k = (\underbrace{(q_{1,x_k}(c_1), \ldots, q_{1,x_k}(c_{m_1}))}_{\text{première variable}}, \ldots, \underbrace{(q_{p,x_k}(c_1), \ldots, q_{p,x_k}(c_{m_p}))}_{p^{ième}\text{variable}})$$

De cette manière, on obtient la matrice des fréquences $\tilde{X}$ de dimension $n \times (m_1 + \ldots + m_p)$:

	Y_1			...	Y_p		
	1	...	m_1	...	1	...	m_p
1	$q_{1,x_1}(c_1)$	...	$q_{1,x_1}(c_{m_1})$	...	$q_{p,x_1}(c_1)$	...	$q_{p,x_1}(c_{m_p})$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
k	$q_{1,x_k}(c_1)$	...	$q_{1,x_k}(c_{m_1})$	...	$q_{p,x_k}(c_1)$	...	$q_{p,x_k}(c_{m_p})$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
n	$q_{1,x_n}(c_1)$	...	$q_{1,x_n}(c_{m_1})$	...	$q_{p,x_n}(c_1)$	...	$q_{p,x_n}(c_{m_p})$

où les $q_{j,x_k}(c_i)$ sont tels que $\sum_{j=1}^{p} \sum_{i=1}^{m_j} q_{j,x_k}(c_i) = p, \quad \forall k = 1, \ldots, n.$

Exemple 2.8.1.
Considérons la matrice de données symboliques suivantes:

x_k	$Y(x_k)$
Namur	{*Match*,*Spar*,*Champion*}
Charleroi	{*Champion*,*Cora*}
Philippeville	{*Champion*,*GB*}

Nous codons alors la matrice de fréquences de cette matrice de données de la manière suivante:

x_k	*Magazins*				
	Match	*Spar*	*Champion*	*Cora*	*GB*
Namur	1/3	1/3	1/3	0	0
Charleroi	0	0	1/2	1/2	0
Philippeville	0	0	1/2	0	1/2

Grâce à cette matrice des fréquences, nous pouvons, pour chaque variable, définir une mesure de dissimilarité sur les $\mathcal{B}_j$.

$$\begin{array}{rccc} \delta_j : & \mathcal{B}_j \times \mathcal{B}_j & \to & \mathbb{R} \\ & (x_{ij}, x_{kj}) & \rightsquigarrow & \delta_j(x_{ij}, x_{kj}) \end{array}$$

Notons $x_{kj}^{(i)}$ la fréquence prise par la variable j pour l'individu x_k concernant la modalité i.
Nous pouvons alors définir trois types de distances :

1. La distance L_1 :
$$\delta_j(x_{kj},x_{\ell j}) = \sum_{i=1}^{|\mathcal{Y}_j|} |x_{kj}^{(i)} - x_{\ell j}^{(i)}|.$$

2. La distance L_2 :
$$\delta_j(x_{kj},x_{\ell j}) = \sum_{i=1}^{|\mathcal{Y}_j|} (x_{kj}^{(i)} - x_{\ell j}^{(i)})^2.$$

3. La distance de De Carvalho :
$$\delta_j(x_{kj},x_{\ell j}) = \sum_{i=1}^{|\mathcal{Y}_j|} (\gamma x_{kj}^{(i)} + \gamma' x_{\ell j}^{(i)}).$$

où
- $\gamma = \begin{cases} 1 & \text{si } x_{kj} \text{ prend la modalité } i \text{ et pas } x_{\ell j} \\ 0 & \text{sinon} \end{cases}$
- $\gamma' = \begin{cases} 1 & \text{si } x_{\ell j} \text{ prend la modalité } i \text{ et pas } x_{kj} \\ 0 & \text{sinon.} \end{cases}$

Nous pouvons, grâce à ces trois distances, nous ramener à une dissimilarité sur l'ensemble E:

$$\begin{array}{rccc} d: & E \times E & \to & \mathbb{R}^+ \\ & (x_i,x_k) & \rightsquigarrow & d(x_i,x_k) = \left(\sum_{j=1}^{p} \delta_j^2(x_{ij},x_{kj}) \right)^{\frac{1}{2}} \end{array}$$

où δ_j est la dissimilarité définie précédemment.

2.8.3 Le cas des variables modales.

Considérons $E = \{x_1, \ldots, x_n\}$ un ensemble de n objets décrits par p variables modales $Y_1, \ldots, Y_p$ respectivement d'espaces d'observation $\mathcal{Y}_1, \ldots, \mathcal{Y}_p$.

Nous avons vu que dans le cas des variables modales, la matrice de données symboliques $X = (Y_j(x_k))_{n\times p} = (x_{kj})_{n\times p}$ était telle que chaque cellule x_{kj} contenait un histogramme.
Comme dans le cas des variables multivaluées, nous allons construire la matrice des fréquences où chaque élément x_k sera représenté par

$$x_k = (\underbrace{(q_{1,x_k}(c_1), \ldots, q_{1,x_k}(c_{m_1}))}_{\text{première variable}}, \ldots, \underbrace{(q_{p,x_k}(c_1), \ldots, q_{p,x_k}(c_{m_p}))}_{p^{ième}\text{variable}})$$

où $q_{j,x_k}(c_s)$ représentera, dans ce cas, la valeur de la distribution (fréquence, probabilité ou poids) associée à la catégorie $c_s (s = 1, \ldots, m_j)$ de Y_j.

Nous pouvons donc, à partir de cette matrice de fréquences, définir les trois mêmes distances que pour les variables multivaluées, ainsi que la mesure de dissimilarité sur E.

Chapitre 3

La classification automatique.

3.1 Introduction.

Dans ce mémoire, le problème qui nous intéresse est de décomposer un ensemble d'individus, appelé population, décrit par des variables en un certain nombres de groupes ayant des caractéristiques communes que nous appelerons des classes.

Intuitivement, une classe est un ensemble composé d'objets qui se ressemblent entre eux et qui sont différents des objets des autres classes.

Considérons un ensemble de n individus $\Omega = \{x_1, \ldots, x_n\}$ décrits par p variables $Y_1, \ldots, Y_p$.

Nous cherchons à déterminer une partition de Ω en k classes où k est un nombre fixé a priori.

Soit donc $\mathcal{P}_k$ l'ensemble de toutes les partitions de Ω en k classes et $P = \{C_1, \ldots, C_k\} \in \mathcal{P}_k$.

Nous définissons alors le critère de classification suivant :

$$\begin{array}{rccc} W: & \mathcal{P}_k & \to & \mathbb{R} \\ & P & \rightsquigarrow & W(P,k) \end{array}$$

La partition optimale $P^* = \{C_1^*, \ldots, C_k^*\}$ de l'ensemble Ω sera telle que :

$$W(P^*,k) = \min_{P \in \mathcal{P}_k} W(P,k) \qquad \text{ou} \qquad W(P^*,k) = \max_{P \in \mathcal{P}_k} W(P,k)$$

Le but de ce chapitre étant de présenter sommairement plusieurs techniques de classification, nous remarquerons que c'est le choix de ce critère qui distingue les différentes méthodes et que, bien souvent, celui-ci est basé sur le calcul de distances ou des dissimilarités entre objets ou groupes d'objets.

3.2 Les méthodes de classification.

De manière générale, il existe deux types de méthode de classification :
- les méthodes hiérarchiques;
- les méthodes non-hiérarchiques (ou de partitionnements).

Les méthodes hiérarchiques peuvent être soit ascendantes (ou agglomératives), soit descendantes et ne nécessitent pas la connaissance a priori du nombre de classes k.

Dans le premier cas, nous commençons avec une partition en n classes formée de singletons et à chaque étape, nous regroupons les deux classes les plus proches au sens du critère choisi jusqu'à obtenir une seule classe.

Dans le second cas, nous commençons avec une partition en une classe et à chaque étape, une nouvelle classe est créée par division. On continue le processus jusqu'à ce que chaque classe soit composée d'un seul objet.

Le principe des méthodes de partitionnement est de partir d'une partition initiale en k classes (où k est connu a priori) et de l'améliorer par itérations successives en essayant de minimiser un critère mathémathique.
Nous présenterons, dans ce chapitre, la méthode des hypervolumes alors que la méthode des nuées dynamiques sera étudiée dans le détail au chapitre 4.

3.3 Les méthodes hiérarchiques agglomératives.

3.3.1 La méthode du lien simple.

La méthode du lien simple est aussi appelée méthode du saut minimum ou du voisin le plus proche.

Dans cette méthode, la distance entre deux classes est définie par la plus petite distance entre deux points de chacune des classes.

On définit donc la distance entre deux classes C_i et C_j :

$$d(C_i,C_j) = \min_{\substack{x\in C_i \\ y\in C_j}} d(x,y)$$

Les avantages de cette méthode sont sa simplicité, son invariance par rapport aux transformations monotones des dissimilarités et son efficacité à retrouver des classes allongées bien séparées grâce à l'effet de chaînage qu'elle induit.
Nonobstant, cet effet de chaînage peut devenir un inconvénient lorsque certains individus forment des ponts entre les classes. Dans ce cas, la méthode ne trouvera pas les classes correctes.

3.3.2 La méthode du lien complet.

La méthode du lien complet est aussi appelée méthode du saut maximum ou du voisin le plus éloigné.

Ici, la distance entre deux classes est définie par la distance maximum entre deux individus de chacune des deux classes.

On définit donc la distance entre deux classes C_i et C_j :

$$d(C_i,C_j) = \max_{\substack{x\in C_i \\ y\in C_j}} d(x,y).$$

Les principaux avantages de cette méthode sont sa simplicité et son invariance par rapport aux transformations monotones des dissimilarités.
Notons également que cette méthode est biaisée par rapport aux classes hypersphériques, c'est-à-dire qu'elle a fortement tendance à retrouver dans les données des classes hypersphériques même si la structure naturelle des données ne s'y prête pas.

3.3.3 La méthode du centroïde.

On définit le centre de gravité de la classe C_ℓ par

$$g^{(\ell)} = (g_1^{(\ell)},g_2^{(\ell)},\ldots,g_p^{(\ell)}) \qquad \text{avec } g_j^{(\ell)} = \frac{1}{n_\ell}\sum_{i=1}^{n_\ell} x_{ij}^{(\ell)}$$

où n_ℓ représente le nombre d'individus dans la classe C_ℓ.

La méthode consiste à regrouper à chaque étape, le deux classes dont les centroïdes sont les plus proches.

Comme pour la méthode du lien complet, cette méthode est biaisée par rapport aux classes hypershériques.

3.3.4 La méthode de Ward.

Cette méthode est l'une des plus utilisées. Elle se base sur le critère des moindres carrés (appelé aussi critère de la variance).

Soit le centroïde d'une classe défini à la section précédente :

$$g^{(\ell)} = (g_1^{(\ell)},g_2^{(\ell)},\ldots,g_p^{(\ell)})$$

On définit alors

- **l'erreur associée à la classe C_ℓ :**

$$e_\ell^2 = \sum_{i=1}^{n_\ell}\sum_{j=1}^{p}(x_{ij}^{(\ell)} - g_j^{(\ell)})^2 = \sum_{i=1}^{n_\ell}\|x_i^{(\ell)} - g^{(\ell)}\|^2$$

où $(x_{ij}^{(\ell)} - g_j^{(\ell)})$ est la déviation de chaque point de la classe C_ℓ par rapport au centroïde de la classe.

- **l'erreur associée à une partition :**

$$E_k^2 = \sum_{\ell=1}^{k} c_\ell^2$$

qui n'est autre que le critère des moindres carrés.

Si nous définissons

$$\Delta E_{rs}^2 = e_t^2 - e_r^2 - e_s^2$$

comme l'accroissement de la valeur de E^2 lorsque nous regroupons les classes C_r et C_s pour former la classe C_t.
Nous décidons de fusionner les classes pour lesquelles la valeur de cet indice est minimale.

Remarquons que le critère des moindres carrés E_k^2 augmente lorsque le nombre de classes diminue et que ce critère est optimisé à chaque étape mais que cela ne garantit pas que la partition finale en k classes soit celle pour laquelle le critère soit minimum.
Notons par ailleurs que le critère de la variance est également biaisé par rapport aux classes hypersphériques.

3.4 La méthode des hypervolumes.

Contrairement aux méthodes présentées précédemment, la méthode des hypervolumes est non-hiérarchique et est basée sur un modèle statistique.

3.4.1 Les processus de Poisson.

Définition. 3.4.1.

*N est un **processus de Poisson** sur $E \subset \mathbb{R}^p$ si et seulement si*

1. *$\forall A_1, \ldots, A_k \subset E$, disjoints 2 à 2, les variables aléatoires $N(A_1), \ldots, N(A_k)$ sont indépendantes avec $N(A_i)$ qui compte le nombre de points dans A_i;*
2. *$\forall A \subset E$ et $\forall k \geq 0$,*

$$P(N(A) = k) = e^{-\mu(A)} \frac{(\mu(A))^k}{k!}$$

où μ est une mesure qui donne une masse finie à tout ensemble borné.

Définition. 3.4.2.

*N est un **processus de Poisson homogène (ou stationnaire) d'intensité** ρ sur $E \subset \mathbb{R}^p$ si et seulement si*

1. *N est un processus de Poisson sur E;*
2. $$\mu(A) = \rho m(A)$$

où

- *$m(A)$ est la mesure de Lebesgue de A,*
- *$\rho \in \mathbb{R}_0^+$ est l'intensité du processus.*

3.4.2 Le modèle statistique.

La méthode des hypervolumes repose sur les hypothèses suivantes :

- les variables alcatoires qui comptent le nombre de points dans des régions disjointes sont indépendantes;
- le nombre moyen de points dans cette région est proportionnel à la mesure de Lebesgue de cette région.

La méthode des hypervolumes suppose donc que les points sont générés par un processus de Poisson homogène dans $D \subset \mathbb{R}^p$ où $D = \dot{\bigcup}_{i=1}^{k} D_i$ où les domaines D_i sont disjoints avec k fixé a priori.

3.4.3 La solution statistique.

Considérons un ensemble de points $\Omega = \{x_1, \ldots, x_n\}$ engendrés par un processus de Poisson homogène dans k domaines disjoints $D_1, \ldots, D_k$.
Notre but, en classification, est d'estimer les frontières de ces domaines.

Soit $x = (x_1, \ldots, x_n)$ le vecteur des réalisations du processus dans $D = \dot{\bigcup}_{i=1}^{k} D_i$.

La fonction de vraisemblance s'écrit alors :

$$L(D,x) = f_D(x) = \left(\frac{1}{m(D)}\right)^n \prod_{i=1}^{n} I_D(x_i)$$

où

- $m(D)$ est la mesure de Lebesgue du domaine D;
- $I_D(x_i)$ est la fonction indicatrice de D.

Nous obtenons donc que le domaine D pour lequel la vraisemblance est maximale est celui, parmi tous ceux qui contiennent tous les points, dont la mesure de Lebesgue est minimale.

Afin d'éviter un certain nombre de solutions triviales à ce problème, il est nécessaire d'ajouter une condition supplémentaire sur la structure de D: la convexité des domaines $D_1,\ldots,D_k$.

Donc le maximum de la fonction de vraisemblance sera atteint par la partition pour laquelle la somme de la mesure de Lebesgue des enveloppes convexes de chacun des k domaines est minimale.

3.4.4 Le critère des hypervolumes.

Formellement, le critère des hypervolumes sera défini de la manière suivante:

$$\begin{array}{rccl} W: & \mathcal{P}_k & \to & \mathbb{R}^+ \\ & P(C_1,\ldots,C_k) & \rightsquigarrow & W(P,k)=\displaystyle\sum_{\ell=1}^{k} m(H(C_\ell)) \end{array}$$

où

- $\mathcal{P}_k$ est l'ensemble des partitions possibles en k classes;
- $m(H(C_\ell))$ est la mesure de Lebesgue de l'enveloppe convexe des points appartenant à la classe C_ℓ.

La partition optimale $P^*=\{C_1^*,\ldots,C_k^*\}\in\mathcal{P}_k$ sera donc telle que

$$W(P^*,k)=\min_{P\in\mathcal{P}_k} W(P,k)=\min_{P\in\mathcal{P}_k}\sum_{\ell=1}^{k} m(H(C_\ell)).$$

Chapitre 4

La méthode des nuées dynamiques.

4.1 Introduction.

Le but de la méthode des nuées dynamiques (introduite par E. Diday en 1972) est de trouver, parmi l'ensemble de toutes les partitions possibles en k classes, celle qui optimise un critère défini a priori.

En théorie, on peut toujours calculer la valeur du critère pour chacune des partitions.
Malgré cela, cette recherche exhaustive de la partition optimale est infaisable en pratique car si nous cherchons la meilleure partition en k classes d'un ensemble E de n individus, il faudrait analyser $\frac{k^n}{k!}$ partitions!

Une des façons de résoudre ce problème est de partir d'une solution initiale acceptable et réalisable que l'on cherche à améliorer par itérations successives.
Cette amélioration est obtenue en changeant la classe d'appartenance de certains individus.

Avant d'introduire la méthode des nuées dynamiques, nous avons besoin d'introduire la notion d'inertie.

4.2 Notion d'inertie.

4.2.1 Inertie par rapport à un point.

Considérons un ensemble de n individus $\Omega = \{x_1, \ldots, x_n\}$ décrits par p variables quantitatives.

La matrice de données s'écrit :

$$\tilde{X} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix}$$

Définition. 4.2.1.
Nous appelons **inertie de Ω par rapport à un point** $a \in \mathbb{R}^p$ *la quantité:*

$$I_a(\Omega) = \sum_{i=1}^{n} d^2(x_i,a)$$

où d est la distance euclidienne.

4.2.2 Théorème de Huygens.

Huygens a établi que le centre de gravité g est le point par rapport auquel l'inertie d'un nuage de points est minimale.

En effet, on a la relation suivante:

$$\forall a \in \mathbb{R}^p, \qquad I_a(\Omega) = I_g(\Omega) + \ n\ d^2(g,a).$$

L'inertie $I_g(\Omega)$, notée T, est définie comme l'inertie totale du nuage de points Ω.

4.2.3 Décomposition de l'inertie.

Inertie associée à une partition.

Considérons $P = \{C_1, \dots, C_k\}$ une partition de Ω.
Le centre de gravité de la classe C_ℓ est donné par:

$$g^{(\ell)} = \frac{1}{n_\ell} \sum_{x_i \in C_\ell} x_i$$

où n_ℓ est le nombre de points dans la classe C_ℓ.

Nous associons trois types d'inertie à cette partition P:
- **l'inertie totale** T:

$$T \equiv I_g(\Omega) = \sum_{i=1}^{n} d^2(x_i,g).$$

Il s'agit de l'inertie par rapport au centre de gravité g du nuage de points Ω.

- **l'inertie intra-classes** W:

$$W = \sum_{\ell=1}^{k} \sum_{x_i \in C_\ell} d^2(x_i, g^{(\ell)}).$$

Cela représente la somme sur chaque classe des inerties de la classe C_ℓ à son centre de gravité $g^{(\ell)}$.

- **l'inertie inter-classes** B:

$$B = \sum_{\ell=1}^{k} n_\ell \, d^2(g^{(\ell)}, g).$$

Celle-ci représente l'inertie du nuage de points formé des k centres de gravité $g^{(\ell)}$ de chacune des classes par rapport au centre de gravité global g.

Les trois inerties définies ci-dessus sont liées par la relation suivante:

$$T = B + W.$$

Etant donné que l'inertie totale T est indépendante de la partition, nous pouvons donc affirmer que plus l'inertie inter-classes B est grande, plus l'inertie intra-classes W est petite.

Décompostion de l'inertie sur les classes et les variables.

Nous allons à présent, décomposer ces trois inerties en fonction des classes et des variables.

L'inertie totale T peut se décomposer comme suit:

$$T = \sum_{\ell=1}^{k} \sum_{j=1}^{p} T_j^{(\ell)} \qquad \text{où} \quad T_j^{(\ell)} = \sum_{x_i \in C_\ell} (x_{ij} - g_j)^2$$

où $T_j^{(\ell)}$ représente l'écart, pour la variable j, des points de la classe C_ℓ au centre de gravité global g_j.

L'inertie intra-classes W peut être définie par:

$$W = \sum_{\ell=1}^{k} \sum_{j=1}^{p} W_j^{(\ell)} \qquad \text{où} \quad W_j^{(\ell)} = \sum_{x_i \in C_\ell} (x_{ij} - g_j^{(\ell)})^2.$$

$W_j^{(\ell)}$ est l'inertie de la classe C_ℓ, pour la variable j, par rapport au centre gravité local $g_j^{(\ell)}$.

L'inertie inter-classes B peut s'écrire:

$$B = \sum_{\ell=1}^{k} \sum_{j=1}^{p} B_j^{(\ell)} \qquad \text{où} \quad B_j^{(\ell)} = (g_j^{(\ell)} - g_j)^2.$$

$B_j^{(\ell)}$ est l'écart, pour la variable j, du centre de gravité local $g_j^{(\ell)}$ de la classe C_ℓ au centre de gravité global g_j.

Donc, si nous notons $\forall j = 1, \ldots, p$:

$$T_j = \sum_{\ell=1}^{k} T_j^{(\ell)} \qquad W_j = \sum_{\ell=1}^{k} W_j^{(\ell)} \qquad B_j = \sum_{\ell=1}^{k} B_j^{(\ell)}$$

nous obtenons:

$$T = \sum_{j=1}^{p} T_j \qquad W = \sum_{j=1}^{p} W_j \qquad B = \sum_{j=1}^{p} B_j$$

et alors, T_j,W_j et B_j expriment la contribution de la variable j pour chacune des différentes inerties.

De manière analogue, si nous notons $\forall \ell = 1, \ldots, k$:

$$T^{(\ell)} = \sum_{j=1}^{p} T_j^{(\ell)} \qquad W^{(\ell)} = \sum_{j=1}^{p} W_j^{(\ell)} \qquad B^{(\ell)} = \sum_{j=1}^{p} B_j^{(\ell)}$$

nous obtenons:

$$T = \sum_{\ell=1}^{k} T^{(\ell)} \qquad W = \sum_{\ell=1}^{k} W^{(\ell)} \qquad B = \sum_{\ell=1}^{k} B^{(\ell)}$$

et alors, $T^{(\ell)}$,$W^{(\ell)}$ et $B^{(\ell)}$ expriment la contribution de la classe ℓ pour chacune des différentes inerties.

4.3 Méthode des nuées dynamiques dans la cas de données classiques[18].

L'algorithme des nuées dynamiques se décompose en deux étapes. La première étape, appelée étape de représentation, consiste à trouver un représentant (ou prototype) pour chacune des k classes.
Ce représentant, ou **noyau** de la classe, peut être :

- un ensemble de points,
- une droite ou encore,
- un centre de gravité.

La seconde étape, appelée étape d'affectation, modifie la classe d'appartenance de chaque individu de E.

L'algorithme procède de la manière suivante :

- l'algorithme commence par choisir k noyaux estimés ou tirés au hasard parmi une famille de noyaux admissibles, appelée espace de représentation et notée $\mathcal{L}$;
- ensuite, les points sont affectés au noyau le plus proche.
  Une nouvelle partition en k classes est ainsi formée et les nouveaux noyaux associés à cette partition sont calculés;
- le procédé est alors recommencé avec les nouveaux noyaux.

En posant certaines conditions sur les fonctions qui permettent d'affecter les points aux noyaux et de calculer les nouveaux noyaux, on peut s'assurer que l'algorithme fait décroître un critère W qui mesure l'adéquation entre les classes et leur noyau respectif.

Le critère W sera défini de la manière suivante :

$$\begin{aligned} W \quad : \quad & \mathcal{P}_k \times \mathcal{L}_k \quad \to \mathbb{R}^+ \\ & (P,L) \quad \rightsquigarrow W(P,L) = \sum_{\ell=1}^{k} D(L_\ell,C_\ell) \end{aligned}$$

où

- $\mathcal{L}_k$ est l'ensemble des k-uples de noyaux $L = (L_1,\ldots,L_k)$ avec
  - $\mathcal{L}_k = \Omega^k$ si $L^\ell \in \Omega$ c'est-à-dire si chaque noyau est un point de Ω,
  - $\mathcal{L}_k = \mathbb{R}^{p\times k} = \underbrace{\mathbb{R}^p \times \cdots \times \mathbb{R}^p}_{k \text{ fois}}$ si $\mathcal{L}_\ell$ est le centre de gravité de la classe C_ℓ.
- $\mathcal{P}_k$ est l'ensemble des partitions $P = (C_1,\ldots,C_k)$ en k classes de Ω.
- D est une mesure d'adéquation entre classe et noyau. Autrement dit, une petite valeur de $D(L_\ell,C_\ell)$ exprime une bonne "ressemblance" entre le noyau L_ℓ et sa classe d'appartenance C_ℓ.

4.3.1 Aspect formel.

Le but de la méthode des nuées dynamiques est de trouver un couple $(P^*,L^*) \in \mathcal{P}_k \times \mathcal{L}_k$ tel que

$$W(P^*,L^*) = \min_{\substack{P\in\mathcal{P}_k \\ L\in\mathcal{L}_k}} W(P,L)$$

où

- $\mathcal{P}_k$ est l'ensemble des partitions de Ω en k classes,
- $\mathcal{L}_k$ est un espace de représentation.

Comme déjà dit précédemment, ce critère est minimisé en utilisant successivement une étape de représentation et une étape d'affectation jusqu'à obtenir la convergence.

La méthode des nuées dynamiques consiste à :

1. Choisir un espace de représentation $\mathcal{L}_k$;
2. Définir un critère $W : \mathcal{P}_k \times \mathcal{L}_k \to \mathbb{R}$ qui permet de mesurer l'adéquation entre toute partition $P \in \mathcal{P}_k$ et toute représentation $L \in \mathcal{L}_k$ de cette partition.
3. Chercher simultanément une partition $P \in \mathcal{P}_k$ et sa représentation $L \in \mathcal{L}_k$ de sorte que P et L aient la meilleure adéquation au sens du critère W.

L'algorithme va utiliser itérativement

- une fonction de représentation $g : \mathcal{P}_k \to \mathcal{L}_k$;
- une fonction d'affectation $f : \mathcal{L}_k \to \mathcal{P}_k$

jusqu'à la convergence du critère.

La fonction g va permettre de calculer, étant donné une partition $P \in \mathcal{P}_k$, le noyau de chaque classe.
Alors que la fonction f, étant donné les noyaux $L = (L_1,\ldots,L_k)$, va affecter chaque individu au noyau dont il est le plus proche.

Il faut bien entendu un point de départ, c'est pourquoi on procède en premier lieu à une étape d'**initialisation** qui

- choisit une partition initiale $P^0 \in \mathcal{P}_k$ aléatoirement ou bien
- choisit une représentation initiale $L^0 \in \mathcal{L}_k$, estimée ou choisie au hasard.

Si on part d'une représentation initiale L^0, on obtiendra successivement

- $f(L^0) = P^0 \qquad g(P^0) = L^1$;
- $f(L^1) = P^1 \qquad g(P^1) = L^2$;
- $f(L^2) = P^2 \qquad g(P^2) = L^3$;
- $\vdots \qquad\qquad \vdots$

On construit donc une suite $u_n = W(v_n)$ avec $v_n = (P^n, L^n)$ où :
- $P^n \in \mathcal{P}_k$ est la partition en k classes obtenue à l'itération n,
- $L^n \in \mathcal{L}_k$ est la représentation de la partition P^n obtenue à l'aide de la fonction g.

Si les fonctions de représentation et d'affectation sont bien choisies, alors:
- les suites u_n et v_n sont convergentes;
- la suite u_n est décroissante.

Ce qui nous permet de conclure que l'algorithme va faire décroître le critère W à chaque itération jusqu'à ce qu'il se stabilise.

4.3.2 Le cas du centre de gravité.

Dans cette section, nous allons détailler l'algorithme des nuées dynamiques dans le cas où le représentant de chaque classe est défini comme étant son centre de gravité.

L'espace de représentation.

Comme nous l'avons déjà fait remarquer plus haut, l'espace de représentation d'une classe est $\mathbb{R}^p$ lorsque les représentants sont des centres de gravité.

La mesure d'adéquation dans ce cas-ci est l'inertie de la partition considérée par rapport à un point $x \in \mathbb{R}^p$:

$$\begin{array}{rccl} D & : & \mathcal{P}_k \times \mathcal{L}_k & \to \mathbb{R}^+ \\ & & (A,x) & \rightsquigarrow D(A,x) = \sum_{a \in A} d^2(a,x) = I_x(A) \end{array}$$

où d est une distance euclidienne.

La fonction de représentation.

Grâce au théorème de Huygens, nous savons que le point x qui minimise l'inertie d'une classe A par rapport à x est son centre de gravité.

On définit donc la fonction de représentation :

$$\begin{array}{rccl} g: & & \mathcal{P}_k & \to \mathcal{L}_k \\ & & (C_1,\ldots,C_k) & \rightsquigarrow (g^{(1)},\ldots,g^{(k)}) \end{array}$$

où $g^{(\ell)}$ est le centre de gravité de la classe C_ℓ.

La fonction d'affectation.

La fonction d'affectation f est définie :

$$\begin{aligned} f : \quad \mathcal{L}_k \quad &\to \mathcal{P}_k \\ (g^{(1)},\ldots,g^{(k)}) \quad &\rightsquigarrow (C_1,\ldots,C_k) \end{aligned}$$

avec, $\forall \ell = 1,\ldots,k$

$C_\ell = \{x \in \mathbb{R}^p \ : \ d(x,g^{(\ell)}) \leq d(x,g^{(m)}), \forall m \in \{1,\ldots,k\}$ et $\ell < m$ en cas d'égalité$\}$.

La classe C_ℓ est donc constituée des individus les plus proches, au sens de la métrique choisie, de son centre de gravité.

Le problème d'optimisation.

Comme nous le savons, l'objectif est de trouver le couple optimal $(P^*,L^*) \in \mathcal{P}_k \times \mathcal{L}_k$ qui minimise le critère W qui mesure l'adéquation entre chaque classe et son représentant.
On définit ce critère:

$$W(P,L) = \sum_{\ell=1}^{k} D(C_\ell,g^{(\ell)}) = \sum_{\ell=1}^{k} \sum_{x_i \in C_\ell} d^2(x_i,g^{(\ell)})$$

où $g^{(\ell)}$ est le centre de gravité de la classe ℓ.

Si nous notons W_ℓ l'inertie de la classe C_ℓ par rapport à son centre de gravité, On peut alors écrire le critère :

$$W(P,L) = \sum_{\ell=1}^{k} W_\ell = W$$

On voit donc que le critère que la méthode des nuées dynamiques cherche à minimiser, dans le cas où les noyaux sont les centres de gravité, n'est autre que l'inertie intra-classes W.

Grâce à la relation annoncée précédemment, $T = W + B$, nous pouvons affirmer que cela revient à maximiser l'inertie inter-classes B.

Convergence du critère.

Si nous considérons :
- les fonctions de représentation et d'affectation f et g définies ci-dessus et
- les suites $v_n = (P^n,L^n)$ et $u_n = W(v_n)$ engendrées par ces deux fonctions.

Nous obtenons les deux propriétés suivantes:

- La suite v_n est stationnaire.

- La suite u_n est décroissante et convergente.

4.4 La méthode des nuées dynamiques dans le cas de données symboliques.

4.4.1 Le cas de variables intervalles.

Considérons un ensemble $E = \{x_1, \ldots, x_n\}$ de n objets symboliques décrits par p variables de type intervalle.
Chaque objet symbolique x_i pourra être représenté par un hyperrectangle dans un espace euclidien à p dimensions:

$$x_i = ([\alpha_{i1}, \beta_{i1}], \ldots, [\alpha_{ip}, \beta_{ip}]) .$$

On appelle le centre de gravité de n objets symboliques décrits par p variables de type intervalle, un **hyperrectangle de gravité**. Il est défini comme suit:

$$\left(\left[\frac{1}{n} \sum_{i=1}^{n} \alpha_{i1} , \frac{1}{n} \sum_{i=1}^{n} \beta_{i1} \right], \ldots, \left[\frac{1}{n} \sum_{i=1}^{n} \alpha_{ip} , \frac{1}{n} \sum_{i=p}^{n} \beta_{ip} \right] \right) .$$

Espace de représentation.

L'espace $\mathcal{I}^p$ des intervalles fermés bornés correspond à la fois à l'espace des objets et à l'espace de représentation $\mathcal{L}$.

La mesure d'adéquation sera définie:

$$\begin{array}{rccl} D & : & \mathcal{P} \times \mathcal{L} & \to \mathbb{R}^+ \\ & & (A, x_i) & \rightsquigarrow D(A, x_i) = \sum_{a \in A} d^2(a, x_i) \end{array}$$

où d est une des mesures de dissimilarités pour des variables intervalles définies au chapitre 2.

La fonction de représentation.

La fonction de représentation g est définie par:

$$\begin{array}{rccl} g & : & \mathcal{P}_k & \to \mathcal{L}_k \\ & & (C_1, \ldots, C_k) & \rightsquigarrow (g^{(1)}, \ldots, g^{(k)}) \end{array}$$

avec $g^{(\ell)}$ l'hyperrectangle de gravité de la classe C_ℓ:

$$g^{(\ell)} = \left(\left[\frac{1}{n_\ell} \sum_{x_i \in C_\ell} \alpha_{i1} , \frac{1}{n_\ell} \sum_{x_i \in C_\ell} \beta_{i1} \right], \ldots, \left[\frac{1}{n_\ell} \sum_{x_i \in C_\ell} \alpha_{ip} , \frac{1}{n_\ell} \sum_{x_i \in C_\ell} \beta_{ip} \right] \right)$$

$g^{(\ell)}$ sera donc le prototype de la classe C_ℓ.

La fonction d'affectation.

La fonction d'affectation f est définie par:

$$\begin{array}{rccc} f & : & \mathcal{L}_k & \to \mathcal{P}_k \\ & & (g^{(1)},\ldots,g^{(k)}) & \rightsquigarrow (C_1,\ldots,C_k) \end{array}$$

avec, $\forall \ell = 1,\ldots,k$

$$C_\ell = \{x \in \mathbb{R}^p \ : \ d(x,g^{(\ell)}) \leq d(x,g^{(m)}),\ \forall m \in \{1,\ldots,k\} \text{ et } \ell < m \text{ en cas d'égalité}\}.$$

C'est-à-dire que la classe C_ℓ est composée des éléments les plus proches, au sens de la métrique choisie, de son hyperrectangle de gravité $g^{(\ell)}$.

Le problème d'optimisation.

Nous devons trouver le couple optimal $(P^*,L^*) \in \mathcal{P}_k \times \mathcal{L}_k$ qui minimise le critère W mesurant l'adéquation entre chaque classe et son prototype.
Le critère est défini:

$$W(P,L) = \sum_{\ell=1}^{k} D(C_\ell,g^{(\ell)}) = \sum_{\ell=1}^{k} \sum_{x_i \in C_\ell} d^2(x_i,g^{(\ell)})$$

où $g^{(\ell)}$ est l'hyperrectangle de gravité de la classe l.

Comme pour les données classiques, si nous notons W_ℓ l'inertie de la classe C_ℓ par rapport à son prototype $g^{(\ell)}$, on peut alors écrire le critère:

$$W(P,L) = \sum_{\ell=1}^{k} W_\ell = W$$

De nouveau, le critère que la méthode des nuées dynamiques cherche à minimiser (respectivement maximiser) est l'inertie intra-classes W (respectivement inertie inter-classes B).

4.4.2 Le cas des variables multivaluées et modales.

Les variables multivaluées et modales étant fort semblables, nous allons traiter ces deux cas simultanément.

Considérons un ensemble $E = \{x_1,\ldots,x_n\}$ de n objets symboliques décrits par p variables multivaluées ou modales $Y_1,\ldots,Y_p$.
Chaque objet symbolique x_i pourra être représenté par un vecteur à $m_1+m_2+\ldots+m_p$ dimensions où m_j est le nombre de modalités que peut prendre la variable Y_j (voir chapitre 2):

$$x_i = ((q_{1,x_k}(c_1),\ldots,q_{1,x_k}(c_{m_1})),\ldots,(q_{p,x_k}(c_1),\ldots,q_{p,x_k}(c_{m_p}))).$$

Dans le cas de variables multivaluées comme dans le cas de variables modales, le prototype $g^{(\ell)}$ d'une classe C_ℓ d'objets sera représenté comme un objet symbolique de type modal contenant la distribution de fréquence ou de probabilité de chacune des catégories pour chacune des p variables :

$$\left(\left(\frac{1}{n_\ell}\sum_{x_i\in C_\ell} q_{1,x_i}(c_1),\ldots,\frac{1}{n_\ell}\sum_{x_i\in C_\ell} q_{1,x_i}(c_{m_1})\right),\ldots,\left(\frac{1}{n_\ell}\sum_{x_i\in C_\ell} q_{p,x_i}(c_1),\ldots,\frac{1}{n_\ell}\sum_{x_i\in C_\ell} q_{p,x_i}(c_{m_p})\right)\right).$$

La mesure d'adéquation entre les classes et leurs noyaux est définie par :

$$\begin{array}{rccl} D & : & \mathcal{P}\times\mathcal{L} & \to \mathbb{R}^+ \\ & & (A,x_i) & \rightsquigarrow D(A,x_i)=\sum_{a\in A} d^2(a,x_i) \end{array}$$

où d est une des mesures de dissimilarités pour des variables multivaluées et modales définies au chapitre 2.

La fonction de représentation.

La fonction de représentation g est définie par :

$$\begin{array}{rccl} g & : & \mathcal{P}_k & \to \mathcal{L}_k \\ & & (C_1,\ldots,C_k) & \rightsquigarrow (g^{(1)},\ldots,g^{(k)}) \end{array}$$

où $g^{(\ell)}$ est le prototype de la classe C_ℓ défini ci-dessus.

La fonction d'affectation.

La fonction d'affectation f est définie par :

$$\begin{array}{rccl} f & : & \mathcal{L}_k & \to \mathcal{P}_k \\ & & (g^{(1)},\ldots,g^{(k)}) & \rightsquigarrow (C_1,\ldots,C_k) \end{array}$$

avec, $\forall \ell = 1,\ldots,k$

$$C_\ell = \{x\in\mathbb{R}^p \ : \ d(x,g^{(\ell)}) \leq d(x,g^{(m)}),\ \forall m\in\{1,\ldots,k\} \text{ et } \ell < m \text{ en cas d'égalité}\}.$$

C'est-à-dire que la classe C_ℓ est composée des éléments les plus proches, au sens de la métrique choisie, de son prototype $g^{(\ell)}$.

La problème d'optimisation.

Comme dans les cas précédents, on cherche un couple $(P^*,L^*) \in \mathcal{P}_k \times \mathcal{L}_k$ qui minimise un critère W qui peut se décomposer comme suit :

$$W(P,L) = \sum_{\ell=1}^{k} W_\ell = \sum_{\ell=1}^{k} D(C_\ell,g^{(\ell)}) = \sum_{\ell=1}^{k} \sum_{x_i \in C_\ell} d^2(x_i,g^{(\ell)})$$

où

- $g^{(\ell)}$ est le prototype de la classe C_ℓ.
- W_ℓ est l'inertie de la classe C_ℓ.

Le critère W que l'on cherche à minimiser est donc l'inertie intra-classes, ce qui revient à maximiser l'inertie inter-classes B grâce à la relation :

$$T = W + B.$$

Chapitre 5

Les méthodes de détermination du nombre de classes.

5.1 Introduction.

La plupart des méthodes de classification automatique, dont la méthode des nuées dynamiques (exposée au chapitre précedent), qui ont pour but de partitionner un ensemble de données Ω en k classes supposent le nombre k fixé **a priori**.

Le but de ce chapitre sera donc de mettre en place des méthodes pour détecter le nombre de classes réellement présentes dans les données.

Dans la littérature, de nombreux critères ont été proposés c'est pourquoi, en 1988, Milligan et Cooper[14] en ont testé une trentaine et ont établi un classement.
Nous allons, dans la suite du chapitre, présenter les cinq meilleures méthodes $(M_1,\ldots,M_5)$ issues de ce classement.

Remarquons déjà que les deux méthodes basées sur des tests d'hypothèses ($M2$ et $M5$) ne seront pas applicables à la méthode des nuées dynamiques car elles nécessitent une hiérarchie de partitions.

5.2 Les méthodes de Milligan et Cooper.

5.2.1 La méthode de Calinski et Harabasz ($M1$). [9]

Considérons un ensemble $\Omega = \{x_1,\ldots,x_n\}$ de n individus décrits par p variables $Y_1,\ldots,Y_p$.

Soit également la matrice de données classiques :

$$\tilde{X} = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1p} \\ x_{21} & x_{22} & \ldots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{np} \end{pmatrix} = \begin{pmatrix} x_1' \\ x_2' \\ \vdots \\ x_n' \end{pmatrix}$$

Nous définissons alors :

- la matrice de dispersion totale M_T :

$$M_T = \sum_{i=1}^{n} (x_i - g)(x_i - g)'$$

- la matrice de dispersion intra-classes M_W:

$$M_W = \sum_{\ell=1}^{k} M_W^{(\ell)} = \sum_{\ell=1}^{k} \sum_{i=1}^{n_\ell} (x_i^{(\ell)} - g^{(\ell)})(x_i^{(\ell)} - g^{(\ell)})'$$

- la matrice de dispersion inter-classes M_B :

$$M_B = \sum_{\ell=1}^{k} \sum_{i=1}^{n_\ell} (g^{(\ell)} - g)(g^{(\ell)} - g)'$$

où

- g est le centre de gravité du nuage de points Ω;
- x_i est le vecteur colonne représentant le $i^{ième}$ individu;
- k est le nombre de classes;
- $x_i^{(\ell)}$ est le vecteur colonne représentant le $i^{ième}$ individu de la classe ℓ;
- n_ℓ est le nombre d'individus dans le classe ℓ;
- $g^{(\ell)} = \frac{1}{n_\ell} \sum_{i=1}^{n_\ell} x_i^{(\ell)}$ est le centre de gravité de la classe ℓ.

On a les relations suivantes :

- $M_T = M_W + M_B$;
- $tr(M_T) = T$ avec T l'inertie totale;
- $tr(M_W) = W$ avec W l'inertie intra-classes;
- $tr(M_B) = B$ avec B l'inertie inter-classes;

où tr représente la trace de la matrice.

L'indice de Calinski et Harabasz est défini par :

$$CH = \frac{tr(M_B)/(k-1)}{tr(M_W)/(n-k)} = \frac{B/(k-1)}{W/(n-k)}.$$

Puisque nous cherchons l'inertie intra-classes qui soit la plus petite possible et l'inertie inter-classes qui soit la plus grande possible, nous déterminerons le nombre k par un maximum relatif ou absolu de l'indice ou par un écart important entre deux de ses valeurs successives.

5.2.2 La méthode de Duda et Hart (M_2). [11]

Considérons un ensemble $\Omega = \{x_1, \ldots, x_n\}$ de n individus décrits par p variables $Y_1, \ldots, Y_p$.

Définissons

$$J_e(k) = \sum_{\ell=1}^{k} \sum_{i=1}^{n_\ell} \sum_{j=1}^{p} (x_{ij}^{(\ell)} - g_j^{(\ell)})^2$$

qui vaut la somme des carrés des erreurs qui surviennent si les n individus sont représentés par le centre de gravité de leur classe respective.

La méthode de Duda et Hart pour la détermination du nombre de classes est basée sur le test d'hypothèses suivant:

$$\begin{cases} H_0 & : \text{Les points sont issus d'une seule population Normale de moyenne } \mu \text{ et de matrice de variance-covariance } \sigma^2 I. \\ H_1 & : \text{Les points sont issus de deux populations Normales.} \end{cases}$$

Règle de décision.

Nous rejetons H_0 au niveau de signification α si

$$\frac{J_e(2)}{J_e(1)} < 1 - \frac{2}{\pi p} - z_{1-\alpha} \sqrt{\frac{2(1 - \frac{8}{\pi^2 p})}{np}}$$

où

- n est le nombre d'individus;
- p est le nombre de variables;
- $z_{1-\alpha}$ est le quantile d'ordre $1 - \alpha$ de la loi Normale.

Cette méthode ne peut s'appliquer qu'à des méthodes de classification hiérarchiques: à chaque niveau de la hiérarchie on vérifie si la valeur de l'expression

$$DH = \frac{-\frac{J_e(2)}{J_e(1)} + 1 - \frac{2}{\pi p}}{\sqrt{\frac{2(1 - \frac{8}{\pi^2 p})}{np}}}$$

est plus grande (rejet de H_0) ou plus petite (non-rejet de H_0) que la valeur choisie pour $z_{1-\alpha}$.
Si nous rejetons H_0 (c'est-à-dire nous refusons une fusion entre deux classes) pour une valeur $k = k_0$, nous devons conclure que k_0+1 classes sont présentes dans les données.

Plusieurs valeurs de $z_{1-\alpha}$ ont été proposées. Milligan et Cooper ont suggéré $z_{1-\alpha} = 3.2$ alors que pour Gordon[13] un choix $z_{1-\alpha} = 4$ donne des meilleurs résultats.

5.2.3 La méthode du C-index (M_3). [19]

Considérons un ensemble $\Omega = \{x_1, \ldots, x_n\}$ de n individus décrits par p variables $Y_1, \ldots, Y_p$.

Définissons :

$$C(x_i,x_j) = \begin{cases} 1 & \text{si } x_i \text{ et } x_j \text{ sont dans la même classe } (i \neq j); \\ 0 & \text{sinon.} \end{cases}$$

et

$$\begin{aligned} \Gamma &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}C(x_i,x_j) \\ &= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}C(x_i,x_j) \\ &= \sum_{\ell=1}^{k}\sum_{\substack{i,j=1 \\ i<j}}^{n_\ell} d(x_i^{(\ell)},x_j^{(\ell)}) \end{aligned}$$

où

- n_ℓ est le nombre d'individus dans la classe ℓ;
- k est le nombre de classes;
- d est une distance euclidienne;
- $d_{ij} = d(x_i,x_j)$.

La normalisation de Γ (proposée par Dalrymphe-Arford[21]) nous donne l'indice :

$$\text{index } C = \frac{\Gamma - \min\Gamma}{\max\Gamma - \min\Gamma}.$$

Si

- t est le nombre total de dissimilarités différentes entre paires d'individus sur l'ensemble de points Ω ($t = \frac{n\ n-1}{2}$) .

- r est la somme sur chacune des k classes des dissimilarités entre paires d'individus de la classe.

Alors,

- $\min\Gamma$ est défini comme la somme des r plus petites dissimilarités entre paires d'individus (parmi les t dissimilarités possibles);
- $\max\Gamma$ est défini comme la somme des r plus grandes dissimilarités entre paires d'individus (parmi les t dissimilarités possibles).

Le nombre de classes est indiqué par la valeur minimale de l'indice.

5.2.4 La méthode Gamma ($M4$). [20]

Considérons un ensemble $\Gamma = \{x_1, \ldots, x_n\}$ composé de n individus et d la distance euclidienne.

Définissons :

$$T_\ell(x_i,x_j) = \begin{cases} 0 & \text{si } x_i \text{ et } x_j \text{ sont dans la même classe } (i \neq j); \\ 1 & \text{sinon.} \end{cases}$$

Si $T_\ell(x_i,x_j) = 0$ c'est-à-dire si x_i et x_j appartiennent à la même classe, nous posons :

$$n_\ell(x_i,x_j) = \#\{\{x_r,x_s\} \ : \ T_\ell(x_r,x_s) = 1 \text{ et } d(x_r,x_s) < d(x_i,x_j)\}$$

qui est le nombre de couples d'objets qui n'appartiennent pas à la même classe mais qui sont plus proches que x_i n'est proche de x_j.

Nous définissons l'indice α_ℓ :

$$\alpha_\ell = \frac{\sum\limits_{i<j} n_\ell(x_i,x_j)}{\max \sum\limits_{i<j} n_\ell(x_i,x_j)}$$

où

- le maximum est pris sur toutes les partitions possibles en k classes en gardant le même nombre d'objets par classe;
- les sommes sont effectuées sur les paires d'objets $\{x_i,x_j\}$ qui appartiennent à la même classe.

Le numérateur de cet indice représente la somme sur toutes les paires d'objets appartenant à la même classe, du nombre d'objets appartenant à des classes différentes et étant plus proche que les paires d'objets considérés sur la somme.
Donc l'indice α_ℓ représente la proportion de paires d'objets placés dans de mauvais groupes dans le sens où deux objets d'une même classe sont censés être plus proches que deux objets de classe différente.

On définit alors l'indice γ:

$$\gamma = 1 - 2\alpha_\ell.$$

On note alors que

- si aucun objet n'est "mal placé", on a $\alpha_\ell = 0$ et donc $\gamma = 1$;
- si un maximum d'objets est mal placé, on a $\alpha_\ell = 1$ et donc $\gamma = -1$.

On obtient alors que $-1 \leq \gamma \leq 1$.

En pratique, nous chercherons la valeur maximale de l'indice, c'est-à-dire la valeur la plus proche de 1.
Si $\gamma = 1$, on parle de partition "parfaite".

5.2.5 La méthode de Beale ($M5$). [10]

La méthode de Beale procède de la même manière que la méthode de Duda et Hart c'est-à-dire qu'à chaque niveau de la hiérarchie, on essaie d'établir, à l'aide d'un test d'hypothèse, si la fusion entre deux groupes est justifiée ou non.

$$\begin{cases} H_0 & : \text{ il y a un groupe dans les données;} \\ H_1 & : \text{ il y a deux groupes dans les données.} \end{cases}$$

Soient

$$W_1 = \sum_{i=1}^{n}\sum_{j=1}^{p}(x_{ij} - g_j)^2 = I(C),$$

l'inertie totale du nuage de points, et

$$W_2 = \sum_{\ell=1}^{2}\sum_{i=1}^{n}\sum_{j=1}^{p}(x_{ij}^{(\ell)} - g_j^{(\ell)})^2 = I(C_1) + I(C_2),$$

qui est la somme des inerties de chacune des classes.

. La statistique du test est donnée par :

$$W = \frac{\frac{W_1 - W_2}{W_2}}{\left(\frac{n-1}{n-2}\right)2^{\frac{2}{p}} - 1} \sim F(p,(n-2)p)$$

où F représente la loi de Fisher-Snedecor à p degrés de liberté au numérateur et $n - 2p$ degrés de liberté au dénominateur.

Si k_0 est la première valeur qui conduit au rejet de la fusion de deux groupes (H_0) c'est-à-dire qu'on a $W > F_{p,(n-2)p,1-\alpha}$, on conclut que le nombre de classes présentes dans les données est $k_0 + 1$.

Généralisation.

Ce test peut être généralisé :

$$\begin{cases} H_0 & : \text{ il y a } k_1 \text{ groupes dans les données;} \\ H_1 & : \text{ il y a } k_2 \text{ groupes dans les données.} \end{cases}$$

On obtient alors :

$$W = \frac{\frac{W(k_1) - W(k_2)}{W(k_2)}}{\left(\frac{n-k_1}{n-k_2}\right)\left(\frac{k_2}{k_1}\right)^{\frac{2}{p}} - 1} \sim F(p(k_2 - k_1),(n-k_2)p)$$

Chapitre 6

Description du programme Sclust.

6.1 Introduction.

Sclust est une méthode de classification d'objets symboliques en cours de développement réalisée de le cadre du contrat européen ASSO[1].
Ce programme a été conçu et codé en C++ par Yves Lechevallier de l'INRIA Rocquencourt.

Le programme Sclust étant basé sur la méthode des nuées dynamiques dans le cas de données classiques, il détermine de façon itérative une série de partitions de l'ensemble des objets en améliorant un critère de classification.
A chaque étape, il calcule les prototypes qui représentent chacune des classes et évalue le critère en fonction d'une mesure de proximité qui détermine la ressemblance entre les objets d'une classe et leur représentant.

Le programme Sclust traite aussi bien les données symboliques à savoir :
- les variables intervalles,
- les variables multivaluées et
- les variables modales;

que les variables quantitatives et qualitatives classiques.

Nous allons à présent décrire plus en détail les parties de ce programme ainsi que le fichier sodas qui est le fichier d'entrée.

1. Analysis System of Symbolic Official data

6.2 Le fichier sodas.

Ce fichier contient tous les renseignements sur le jeu de données que le programme Sclust a besoin de connaître pour effectuer la classification.
Ce fichier est généré à partir d'une base de données présentée sous forme d'un fichier Excel ou Access à l'aide du module DB2SO du logiciel SODAS.

Chaque fichier sodas contient au moins les six blocs suivants :

- `CONTAINS` reprend la liste de tous les blocs contenus dans le fichier sodas;
- `HEADER` répertorie
  - le titre et le sous-titre du jeu de données ,
  - le nombre total de variables et d'individus dans le jeu de données,
  - le nombre de variables de chaque type,
  - le nombre de données manquantes;
- `INDIVIDUALS` qui affecte à chaque individu un numéro et un libellé;
- `VARIABLES` qui affecte à chaque variable un numéro, son type et un libellé;
- `RECTANGLE_MATRIX` qui correspond à la matrice de données symboliques. Elle contient donc les caractéristiques de chaque objet. La $i^{ième}$ ligne représente la description du $i^{ième}$ individu alors que la $j^{ième}$ colonne donne les valeurs prises par la $j^{ième}$ variable pour chacun des individus.

En plus de ces six blocs obligatoires, il peut y avoir des blocs supplémentaires :

- `RULES` qui spécifie la restriction des valeurs que peut prendre une certaine variable Y_k si celle-ci est dépendante d'une autre variable Y_l;
- `TRIANGLE_MATRIX` qui est la matrice triangulaire inférieure des dissimilarités entre tous les individus du jeu de données;
- `HIERARCHIE`
  ⋮

Exemple 6.2.1.

Voici le fichier sodas d'un jeu de données artificielles contenant 6 individus décrits par une variable intervalle, une variable multivaluée et une variable modale.

```
SODAS = (
 CONTAINS = (
 FILES, HEADER, INDIVIDUALS, VARIABLES, RECTANGLE_MATRIX),
FILE =  (
 procedure_name = "EDITOR",
 version = "sans",
 create_date = "13/4/2005",
),
HEADER = (
title = "Titre",
sub_title = "Sous-titre",
indiv_nb = 6,
```

```
var_nb = 3,
rules_nb = 0,
nb_var_set = 0,
nb_indiv_set = 0,
nb_var_nom = 0,
nb_var_cont = 0,
nb_var_text = 0,
nb_var_cont_symb = 1,
nb_var_nom_symb = 0,
nb_var_nom_mod = 2,
nb_na = 0,
nb_nu = 0,
nb_hierarchies = 0
),
INDIVIDUALS = (
(1,"O1","obj1"),
(2,"O2","obj2"),
(3,"O3","obj3"),
(4,"O4","obj4"),
(5,"O5","obj5"),
(6,"O6","obj6")
),
VARIABLES =  (
(1,inter_continue,"","V1","var_inter",0,0,22,98),
(2,mult_nominal,"","V2","var_multi",0,0,3,(
     (1,"V2_1","var_multi1",0),
     (2,"V2_2","var_multi2",0),
     (3,"V2_3","var_multi3",0))
),
(3,mult_nominal_Modif,"","V3","var_modal",0,0,3,(
     (1,"V3_1","var_modal1",0),
     (2,"V3_2","var_modal2",0),
     (3,"V3_3","var_modal3",0))
)
),
RECTANGLE_MATRIX = (
((34.00:56.00),(1,2),(1(0.5),  2(0.5),  3(0) )),
((22.00:76.00),(2),(1(0.3),  2(0.4),  3(0.3) )),
((78.00:87.00),(1,2,3),(1(0.5),  2(0.4),  3(0.1) )),
((57.00:98.00),(3),(1(0.3),  2(0),  3(0.7) )),
((32.00:45.00),(1,3),(1(1),  2(0),  3(0) )),
((75.00:85.00),(2,3),(1(0.5),  2(0.4),  3(0.1) ))
)
)END
```

6.3 Le fichier Scluster.h.

En C++, les structures permettent de regrouper des objets (des variables) au sein d'une entité repérée par un seul nom de variable. Le fichier **Scluster.h** contient une structure appelée **scluster** qui permet de stocker différents types de données, entre autres :

- le nombre d'objets dans le fichier sodas;
- le nombre de variables dans le fichier sodas;
- le nombre d'objets sélectionnés;
- le nombre de variables séléctionnées;
- le nombre de classes;
- le nombre d'itérations;
- le nombre d'essais;
- le code des différentes distances pour chaque type de variable;
- le code de l'initialisation (par prototypes aléatoires ou par partition aléatoire);
- ⋮

Notons que les valeurs prises par chacune de ces variables dépendent à la fois du fichier sodas séléctionné et des choix faits par l'utilisateur lui-même (type de distance,...) et restent inchangées tout au long de l'exécution du programme.
Notons également que, en C++, un commentaire est introduit par // et se termine à la fin de la ligne.

```
struct scluster {
char gpath [maxlenfile+2]; // name of the local path
char listing [maxlenfile+2]; // name of listing file
char file_log[maxlenfile+2]; // name of log file
char clustersds[maxlenfile +2];// name of clusters sds file
char prototypesds[maxlenfile +2];// name of prototype sds file
char nombase[maxlenfile+2]; // nmae of sds file
char file_tex[maxlenfile+2];// name of latex file
char file_VSTAR[maxlenfile+2];// name of VSTAR file
char file_SOV[maxlenfile+2];// name of sov file
char file_VPLOT[maxlenfile+2];// name of VPLOT file
char file_CLINT[maxlenfile+2];// name of CLINT file
long nb_objects_base; // number of objects in sds file
long nb_variables_base;// number of variables in sds file
long nb_objects;// number of objects selected
long nb_variables;// number of variables selected
int nb_class;// number of classes
int nb_iter;// max of iterations
int nb_run;// number of runs
```

```
int distance_quant;// code of distance var quantitatives and intervals
int distance_qual;// code of distance nominal
int distance_modal;// code of distance var multi-valuées and modales
int normalize; // code de normalisation
int init;// code of initialisation
int nb_mod;// total of modalities
int maxlibvar;// max lenght of libele of variables
int maxlibobj;// max lenght of libele of objects
int nb_col;// number of columns for list of objects
int debug;// code of debug
int nbclust; // 1 = determination of the number of clusters
int stabclust; // 1 = Stability of an individual cluster
};
```

Exemple 6.3.1.

Si nous reprenons les valeurs du fichier sodas présenté à l'exemple 6.2.1, les différentes variables de la structure prendront les valeurs suivantes :

- `nb_objects` = 6
- `nb_variables` = 3
- `nb_run` = 10
- `nb_iter` = 10
- `nb_class` = 2
- `distance_quant` = 0
- `distance_modal` = 0
- `init` = 0

si l'utilisateur souhaite effectuer 10 essais de recherche de la partition optimale en 2 classes des 6 objets décrits par les 3 variables intervalles, multivaluées et modales en procédant à une initialisation par prototype.
La distance choisie pour le variable intervalle étant la distance de Hausdorff et celle choisie pour les variables multivaluées et modales étant la distance de De Carvalho.

6.4 Le fichier calcul_scluster.cpp .

Le fichier `calcul_scluster.cpp` est celui qui contient, entre autre, les 4 fonctions correspondant aux 4 étapes de la méthode des nuées dynamiques dans le cas d'objets symboliques.

Il s'agit donc d'un des fichier les plus importants du programme Sclust.
Dans les sections qui suivent, nous allons décrire ces 4 fonctions ainsi que la fonction qui contient l'algorithme des nuées dynamiques.

6.4.1 La fonction Init_scluster.

Cette fonction permet l'initialisation de la partition. Notons que le programme Sclust permet deux types d'initialisations :

- une initialisation par partition aléatoire et
- une initialisation par prototypes.

Initialisation par une partition aléatoire.

Etant donné que la fonction connait, par l'intermédiaire de la structure scluster, le nombre d'objets dans le jeu de données et le nombre de classes demandées, `Init_scluster` est en mesure d'associer aléatoirement à chaque individu un nombre réel strictement compris entre 0 et `nb_class`. Ce nombre est ensuite arrondi à l'entier supérieur. Les nombres ainsi obtenus détermineront alors les classes initiales d'appartenance de chacun des individus.

Par la suite, le nombre d'individus dans chaque classe ainsi que le représentant (appelé noyau ou prototype) de chacune des classes sont déteminés. La fonction `Proto_scluster`, que nous analyserons à la section 6.4.4, construit les prototypes de chaque classe à partir des individus qui la composent.

Pour terminer, la fonction calcule la valeur du critère à l'aide de la fonction `W_scluster` qui sera présentée à la section 6.4.3. Voici le code C++ qui permet l'initialisation de l'algorithme par une partition aléatoire:

```
if(para.init==1)  // initialisation par une partition aleatoire
{
for (i=1;i<=imax;i++)
{
  x1=float(rand())/float(RAND_MAX); //0< x1 <1
  x1=x1*(float)kmax; // 0< x1 < kmax
  k=int (x1)+1; // x1 arrondi a l'entier supérieur k
  classe[i]=k; // l'individu est affecte a la classe k
  Nt[k]=Nt[k]+1; //le nombre d'individus de la classe k est incrémenté
};

// construction des prototypes
Proto_scluster(lis,para,M,vsel,classe,Proto);

//calcul de la valeur du critère
ww=W_scluster(lis,para,M,vsel,classe,Pw,Proto);
}
```

Initialisation par prototypes.

Dans ce type d'initialisation, le fonction associe aléatoirement à chaque classe un nombre entre 0 et `nb_objects`. Ce nombre est ensuite arrondi à l'entier supérieur. Les k nombres (où k est le nombre de classes) ainsi obtenus détermineront les individus qui serviront de prototypes pour chacune des classes.
Notons que deux classes distinctes ne peuvent pas être représentées par le même prototype.

6.4.2 La fonction Affect_scluster.

Une fois que chaque classe est représentée par un prototype, la fonction `Affect_scluster` permet de trouver, pour chaque individu, le prototype le plus proche au sens des distances choisies, et affecte l'individu considéré à la classe repésentée par ce prototype. `Affect_scluster` correspond au fait à la fonction d'affectation f dans la méthode des nuées dynamiques.
Voici le code C++ de cette fonction :

```
for (i=1;i<=imax;i++) // i individuals
{
 for (k=1;k<=kmax;k++) // k classes
 {
  dd=0.;l=0;
  for (j=0;j<nvar;j++) // j variables
  {
   num=vsel[j];
   (...)
   switch(M->gettypevar(num))
   {
    case continu:
    {
     l++;xx=M->index(i,num).getfval();
     dd0=fabs(Proto[k][l] - xx);
     switch(para.distance_quant)
     {
      case 0: //hausdorff
       dd=dd+Pw[j+1]*dd0;
      break;
      case 1: // L1
       dd=dd+Pw[j+1]*dd0;
      break;
      case 2: // L2
       dd=dd+Pw[j+1]*dd0*dd0;
      break;
      default :
```

```
    dd=dd+Pw[j+1]*dd0;
  }; //end switch para.distance_quant
 }; //end case continu
 break;
 case intercont:
 {
  l++;xmin=fabs(M->index(i,num).getinter()->getmin()-Proto[k][l]);
  l++;xmax=fabs(M->index(i,num).getinter()->getmax()-Proto[k][l]);
  switch(para.distance_quant)
  {
   case 0: //hausdorff
    dd=dd+Pw[j+1]*maxi(xmin,xmax);
   break;
   case 1: // L1
    dd=dd+Pw[j+1]*(xmin+xmax);
   break;
   case 2: // L2
    dd=dd+Pw[j+1]*(xmin*xmin+xmax*xmax);
   break;
   default :
   dd=dd+Pw[j+1]*maxi(xmin,xmax);
  }; //end switch para.distance_quant
 }; //end case intercont
 break;
 case nominal:
 {
  nmod=M->getnbcateg(num);
  for (kk=1;kk<=nmod;kk++)
  {
   l++;
   if(kk==M->index(i,num).getcateg()) xx=1.;
   else   xx=0.;
   switch(para.distance_qual)
   {
    case 0: //De Carvalho
     if ((int)100.0*xx==0.) dd=dd+Pw[j+1]*Proto[k][l];
     if ((int)100.0*Proto[k][l]==0.) dd=dd+Pw[j+1]*xx;
    break;
    case 1: //L1
     dd=dd+Pw[j+1]*fabs(Proto[k][l] - xx);
    break;
    case 2:  //L2
     dd=dd+Pw[j+1]*(Proto[k][l] - xx)*(Proto[k][l] - xx);
    break;
```

```
    default :
     if ((int)100.0*xx==0.) dd=dd+Pw[j+1]*Proto[k][l];
     if ((int)100.0*Proto[k][l]==0.) dd=dd+Pw[j+1]*xx;
   };// switch para.distance_qual
  }; //fin de la boucle sur le nombre de modalités
 };//end case nominal
 break;
 case multnomin:
 {
  nmod=M->getnbcateg(num);total=((dim_Mat*)M)->nomicard(i,num);
  for (kk=1;kk<=nmod;kk++)
  {
   l++;
   if(total == 0)
    xx=M->index(i,num).getnomin()->index(kk);
   else
    xx=((double)M->index(i,num).getnomin()->index(kk))/total;
   switch(para.distance_modal)
   {
    case 0: //De Carvalho
     if (xx==0.) dd=dd+Pw[j+1]*Proto[k][l];
     if (Proto[k][l]==0.) dd=dd+Pw[j+1]*xx;
    break;
    case 1: //L1
     dd=dd+Pw[j+1]*fabs(Proto[k][l] - xx);
    break;
    case 2: //L2
     dd=dd+Pw[j+1]*(Proto[k][l] - xx)*(Proto[k][l] - xx);
    break;
    default :
     if ((int)100.0*xx==0.) dd=dd+Pw[j+1]*Proto[k][l];
     if ((int)100.0*Proto[k][l]==0.) dd=dd+Pw[j+1]*xx;
   };//end switch para.distance_modal
  };//end boucle sur le nombre de modalités
 };//end case multinomin
 break;
 case multnominm:
 {
  nmod=M->getnbcateg(num);total=((dim_Mat*)M)->fuzzycard(i,num);
  for (kk=1;kk<=nmod;kk++)
  {
   l++;
   if(total == 0)
    xx=M->index(i,num).getnominM()->indexm(kk);
```

```
        else
         xx=M->index(i,num).getnominM()->indexm(kk)/total;
        switch(para.distance_modal)
        {
         case 0://De Carvalho
          if ((int)100.0*xx==0.) dd=dd+Pw[j+1]*Proto[k][l];
          if ((int)100.0*Proto[k][l]==0.) dd=dd+Pw[j+1]*xx;
         break;
         case 1: //L1
          dd=dd+Pw[j+1]*fabs(Proto[k][l] - xx);
         break;
         case 2: //L2
          dd=dd+Pw[j+1]*(Proto[k][l] - xx)*(Proto[k][l] - xx);
         break;
         default :
          if ((int)100.0*xx==0.) dd=dd+Pw[j+1]*Proto[k][l];
          if ((int)100.0*Proto[k][l]==0.) dd=dd+Pw[j+1]*xx;
         };//end switch para.distance_modal
        };//end boucle sur le nombre de modalités
       };//end case multinomin
       break;
      }; //end witch(M->gettypevar(num))
     }; //end boucle sur les variables
     if(k==1)
      dmin=dd;nclass=1;
     else
     {
      if(dmin > dd) dmin=dd;nclass=k;
     };
    }; //end boucle sur le nombre de classes k
    if(nclass != classe[i]) n_diff++;
    classe[i]=nclass;ww=ww+dmin;
   }; //end boucle sur nombre d'individus
  return n_diff;
```

Commentaires sur l'algorithme.

Remarquons premièrement que la partie qui gère les données manquantes n'a pas été affichée et a été remplacée par le symbole (...).

Supposons fixés l'individu i et la classe k, nous allons nous préoccuper de la boucle plus interne sur les variables.

Dans le cas de variables quantitatives (`case continu`) et intervalles (`case intercont`),

l'utilisateur a le choix entre 3 types de distances :

- la distance de Hausdorff (`case 0 :`);
- la distance L1 (`case 1 :`);
- la distance L2 (distance euclidienne) (`case 2 :`).

Dans le cas de variables nominales (`case nominal`) comme dans le cas de variables multivaluées (`case multnomin`) et modales (`case multnominm`), l'utilisateur a les choix entre 3 types de distances :

- la distance de De Carvalho (`case 0`);
- la distance L1 (`case 1`);
- la distance L2 (euclidienne)(`case 2`).

L'algorithme va sommer sur chacune des variables les distances entre l'individu i et la valeur du prototype de la classe k pour la variable considérée.

Une fois sorti de la boucle sur les variables, la distance totale entre l'individu i et le prototype de la classe k, c'est-à-dire la variable `dd`, est comparée avec la distance minimale (la variable `ddmin`) entre ce même individu i et une des $k-1$ classes considérées précédemment.

Si `dd<ddmin`, alors on modifie la valeur de `ddmin` ainsi que la classe d'affectation de l'individu i.
Si l'individu i a été changé de classe d'appartenance, on incrémente la variable `n_diff`.

Chaque individu est donc réaffecté à sa classe la plus proche et la valeur de `n_diff` qui correspond au nombre total de changements de classes effectués est renvoyée au programme principal.

6.4.3 La fonction W_scluster.

Cette fonction calcule le critère de la méthode des nuées dynamiques qui permet de mesurer l'adéquation entre toute partition et toute représentation de cette partition.

Dans Sclust, le critère est défini de la manière suivante:

$$W(P,L) = \sum_{\ell=1}^{k} \sum_{x_i \in C_\ell} d^2(x_i, L_\ell)$$

où

- k est le nombre de classes;
- C_ℓ est la classe ℓ;

– L_ℓ est le représentant (prototype) de la classe ℓ.

Comme nous l'avons déjà signalé dans les chapitres précédents, le critère que nous cherchons à minimiser est au fait l'inertie intra-classes W du nuage de points.

Voici le code C++ de cette fonction :

```
for (i=1;i<=imax;i++)
{
 dd=0.;dd0=0.;l=0;k=classe[i];
 for (j=0;j<nvar;j++)
 {
  num=vsel[j];
  switch(M->gettypevar(num))
  {
   (...)
  };// end switch M->gettypevar(num)
 };// end variables
 ww=ww+dd; num=classe[i];Wt[num]=Wt[num]+dd;
}; // end individuals
return ww;
```

Commentaires sur l'algorithme.

La fonction `W_scluster` est semblable à la fonction `Affect_scluster` décrite à la section 6.4.2 mis à part qu'une boucle sur les classes n'est plus nécéssaire car nous connaissons la classe d'appartenance de chaque individu.

En effet, en sortant de la boucle sur les variables, `dd` contient la distance totale entre l'individu i et et le prototype de sa classe d'affectation.

La valeur du critère `ww` est renvoyée au programme principal. Notons également qu'étant donné que l'inertie intra-classes peut se décomposer de manière additive sur les classes, le vecteur `Wt[]` contiendra les contributions de chacune des classes à l'inertie intra-classes du nuage de points.

6.4.4 La fonction Proto_scluster.

Dans la méthode des nuées dynamiques, une fois que chaque individu a été affecté à une classe, on doit calculer les nouveaux représentants (prototypes) de chaque classe grâce à la fonction de représentation g.

Dans Sclust, c'est la fonction `Proto_scluster` qui correspond à la fonction g.

Dans cette section, nous allons décrire les principales étapes de cette fonction.

Premièrement, on procède à l'initialisation des matrices `Proto[][]` et `Pt[][]`.
Ensuite on rentre dans une boucle extérieure sur les variables (supposons la variable j fixée).

- Si la variable est de type intervalle :

  Pour chaque individu i, la fonction met sa classe d'appartenance dans une variable `nclas`, puis récupère les extrémités inférieure `a` et supérieure `b` de l'intervalle :

  ```
  a=M->index(i,num).getinter()->getmin();
  b=M->index(i,num).getinter()->getmax();
  ```

  On met alors à jour la matrice `Pt[][]` qui contient pour chacune des k classes le nombre d'individus qui la composent:

  ```
  Pt[nclas][j]=Pt[nclas][j]+1;
  ```

  La matrice `Proto[][]` est également mise à jour:

  ```
  Proto[nclas][l]=Proto[nclas][l]+a;
  Proto[nclas][l+1]=Proto[nclas][l+1]+b;
  ```

  Après avoir analysé tous les individus,
  - `Proto[nclass][l]` contient la somme de toutes les extrémités inférieures `a` de la variable de type intervalle j décrivant les individus de la classe `nclass` et
  - `Proto[nclass][l+1]` contient la somme de toutes les extrémités supérieures `b` de la variable de type intervalle j décrivant les individus de la classe `nclass`.

- si la variable est de type multivaluée ou modale :

  On procède de la même manière que pour des variables intervalles sauf qu'il y a une boucle sur le nombre de modalités que la variable j peut prendre (pour les variables intervalles, il y avait 2 modalités : la borne inférieure et la borne supérieure).
  On a donc que chaque colonne de la matrice `Proto[][]` correspond à une des modalités que peut prendre l'individu i pour chaque variable.

  Après avoir analysé tous les individus, `Proto[nclass][l1]` contient la somme des distributions associées à chaque catégorie que prend la variable j pour décrire les individus appartenant à la classe `nclass`.

En dernier lieu, on normalise les éléments de la matrice `Proto[][]`, c'est-à-dire que les éléments de chaque ligne de la matrice (correspondant à chaque classe) sont

divisés par le nombre d'individus de cette classe.

```
for (i=1;i<=nmod;i++)
{
 l1++;
 for (k=1;k<=kmax;k++) //boucle sur les classes
 {
  if(Pt[k][j] != 0)
   Proto[k][l1]=Proto[k][l1]/(double)Pt[k][j];
 };
};
```

La matrice `Proto` contient alors les prototypes de chaque classe.

6.4.5 La fonction `main_scluster`.

La fonction `main_scluster` implémente l'algorithme principal des nuées dynamiques en appelant les différentes fonctions vues précédemment.

Supposons que la structure `Scluster` soit initialisée, voici le code C++ de cette fonction:

```
for (n=1;n<=n_run;n++) // boucle sur le nombre de runs
{
 Init_scluster(lis,para,M,vsel,classe, Nt, Proto,Pw);//initialisation de la méthode
 iter=0;n_diff=imax;
 while( (iter < iter_max)  &&  (n_diff != 0))//boucle sur les iterations
 {
  iter=iter+1;

  /************* Etape d'affectation ***************************/

  n_diff=Affec_scluster(lis,para,M,vsel,classe,Proto,Pw);//Calcul de la partition
  ww=W_scluster(lis,para,M,vsel,classe,Pw,Proto);// Calcul du critère

  /************* Etape de représentation ***********************/

  // Calcul des prototypes
  Proto_scluster(lis,para,M,vsel,classe,Proto);

  // Calcul du critère
  ww=W_scluster(lis,para,M,vsel,classe,Pw,Proto);
 };// end boucle sur les iterations

 // stockage des informations sur ce run
```

```
W_opt[n]=ww;N_iter[n]=iter;Nb_classe[n]=0;
for (k=1;k<=kmax;k++)
Nt[k]=0;
for (i=1;i<=imax;i++)
Nt[classe[i]]++;
for (k=1;k<=kmax;k++)
if(Nt[k] != 0) Nb_classe[n]++;

/********** Si solution optimale stockage du run **************/

if(n==1) //si c'est le premier run
{
 max_run=1;w_max=ww;
 for (i=1;i<=imax;i++)
  classe_opt[i]=classe[i];
 for (k=1;k<=kmax;k++)
 {
  for (m=0;m<=mod_max;m++)
  {
   Proto_opt[k][m]=Proto[k][m];
  }; //end boucle sur m
 }; //end boucle sur k
} //end if(n==1)
else
{
 if(ww < w_max)
 {
  max_run=n;w_max=ww;
  for (i=1;i<=imax;i++)
   classe_opt[i]=classe[i];
  for (k=1;k<=kmax;k++)
  {
   for (m=0;m<=mod_max;m++)
    {
     Proto_opt[k][m]=Proto[k][m];
    }; //end boucle sur m
   }; //end boucle sur k
  };// end if(ww < w_max)
 };// end else
};// end boucle sur le nombre de runs
```

Commentaires sur l'algorithme.

Si on suppose fixé un essai (`run`) de recherche de partition optimale en `k` classes, l'algorithme commence par calculer la partition initiale et le prototype de chacune des classes grâce à la fonction `Proto_scluster`.
Ensuite, à chaque itération, l'algorithme va successivement appeler :

- la fonction `Affect_scluster` qui va affecter chaque individu au prototype dont il est le plus proche;
- la fonction `Proto_scluster` qui va calculer le nouveau prototype de chaque classe;
- la fonction `W_scluster` qui va calculer la valeur du critère de classification.

Les itérations se terminent quand :

- le nombre d'itérations maximal (`n_iter`) défini par l'utilisateur est atteint;
- le nombre de changements de classes d'appartenance `n_diff` (renvoyé par la fonction `W_scluster`) est égale à 0.

Cette condition vient du fait que la suite $v_i = (P_i, L_i)$ composée de la partition et des représentants à l'itération i est stationnaire (voir chapitre 4), on a donc :

$$\exists N \text{ tel que } \forall i > N \ , \ v_i = v_{i+1}.$$

Il est donc inutile de continuer les itérations une fois que la valeur `n_diff` a atteint 0.

Les résultats de ce `run` sont alors stockés:

- la valeur optimale du critère dans `W_opt[n]`;
- le nombre d'itérations effectuées pour obtenir la solution dans `N_iter[n]`;
- le nombre d'individus par classe dans `Nt[classe[i]]`;
- le nombre de classes dans `Nb_classe[n]`.

Si le `run` considéré n donne une valeur de critère inférieure à celle des $n-1$ `run` déjà exécutés, on stocke en plus :

- la valeur du critère `ww` dans `wmax`;
- la partition optimale dans `classe_opt[i]`;
- les prototypes de chaque classe dans `Proto_opt[k][m]`.

6.5 Le fichier listing.

Le fichier listing est le fichier dans lequel les résultats donnés par Sclust sont enregistrés.

Nous allons à présent analyser le fichier listing que Sclust nous a fourni lorsque nous avons exécuté le programme sur le jeu de données artificielles présenté à l'exemple 6.2.1.

Dans un premier temps, Sclust affiche les valeurs prises par les principales variables définies dans la structure `scluster`.
Notons que ces valeurs seront inchangées pendant toute l'exécution du programme.
La valeur de l'inertie totale est également indiquée.

```
Learning Set          :     6
Number of variables   :     3
Number of iterations  :    20
Number of classes     :     2
Initialisation        :     0  random prototypes
Number of runs        :    10
Quantitative distance:      0  Hausdorff Distance
Boolean distance      :     0  De Carvalho Distance
Modal distance        :     0  De Carvalho Distance
Normalize             :     1  No
NBCLUST procedure     :     1  No
STABCLUST procedure   :     1  No

Initial Criterion : 153.188889
```

Deuxièmement, Sclust indique pour chaque variable sélectionnée:

- son numéro;
- les valeurs

$$100.\frac{T_j}{T} \quad \text{et} \quad 100.\frac{W_j}{W}$$

  qui sont les contributions de la variable respectivement à l'inertie totale et intra-classes;
- son poids;
- son nom;
- son type et, si nécessaire, le nombre de modalités.

Sclust indique également la liste des objets symboliques.

```
GROUP OF SELECTED VARIABLES :
=============================

( Pos )     Tj          Tj        Weight      Name            Type
          initial     used

(   1 )    97.92      97.92    1.000000      var_inter       INTERVAL
(   2 )     1.45       1.45    1.000000      var_multi       MULTINOMINAL   3 Modalities
(   3 )     0.63       0.63    1.000000      var_modal       MODAL   3 Modalities


LIST OF SYMBOLIC OBJECTS IN THE SET :
=====================================

obj1      obj2      obj3      obj4      obj5      obj6
```

Ensuite, pour chaque essai (**run**), Sclust donne, pour toutes les itérations, le nombre de changements de classes d'affectation de tous les individus ainsi que la valeur du critère.

```
RUN NUMBER :     1
================

Iteration Permutation Criterion
    1          6      74.266667
    2          0      54.433333

RUN NUMBER :     2
================

Iteration Permutation Criterion
    1          6     141.233333
    2          0     124.625000
```

Une fois que tous les essais sont terminés, Sclust affiche l'essai pour lequel la valeur optimale du critère a été obtenue.

```
OPTIMAL SOLUTION
================

RUN NUMBER :  1
CRITERION  :  54.433333
```

Après cela, Sclust affiche la partition optimale ainsi que le nombre d'individus dans chaque classe et la distance entre chaque individu et le représentant de sa classe (en unité de distance moyenne).

```
EDITION OPTIMAL PARTITION
=========================


Classe :   1  Cardinal :      3
================================

(  0) obj1     [0.0]  (  1) obj2     [1.9]  (  4) obj5     [1.1]

Classe :   2  Cardinal :      3
================================

(  2) obj3     [0.4]  (  3) obj4     [2.6]  (  5) obj6     [0.0]
```

La description de la partition nous donne les quantités suivantes :

- la dispersion totale du nuage de points qui est égale à l'inertie totale T du nuage de points;
- la valeur du critère qui est l'inertie intra-classes W;
- le pourcentage de dispersion expliquée donné par

$$100.\frac{T-W}{T} = 100.\frac{B}{T}$$

qui représente la part d'inertie que l'on obtient si l'on remplace chaque individu par le représentant de sa classe.

```
PARTITION DESCRIPTION
=====================

INITIAL CRITERION     : 153.188889
FINAL CRITERION   : 54.433333
Percentage of the explained criterion :     64.47
```

Viens alors la description des variables. En effet, pour chacune d'entre elles, Sclust nous donne les quantités suivantes:

- $100.\frac{B_j}{T_j}$ qui représente la part d'inertie de la variable j prise en compte par la partition;
- $100.\frac{W_j}{W}$ qui mesure la contribution relative de la variable j à l'inertie intra-classes;

- $100.\frac{T_j}{T}$ qui mesure la contribution relative de la variable j à l'inertie totale;
- la "Qualité" de la variable qui est calculée à partir de la formule

$$\left(\frac{100.\frac{B_j}{T_j}}{P}\right)$$

où P est le pourcentage de dispersion expliquée présenté précédemment.

```
VARIABLES DESCRIPTION
=====================

Position!     Name          Bj/Tj      Wj/W      Tj/T    Quality
(   1 )       var_inter     65.33     99.23     97.92       1.01
(   2 )       var_multi     25.00      0.56      1.45       0.39
(   3 )       var_modal     20.69      0.20      0.63       0.32
```

Sclust nous présente alors une description de chaque classe:

- $\frac{B_k}{T_k}$ qui représente la part d'inertie pour la classe k prise en compte par la partition;
- $\frac{W_k}{W}$ qui représente la contribution de la classe k à l'inertie intra-classes;
- $\frac{T_k}{T}$ qui représente la contribution de la classe k à l'inertie totale;
- $\frac{B_k}{N_k.B}$;
- $\frac{W_k}{N_k.W}$;

où N_k est le nombre d'individus dans la classe k.

```
CLUSTER DESCRIPTION
===================

Cluster  Size(Nk)  Bk/Tk       Wk/W      Tk/T     Bk/Nk.B    Wk/Nk.W
   1         3     21.42      60.01     27.14      20.004      9.046
   2         3     80.50      39.99     72.86      13.329     24.287
```

L'édition des prototypes variable par variable nous renseigne alors sur le prototype de chaque classe en indiquant

- si la variable est de **type intervalle**, la borne inférieure (`minimum`) et la borne supérieure (`maximum`) du représentant;
- si la variable est **de type multivaluée ou modale**, pour chaque modalité, la probabilité prise par le représentant de chaque classe.

Notons également que dans le cas de variables intervalles, Sclust fournit aussi les valeurs :

$$\frac{W_j^{(k)}}{T_j^{(k)}} \quad \text{et} \quad \frac{W_j^{(k)}}{W_j}$$

```
EDITION PROTOTYPES BY VARIABLES
===============================

Variable ( 1 ) var-inter

Cluster     Minimum       Maximum  Wkj/Tkj  Wkj/Wj
Set           42.50         55.50
   1          34.00         56.00    78.48   59.62
   2          75.00         85.00    19.00   40.38


Variable ( 2 ) var-multi

 Set      1       2      <= Cluster
 0.22   0.33    0.11    var_multi1
 0.39   0.50    0.28    var_multi2
 0.39   0.17    0.61    var_multi3


Variable ( 3 ) var-modal

 Set      1       2      <= Cluster
 0.52   0.60    0.43    var_modal1
 0.28   0.30    0.27    var_modal2
 0.20   0.10    0.30    var_modal3
```

Sclust termine le fichier listing en résumant pour chaque essai :

- le nombre d'itérations effectuées;
- le nombre de classes;
- la valeur du critère;

et fournit, à partir de ces valeurs les statistiques suivantes :

- la valeur minimale du critère;
- la moyenne de tous les critères;
- la valeur maximale du critère;
- l'écart-type.

```
CRITERION
=========

Run    Iteration Class    Criterion
 1      2          2       54.433333
 2      2          2       124.625000
 3      2          2       124.625000
 4      3          2       54.433333
 5      2          2       124.625000
 6      2          2       54.433333
 7      2          2       54.433333
 8      2          2       54.433333
 9      2          2       54.433333
10      2          2       54.433333

Statistics on criterion distribution :

Minimum              54.433333
Means                75.490833
Maximum              124.625000
Standard deviation   32.165863
```

Chapitre 7

Adaptation des méthodes de détermination du nombre de classes aux données symboliques.

7.1 Introduction.

Initialement, le programme Sclust permettait, grâce à la méthode des nuées dynamiques pour des données symboliques, de déterminer la partition optimale de n objets décrits par p variables en un nombre fixé de classes.

En 2002, S. Delogne a incorporé au programme Sclust un module appelé `NBCLUST`. Ce module permet de calculer les indices de détermination du nombre de classes uniquement pour des données de type intervalles en utilisant la modélisation milieu-longueur (voir chapitre 2).

L'année suivante, S. Collès a développé des sous-routines permettant de calculer les mêmes indices uniquement pour des variables multivaluées et modales.

L'objectif de ce mémoire était, dans un premier temps, d'incorporer les sous-routines de S. Collès au module `NBCLUST` et de créer une fonction permettant de calculer les distances entre objets symboliques de types intervalles sans utiliser la modélisation milieu-longueur mais en utilisant les distances de Hausdorff, L1 et L2 . Dans un second temps, nous avons adapté le code C++ des différentes fonctions du module afin de pouvoir calculer les indices de détermination du nombre de classes pour des objets symboliques décrits par des combinaisons de variables quantitatives, qualitatives, intervalles, multivaluées et modales.

7.2 Implémentation.

Les différentes fonctions nécéssaires au calcul des indices de détermination du nombre de classes sont rassemblées dans le fichier `classes.cpp` présenté en annexe.

Si le module `NBCLUST` est désactivé, Sclust calcule seulement la partition optimale en k classes où k est un nombre fixé a priori par l'utilisateur.

Par contre, lorsque `NBCLUST` est activé, Sclust calcule les différentes partitions optimales en 2,3,... et k classes.
Ensuite, il fait appel à la fonction `codage_all` du fichier `classes.cpp` afin de coder la matrice de données et la matrice de dissimilarités.

Cette matrice de dissimilarités est alors passée en argument à la fonction `milligan_scluster_m` qui calcule les indices

- **C-H** (Calinski et Harabasz),
- **C** (C-index) et
- **Gamma**

pour chaque nombre de classes allant de 2 à k.

La fonction `codage_all`.

Cette fonction permet de calculer la matrice de dissimilarités pour des objets symboliques décrits par n'importe quelle combinaison de types de variables.

Pour chaque couple d'individus, la fonction va sommer sur chaque variable les dissimilarités entre les 2 individus obtenues en utilisant les distances

- de Hausdorff,
- L1 et
- L2

pour des variables quantitatives et intervalles, et en utilisant les distances

- de De Carvalho,
- L1 et
- L2

pour des variables qualitatives, multivaluées et modales.

La fonction `milligan_scluster_m`.

Pour calculer les indices de Milligan et Cooper, cette fonction a simplement besoin de connaître la matrice de dissimilarités `d[][]` (calculée par `codage_all`) ainsi que la matrice `t_classe[][]` où `t_classe[i][k]` donne la classe d'affectation de l'individu `i` lors d'un partitionnement en `k` classes.

Une fois que ces indices sont calculés pour chaque nombre de classe, `milligan_scluster_m` imprime les résultats obtenus dans le fichier listing.

7.3 Sortie dans le fichier listing.

Les résultats obtenus par la fonction `milligan_scluster_m` sont présentés de la manière suivante:

```
============================================================
MILLIGAN et COOPER sur les partitions générées par Sclust
============================================================

Ngps        C-H         C-index       Gamma

  9       50.01377       0.03190      0.92316
  8       44.27354       0.05798      0.90819
  7       41.83026       0.05855      0.87005
  6       58.72534       0.03472      0.87875
  5       65.79971       0.03316      0.87556
  4       61.14946       0.05364      0.86495
  3       71.54385       0.03974      0.88908
  2       92.73942       0.00342      0.99981
  1          -              -            -

------- END OF PROGRAM SCLUST ---------
```

Dans cet exemple, l'utilisateur a choisi de chercher la partition optimale en 9 classes.
Sclust a donc calculé les partitions optimales en 2,3,4,...,8 et 9 classes.
Le module `NBCLUST` a ensuite calculé les indices de Milligan et Cooper pour chacun des nombres de classes.

Comme nous l'avons expliqué dans le chapitre 5, le nombre optimal de classes est celui pour lequel l'indice de Calinski et Harabasz est maximal, l'indice du C-index est le plus proche de 0 et l'indice Gamma est le plus proche de 1.

Il semble alors évident que, dans cet exemple, le nombre optimal de classes est 2.

Chapitre 8

Applications.

8.1 Introduction.

Le but de ce chapitre est d'appliquer le module NBCLUST à un certain nombre de jeux de données artificiels et réels afin de montrer qu'à présent, il fonctionne pour toutes les combinaisons de type de variables.

Pour chacun de ces jeux de données, nous allons tester les méthodes de détermination du nombre de classes avec les différentes distances de Hausdorff, de De Carvalho, L1 et L2.

8.2 Paramétrisation de Sclust.

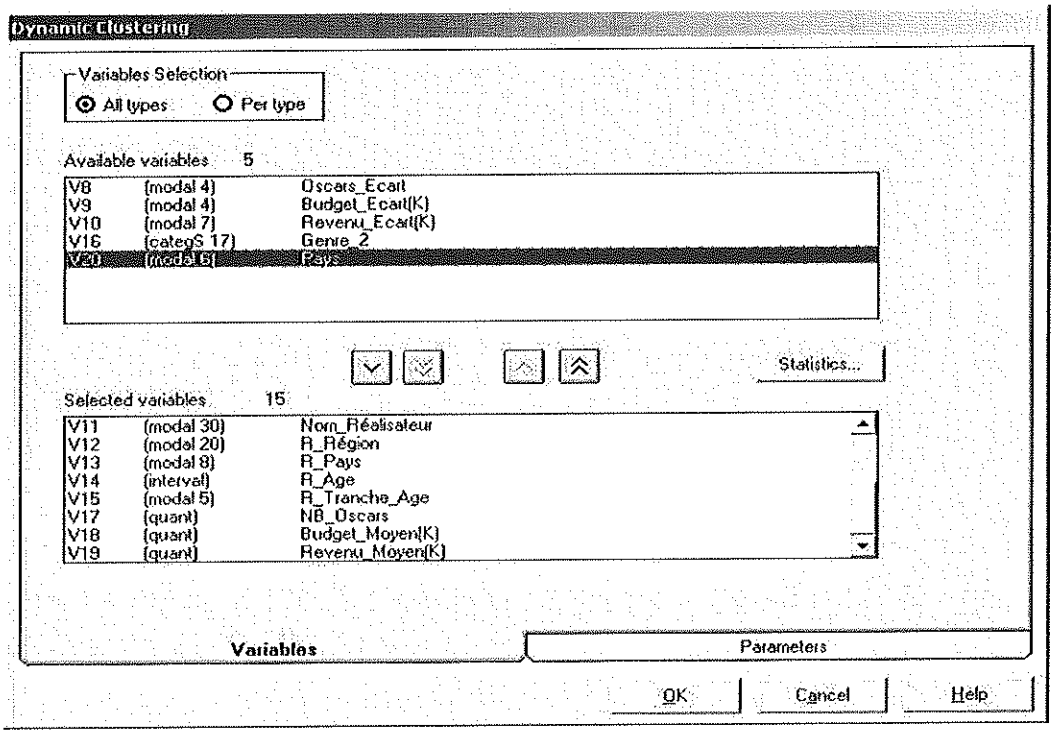

FIG. 8.1 – *Fenêtre de sélection des variables du programme Sclust.*

En premier lieu, Sclust nous donne la possibilité de choisir les variables que nous allons utiliser pour procéder à la classification (voir figure 8.1).

La fenêtre permettant de régler les différents paramètres du programme Sclust est présentée à la figure 8.2.
Dans cet exemple, l'utilisateur

- définit le nombre de classes = 8;
- définit le nombre d'essais = 10;
- définit le nombre maximal d'itérations = 20;
- choisit une initialisation par prototype (voir chapitre 6);
- désactive la normalisation des variables;
- active le module `NBCLUST`;
- désactive la procédure de validation;
- choisit comme distance quantitative (pour des variables quantitatives classiques ou intervalles) la distance de Hausdorff;
- choisit comme distance booléenne (pour des variables qualitatives classiques) la distance de De Carvalho et
- choisit comme distance modale (pour des variables multivaluées et modales) la distance Euclidienne (L2).

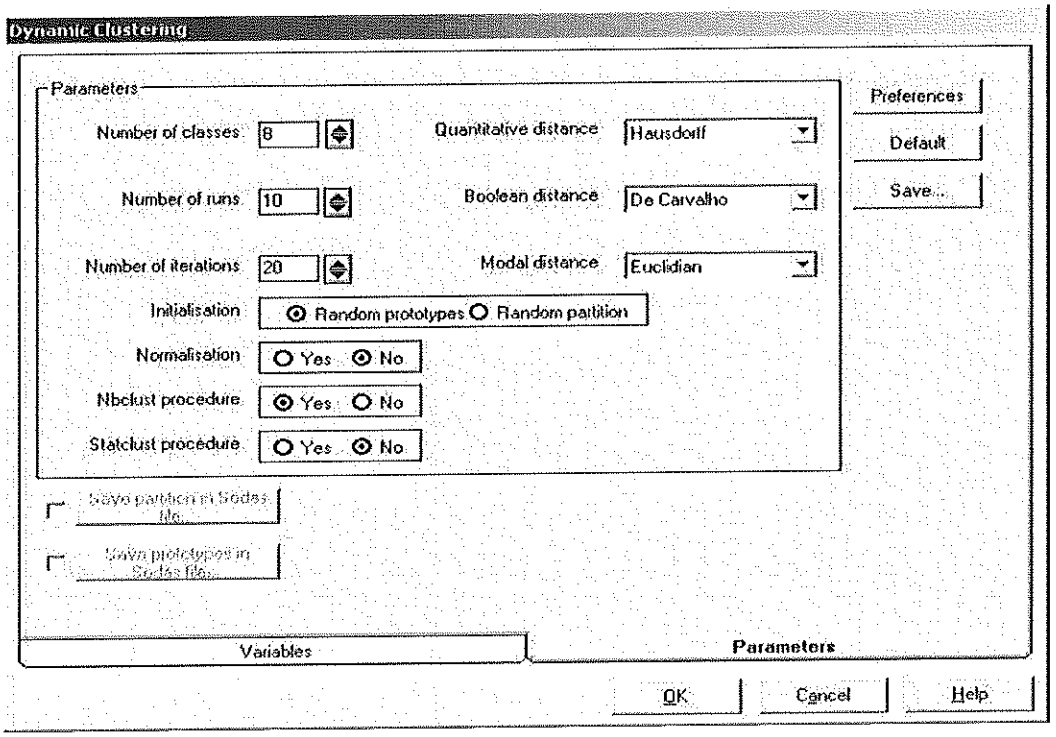

FIG. 8.2 – *Fenêtre de paramétrisation du programme Sclust.*

8.3 Présentation des résultats.

Pour chacun des exemples, nous commencerons par présenter brièvement les données ainsi que les variables mesurées sur chaque objet.

Nous analyserons ensuite les résultats obtenus par le programme Sclust et le module NBCLUST lorsque toutes les variables disponibles sont sélectionnées et la normalisation des variables est désactivée.
Cela nous permettra d'avoir une première appréciation sur une éventuelle classification des données.

Les résultats du module NBCLUST seront présentés sous la forme d'un tableau à double entrée (voir tableau 8.1).

		C-H	C	Γ
SCLUST				

TAB. 8.1 – *Tableau utilisé pour présenter les résultats des méthodes de Milligan et Cooper sur Sclust.*

Les trois dernières colonnes correspondent aux méthodes de détermination du nombre de classes de Milligan et Cooper applicables aux méthodes de classification non-hiérarchiques c'est-à-dire les méthodes :

- de Calinski et Harabasz;
- du C-index et
- Gamma.

Le chiffre contenu à l'intersection de la ligne SCLUST et d'une des trois colonnes indiquera le nombre de classes suggéré par la méthode de détermination du nombre de classes considérée, appliquée aux partitions générées par Sclust.

La deuxième colonne quant à elle, nous indiquera si Sclust retrouve les classes naturelles présentes dans les données. Un signe "+" signalera une classification correspondant aux classes naturelles et un signe "-" le cas contraire.

Après cela, nous présenterons un tableau contenant les valeurs des trois indices de Milligan et Cooper pour différents nombres k de classes (voir tableau 8.2).

Par exemple, la valeur de l'indice Gamma lorsque les données sont partitionnées en 6 classes sera donné par l'intersection de la ligne $k = 6$ et de la colonne Γ.

Rappelons déjà que le nombre k de classes est celui pour lequel

- la valeur de l'indice de Calinski et Harabasz (C-H) atteint un maximum relatif ou absolu;
- la valeur du C-index (C) est la plus proche de 0;
- l'indice Gamma est le plus proche de 1.

SCLUST	C-H	C	Γ
k=8			
k=7			
k=6			
k=5			
k=4			
k=3			
k=2			
k=1			

TAB. 8.2 – *Valeurs des indices de Milligan et Cooper pour la méthode Sclust.*

Après avoir comparé les résultats obtenus avec les différentes distances, nous mettrons en évidence les variables qui contribuent le plus à l'inertie intra-classes de la partition.
Ensuite, nous analyserons les résultats obtenus par Sclust et NBCLUST lorsque la normalisation des variables est activée.

Pour terminer, nous tenterons de valider les résultats obtenus avec une autre méthode de classification symbolique du logiciel SODAS : la méthode DIV [17].

Il s'agit d'une méthode descendante hiérarchique (voir chapitre 3) qui à chaque étape, divise une classe en deux selon une question binaire.
Notons que cette méthode est seulement applicable pour des variables quantitatives classiques et intervalles.

8.4 Les données de Ruspini.[6]

Dans cette section, le but est de montrer que le module NBCLUST fonctionne également sur des données quantitatives classiques.

8.4.1 Présentation des données.

Ce jeu de données est artificiel et est composé de 75 individus décrits par 2 variables quantitatives. Une représentation graphique dans le plan des données de Ruspini est donnée à la figure 8.3.

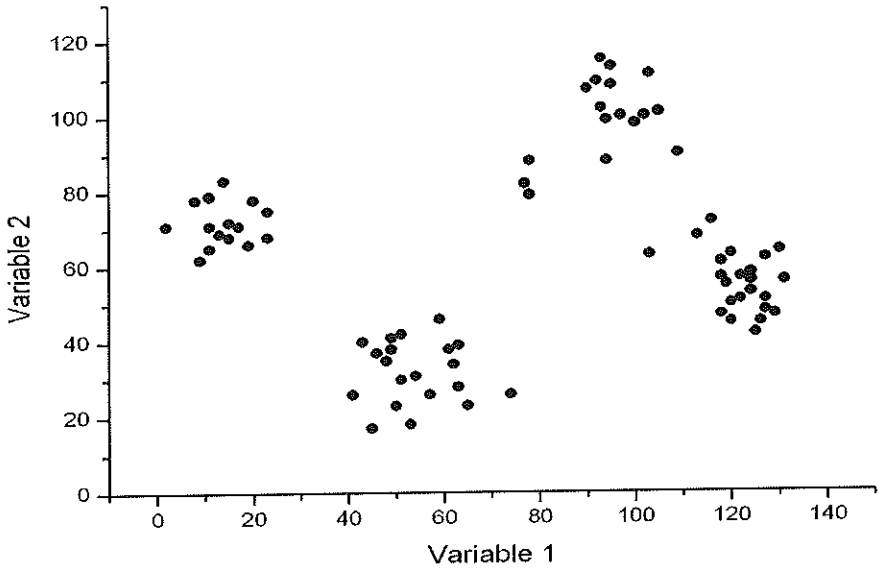

FIG. 8.3 – *Données de Ruspini.*

8.4.2 Distance de Hausdorff.

Les méthodes de détermination du nombre de classes de Milligan et Cooper appliquées aux partitions produites par Sclust donnent les résultats suivants :

		C-H	C	Γ
SCLUST	+	4	4	4

TAB. 8.3 – *Résultats obtenus avec la distance de Hausdorff.*

Les valeurs des indices des Milligan et Cooper sont données dans le tableau ci-dessous.

SCLUST	C-H	C	Γ
k=10	336.63267	0.00452	0.94107
k=9	351.05628	0.00446	0.96640
k=8	314.11908	0.00665	0.96088
k=7	357.82827	0.00591	0.96422
k=6	378.34155	0.00421	0.99317
k=5	358.04521	0.00681	0.99370
k=4	**429.72368**	**0.00093**	**0.99885**
k=3	139.07264	0.03473	0.90994
k=2	128.17871	0.02789	0.92905
k=1	-	-	-

TAB. 8.4 – *Valeurs des indices de Milligan et Cooper pour la méthode Sclust.*

Le signe "+" dans la deuxième colonne du tableau 8.3 indique que Sclust retrouve bien la structure naturelle des données en 4 classes (voir figure 8.4).

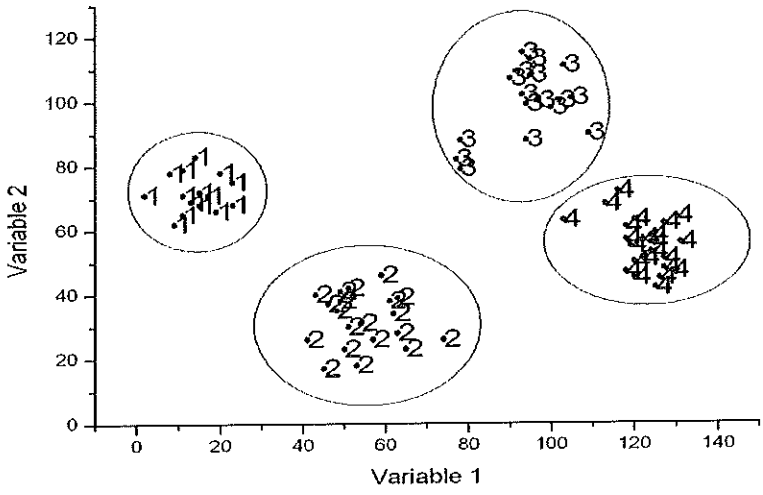

FIG. 8.4 – *Partitionnement des données de Ruspini en 4 classes par la méthode Sclust.*

8.4.3 Distance L1.

Comme pour la distance de Hausdorff, les valeurs des indices de Milligan et Cooper (voir tableau 8.6) indiquent la présence de 4 classes dans les données :

		C-H	C	Γ
SCLUST	+	4	4	4

TAB. 8.5 – *Résultats obtenus avec la distance L1.*

SCLUST	C-H	C	Γ
k=10	332.35155	0.00453	0.93768
k=9	348.53447	0.00438	0.96224
k=8	372.67354	0.00436	0.96726
k=7	379.09199	0.00502	0.97016
k=6	377.05494	0.00388	0.99369
k=5	361.06901	0.00943	0.99885
k=4	**429.72368**	**0.00093**	**0.99885**
k=3	139.07264	0.03473	0.90994
k=2	128.17871	0.02789	0.92905
k=1	-	-	-

TAB. 8.6 – *Valeurs des indices de Milligan et Cooper pour la méthode Sclust.*

Par ailleurs, la partition en 4 classes engendrée lorsque Sclust utilise la distance L1 est la même que celle engendrée lorsqu'il utilise la distance de Hausdorff (voir figure 8.4)

8.4.4 Distance L2.

Les résultats obtenus lorsque Sclust utilise la distance euclidienne sont les suivants :

		C-H	C	Γ
SCLUST	+	4	4	4

TAB. 8.7 – *Résultats obtenus avec la distance L2.*

Comme nous pouvons le voir dans le tableau 8.8, NBCLUST détecte bien la présence de 4 classes dans les données sauf pour l'indice de Calinski et Harabasz où on remarque 3 maxima relatifs en $k = 4$, $k = 7$ et $k = 9$.

Cependant, on remarque qu'un saut important [1] est effectué à la valeur de $k = 4$.

1. passage de 551.09 pour $k = 3$ à 7814.06 pour $k = 4$

On conclut donc que 4 classes sont présentes dans les données.

SCLUST	C-H	C	Γ
k=10	14078.74993	0.00011	0.94512
k=9	**15667.17626**	0.00010	0.96684
k=8	9345.91464	0.00025	0.98324
k=7	**11152.26517**	0.00021	0.99063
k=6	9359.40653	0.00035	0.99793
k=5	7387.51750	0.00034	0.99877
k=4	**7814.05544**	**0.00013**	**0.99881**
k=3	551.08730	0.01864	0.90681
k=2	566.84473	0.01387	0.94735
k=1	-	-	-

TAB. 8.8 – *Valeurs des indices de Milligan et Cooper pour la méthode Sclust.*

La partition en 4 classes engendrée par Sclust en utilisant la distance L2 est identique à celles engendrées en utilisant les distances de Hausdorff et L1 (voir figure 8.4).

8.4.5 La méthode DIV.

Si nous appliquons la méthode DIV aux données de Ruspini en choisissant une classification en 4 classes, voici les résultats obtenus :

```
THE CLUSTERING TREE :
---------------------
   - the number noted at each node indicates
     the order of the division
   - Ng <-> yes and Nd <-> no

          +----  Classe 1 (Ng=20)
          !
      !----2- [var2 <= 54.000000]
      !   !
      !   +----  Classe 3 (Nd=15)
      !
 !----1- [var1 <= 75.500000]
      !
      !   +----  Classe 2 (Ng=23)
      !   !
      !----3- [var2 <= 75.500000]
          !
          +----  Classe 4 (Nd=17)
```

- Les éléments de la classe 1 sont ceux pour lesquels :
  - la variable 1 est inférieure ou égale à 75.5;
  - la variable 2 est inférieure ou égale à 54.0.
- Les éléments de la classe 2 sont ceux pour lesquels :
  - la variable 1 est supérieure à 75.5;
  - la variable 2 est inférieure ou égale à 75.5.
- Les éléments de la classe 3 sont ceux pour lesquels :
  - la variable 1 est inférieure ou égale à 75.5;
  - la variable 2 est supérieure à 54.0.
- Les éléments de la classe 4 sont ceux pour lesquels :
  - la variable 1 est supérieure à 75.5;
  - la variable 2 est supérieure à 75.5.

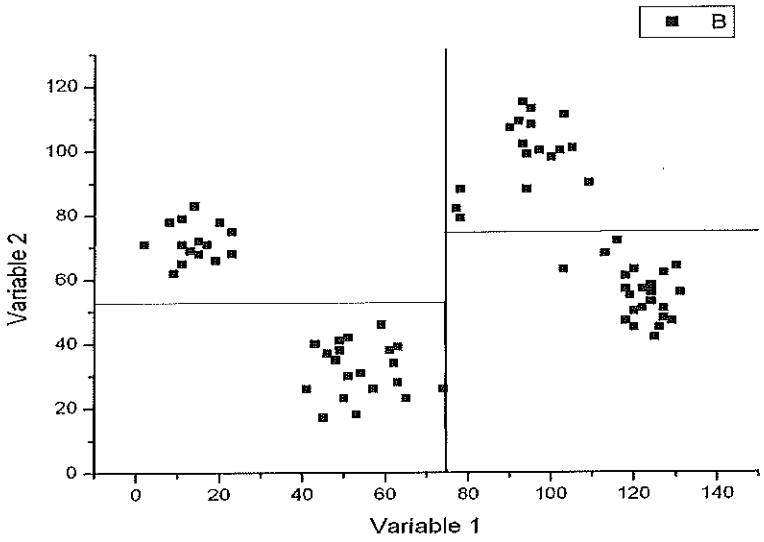

FIG. 8.5 – *Partitionnement des données de Ruspini en 4 classes par la méthode DIV.*

8.4.6 Conclusion.

Commençons par signaler que le module NBCLUST détecte, pour n'importe quelle distance (Hausdorff, L1 et L2), la présence de 4 classes dans les données.

De plus, les méthodes de classification symboliques Sclust et DIV retrouvent bien la partition naturelle des données.

8.5 Données sur le cinéma.

8.5.1 Présentation du jeu de données.

Cette base de données peut être obtenue à l'adresse suivante :
`http://www.ceremade.dauphine.fr/~touati/exemples.htm`.

Elle contient un ensemble de 17 catégories de films décrits par 19 variables de différents types.
Voici la liste des 17 individus :

1. science-fiction;
2. Suspense;
3. Fantastique;
4. Comédie dramatique;
5. Drame;
6. Film historique;
7. Catastrophe;
8. Policier;
9. Action;
10. Comédie fantastique;
11. Animation;
12. Comédie;
13. Aventure;
14. Film de guerre;
15. Western;
16. Espionnage;
17. Comédie musicale.

Ces individus sont décrits par 3 variables quantitatives classiques, 5 variables intervalles et 11 variables modales.
Le tableau 8.9 donne une brève description de chacune des variables ainsi que le nombre de modalités que les variables modales peuvent prendre.

Nous allons, à présent, tester les méthodes de détermination du nombre de classes de Milligan et Cooper sur les partitions de ce jeu de données générées par Sclust en utilisant :

- les distances de Hausdorff et de De Carvalho;
- la distance L1 et
- la distance L2.

Notons également que nous ne connaissons rien sur une éventuelle classification des données.

Nom de la variable	Type de variable	Description	Nombre de modalités
Année	intervalle	Années de sortie des films pour une catégorie	-
Acteur_Principal	modale	noms des différents acteurs de la catégorie de films	76
Lieu_Tournage	modale	lieux de tournage des films de la catégorie	25
Continent_Tournage	modale	continents de tournage des films de la catégorie	6
Nombre_Oscars	intervalle	nombre d'Oscars gagnés par les films de la catégorie	-
Budget	intervalle	budget pour une catégorie de films	-
Revenu	intervalle	revenu pour une catégorie de films	-
Oscars_Ecart	modale	écart[2]des nombres d'Oscars pour une catégorie de films	4
Budget_Ecart	modale	écart des budgets pour une catégorie de films	4
Revenu_Ecart	modale	écart des revenus pour une catégorie de films	7
Nom_Réalisateur	modale	nom des différents réalisateurs pour une catégorie de films	30
R_Région	modale	région des réalisateurs de la catégorie de films	20
R_Pays	modale	pays des réalisateurs de la catégorie de films	8
R_Age	intervalle	age des réalisateurs de la catégorie de films	-
R_Tranche_Age	modale	tranche d'âge des réalisateurs de la catégorie de films	5
NB_Oscars	quantitative	nombre total d'Oscars pour la catégorie de films	1
Budget_Moyen	quantitative	budget moyen pour la catégorie de films	1
Revenu_Moyen	quantitative	revenu moyen pour la catégorie de films	1
Pays	modale	nationalité de la catégorie de films	6

TAB. 8.9 – *Variables du jeu de données sur le cinéma.*

8.5.2 Distances de Hausdorff et de De Carvalho.

Les résultats obtenus en utilisant la distances de Hausdorff pour les variables quantitatives et intervalle et la distance de De Carvalho pour les variables modales sont donnés dans le tableau ci-dessous :

	C-H	C	Γ
SCLUST	5	5	5

TAB. 8.10 – *Résultats obtenus avec les distances de Hausdorff et de De Carvalho.*

En effet, nous pouvons remarquer dans le tableau 8.11 que l'indice C-H atteint un maximum pour $k = 5$ et que l'indice du C-index et l'indice Gamma sont également favorables à un partitionnement en 5 classes.

Dès lors, il semble correct d'opter pour une partition en 5 classes.

2. écart=min-max

Voici la répartition des catégories de films fournie par Sclust :

classe 1 : Science-fiction et Fantastique;

classe 2 : Comédie, Comédie musicale, Western et Espionnage;

classe 3 : Catastrophe;

classe 4 : Drame;

classe 5 : Film de guerre, Film historique, Action, Comédie fantastique, Suspense, Policier, Animation, Aventure et Comédie dramatique.

SCLUST	C-H	C	Γ
k=8	42.13578	0.01089	0.82292
k=7	49.13090	0.01000	0.84975
k=6	41.81755	0.01515	0.83357
k=5	**49.79346**	**0.00959**	**0.94271**
k=4	36.59247	0.01724	0.91116
k=3	12.10472	0.08955	0.82041
k=2	16.46934	0.08943	0.84249
k=1	-	-	-

TAB. 8.11 – *Valeurs des indices de Milligan et Cooper pour la méthode Sclust.*

8.5.3 Distance L1.

Voici les résultats obtenus lorsqu'on utilise la distance L1 pour chaque type de variable:

	C-H	C	Γ
SCLUST	3,9	9	9

TAB. 8.12 – *Résultats obtenus avec la distance L1.*

En consultant, le tableau 8.13, on remarque que l'indice de Calinski et Harabasz atteint un maximum local en $k = 3$ et en $k = 9$.
On constate également que les indices C et Γ donnent de très bonne valeur pour une partition en 9 classes.

Voici la répartition des catégories de film si on choisit une partition en 9 classes:

classe 1 : Western et Espionnage;

classe 2 : science-fiction et Fantastique;

classe 3 : Comédie et Comédie musicale;

classe 4 : Film de guerre;

classe 5 : Film historique, Action et Comédie fantastique;
classe 6 : Drame;
classe 7 : Comédie dramatique;
classe 8 : Catastrophe;
classe 9 : Suspense, Policier, Animation et Aventure.

SCLUST	C-H	C	Γ
k=12	242.51096	0.00013	0.99487
k=11	205.82117	0.00013	0.99683
k=10	194.43012	0.00011	1.00000
k=9	**209.78986**	**0.00000**	**1.00000**
k=8	149.40658	0.00011	0.99669
k=7	63.46114	0.00426	0.94068
k=6	58.84312	0.00519	0.92133
k=5	35.22321	0.01915	0.87333
k=4	15.11325	0.04362	0.90447
k=3	**19.66938**	0.09307	0.83112
k=2	8.20561	0.17937	0.86771
k=1	-	-	-

TAB. 8.13 – *Valeurs des indices de Milligan et Cooper pour la méthode Sclust.*

8.5.4 Distance L2.

Voici les résultats obtenus avec la distance euclidienne:

	C-H	C	Γ
SCLUST	5,8	8	8

TAB. 8.14 – *Résultats obtenus avec la distance L2.*

Comme nous pouvons le voir dans le tableau 8.15, les indices de Calinski et Harabasz sont croissants avec le nombre de classes. Cependant, on remarque un saut important pour $k = 5$ ainsi que pour $k = 8$.
Les valeurs des indices C et Γ donnent de très bons résultats pour $k = 8$.

Notons également que les valeurs de ces indices pour $k = 5$ sont aussi acceptables. La partition en 5 classes générée par Sclust en utilisant la distance L2 est identique à celle générée lorsque Sclust utilise les distances de Hausdorff et de De Carvalho au point près que l'individu "comédie dramatique" change de classe et rejoint les films de sciences-fictions et les films fantastiques.

SCLUST	C-H	C	Γ
k=10	139821.04408	0.00000	1.00000
k=9	49181.86670	0.00000	0.98656
k=8	**47476.47481**	**0.00000**	**0.99890**
k=7	10480.97451	0.00002	0.96853
k=6	3062.29980	0.00012	0.93151
k=5	**2681.30750**	0.00015	0.98562
k=4	229.31511	0.00430	0.89921
k=3	303.54779	0.00988	0.82051
k=2	109.39259	0.02401	0.89742
k=1	-	-	-

TAB. 8.15 – *Valeurs des indices de Milligan et Cooper pour la méthode Sclust.*

Par contre, si on partitionne les données en 8 classes, on obtient la répartition suivante :

classe 1 : Comédie et Comédie musicale;
classe 2 : science-fiction et Fantastique;
classe 3 : Suspense, Policier, Animation et Aventure;
classe 4 : Catastrophe;
classe 5 : Film historique, Film de guerre, Action et Comédie fantastique;
classe 6 : Drame;
classe 7 : Western et Espionnage;
classe 8 : Comédie dramatique.

8.5.5 Analyse complémentaire.

En analysant plus en profondeur le fichier listing produit par Sclust, on remarque que la contribution à l'inertie intra-classes de chacune de variables (coéfficient `Wj/W`) est nulle sauf pour les variables `Budget`, `Revenu`, `Budget_Moyen` et `Revenu_Moyen`.

Ce sont donc ces variables qui influencent la classification.

Donc, si nous n'activons pas la normalisation proposée par Sclust, la classification est uniquement influencée par quatre variables. C'est pourquoi nous allons, à présent, analyser les résultats obtenus par Sclust lorsque les variables ont été normalisées au préalable.

Voici les résultats obtenus en utilisant toutes les variables avec les distances de Hausdorff et de De Carvalho et en activant la normalisation:

On voit bien dans la table 8.16 que les indices de Milligan et Cooper suggèrent une partition en 5 classes.

SCLUST	C-H	C	Γ
k=9	2.79599	0.08670	0.72596
k=8	2.60144	0.12803	0.67950
k=7	2.86624	0.13128	0.68385
k=6	2.68392	0.16963	0.63340
k=5	**3.67840**	**0.09178**	**0.75508**
k=4	3.09304	0.19055	0.72135
k=3	2.66944	0.26949	0.58951
k=2	2.25608	0.42516	0.42351
k=1	-	-	-

TAB. 8.16 – *Valeurs des indices de Milligan et Cooper pour la méthode Sclust.*

Cependant, la partition en 5 classes est différente de celle obtenue lorsqu'on n'utilisait pas la normalisation :

classe 1 : Suspense, Comédie dramatique, Policier, Action, Animation, Comédie, Aventure;
classe 2 : Comédie musicale;
classe 3 : Western, Espionnage;
classe 4 : Catastrophe, Comédie fantastique;
classe 5 : Science-fiction, Fantastique, Drame, Film historique, Film de guerre.

Dans ce cas, les variables qui contribuent le plus à l'inertie intra-classes de la partition sont `Budget_Ecart`, `Continent_Tournage`, `R_Age` et `R_Tranche_Age`.
Cela explique le fait que les catégories Western et Espionnage soient dans la même classe. En effet, les valeurs des 4 variables les plus importantes pour ces 2 individus sont quasiment identiques.
L'isolation de la catégorie Comédie musicale peut être expliquée par le fait qu'elle est la seule dont la variable `Continent_Tournage` prend la modalité "Océanie".

8.5.6 La méthode DIV.

Etant donné que dans la plupart des cas, le module NBCLUST nous suggère une partition en 5 classes, nous avons appliqué la méthode DIV aux données sur le cinéma en sélectionnant seulement les variables quantitatives classiques et intervalles.
Voici les résultats obtenus :

```
THE CLUSTERING TREE :
---------------------
   - the number noted at each node indicates
     the order of the division
   - Ng <-> yes and Nd <-> no
```

```
          +----  Classe 1 (Ng=4)
          !
      !----3- [Revenu(K) <= 134125.000000]
      !   !
      !   !    +----  Classe 4 (Ng=9)
      !   !    !
      !   !----4- [Année <= 1995.750000]
      !        !
      !        +----  Classe 5 (Nd=1)
      !
!----1- [Revenu(K) <= 467900.000000]
      !
      !   +----  Classe 2 (Ng=2)
      !   !
      !----2- [Revenu(K) <= 704344.750000]
          !
          +----  Classe 3 (Nd=1)
```

On voit que :

- les éléments de la classe 1 sont ceux pour lesquels :
  - la variable **Revenu** est inférieure ou égale à 134125;
- les éléments de la classe 2 sont ceux pour lesquels :
  - la variable **Revenu** est comprise entre 467900 et 704344.75;
- les éléments de la classe 3 sont ceux pour lesquels :
  - la variable **Revenu** est supérieure 704344.75;
- les éléments de la classe 4 sont ceux pour lesquels :
  - la variable **Revenu** est comprise entre 134125 et 467900;
  - la variable **Année** est inférieure à 1995.75;
- les éléments de la classe 5 sont ceux pour lesquels :
  - la variable **Revenu** est comprise entre 134125 et 467900;
  - la variable **Année** est supérieure à 1995.75.

On obtient donc la partition suivante :

classe 1 : Comédie, Comédie musicale, Western, Espionnage;

classe 2 : Science-fiction, fantastique;

classe 3 : Drame;

classe 4 : Suspense, Comédie dramatique, Film historique, Policier, Action, Comédie fantastique, Animation, Aventure, Film de guerre;

classe 5 : Catastrophe.

Notons que cette partition en 5 classes est identique à celle engendrée par Sclust lorsque les distances de Hausdorff et de De Carvalho sont utilisées et que la normalisation est désactivée.

8.5.7 Conclusion.

Au cours de cette étude, nous avons remarqué que les méthodes de détermination du nombre de classes donnaient des résultats différents selon les distances utilisées. En effet, les distances de Hausdorff et de De Carvalho suggèrent de façon évidente une partition en 5 classes alors que les distance L1 et L2 suggèrent un partitionnement respectivement en 9 et 8 classes.
Cependant, en analysant les résultats obtenus avec la distance L2, on remarque que les valeurs des indices de Milligan et Cooper pour une partition en 5 classes sont également satisfaisants.

Nous avons également remarqué que sans la normalisation des variables, la classification était principalement influencée par les quatre variables suivantes : `Budget`, `Revenu`, `Budget_Moyen` et `Revenu_Moyen`.
Si nous activons la normalisation des variables, NBCLUST nous suggère une partition en 5 classes mais celle-ci est différente de celle générée sans normalisation.

Par contre, la partition en cinq classes engendrée par la méthode DIV est identique à celle engendrée par Sclust lorsque les distances de Hausdorff et de De Carvalho sont utilisées.

De manière générale, nous pouvons aussi affirmer que certaines catégories de films sont souvent associées l'une à l'autre (comédie avec comédie musicale, science-fiction avec fantastique, Western avec Espionnage), et que d'autres catégories sont isolées (drame et catastrophe).

8.6 Données sur la musique.

8.6.1 Présentation du jeu de données

Ce jeu de données est composé de 33 individus décrits par 14 variables de différents types.
Il peut être téléchargé à partir du site web suivant :
`http://www.ceremade.dauphine.fr/~touati/exemples.htm`.

Les individus représentent les chanteurs suivants :

Eminem	Madonna
Mickael Jackson	Janet Jackson
Britney Spears	Mariah Carey
Lenny Kravitz	Whitney Houston
Lauryn Hill	Craig David
Florent Pagny	Jean-Louis Aubert
Jean-Jacques Goldman	Marc Lavoine
Laurent Voulzy	Zazie
Johnny Hallyday	Pascal Obispo
Hélène Segara	Francis Cabrel
Paricia Kaas	Nathalie Imbruglia
Kylie Minogue	Shakira
Céline Dion	Mylène Farmer
Twain Shaniah	Alanis Morissette
Patrick Bruel	Robbie Williams
Axelle Red	Lara Fabian
Gérald De Palmas	

Chacun de ces individus est décrit par 2 variables quantitatives classiques, 2 variables intervalles, 4 variables qualitatives classiques et 6 variables modales. Une description de ces différentes variables est donnée dans le tableau 8.17.

Comme pour le jeu de données précédent, nous ne connaissons rien sur une éventuelle classification des données.

Remarque:

Le jeu de données original comportait 17 variables, cependant certaines de ces variables étaient redondantes entres elles. Nous avons donc supprimé 3 variables du jeu de données original avant de commencer notre analyse.

Nom de la variable	Type de variable	Description	Nombre de modalités
date_sortie	modale	dates de sortie des chansons de l'artiste	40
genre	modale	genres des chansons de l'artiste	7
recomp_chanson	intervalle	récompenses obtenues par les chansons de l'artiste	-
duree	modale	durées des chansons de l'artiste	40
date_naissance	qualitative	date de naissance de l'artiste	33
region	qualitative	région de l'artiste	25
pays	qualitative	pays d'origine de l'artiste	9
editeur	modale	éditeurs de l'artiste	14
distributeur	modale	distributeur de l'artiste	7
vente	modale	succès des chansons de l'artiste	4
année	intervalle	année des chansons de l'artiste	-
nationalite_1	qualitative	nationalité de l'artiste	9
recomp_chanteur_1	quantitative	nombre de récompenses obtenues par l'artiste	-
nb_album	quantitative	nombre d'albums de l'artiste	-

TAB. 8.17 – *Variables du jeu de données sur la musique.*

8.6.2 Distance de Hausdorff et de De Carvalho.

Les résultats fournis par le module NBCLUST sont présentés dans le tableau 8.19 et sont résumés dans le tableau ci dessous :

	C-H	C	Γ
SCLUST	5	6	3

TAB. 8.18 – *Résultats obtenus avec la distance de Hausdorff et de De Carvalho.*

On remarque un maximum dans la colonne de l'indice C-H pour $k = 5$, alors que pour le C-index, la valeur la plus proche de 0 est atteinte quand $k = 6$.
En ce qui concerne l'indice Gamma, la valeur maximale est atteinte en k=3, néanmoins cette valeur n'est pas proche de 1 ce qui indique la mauvaise qualité de l'indice.

Nous ne pouvons donc pas conclure, en vue de ces seuls indices, qu'un nombre de classes est préférable à un autre.

SCLUST	C-H	C	Γ
k=10	4.96893	0.02882	0.56061
k=9	3.50390	0.06332	0.46806
k=8	9.97788	0.06759	0.45638
k=7	10.69747	0.04309	0.45268
k=6	14.87098	**0.04196**	0.48826
k=5	**17.31893**	0.04875	0.59362
k=4	13.63863	0.05137	0.67948
k=3	10.93413	0.10101	**0.69963**
k=2	7.00281	0.39328	0.61613
k=1	-	-	-

TAB. 8.19 – *Valeurs des indices de Milligan et Cooper pour la méthode Sclust.*

8.6.3 Distance L1.

Les résultats obtenus en utilisant la distance L1 sont résumés ci-dessous :

	C-H	C	Γ
SCLUST	4	9	5

TAB. 8.20 – *Résultats obtenus avec la distance L1.*

SCLUST	C-H	C	Γ
k=10	6.64215	0.02667	0.59377
k=9	8.20099	**0.02206**	0.70343
k=8	9.31212	0.02904	0.73009
k=7	15.62529	0.03294	0.71024
k=6	17.84140	0.02963	0.69745
k=5	14.52756	0.03236	**0.75769**
k=4	**21.06847**	0.08011	0.65057
k=3	11.02059	0.15976	0.62463
k=2	6.50409	0.40345	0.50358
k=1	-	-	-

TAB. 8.21 – *Valeurs des indices de Milligan et Cooper pour la méthode Sclust.*

L'indice de Calinski et Harabasz suggère une partition en 4 classes alors que le C-index favorise une partition en 9 classes et que l'indice Γ atteint son maximum en $k = 5$.

De nouveau, on remarque que les indices de Milligan et Cooper n'indiquent pas, de manière significative, qu'un certain nombre de classes est préférable.

8.6.4 Distance L2.

Les méthodes de détermination du nombre de classes de Milligan et Cooper nous donnent les résultats suivant :

	C-H	C	Γ
SCLUST	8	2	2

TAB. 8.22 – *Résultats obtenus avec la distance L2.*

SCLUST	C-H	C	Γ
k=10	2083.54474	0.00015	0.75172
k=9	1479.51475	0.00037	0.72302
k=8	**2284.30410**	0.00018	0.75814
k=7	1870.73213	0.00034	0.74931
k=6	1895.15286	0.00043	0.77818
k=5	1989.91216	0.00067	0.86386
k=4	1173.38009	0.00331	0.80460
k=3	684.55858	0.00432	0.97045
k=2	264.15292	**0.00000**	**1.00000**
k=1	-	-	-

TAB. 8.23 – *Valeurs des indices de Milligan et Cooper pour la méthode Sclust.*

Si on utilise la distance L2, le maximum de la colonne de l'indice C-H est atteint pour $k = 8$.
Par contre, le C-index et de l'indice Gamma donnent des valeurs "parfaites" lorsque l'on partitionne les données en 2 classes. En effet, la valeur du C-index vaut 0 et celle de l'indice Gamma vaut 1.

En vue de ces valeurs, nous allons nous attarder un peu sur le résultat obtenu par Sclust lors d'un partitionnement en 2 classes.

En consultant le fichier listing, on voit que la première classe est composée d'un seul élément : Johnny Hallyday.
La deuxième classe est donc composée des 32 autres chanteurs de notre jeu de données.

Le fait que Johnny Hallyday constitue un chanteur à part peut être expliqué par le fait que la variable `nb_album` vaut 58!
En effet, le deuxième chanteur ayant le plus d'albums est Jean-Jacques Goldman avec 20 albums. L'isolation de Johnny Hallyday est donc principalement due au fait qu'il compte beaucoup plus d'albums que les autres chanteurs.

8.6.5 Analyse complémentaire.

En analysant le fichier listing, on remarque, de manière générale, que seulement 4 variables contribuent de façon significative à l'inertie intra-classes de la partition. Il s'agit des variables `recomp_chanson`, `année`, `recomp_chanteur_1` et `nb_albums`.

Cela renforce le fait que Johny Hallyday forme souvent une classe à lui tout seul puisqu'il se détache des autres individus par son nombre d'albums.
Lorsque l'individu Johnny Hallyday n'est pas seul dans une classe, il est souvent accompagné de Jean-Jacques Goldman. En effet, ces deux individus ont des valeurs de variables très semblables.

Nous avons aussi remarqué que Alanis Morisette était souvent l'unique individu d'une classe. Cela est dû à la valeur prise par la variable `année` qui est fort différente de celle des autres individus.

Voyons à présent les résultats obtenus lorsqu'on active la normalisation et qu'on utilise les distances de Hausdorff et de De Carvalho :

	C-H	C	Γ
SCLUST	2	?	?

TAB. 8.24 – *Résultats obtenus avec la distance de Hausdorff et de De Carvalho.*

SCLUST	C-H	C	Γ
k=9	2.06715	0.14798	0.61010
k=8	2.57534	0.11004	0.59801
k=7	2.07313	0.18953	0.58668
k=6	2.37428	0.19768	0.54634
k=5	2.01224	0.28089	0.51183
k=4	2.54105	0.26474	0.51033
k=3	2.90174	0.25935	0.51809
k=2	**3.18099**	0.37776	0.45648
k=1	-	-	-

TAB. 8.25 – *Valeurs des indices de Milligan et Cooper pour la méthode Sclust.*

On remarque, dans le tableau 8.25, un maximum pour l'indice de Calinski et Harabasz pour $k = 2$. Par contre, les indices C et Γ ne donnent pas des résultats significatifs.
Lors d'un partitionnement en 2 classes, Sclust regroupe dans la première classe les chanteurs francophones et dans la seconde classe les chanteurs anglophones.

8.6.6 La méthode DIV.

Pour ce jeu de données, le module NBCLUST n'a pas permis de trancher en faveur d'un nombre de classes. Cependant, à plusieurs reprises, le module a suggéré un partitionnement en 2 classes.
C'est pourquoi nous avons décidé, en premier lieu, d'analyser les résultats obtenus par la méthode DIV lorsque le nombre de classes est fixé à 2.
Nous analyserons ensuite les résultats obtenus lors d'une classification en 5 classes.

Pour rappel, seuls les variables quantitatives classiques et intervalles peuvent être sélectionnées pour la classification.

Voici les résultats obtenus lors d'un partitionnement en 2 classes :

```
THE CLUSTERING TREE :
---------------------
   - the number noted at each node indicates
     the order of the division
   - Ng <-> yes and Nd <-> no

      +----  Classe 1 (Ng=32)
      !
 !----1- [nb_albums <= 39.000000]
      !
      +----  Classe 2 (Nd=1)
```

Donc,

- les éléments de la classe 1 sont ceux pour lesquels :
  - la variable `nb_albums` est inférieure ou égale à 39;
- les éléments de la classe 2 sont ceux pour lesquels :
  - la variable `nb_album` est supérieure à 39.

Comme on pouvait s'y attendre, l'unique individu se trouvant dans la classe 2 est Johnny Hallyday.
La classe 1 est donc composée de tous les autres chanteurs.

On remarque que la variable qui "décide" de l'appartenance à un groupe est `nb_albums` ce qui vient confirmer le fait que cette variable contribue fortement à l'inertie intra-classes de la partition (voir section précédente).

A présent, voici les résultats obtenus par la méthode DIV lorsqu'on demande une partition en 5 classes :

```
THE CLUSTERING TREE :
---------------------
   - the number noted at each node indicates
     the order of the division
   - Ng <-> yes and Nd <-> no

                +----  Classe 1 (Ng=1)
                !
           !----3- [annee <= 1985.500000]
           !    !
           !    !    +----  Classe 4 (Ng=6)
           !    !    !
           !    !----4- [annee <= 1997.500000]
           !         !
           !         +----  Classe 5 (Nd=13)
           !
      !----2- [nb_albums <= 8.000000]
      !    !
      !    +----  Classe 3 (Nd=12)
      !
 !----1- [nb_albums <= 39.000000]
      !
      +----  Classe 2 (Nd=1)
```

Ici,

- les éléments de la classe 1 sont ceux pour lesquels :
  - la variable `nb_albums` est inférieure ou égale à 8;
  - la variable `année` est inférieure ou égale à 1985;
- les éléments de la classe 2 sont ceux pour lesquels :
  - la variable `nb_albums` est supérieure à 39;
- les éléments de la classe 3 sont ceux pour lesquels :
  - la variable `nb_albums` est comprise entre 8 et 39;
- les éléments de la classe 4 sont ceux pour lesquels :
  - la variable `nb_album` est inférieure à 8;
  - la variable `année` est comprise entre 1985 et 1997;
- les éléments de la classe 5 sont ceux pour lesquels :
  - la variable `nb_albums` est inférieure à 8;
  - la variable `année` est supérieure à 1997.

On voit que les 2 variables qui influencent la classification sont `nb_albums` et `année`.

La partition générée est la suivante :

classe 1 : Alanis Morisette;

classe 2 : Johnny Hallyday;

classe 3 : Madonna, Janet Jackson, Mariah Carey, Whitney Houston, Florent Pagny, Jean-Jacques Goldman, Francis Cabrel, Paricia Kaas, Kylie Minogue, Céline Dion, Mylène Farmer, Patrick Bruel;

classe 4 : Eminem, Britney Spears, Lenny Kravitz, Craig David, Marc Lavoine, Zazie;

classe 5 : Mickael Jackson, Lauryn Hill, Jean-Louis Aubert, Laurent Voulzy, Pascal Obispo, Hélène Segara, Nathalie Imbruglia, Shakira, Twain Shaniah, Robbie Williams, Axelle Red, Lara Fabian, Gérald De Palmas.

Ces résultats confirment ce que nous avons affirmé lors de la section précédente, à savoir que Johnny Hallyday est isolé à cause de sa grande valeur pour le variable `nb_albums` et que Alanis Morisette est isolée à cause de se petite valeur pour la variable `année`.

8.6.7 Conclusion.

En vue des résultats obtenus grâce aux indices de détermination du nombre de classes de Milligan et Cooper, nous ne pouvons pas affirmer qu'il existe une réelle structure classificatoire dans les données.

Nonobstant, les résultats obtenus à l'aide de la distance L2 ont permis de mettre en évidence le fait que certains chanteurs "sortaient du lot", le plus évident d'entre eux étant Johnny Hallyday.
Nous avons également remarqué que Alanis Morissette était régulièrement l'unique individu d'une classe.
Ces observations ont été confirmées par la méthode DIV.

Lorsque la normalisation des variables est activée, Sclust crée 2 classes : l'une composée des chanteurs francophones et l'autre composée des chanteurs anglophones.

Nous pouvons conclure cette analyse en affirmant qu'il n'y pas de structure classificatoire nette dans ce jeu de données.

8.7 Données sur le vin.[7]

8.7.1 Présentation des données.

Ce jeu de données est composé de 23 vins qui sont décrits par 21 variables intervalles, 1 variable modales, 2 variables qualitatives classiques et 2 variables quantitatives classiques.

Voici la liste des différents vins :

Ausone	Cheval Blanc	Cos d'Estournel
Ducru-Beaucaillou	Haut-Brion	L'Evangile
Lafite-Rothschild	Lafleur	Latour
Léoville Las Cases	Lynch-Bages	Margaux
Mission Haut-Brion	Montrose	Mouton-Rothschild
Petit Village	Petrus	Pichon C. de Lalande
Pichon Longueville	Sassicaia	Sociando Mallet
Trotanoy	Vieux Château Certan.	

Les 21 variables intervalles correspondent à des appréciations (en pourcentage) de la qualité de chaque vins sur différentes années, attribuées par 21 goûteurs,
La variable modale indique les différents cépages (cabernet franc, merlot, cabernet sauvignon et petit verdot) contenus dans le vin. Elle possède donc 4 modalités.
Les variables qualitatives classiques décrivent :

- la *région* de provenance du vin (4 modalités) : Monde, Graves, Médoc et Libournais et
- l'*appelation* du vin (9 modalités) : Etranger, Graves, Haut Médoc, Margaux, Paulliac, Pomerol, Saint-Emilion, Saint-Estèphe et Saint-Julien.

Les variables quantitatives classiques, quant à elles, décrivent :

- la *superficie* des terres de provenance du vin et
- le *nombre de caisses* de vin fabriquées pour les différentes années.

Nous signalons la présence de données manquantes dans le jeu de données. En effet, les valeurs des variables *cépages, superficie et nombre de caisses* pour l'individu **Sassicaia** ne sont pas connues.

Notons également que nous ne connaissons rien sur une éventuelle classification des données.

8.7.2 Distance de Hausdorff et de De Carvalho.

Si nous utilisons les distances de Hausdorff et de De Carvalho, les indices de détermination du nombre de classes de Milligan et Cooper nous donnent les résultats suivants :

	C-H	C	Γ
SCLUST	7	7	2

TAB. 8.26 – *Résultats obtenus avec les distances de Hausdorff et de De Carvalho.*

SCLUST	C-H	C	Γ
k=9	5.90721	0.10768	0.79725
k=8	7.22341	0.08319	0.85755
k=7	**256.11884**	**0.00257**	0.86090
k=6	138.97122	0.00696	0.85892
k=5	14.09711	0.08097	0.86872
k=4	13.05199	0.13629	0.81202
k=3	18.41182	0.12390	0.81185
k=2	**63.91472**	0.05016	**0.88726**
k=1	-	-	-

TAB. 8.27 – *Valeurs des indices de Milligan et Cooper pour la méthode Sclust.*

En consultant le tableau 8.27, on peut voir que le maximum pour l'indice de Calinski et Harabasz est atteint en $k = 7$ et que la valeur la plus proche de 0 pour le C-index est obtenue lors d'un partitionnement en 7 classes.
L'indice Gamma, quant à lui, suggère une partition de 2 classes.
Lorsqu'on examine plus attentivement ces résultats, 2 possibilités viennent à nous :

- un partitionnement en 2 classes et
- un partitionnement en 7 classes.

En effet, même si le maximum global de l'indice C-H est atteint lorsque $k = 7$, on remarque tout de même la présence d'un maximum local en $k = 2$. De plus la valeur du C-index pour $k = 2$ est fort proche de 0.

D'un autre côté, même si la valeur optimale de l'indice Gamma est obtenue lors d'une partition en 2 classes, la valeur de cet indice pour $k = 7$ est encore acceptable.

Nous présentons ensuite les résultats obtenus par le programme Sclust pour des partitions en 2 et 7 classes :

Partition en 2 classes :

classe 1 : Cos d'Estournel, Ducru-Beaucaillou, Lafite-Rothschild, Léoville Las Cases, Lynch-Bages, Margaux, Montrose, Mouton-Rothschild, Pichon C. de Lalande;

classe 2 : Ausone, Cheval Blanc, Haut-Brion, L'Evangile, Lafleur, Latour, Mission Haut-Brion, Petit Village, Petrus, Pichon Longueville, Sassicaia, Sociando Mallet, Trotanoy, Vieux Château Certan.

Partition en 7 classes :

classe 1 : Latour, Pichon Longueville, Sociando Mallet;
classe 2 : Lafite-Rothschild, Lynch-Bages;
classe 3 : L'Evangile, Mission Haut-Brion, Petit Village, Petrus, Vieux Château Certan;
classe 4 : Léoville Las Cases, Margaux, Montrose, Pichon C. de Lalande;
classe 5 : Ausone, Lafleur, Sassicaia, Trotanoy;
classe 6 : Cheval Blanc, Haut-Brion;
classe 7 : Cos d'Estournel, Ducru-Beaucaillou, Mouton-Rothschild.

8.7.3 Distance L1.

Voici les résultats obtenus par les indices de Milligan et Cooper :

	C-H	C	Γ
SCLUST	7	7	2

TAB. 8.28 – *Résultats obtenus avec la distance L1.*

En regardant le tableau 8.29, on voit que les résultats sont similaires à ceux obtenus aves les distances de Hausdorff et de De Carvalho.
De nouveau, on a le choix entre un partitionnement en 2 classes ou un partitionnement en 7 classes.

Lors d'une partition en 2 classes, la répartition des différents vins est identiquement la même que celle présentée à la section précédente.

Par contre, lors d'une classification en 7 classes, il y a quelques changements. En effet, les classes 1, 2, 4, 6 et 7 restent inchangées alors que Sassicaia rejoint la classe 3 et que l'Evangile, Petit Village, Petrus et Vieux Château Certan sont affectés à la classe 5.

SCLUST	C-H	C	Γ
k=9	5.92380	0.14304	0.68889
k=8	122.17468	0.00444	0.73999
k=7	**124.92435**	**0.00347**	0.81282
k=6	7.59167	0.07816	0.74210
k=5	14.09671	0.08097	0.78997
k=4	42.93876	0.05024	0.71801
k=3	18.41138	0.12390	0.73729
k=2	**63.91185**	0.05016	**0.86690**
k=1	-	-	-

TAB. 8.29 – *Valeurs des indices de Milligan et Cooper pour la méthode Sclust.*

8.7.4 Distance L2.

	C-H	C	Γ
SCLUST	7	7	2

TAB. 8.30 – *Résultats obtenus avec la distance L2.*

SCLUST	C-H	C	Γ
k=9	4.09307	0.11419	0.69492
k=8	225.79706	0.00206	0.68981
k=7	**2230.56625**	**0.00023**	0.74972
k=6	7.94835	0.09027	0.78731
k=5	10.50272	0.07762	0.80801
k=4	5.86015	0.15547	0.76135
k=3	10.69021	0.15751	0.79036
k=2	**341.83529**	0.01788	**0.88790**
k=1	-	-	-

TAB. 8.31 – *Valeurs des indices de Milligan et Cooper pour la méthode Sclust.*

Comme dans les deux sections précédentes, le maximum global de l'indice C-H est atteint en $k = 7$ et un maximum local existe en $k = 2$.
Les indices Gamma et du C-index conduisent également aux mêmes interprétations.

En consultant le fichier listing de Sclust, on voit que la partition en 2 classes est toujours constituée des mêmes individus alors que la partition en 7 classes est différentes.

8.7.5 Analyse complémentaire.

Comme pour les autres exemples, nous allons, dans cette section, commencer par regarder la contribution de chaque variable à l'inertie intra-classes. Cela va nous permettre de détecter les variables qui influencent le plus la classification.

Ici, la variable `nb_caisse` explique, dans tous les cas, presque 100% de l'inertie intra-classes.
On peut donc dire que la classification est influencée uniquement par cette variable.

Voici les résultats obtenus lorsqu'on active la normalisation et qu'on utilise les distances de Hausdorff et de De Carvalho :

L'indice C-H suggère une partition en 2 classes alors que les 2 autres indices penchent en faveur d'une partition en 9 classes.
Notons que cette partition est différente de celle engendrée par Sclust lorsque la normalisation des variables était désactivée.

SCLUST	C-H	C	Γ
k=11	2.69652	0.10151	0.73990
k=10	2.68513	0.13257	0.73532
k=9	3.08881	**0.06964**	**0.77530**
k=8	3.10113	0.12527	0.71793
k=7	3.44844	0.10761	0.72403
k=6	3.13342	0.13025	0.69074
k=5	3.51329	0.16368	0.66498
k=4	4.06101	0.17021	0.62323
k=3	4.69024	0.20867	0.59229
k=2	**5.41475**	0.28089	0.44551
k=1	-	-	-

TAB. 8.32 – *Valeurs des indices de Milligan et Cooper pour la méthode Sclust.*

8.7.6 La méthode DIV.

La méthode DIV ne pouvant pas être appliquée lorsqu'il y a des valeurs manquantes dans les données, nous avons dû remplacer les valeurs des variables `superficie` et `nb_caisse` de l'individu `Sassicaia` par la moyenne de ces variables sur les autres individus.

A plusieurs reprises, les méthodes de détermination du nombre de classes de Milligan et Cooper ont suggéré un partitionnement en 2 classes, c'est pourquoi nous commençons par lancer la méthode DIV en fixant le nombre de classes à 2.

Voici les résultats obtenus :

```
THE CLUSTERING TREE :
---------------------
   - the number noted at each node indicates
     the order of the division
   - Ng <-> yes and Nd <-> no

      +----  Classe 1 (Ng=14)
      !
 !----1- [nb_caisses <= 17798.500000]
      !
      +----  Classe 2 (Nd=9)
```

Comme on pouvait s'y attendre, la variable qui "décide" de la partition est la variable `nb_caisse`. Les individus formant le premier groupe sont des "petits" producteurs de vins alors que ceux composant le deuxième groupe sont des "gros" pro-

ducteurs.

Lorsqu'il ne nous suggèrait pas une classification en 2 groupes, le module NBCLUST penchait le plus souvent en faveur d'un partitionnement en 7 classes.
Voici les résultats obtenus par la méthode DIV lorsqu'on demande 7 classes :

```
THE CLUSTERING TREE :
---------------------
   - the number noted at each node indicates
     the order of the division
   - Ng <-> yes and Nd <-> no

                +----  Classe 1 (Ng=4)
                !
             !----5- [nb_caisses <= 4125.000000]
             !    !
             !    +----  Classe 6 (Nd=4)
             !
         !----2- [nb_caisses <= 10000.000000]
         !    !
         !    +----  Classe 3 (Nd=6)
         !
  !----1- [nb_caisses <= 17798.500000]
         !
         !          +----  Classe 2 (Ng=3)
         !          !
         !    !----6- [nb_caisses <= 22500.000000]
         !    !    !
         !    !    +----  Classe 7 (Nd=1)
         !    !
         !----3- [nb_caisses <= 26500.000000]
              !
              !    +----  Classe 4 (Ng=3)
              !    !
              !----4- [nb_caisses <= 32500.000000]
                   !
                   +----  Classe 5 (Nd=2)
```

Dans ce cas,

- les éléments de la classe 1 sont ceux pour lesquels :
  - la variable `nb_caisse` est inférieure ou égale à 4125;
- les éléments de la classe 2 sont ceux pour lesquels :
  - la variable nb_caisse est comprise entre 17798 et 22500;

- les éléments de la classe 3 sont ceux pour lesquels :
  - la variable nb_caisse est comprise entre 10000 et 17798;
- les éléments de la classe 4 sont ceux pour lesquels :
  - la variable nb_caisse est comprise entre 26500 et 32500;
- les éléments de la classe 5 sont ceux pour lesquels :
  - la variable nb_caisse est supérieure à 32500;
- les éléments de la classe 6 sont ceux pour lesquels :
  - la variable nb_caisse comprise entre 4125 et 10000;
- les éléments de la classe 7 sont ceux pour lesquels :
  - la variable nb_caisse est comprise entre 22500 et 26500.

Cela nous donne la partition suivante :

Partition en 7 classes :

classe 1 : Ausone, Lafleur, Petrus, Trotanoy;
classe 2 : Cos d'Estournel, Ducru-Beaucaillou, Mouton-Rothschild;
classe 3 : Cheval Blanc, Haut-Brion, Latour, Pichon Longueville, Sassicaia, Sociando Mallet;
classe 4 : Léoville Las Cases, Montrose, Pichon C.de Lalande;
classe 5 : Lafite-Rothschild, Lynch-Bages;
classe 6 : L'Evangile, Mission Haut-Brion, Petit Village, Vieux Château Certan;
classe 7 : Margaux.

Lors d'un partitionnement en 7 classes, la méthode DIV divise simplement les données en fonction de la variable `nb_caisses`.
Remarquons aussi que l'individu `Sassicaia` pourrait être placé dans une autre classe puisqu'en réalité nous ne connaissons pas sa valeur pour la variable `nb_caisse`.

8.7.7 Conclusion.

Comme nous l'avons constaté, pour ce jeu de données, les indices de détermination du nombre de classes de Milligan et Cooper suggèrent un partitionnement en 2 ou 7 classes.

Si l'on choisit de diviser les données en 7 classes, les résultats obtenus lors de la classification dépendent des distances et de la méthode choisie (Sclust ou DIV). En effet, la répartition des individus ne sera pas la même avec la distance L1 qu'avec la distance L2,...

Par contre, si l'on choisit une partition en 2 classes, la répartition des individus est invariante par rapport aux distances et aux méthodes utilisées.

C'est cet argument qui va nous amener à opter pour un partitionnement en 2 classes plutôt qu'en 7 classes.

La séparation en 2 groupes est principalement expliquée par la variables `nb_caisses`. En effet, la première classe (voir 8.7.2) est composée des individus pour lesquels les valeurs de ces variables sont élevées. Alors que les individus de la seconde classe ont des valeurs de `nb_caisses` plus petites. Cette observation a également été confirmée par la méthode DIV.

Pour terminer, nous pouvons dire qu'une partition en 2 classes semble adaptée à notre jeu de données.
La première classe étant composée de "gros" producteurs de vins, la seconde étant composée de vins issus de plus petits vignobles.

Conclusion.

Le principal objectif de ce mémoire fut d'adapter les méthodes classiques de détermination du nombre de classes à des objets symboliques décrits par des combinaisons de variables quantitatives classiques, qualitatives classiques, intervalles, multivaluées et modales.

La première partie de ce mémoire fut consacrée à implémenter une procédure permettant de calculer une matrice de dissimilarités entre objets symboliques décrits par tous les types de variable en utilisant les distances de Haussdorff, de De Carvalho, L1 et L2.

Nous avons, dans la deuxième partie du mémoire, appliqué les méthodes de détermination du nombre de classes de Milligan et Cooper aux partitions générées par le programme de classification symbolique Sclust sur différents jeux de données. Ensuite, nous avons comparé les résultats obtenus par Sclust avec ceux obtenus par une autre méthode de classification symbolique : la méthode DIV.

En premier lieu, nous nous sommes intéressés à un jeu de données artificielles bien connu dans la littérature : les données de Ruspini.
Le but de cette expérience était de démontrer que les méthodes développées dans ce mémoire s'appliquaient aussi bien à des données symboliques qu'à des données classiques.
Les résultats furent concluants. En effet, chaque méthode de détermination du nombre de classes de Milligan et Cooper suggéra une partition en 4 classes ce qui correspond à la structure naturelle des données. De plus, Sclust engendra la bonne partition en 4 classes. Les résultats obtenus avec la méthode DIV furent identiques.

Dans un second temps, nous nous sommes interressés à un certain nombre de jeux de données réelles. Ici, les individus étaient décrits par des combinaisons de n'importe quels types de variables.

Notre premier jeu de données traitait des catégories de films. Les indices de détermination du nombre de classes étaient variables en fonction de la distance utilisée, cependant en comparant les résultats, nous sommes arrivés à la conclusion qu'une partition en 5 classes semblait acceptable.
Les partitions en 5 classes engendrées par Sclust et DIV étaient légèrement différentes.

Les données concernant les chanteurs n'ont pas fourni de bons résultats. En effet les indices de Milligan et Cooper ne permettaient pas de trancher en faveur d'un nombre de classes en particulier. Nous avons tout de même remarqué que certains individus étaient isolés.

Pour le troisième jeu de données, les méthodes de détermination du nombre de classes ont simplement permis de détecter deux groupes; le premier composé des grands producteurs de vins, le second de petits producteurs. La méthode DIV a permis de confirmer de cette observation.

A l'heure actuelle, le module NBCLUST permet de calculer les indices de Milligan et Cooper pour un ensemble d'individus décrits par n'importe quelle combinaison de variables.

Une perspective intéressante serait d'appliquer ce module à d'autres méthodes de classification symbolique que le programme Sclust.
On pourrait également comparer les résultats obtenus avec les distances disponibles dans NBCLUST et celles disponibles dans le module DISS[16].

Annexe A

Listing du fichier classes.cpp

```
//#include "stdafx.h"
#include <fstream.h>
#include <iostream.h>
#include <math.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include "milligan.h"
//#include <iostream>


/*=====================================================================*/
void codage_all(FILE *lis,scluster para,CompSymbMat *M,int vsel[],
double **x,double **d,double Pw[])
/*=====================================================================*/
/*                                                                     */
/*Objectif :                                                           */
/*                                                                     */
/*Fonction qui calcule matrice de données pour tous les types          */
/*de variables et qui calcule la matrice de distances associée         */
/*                                                                     */
/*=====================================================================*/

{
  int imax,kmax,nvar;
  int i,j,k,l,kk,mod_max,nmod,num,testmanq;
  double modalite,nb_mod,a,b;
  int i1;
  double tot,dd,ddt,total;
```

```
imax=para.nb_objects;
kmax=para.nb_class;
nvar=para.nb_variables;
mod_max=para.nb_mod;

// Initialisation
for (i=0;i<=imax;i++)
{
  for (j=0;j<=mod_max;j++)
  {
    x[i][j]=0.0;
  };
};


// Initialisation de la matrice de distances
for (i=1;i<=imax;i++)
{
  for (j=1;j<=imax;j++)
    d[i][j]=0.0;
};

for (i=1;i<=imax;i++) // Boucle sur les individus
{
  l=0;
  for (j=0;j<nvar;j++) // Boucle sur les variables
  {
    num=vsel[j];
    if (( M->index(i,num).na() || M->index(i,num).nu() ) &&
    (M->gettypevar(num) == continu || M->gettypevar(num) == intercont ||
    M->gettypevar(num) == nominal))
    {
      testmanq=1;
      if(M->gettypevar(num) == continu || M->gettypevar(num) == nominal)
        l++;
      else
        if(M->gettypevar(num) == intercont)
          l=l+2;
        else
          l=l+M->getnbcateg(num);
    }
    else
    {
      testmanq=0;
```

```
switch(M->gettypevar(num))
{
case continu:
  {
    l++;x[i][l]=(float)M->index(i,num).getfval();
  };
  break;
case intercont:
  {
    a=M->index(i,num).getinter()->getmin();
    b=M->index(i,num).getinter()->getmax();
    l++;x[i][l]=a;
    l++;x[i][l]=b;
  };
  break;
case nominal:
  {
    nmod=M->getnbcateg(num);
    for (kk=1;kk<=nmod;kk++)
    {
      l++;
      if(kk==M->index(i,num).getcateg())
        x[i][l]=1.;
      else
        x[i][l]=0.;
    };
  };
  break;
case multnomin:
  {
    nmod=M->getnbcateg(num);
    nb_mod=0;
    for(k=1;k<=nmod;k++)
    {
      modalite=M->index(i,num).getnomin()->index(k);
      nb_mod=nb_mod+modalite;
    };
    for(k=1;k<=nmod;k++)
    {
      modalite=M->index(i,num).getnomin()->index(k);
      l++;
      x[i][l]=modalite/nb_mod;
    };
  };
```

```
        break;
      case multnominm:
        {
          nmod=M->getnbcateg(num);
          for (k=1;k<=nmod;k++)
          {
            l++;
            total=((dim_Mat*)M)->fuzzycard(i,num);
            if( total == 0)
              x[i][l]=M->index(i,num).getnominM()->indexm(k);
            else
              x[i][l]=M->index(i,num).getnominM()->indexm(k)/total;
          };
        };
        break;
      };
    };
  };// Fin de la boucle sur les variables
}; // Fin de la boucle sur les individus

if(para.debug < 1)
{ fprintf(lis," Matrice des données \n");
  for (i=1;i<=imax;i++)
  {
    fprintf(lis," %d ",i);
    for (j=1;j<=mod_max;j++)
      fprintf (lis," %f ",x[i][j]);
    fprintf(lis," \n ");
  };
};

// Calcul de la matrice de distances
for (i=1;i<=imax;i++) // Boucle sur les individus
{
  i1=i-1;
  for (j=1;j<=i1;j++) // Boucle sur les individus
  {
    kk=0; //variable d'increment pour faire avancer l'indice dans la matrice x
    tot=0.0;
    ddt=0.0;
    for(l=0;l<nvar;l++) // Boucle sur les variables
    {
      dd=0.0;
      num=vsel[l];
```

```
switch(M->gettypevar(num))
{
  case continu :
    {
      switch(para.distance_quant)
      {
        case 0://distance de Hausdoff
          kk++;
          dd = Pw[l+1]*maxdou(fabs(x[j][kk] - x[i][kk]),
            fabs(x[j][kk] - x[i][kk]));
          //les valeurs sont les memes donc ça revient a la distance L1
        break;
        case 1://distance L1
          kk++;
          dd = Pw[l+1]*(fabs(x[j][kk] - x[i][kk]));
        break;
        case 2://distance L2
          kk++;
          dd =Pw[l+1]* ((x[j][kk]-x[i][kk])*(x[j][kk]-x[i][kk]));
        break;
        default:
          kk++;
          dd = Pw[l+1]*maxdou(fabs(x[j][kk] - x[i][kk]),
            fabs(x[j][kk] - x[i][kk]));
        break;
      };//fin switch type de distance
    break;
  };//fin switch type de donnée continu
  case intercont :
    {
      switch(para.distance_quant)
      {
        case 0://distance de Hausdoff
          kk++;
          dd = Pw[l+1]*maxdou(fabs(x[j][kk] - x[i][kk]),
            fabs(x[j][kk+1] - x[i][kk+1]));
          kk++;
        break;
        case 1://distance L1
          kk++;
          dd = Pw[l+1]*(fabs(x[j][kk] - x[i][kk])+
            fabs(x[j][kk+1] - x[i][kk+1]));
          kk++;
```

```
        break;
        case 2://distance L2
          kk++;
          dd =Pw[l+1]* ((x[j][kk]-x[i][kk])*(x[j][kk]-x[i][kk]))+
          ((x[j][kk+1]-x[i][kk+1])*(x[j][kk+1]-x[i][kk+1]));
          kk++;
        break;
        default:
          kk++;
          dd = Pw[l+1]*maxdou(fabs(x[j][kk] - x[i][kk]),
            fabs(x[j][kk+1] - x[i][kk+1]));
          kk++;
        break;
      };//fin switch type de distance
    break;
  };//fin switch type de donnée intercont
  case nominal :
    {
      nmod=M->getnbcateg(num);
      for (k=1;k<=nmod;k++)
      {
        kk++;
        switch(para.distance_qual)
        {
          case 0:
            if ((int)100.0*x[i][kk]==0.) dd=dd+Pw[l+1]*x[j][kk];
            if ((int)100.0*x[j][kk]==0.) dd=dd+Pw[l+1]*x[i][kk];
          break;
          case 1:
            dd=dd+Pw[l+1]*fabs(x[j][kk] - x[i][kk]);
          break;
          case 2:
            dd=dd+Pw[l+1]*(x[j][kk] - x[i][kk])*(x[j][kk] - x[i][kk]);
          break;
          default :
            if ((int)100.0*x[i][kk]==0.) dd=dd+Pw[l+1]*x[j][kk];
            if ((int)100.0*x[j][kk]==0.) dd=dd+Pw[l+1]*x[i][kk];
          break;
        };//fin switch type distance
      };//fin boucle nbre de modalités
    break;
  };//fin type variable nominal
  case multnomin :
  case multnominm :
```

```
            {
              nmod=M->getnbcateg(num);
              for (k=1;k<=nmod;k++)
              {
                kk++;
                switch(para.distance_modal)
                {
                  case 0:
                    if ((int)100.0*x[i][kk]==0.) dd=dd+Pw[l+1]*x[j][kk];
                    if ((int)100.0*x[j][kk]==0.) dd=dd+Pw[l+1]*x[i][kk];
                  break;
                  case 1:
                    dd=dd+Pw[l+1]*fabs(x[j][kk] - x[i][kk]);
                  break;
                  case 2:
                    dd=dd+Pw[l+1]*(x[j][kk] - x[i][kk])*(x[j][kk] - x[i][kk]);
                  break;
                  default :
                    if ((int)100.0*x[i][kk]==0.) dd=dd+Pw[l+1]*x[j][kk];
                    if ((int)100.0*x[j][kk]==0.) dd=dd+Pw[l+1]*x[i][kk];
                  break;
                };//fin switch type distance
              };//fin boucle nbre de modalités
            break;
          };//fin type variable modale et multivaluée
        };//fin switch type de variable
        ddt=ddt+ (dd*dd);
      }; // Fin de la boucle sur les variables
      d[i][j]=ddt;
      d[j][i]=d[i][j];
    };// Fin de la boucle sur les individus j
  };  // Fin de la boucle sur les individus i

  if(para.debug < 1)
  {  fprintf(lis,"\n MATRICE de dissimilarité \n");
     for (i=1;i<=imax;i++)
     {
       fprintf(lis," %d ",i);
       for (j=1;j<=imax;j++)
       fprintf (lis," %f ",d[i][j]);
       fprintf(lis," \n ");
     };
  };
};//fin codage_all
```

```
/* ================================================================== */
void milligan_scluster_m(FILE *lis,scluster para,double **d,int **t_classe)
/* ================================================================== */
/*                                                                    */
/* Objectif :                                                         */
/*                                                                    */
/* Calcul des indices des méthodes de détermination du nombre         */
/* de classes de Milligan et Cooper appliquées aux partitions         */
/* générées par SCLUST                                                */
/*                                                                    */
/* ================================================================== */
{

  // Déclaration et initialisation des paramètres de la structure
  int imax,nvar,kmax;
  imax=para.nb_objects;
  nvar=para.nb_variables;
  kmax=para.nb_class;

  int i,j,k; // Variables locales, indices de boucles
  int ii,jj;
  int i1;

  double **BW; // Matrice liée au calcul de l'indice Gamma
    BW= new double* [imax+1];
    for (i=0;i<=imax;i++)
    BW[i]= new double [imax+1];

  double ttot=0.; // Inertie totale
  double *DLO=new double[imax*imax/2]; // Vecteur de distances
  int NC2;
  double ch=0.; // Indice de Calinski et Harabasz
  int *P=new int[imax+1]; // Vecteur tq P[i]=classe de l'individu i
  int numvar=1; // Numéro de la variable symbolique considérée

  int nbr=0; // Nombre de classes considérées

  int *Nbr_obj=new int[kmax+1];
  // Vecteur tq Nbr_obj[k]=nombre d'individus de la classe k

  double *ssq=new double[kmax+1];
  // Vecteur tq ssq[k]=inertie intra-classes de la classe k
```

```
// Variables liées au calcul de l'indice Gamma
int kin=0,kout=0;
double *BB=new double[imax*imax/2];
double *W=new double[imax*imax/2];
double u=0.0,dd=0.0;
int kk=0,ll=0;
double din=0.0;
double denmr=0.0;
double gamma=0.0; // Indice Gamma

// Variables liées au calcul du C-index
int *Nbr_comb=new int[kmax];
double c_min,c_max,c_index;
double total;
double intra;
double vec=0.;

fprintf(lis,"\n========================================================== \n");
fprintf(lis,"MILLIGAN et COOPER sur les partitions générées par Sclust \n");
fprintf(lis,"========================================================== \n \n");

// Initialisation de BW
for (i=2;i<=imax;i++)
{
  i1=i-1;
  for (j=1;j<=i1;j++)
    BW[i][j]=0.0;
}

ttot=0.0;
k=0;

// Calcul de ttot
for (i=1;i<=imax-1;i++)
{
  for (j=i+1;j<=imax;j++)
  {
    k=k+1;
    DLO[k]=d[j][i];
    ttot=ttot+d[j][i];
  }
};
```

```
ttot=ttot/double(imax); // ttot=Inertie TOTALE

NC2=imax*(imax-1)/2;

// Tri par ordre croissant des distances (de la plus petite à la plus grande)
for (i=1;i<=NC2-1;i++)
{
  vec=DLO[i];
  for (j=i+1;j<=NC2;j++)
  {
    if (vec-DLO[j] > 0)
    {
      DLO[i]=DLO[j];
      DLO[j]=vec;
      vec=DLO[i];
    };
  };
};

k=0;
ch=0.;

nbr=kmax;

fprintf(lis,"Ngps       C-H        C-index      Gamma\n \n");

for (nbr=kmax;nbr>=1;nbr--) //Boucle sur le nombre de classes (on commence à kmax)
{
  ssq[0]=0; // Initialisation inertie intra-classes
  ch=0.; // Initialisation indice C-H

  // Initialisation
  for(i=1;i<=imax;i++) P[i]=0;
  for(k=1;k<=kmax;k++) Nbr_obj[k]=0;
  for(k=1;k<=kmax;k++) Nbr_comb[k]=0;

  for(i=1;i<=imax;i++)
  {
    // P[i] contient la classe d'affectation de l'individu i
    P[i]=t_classe[i][nbr];
    // Nbr_obj[P[i]] contient le nombre d'objets présents dans la classe P[i].
    Nbr_obj[P[i]]+=1;
  };
```

```
for (k=1;k<=nbr;k++) ssq[k]=0.;

// Calcul de l'inertie intra-classes
for (k=1;k<=nbr;k++)
{
  for (i=1;i<=imax-1;i++)
  {
    if (P[i]==k)
    {
      for (j=i+1;j<=imax;j++)
      {
        if (P[j]==k)
        {
          ssq[k]+=d[i][j];
        };
      };
    };
  };
};

intra=0.;
for (k=1;k<=nbr;k++)
  intra+=ssq[k];

// Normalisation
for (k=1;k<=nbr;k++)
{
  ssq[k]=ssq[k]/double(Nbr_obj[k]);
  if (Nbr_obj[k]==0)
  {
    ssq[k]=0;
  };
};

// ssq[0]=somme des inerties intra-classes
ssq[0]=0;
for (k=1;k<=nbr;k++)
  ssq[0]+=ssq[k];

// Mise à jour de la matrice BW
for (k=1;k<=nbr;k++)
{
  for (j=1;j<=imax;j++)
  {
```

```
    if (P[j]==k)
    {
      for (i=1;i<=imax;i++)
      {
        if (P[i]==k)
        {
          ii=maxint(i,j);
          jj=minint(i,j);
          BW[ii][jj]=1.0;
        };
      };
    };
  };
};

// Calcul de l'indice de C-H
if (nbr==1) ch=0.;
ch=(((ttot-ssq[0])*double(imax-nbr))/(ssq[0]*double(nbr-1)));

// Calcul de l'indice Gamma
kin=0;
kout=0;

for (i=2;i<=imax;i++)
{
  i1=i-1;
  for (j=1;j<=i1;j++)
  {
    if (BW[i][j]<=0.0)
    {
      kout=kout+1;
      BB[kout]=d[i][j];
    }
    else
    {
      kin=kin+1;
      W[kin]=d[i][j];
    };
  };
};

u=0.;
dd=0.;
```

```
    for (kk=1;kk<=kin;kk++)
    {
      din=W[kk];
      for (ll=1;ll<=kout;ll++)
      {
        if ((din-BB[ll]) ==0) dd=dd+0.5;
        if ((din-BB[ll]) > 0) u=u+1.;
      };
    };

    denmr=(double(kin)*double(kout)/2.)-dd;

    gamma=(1.-u/denmr);

    // Calcul du C-index
    for (i=1;i<=nbr;i++) Nbr_comb[i]=((Nbr_obj[i])*(Nbr_obj[i]-1)/2);

    Nbr_comb[0]=0;
    for (k=1;k<=nbr;k++) Nbr_comb[0]+=Nbr_comb[k];
    total=Nbr_comb[0];

    c_min=0.;
    c_max=0.;
    c_index=0.;

    for (j=1;j<=total;j++)
    {
      c_min=c_min+DLO[j];
      c_max=c_max+DLO[NC2-Nbr_comb[0]+j];
    };

    c_index=((intra-c_min)/(c_max-c_min));

    // Edition des résultats
    if (nbr!=1) fprintf(lis," %2d   %10.5f    %8.5f    %8.5f \n",nbr,ch,c_index,gamm
    else fprintf(lis," %2d         -             -            -\n\n",nbr);

  }; // Fin boucle sur le nombre de classes

  delete DLO;
  delete P;
  delete Nbr_obj;
  delete ssq;
};
```

```
/* ========================================================================== */
double maxdou(double a, double b)
/* ========================================================================== */
/*                                                                            */
/* Objectif :                                                                 */
/*                                                                            */
/* Déterminer le maximum entre 2 valeurs réelles                              */
/*                                                                            */
/* ========================================================================== */
{

  double m4;

  m4 = (a < b) ? b : a;

  return m4;

};
```

Bibliographie

[1] S. DELOGNE. *Méthodes de détermination du nombre de classes pour des objets symboliques.* Faculté Universitaire Notre-Dame de la Paix, Namur, 2002.

[2] S. COLLES. *Méthodes de détermination du nombre de classes pour des objets symboliques.* Faculté Universitaire Notre-Dame de la Paix, Namur, 2003.

[3] A.D. GORDON. *Classification.* Monographs on Statistics and Applied Probability, $2^{ième}$ edition, 1999.

[4] L. KAUFMAN and P.J. ROUSSEUW. *Finding groups in data : An Introduction to Cluster Analysis.* Wiley-Interscience, 1990.

[5] B. EVERITT. *Cluster Analysis.* Halsted Press, $2^{ième}$ edition, 1980.

[6] E. H. RUSPINI. A new approach to clustering. *Information and Control*, vol 15:22–32, 1979.

[7] Ecole SODAS. Cd rom. Porto, 2001.

[8] H.H. BOCK and E. DIDAY. *Analysis of symbolic data. Exploratory methods for extracting statistical information from complex data.* Springer-Verlag, 2000.

[9] T. CALINSKI and J. HARABASZ. A dendrite method for cluster analysis. *Communication in Statistics*, (3):1–27, 1974.

[10] E.M.L. BEALE. *Cluster analysis.* London : Scientific Control System, 1969.

[11] R.O. DUDA and P.E. HART. *Pattern classification and scene analysis.* Wiley-Interscience, New York, 1973.

[12] E. DIDAY. *Nouvelles méthodes et nouveaux concepts en classification automatique et reconnaissance des formes.* PhD thesis, Université de Paris VI, 1972.

[13] A.D. GORDON. How many clusters? an investigation of five procedures for detecting nested cluster structure. *Data Science, Classification, and Related Methods*, pages 109–116, 1998.

[14] G.W. MILLIGAN and M.C. COOPER. An examination of procedures for determining the number of clusters in data set. *Psychometrika*, 50(2):159–179, Juin 1985.

[15] A. HARDY. On the number of clusters. *Computational Statistics and Data Analysis*, (23):83–96, 1996.

[16] J. TROCLET. *Méthode de détermination du nombre de classes pour des objets symboliques.* Faculté Universitaire Notre-Dame de la Paix, Namur, 2004.

[17] M. CHAVENT. *Analyse de données symboliques. Une méthode divisive de classification.* PhD thesis, Université Paris IX - Dauphine, 1997.

[18] G. CELEUX, E. DIDAY, G. GOVAERT, Y. LECHEVALLIER, and H. RALAMBONDRAINY. *Classification automatique des données.* Dunod-Informatique, Bordas, 1989.

[19] L.J. HUBERT and J.R. LEVIN. A general statistical framework for assessing categorical clustering in free recall. *Psychological Bulletin*, 83(6):1072–1080, 1982.

[20] F.B. BAKER and L.J. HUBERT. Measuring the power of hierarchical cluster analysis. *Journal of the American Statistical Association*, 70:31–38, 1975.

[21] E.C. DALRYMPHE-ARFORD. Measurement of clustering in free recall. *Psychological Bulletin*, 74(1):32–34, 1970.