UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN MATHEMATIQUES ET INFORMATIQUE
APPLIQUEES

PAR
NDIOUGA GUEYE

Exploration des liens formels entre les méthodes statistiques et neuronales en classification

Novembre 2019

Résumé

Ce mémoire s'inscrit dans le cadre d'une étude comparative soulevant des liens entre les méthodes statistiques et neuronales en classification non supervisée. La classification est un outil d'analyse de données multidimensionnelles permettant de regrouper des individus en plusieurs classes homogènes. Plusieurs méthodes de classification existent dans la littérature, entre autres, des méthodes classiques et celles neuronales. Ces méthodes ont le même objectif, mais leurs approches diffèrent. Certaines méthodes exigent a priori une connaissance préalable du nombre de classes et d'autres ne l'exigent pas.

Ce document vise plus particulièrement une étude comparative entre, d'une part, la méthode ascendante hiérarchique, celle des k-means et les modèles gaussiens, et d'autre part la carte auto-organisatrice de Kohonen (SOM). En outre, nous proposons une comparaison de ces différentes méthodes sur la base des formes fortes (i.e. ensemble d'éléments classés ensemble à plusieurs reprises). En effet, Il ressort que la méthode neuronale est plus proche des méthodes à approche géométrique (méthode hiérarchique et k-means), où la présence des formes fortes est plus importante, que la méthode à approche probabiliste (modèle gaussien) où la présence des formes fortes l'est moins.

Avant-propos

À travers ce document, j'exprime ma gratitude à toutes les personnes qui ont contribué à l'accomplissement de ce travail.

Je tiens tout d'abord à remercier profondément ma directrice de recherche, professeure à l'UQTR, Madame Nadia Ghazzali pour l'aide scientifique qu'elle m'a apporté, son soutien financier, sa rigueur durant tout l'encadrement ainsi que ses encouragements.

Je tiens à remercier également ma famille pour leur accompagnement et leur soutien moral depuis le début de mes études.

Mes remerciements vont aussi à tous les professeurs de l'UQTR qui ont contribué à notre réussite et qui nous ont facilité l'intégration dans le système éducatif québécois. Parmi ces professeurs, je me permets de citer Monsieur Jean-François Quessy et Monsieur Ismaïl Biskri pour avoir accepté d'évaluer mon mémoire de maitrise.

Je ne pourrai terminer les remerciements sans penser à mes camarades, promotionnaires, amis et anciens collègues, qui sont Monsieur Babacar Ndoye, Monsieur El hadji Diaraff Diégane Diagne et Monsieur Mamadou Lamine Cissé. Nous avons affronté la maitrise ensemble en traversant plusieurs moments de pression et de sacrifices afin de maintenir le cap vers l'excellence.

Table des matières

Liste des tableaux

Table des figures

Introduction

L'analyse factorielle cherche à représenter graphiquement dans un espace de faible dimension, des informations qui ne peuvent être résumées compte tenu de la taille et de la complexité des données. Il est parfois nécessaire de compléter une analyse factorielle par une typologie en classes homogènes, telle que les individus d'une même classe se ressemblent le plus possible et ceux appartenant à des classes différentes se ressemblent le moins : c'est l'objet des méthodes de classification. Ces dernières cherchent à dégager les ressemblances entre les individus et à les classer selon ce qu'ils ont de commun.

Dans le but de contribuer à faire le lien entre les méthodes de classification classiques et neuronales, ce mémoire présente une analyse comparative entre ces deux catégories de méthodes de classification non supervisée sur un jeu de données réelles et ce, en utilisant le logiciel libre R. Concernant les méthodes statistiques, nous considérons d'une part les méthodes à approche géométrique s'apparentant à la classification ascendante hiérarchique (CAH) et celle des k-means. La première fournit des classes homogènes en une hiérarchie de partitions tandis que la deuxième est une méthode itérative d'optimisation où le nombre de classe est connu a priori et que dans chaque itération, deux étapes sont effectuées (recentrage et réallocation). D'autre part, nous considérons les méthodes à approche probabiliste basée sur les mélanges gaussiens. Cette approche suppose que les données constituent un échantillon indépendant tiré à partir d'un mélange de plusieurs distributions gaussiennes et, par conséquent, les individus de la même classe sont ceux qui appartiennent probablement à la même

distribution.

A ces méthodes statistiques, s'ajoute la méthode de Kohonen caractérisée par une carte auto-organisatrice connue sous le nom de *Self Organizing Map (SOM)* et basée sur les réseaux de neurones artificiels. L'historique de ces derniers a commencé en 1943 avec Warren McCulloch et Walter Pitts qui ont créé un modèle de réseaux de neurones basé sur des algorithmes afin de modéliser le fonctionnement du neurone biologique. C'est dans cette logique que Frank Rosenblat a créé, en 1958, le premier perceptron capable de reconnaitre une forme par le biais d'apprentissage supervisé. Cependant, les premiers pas de l'apprentissage non supervisé avec les réseaux de neurones artificiels, ont commencé avec le réseau de Teuvo Kohonen en 1982. Ce dernier est un type de réseau de neurones artificiels dont l'apprentissage se fait de manière non supervisée. Il est représenté par une carte composée d'un ensemble de neurones disposés sur une grille à faible dimension où chaque neurone correspond à un vecteur prototype qui se spécialise pour représenter une classe des données selon les caractéristiques qui les rassemblent.

Ce document est organisé en trois chapitres. Le premier présente les méthodes statistiques de classification en passant en revue les méthodes à approche géométrique (CAH et k-means) et probabiliste (modèle gaussien) ainsi que leur mise en application dans le logiciel R. Le deuxième chapitre est consacré aux méthodes de classification neuronales, plus particulièrement, la carte auto-organisatrice de Kohonen ainsi que son fonctionnement dans le logiciel R. Le troisième chapitre, quant à lui, concerne l'étude comparative des méthodes statistiques et neuronale en question tout en décrivant les données sur lesquelles seront appliquées les différentes méthodes de classification concernées et discute des résultats obtenus en utilisant le logiciel libre R ainsi que quelques perspectives.

Chapitre 1

Méthodes statistiques de classification

Ce chapitre présente un rappel des méthodes statistiques de classification ainsi que leur mise en œuvre sous le logiciel R.

Les méthodes de classification non supervisée ont un objectif exploratoire dans la mesure où elles sont à la découverte de partitions d'un ensemble d'individus (Observations décrites par les variables) en plusieurs classes. Il s'agit plus précisément de la recherche d'une typologie des individus en classes homogènes. Ceci est fait en optimisant un critère visant à regrouper les individus dans des classes selon ce qu'ils ont de commun, chacune la plus homogène possible et, entre elles, les plus dissemblant possible.

Cependant, la classification de variables peut être intéressante afin de réduire leur nombre en cas de redondances ou dans le cas où les individus sont décrits par un très grand nombre de variables. Dans ce document, nous nous limiterons à la classification des individus. Ainsi, pour faire une classification, il faut au préalable se donner un indice de similarité, de dissimilarité ou de distance afin de mesurer la ressemblance entre individus, et un critère d'agrégation afin de mesurer la proximité entre classes.

1.1 Rappels sur les mesures de ressemblance

1.1.1 Dissimilarité et distance

Une dissimilarité est une application d sur un ensemble $M = R^p$ vérifiant :

$$M \times M \to]0, \infty]$$

$$(x, y) \mapsto d(x, y)$$

$$avec \quad x = (x_1, ..., x_p) \quad et \quad y = (y_1, ..., y_p)$$

telle que :

i) pour tout $(x, y) \in M^2$, on a $d(x, y) = 0$ si, et seulement si, $x = y$,

ii) pour tout $(x, y) \in M^2$, on a $d(x, y) = d(y, x)$ (d est symétrique),

iii) pour tout $(x, y, z) \in M^3$, on a :

$$d(x, y) \leq d(x, z) + d(z, y) \text{ (Inégalité triangulaire)}$$

Une distance peut être définie à partir de la dissimilarité. Elle est une dissimilarité vérifiant la propriété de l'inégalité triangulaire.

Une distance entre deux individus permet de mesurer à quel point ces individus se ressemblent. Elle permet de décider si deux individus sont semblables pour être regroupés au sein d'une même classe.
Plusieurs exemples de distance sont énumérés ci-dessous, parmi lesquels, le plus couramment utilisé est celui de la distance euclidienne.

Exemple 1 : distance euclidienne

On appelle distance euclidienne entre x et y la distance :

$$d(x,y) = \sqrt{\sum_{i=1}^{p}(x_i - y_i)^2}$$

Le carré de la distance euclidienne est parfois utilisé pour mesurer la ressemblance entre deux individus. Mais, dans ce cas, il s'agit d'une dissimilarité plutôt que d'une distance car la propriété de l'inégalité triangulaire n'est pas vérifiée.

Exemple 2 : distance de Manhattan

On appelle distance de Manhattan entre x et$y$ la distance :

$$d(x,y) = \sum_{i=1}^{p} \mid x_i - y_i \mid$$

Exemple 3 : distance de Minkowski

On appelle distance de Minkowski entre x et y la distance :

$$d(x,y) = (\sum_{i=1}^{p} \mid x_i - y_i \mid^q)^{\frac{1}{q}}$$

Remarque : On retrouve les deux distances précédentes pour $q = 2$ et $q = 1$.

Exemple 4 : distance de Canberra

On appelle distance de Canberra entre x et y la distance :

$$d(x,y) = \sum_{i=1}^{p} \frac{\mid x_i - y_i \mid}{\mid x_i + y_i \mid}$$

Exemple 5 : distance maximum

On appelle distance maximum entre x et y la distance :

$$d(x,y) = \sup_{i \in \{1,\ldots,p\}} \mid x_i - y_i \mid$$

1.1.2 Similarité

Un indice s de similarité entre deux individus x et y sur un ensemble $M = R^p$ est une application de

$$M \times M \to [0, 1]$$
$$(x, y) \mapsto s(x, y)$$

telle que :

i) $s(x, y) \geq 0$

ii) $s(x, y) = s(y, x)$ (s est symétrique)

iii) $s(x, x) = s(y, y) = 1 \geq s(x, y)$

La similarité est généralement utilisée dans le cas des tableaux binaires. Dans le cas des tableaux non binaires, la similarité $s(x, y)$ est obtenue à partir de la dissimilarité $d(x, y)$ comme suit :

$$s(x, y) = \frac{1}{1 + d(x, y)}$$

Plusieurs méthodes de calculs d'indice de similarité ont été développées, entre autres, celles énumérées ci-dessous :

Par exemple, pour deux individus x et y caractérisés par une variable z prenant les valeurs 0 ou 1, considérons le tableau 1.1 :

- soit a le nombre de fois où $x_z = y_z = 1$,
- soit b le nombre de fois où $x_z = 0$ et $y_z = 1$,
- soit c le nombre de fois où $x_z = 1$ et $y_z = 0$,
- soit d le nombre de fois où $x_z = y_z = 0$.

	1	**0**	
1	a	c	a+c
0	b	d	b+d
	a+b	c+d	a+b+c+d

Tableau 1.1 - *Tableau occurrence/similarité*

Différentes mesures de similarité ont été proposées :

- Indice de Jaccard

$$s_1(x,y) = \frac{a}{a+b+c}$$

- Indice de Russel et Rao :

$$s_2(x,y) = \frac{a}{a+b+c+d}$$

- Indice de Dice

$$s_3(x,y) = \frac{2a}{2a+b+c}$$

- Indice de Sokal et Sneath

$$s_4(x,y) = \frac{a}{a+2b+2c}$$

- Indice de Sokal et Michener

$$s_5(x,y) = \frac{a+d}{a+b+c+d}$$

- Indice de Yule

$$s_6(x,y) = \frac{ad-bc}{ad+bc}$$

- Indice de Pearson

$$s_7(x,y) = \frac{ad-bc}{(a+d)(b+c)(a+c)(b+d)}$$

Exemple :

Soit deux individus x et $y \in M = R^5$ tels que :

$$x = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

$$y = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Le tableau d'occurrences donne :

	1	**0**	
1	a=1	c=2	a+c=3
0	b=2	d=0	b+d=2
	a+b=3	c+d=2	a+b+c+d=5

Tableau 1.2 - *Exemple de tableau occurrence/similarité*

Alors $s_1(x,y) = \frac{1}{1+2+2} = \frac{1}{5}$, $s_2(x,y) = \frac{1}{1+2+2+0} = \frac{1}{5}$,
$s_3(x,y) = \frac{2\times 1}{(2\times 1)+2+2+0} = \frac{1}{6}$ $s_4(x,y) = \frac{1}{1+(2\times 2)+(2\times 2)} = \frac{1}{9}$
$s_5(x,y) = \frac{1+0}{1+2+2+0} = \frac{1}{5}$ $s_6(x,y) = \frac{1\times 0-2\times 2}{1\times 0+2\times 2} = -1$
$s_7(x,y) = \frac{1\times 0-2\times 2}{(1+0)(2+2)(1+2)(2+0)} = -\frac{1}{9}$
Il y a plus de similarité entre x et y au sens de l'indice de Jaccard et celui de Russel et Rao.

1.1.3 Mesure de proximité entre classes

Pour un regroupement de plusieurs classes en une seule, il faut également se baser sur un indice de proximité défini sur l'ensemble des classes. Cet indice est appelé indice d'agrégation, il est construit à partir de la distance ou dissimilarité d ou de l'indice de similarité s.

a) Définition :

L'indice d'agrégation D entre deux classes A et B, défini à partir de la distance, de la

dissimilarité d ou de l'indice de similarité s, vérifie les propriétés suivantes :

(1) $D(A,B) = D(B,A)$ (Symétrie)

(2) $D(\{x\},\{y\}) = d(x,y) = s(x,y)$ (évaluer l'indice d'agrégation entre deux singletons revient à évaluer leur distance ou leur similarité.)

b) Exemples en termes de dissimilarité ou distance d :

- Méthode du plus proche voisin (Saut minimum) :

L'agrégation D_1 entre deux classes A et B est celle définie par les deux éléments les plus proches appartenant à ces classes.

$$D_1(A,B) = min\{d(x,y), x \in A \quad et \quad y \in B\} = max\{s(x,y), x \in A \quad et \quad y \in B\}$$

- Méthode du voisin le plus éloigné (Saut maximum) :

L'agrégation D_2 entre deux classes A et B est celle définie par les deux éléments les plus éloignées.

$$D_2(A,B) = max\{d(x,y), x \in A \quad et \quad y \in B\} = min\{s(x,y), x \in A \quad et \quad y \in B\}$$

- Méthode du saut moyen :

L'agrégation D_3 entre deux classes A et B est la moyenne des distances entre tous les individus de la classe A et tous les individus de classe B.

$$D_3(A,B) = \frac{\sum_{x \in A, y \in B} d(x,y)}{n_A n_B}$$

où n_A : effectif de la classe A et n_B : effectif de la classe B

- Méthode du centroide :

$$D_4(A,B) = d(\bar{x}_A, \bar{x}_B)$$

Avec $\bar{x}_A = \frac{1}{n_A}\sum_{i \in A} x_i$, $\bar{x}_B = \frac{1}{n_B}\sum_{i \in B} x_i$ et $\bar{x}_{AB} = \frac{n_A \bar{x}_A + n_B \bar{x}_B}{n_A + n_B}$

- **Méthode de la médiane** :

$$D_5(A, B) = d(\bar{x}_A, \bar{x}_B)$$

avec $\bar{x}_{AB} = \frac{\bar{x}_A + \bar{x}_B}{2}$

- **Méthode de Ward** :

Cette méthode est basée sur la perte d'inertie expliquée, résultant de l'agrégation des classes A et B. Elle impose l'utilisation du carré de la distance euclidienne et donnée par la formule :

$$D_6(A, B) = \frac{n_A n_B}{n_A + n_B} d^2(x_A, x_B)$$

1.2 Rappels sur les méthodes classiques de classification

A l'aide d'un calcul combinatoire, on peut montrer que le nombre de partitions possibles d'un ensemble de n éléments augmente considérablement voire exponentiellement avec n. Ainsi, pour un nombre n d'individus, l'arbitrage entre les différentes partitions possibles peut être difficile. Il ne sera donc pas question de chercher à optimiser le critère d'homogénéité sur toutes les partitions possibles. Pour pallier à cette difficulté, il s'agira de procéder étape par étape :

- soit en déterminant une série de partitions emboîtées représentée par un arbre.
- soit en améliorant une partition initiale à l'aide d'algorithmes itératifs qui convergent vers une classification optimale selon un critère à préciser. Dans ce cas, on se fixera a priori le nombre k de classes.

Pour cela, deux grandes méthodes de classification peuvent être définies : Les méthodes dites hiérarchiques et les méthodes non hiérarchiques ou de partitionnement.

1.2.1 Méthodes hiérarchiques

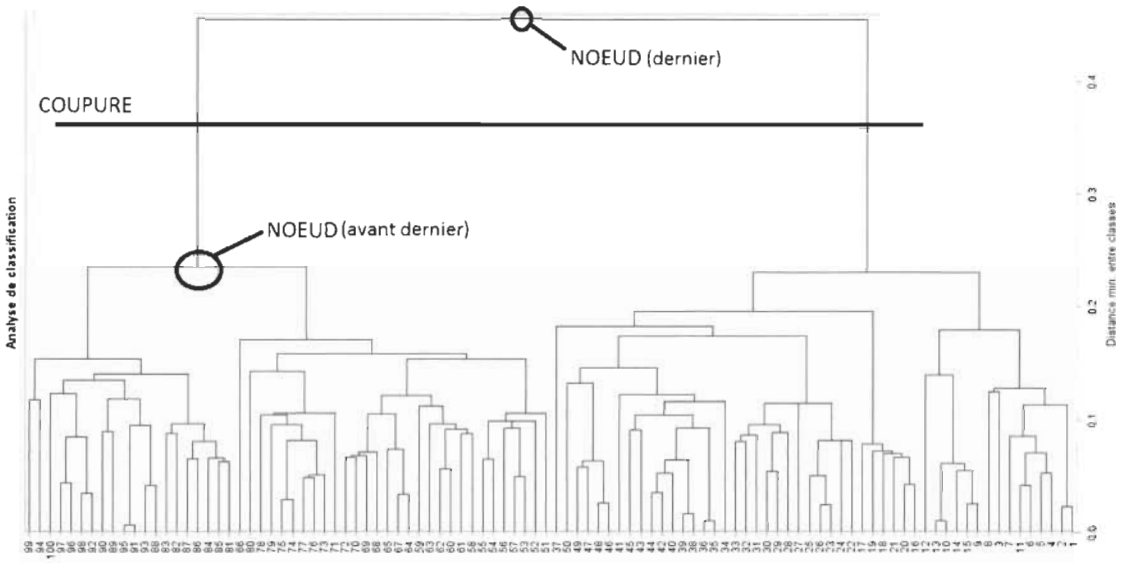
Elles fournissent une typologie des individus en classes homogènes en une hiérarchie de partitions. L'objectif étant d'élaborer la meilleure structure des éléments à classifier. Deux approches différentes peuvent être définies :
(1) Celles dites **ascendantes** ou agglomératives qui, dans la version la plus simple, procèdent par agglomérations successives des classes deux à deux pour former une hiérarchie de partitions. Cette approche part d'une partition la plus fine, où chaque individu est dans une classe, vers une partition moins fine en agrégeant successivement les individus les plus proches tout en respectant les mesures de ressemblances et les critères d'agrégations prédéfinis ;
(2) Celles dites **descendantes** procèdent par une division successive de l'ensemble des éléments. Elles partent d'une classe regroupant tous les éléments ; celle-ci est scindée en deux parties qui sont à leur tour scindées en deux, etc...jusqu'à ce que toutes les sous-classes obtenues correspondent à un seul élément.

Dans la suite du document, nous nous limiterons à l'approche **ascendante** hiérarchique (CAH) qui utilise un algorithme itératif dont l'idée est de créer, à chaque étape, une partition en regroupant les deux éléments les plus proches ou plusieurs paires d'ex aequo :

- On part de la partition P_0 où chaque individu constitue une classe ;
- à l'étape initiale, on agrège les deux individus les plus proches ou plusieurs paires d'ex aequo, au sens d'une mesure de ressemblance (dissimilarité, distance ou similarité) pré-définie ;
- on met à jour le tableau de la mesure de ressemblance considérée (dissimilarité, distance ou similarité) suivant un critère d'agrégation bien défini. On poursuit l'agrégation jusqu'à ce que l'on obtienne la classe formée de tous les individus.

Les partitions obtenues à chaque étape de la CAH peuvent être visualisées à l'aide d'un arbre de classification appelé dendrogramme. Sur un axe horizontal apparait les individus à regrouper et sur l'axe vertical sont indiquées les ressemblances correspondants

aux différents niveaux de regroupement. La figure 1.1 présente un dendrogramme avec la mise en évidence des nœuds.



Graphique 1.1 – *Exemple de dendrogramme*

a) Choix du nombre de classes en CAH

Une coupure du dendrogramme entre deux niveaux de nœuds fournit un nombre de classes. Dans le graphique 1.1, la partition en deux classes est obtenue en coupant l'arbre entre le dernier et l'avant dernier noeud.

Le critère généralement utilisé pour la détermination de la bonne partition est empirique nommé critère du « coude ». En fait selon ce critère, la partition pertinente sera indiquée, à l'observation du dendrogramme, par une décroissance brusque de la mesure de ressemblance (dissimilarité ou distance) entre classes. Ainsi, le bon nombre de classes est obtenu en coupant entre deux noeuds du dendrogramme où l'on observe un saut élevé.

1.2.2 Méthodes non hiérarchiques ou de partitionnement

Elles cherchent à construire directement des partitions. Dans ces méthodes, le nombre k de classes de la partition finale est connu ou fixé a priori. Le principe de ces méthodes est le suivant :

- on se fixe le nombre k de classes de la partition ;
- on part d'une partition P_0 en q classes et on cherche à l'aide d'un algorithme pour améliorer la partition P_0 initiale ;
- Par cet algorithme, on obtient finalement une partition P qui dépend du choix de P_0.

Le nombre k de classes fixé initialement peut être choisi au hasard ou déterminé à partir des connaissances que l'on a des données ou par d'autres algorithmes (Exemple : la classification ascendante hiérarchique).

1.2.2.1 Méthode des centres mobiles

C'est une méthode itérative d'optimisation où dans chaque itération, deux étapes sont effectuées (recentrage et réallocation). Le nombre k de classes, fixé a priori, correspond aux k centres de classes choisis arbitrairement. Le principe des centres mobiles, proposé par Forgy, 1965[1], est le suivant :

Etape 0 : on choisit, au hasard ou par la connaissance des données, k points qui vont définir des centres C_{01},C_{02},...,C_{0k}. On affecte chaque individu à la classe avec le centre de classe qui lui est le plus proche. On constitue donc k classes provisoires N_{01},N_{02},...,N_{0k}.

Etape 1 : On obtient k nouveaux centres de classes, C_{11},C_{12},...,C_{1k}, en choisissant les centres de gravité des classes précédentes c'est-à-dire des classes N_{01},N_{02},...,N_{0k}. Ces

nouveaux centres de classes conduisent à une nouvelle partition construite de la même façon qu'à l'étape 0 (c'est-à-dire, affecter chaque individu à la classe avec le centre qui lui est plus proche). Cette nouvelle partition sera formée des classes N_{11},N_{12},...,N_{1k}.
Etape m : On détermine k nouveaux centres C_{m1},C_{m2},...,C_{mk} en prenant les centres de gravité des classes $N_{(m-1)1}$,$N_{(m-1)2}$,...,$N_{(m-1)k}$. Ces nouveaux centres de gravité conduisent à une nouvelle partition formée des classes N_{m1},N_{m2},...,N_{mk}.
L'algorithme va alors s'arrêter si deux itérations successives donnent la même partition c'est à dire si les centres ne sont plus mobiles. La partition finale obtenue dépendra du choix initial des centres à l'étape 0.

D'autres méthodes similaires peuvent être également présentées. Il s'agit de celles des k-means et des nuées dynamiques.

1.2.2.2 Méthode des k-means

C'est une méthode développée par McQueen (1967)[2], elle est une variante de la méthode des centres mobiles. Cette méthode consiste à ne pas attendre d'agréger tous les individus de la classe pour calculer le centre de gravité. Ainsi, dès qu'on affecte un individu à une classe donnée, on calcule le centre de gravité de cette dernière et c'est par rapport à ce centre qu'on agrège d'autres individus. La position du centre de gravité est donc modifiée à chaque affectation.
La partition finale ainsi obtenue dépend de la partition initiale. C'est pourquoi certains statisticiens tel que Rao (1971) proposent de choisir au hasard les k centres de classes initiaux.

1.2.2.3 Méthode des nuées dynamiques

Cette méthode, développée par Diday (1971)[3], est une généralisation de la méthode des centres mobiles. La différence se trouve au niveau de la réaffectation des individus à chaque classe.

Dans cette méthode, après avoir déterminé les centres initiaux, un représentant est déterminé pour chaque classe. Le représentant ou prototype d'une classe serait plus général que la moyenne des observations et pouvant être de nature quelconque : un individu de cette classe, un groupe d'individus, une loi de probabilité, un segment, une droite, un histogramme, etc.

Ce représentant est défini comme étant l'individu le plus proche du centre de chaque classe. La réaffectation se fera alors, non pas, en fonction de la distance des autres individus par rapport aux centres mais par rapport aux représentants de chaque classe. En désignant par u_t le représentant de la classe t, l'algorithme des nuées dynamiques cherche à trouver, en partant d'une partition initiale quelconque pour un ensemble d'observations, la partition optimale $P = \{C_1, ..., C_k\}$ en minimisant de façon itérative le critère suivant :

$$G = \sum_{t=1}^{k} \sum_{x_i \in C_t} d(x_i, u_t)$$

Où d est une distance pour mesurer la proximité entre les individus x_i et les vecteurs représentants u_t. La minimisation du critère G correspond à sa minimisation pour chaque classe t. Ainsi, le vecteur prototype u_t est calculé en minimisant le critère g_t :

$$g_t = \sum_{x_i \in C_t} d(x_i, u_t)$$

Cependant, on parle de la généralisation des méthodes de centres mobiles dans la mesure où la distance d est substituée par le carré de la distance euclidienne et que les représentants des classes sont leurs centres de gravité. Dans ce cas, on retrouve la méthode des centres mobiles.

1.2.3 Mise en œuvre des méthodes classiques sous R

Cette partie traite la mise en application des méthodes classiques les plus connues et couramment utilisées en pratique. Il s'agit des méthodes de classification ascendante hiérarchique (CAH) et celle des k-means.

1.2.3.1 Classification ascendante hiérarchique (CAH)

Dans le logiciel R, la CAH peut être réalisée sous deux grandes méthodes : une CAH « simple » et une CAH sur composantes principales.

A) **CAH « simple »**

La spécificité de cette méthode est que la CAH est directement appliquée sur les données transformées sous forme de tableau de dissimilarités ou distances. Deux alternatives peuvent être soulignées par le biais de deux fonctions : **«hclust»** et **«agnes»**.

A.1) « hclust »

Il s'agit d'une fonction implémentée dans le package « stats ». Elle permet de réaliser une CAH à partir du tableau de distances entre individus. Ce dernier peut être obtenu, sous le logiciel R, à l'aide de la commande « dist » du package « stats ». La fonction « hclust » permettra alors de produire les résultats de la classification, y compris le dendrogramme.

La commande « dist » utilise, par défaut, la distance euclidienne pour le calcul de la matrice des distances. Aussi est-il possible de spécifier le type de dissimilarité ou distance que l'on souhaite calculer grâce à la commande suivante :

```
library(stats)
d<-dist(x, method = "manhattan") (matrice de distance de Manhattan),
```

où x représente les données.

De la même manière, on peut aussi utiliser les mesures de ressemblances « maximum » « canberra » ou « minkowski ». Pour l'agrégation des classes, on utilise, par exemple, la méthode de Ward via la commande :

```
cah<-hclust(d, method="ward.D2")
```

où **d** représente les données sous la forme d'une matrice de distance. Sous R, c'est la distance euclidienne qui est utilisée par défaut ;
method : le choix du critère d'agrégation entre les classes.
Si la méthode d'agrégation n'est pas spécifiée, « hclust » utilise par défaut la méthode « complete » ou critère du saut maximum.
D'autres critères d'agrégation peuvent être appliqués tels que : **average** (saut moyen), **median** (méthode de la médiane), **centroid** (méthode du centroïde), **ward.D2** (critère de Ward), **single** (saut minimum) etc.
Cependant, faisant suite à l'application de « hclust », le choix du nombre de classes est fait grâce à la fonction « cutree » pour la coupure du dendrogramme. Ainsi, le dendrogramme sera coupé, soit :
- en précisant le nombre de classes k souhaité.

```
Coup1 <-cutree(cah, k=2)  # 2 classes par exemple
```

ou
- en indiquant la hauteur h du dendrogramme à laquelle on souhaite couper (valeurs en ordonnée)

```
Coup2 <- cutree(cah, h=500) # hauteur 500
```

A.2) « agnes »

C'est une fonction implémentée dans le package « cluster », assez similaire à celui de

« hclust ». Cependant, la fonction « agnes » apporte un plus sur les résultats de la classification. En effet, il s'agit du coefficient d'agglomération proposé par Kaufman et Rousseeuw (1990)[4], qui permet de mesurer la qualité de la structure de classification pour une méthode d'agrégation donnée.

Pour chaque individu x_i $i = 1, ..., n$, sa valeur de son indice d'agrégation D (voir section 1.1.3) avec le premier élément avec lequel elle est fusionnée, est divisée par la valeur de l'agrégation effectuée dans la dernière étape de l'algorithme, notée m_i. Le coefficient d'agglomération AC est la moyenne de tous les $1 - m_i$ (Maechler et al., 2005)[5]. Il est donné par le réel :

$$AC = 1 - m_i = \frac{1}{n}\sum_{i=1}^{n}(1 - \frac{D(x_i, A_i)}{D(X, Z)})$$

où A_i désigne le premier élément avec lequel x_i a été regroupé

- X et Z désignent les deux dernières classes agrégées à l'étape finale de l'algorithme.
- $AC \in]0, 1[$

Plus la valeur AC est proche de 1, plus la structure des classes est bonne.

Ces lignes de commandes servent à faire une CAH avec le critère du saut maximum ainsi que le calcul du coefficient d'agglomération correspondant.

```
# CAH
library(cluster)
cah <- agnes(x, method = "complete")

# Calcul du coefficient d'agglomération
cah$ac
```

Où x représente l'ensemble des données. La fonction « agnes » peut aussi prendre une matrice de dissimilarité en entrée.

B) **CAH sur composantes principales**

Une CAH sur composantes principales est une méthode de classification effectuée suite à une analyse factorielle des données. Elle s'applique sur R grâce à la fonction « **HCPC** » du package **FactoMineR** (Husson, et al. 2006)[6].

Dans cette méthode, une première étape consiste à faire une analyse factorielle sur les données pour ensuite utiliser les coordonnées des individus sur les composantes principales pour la classification ascendante hiérarchique. Elle se réalise avec les lignes de commandes suivantes :

```
#Chargement du package « FactoMineR »
library(FactoMineR)
#Application d'une méthode d'analyse factorielle (ACP par exemple)
res.pca <- PCA(x)
#Classification ascendante hierarchique sur les résultats de l'ACP
res.hcpc <- HCPC (res.pca)
```

Ainsi, les résultats sont fournis avec une suggestion systématique du nombre de classes par le biais de la coupure du dendrogramme.

1.2.3.2 Méthode des k-means

C'est une variante de la méthode des centres mobiles qui permet d'effectuer une classification des individus en k classes avec une valeur de k fixée au départ.

Sous R, la fonction est connue sous le nom de « kmeans » et elle est incluse dans le package « cluster ».

Par exemple, une partition en quatre (4) classes d'un ensemble de données représenté sous forme d'une matrice de données x, par la méthode des k-means, est donnée par la fonction :

```
kms <- kmeans(x, 4)
```

Plusieurs informations peuvent être obtenues avec les sorties de « kms », parmi lesquelles :

```
kms\$cluster  indique le numéro de la classe de chaque individu, \\
kms\$centers fournit la moyenne des variables dans chacune des classes,\\
kms\$size donne les tailles des classes obtenues. \\
```

1.3 Approche probabiliste de la classification

Dans cette approche, les données sont considérées comme provenant d'un mélange de distributions de probabilités, une notion qui sera définie ultérieurement.
Cette approche est basée sur une étude de modèles de mélanges de lois de probabilités qui nécessite une bonne connaissance en inférence statistique.
Ces modèles de mélanges sont adaptés, tant au but exploratoire (découvrir une partition dans un ensemble d'individus) qu'au but décisionnel (affecter un tout nouvel individu dans des classes préalablement définies) de la classification.
Dans la première partie, il s'agira de souligner la problématique de la classification selon qu'elle soit supervisée ou non. Il s'en suivra, une présentation globale d'un modèle de mélanges de lois de probabilité, avant de se focaliser sur les estimations des paramètres assujettis au modèle. Par la suite, les critères de choix de modèles spécifiques seront traités. Enfin, il sera proposé, la mise en œuvre de la classification par les modèles de mélange gaussiens sur R.

1.3.1 Problématique de la classification dans un cadre probabiliste

1.3.1.1 Classification supervisée

Nous considérons ici que nos données sont constituées par un couple (x, z) où $x = (x_1, ..., x_n)$ représente l'ensemble des individus de cardinal n et leur partition en k classes G_1,...,G_k. Soit z=(z_1,...,z_i,...,z_n) où z_i=(z_{i1},...,z_{ik}) tel que z_{ik} est une variable qui vaut 1 si l'individu appartient à la classe k et qui vaut 0 sinon.
Nos données sont donc complètes vu que nous connaissons l'appartenance de chaque individu à une classe. La question qui se pose est alors : A quelle classe va appartenir un nouvel individu x_{n+1} ?
Il s'agira alors de déterminer une règle de classement pour affecter tout nouvel individu aux classes préalablement définies, donc faire de la prédiction.

1.3.1.2 Classification non supervisée

Dans la classification non supervisée, on fait plutôt une recherche à l'aveugle. Autrement dit, seuls les individus représentés par x=(x_1, ..., x_n) sont connus, la partition z de ces individus est alors inconnue. Le but sera donc d'estimer la partition z de x.
Dans le but d'établir les liens avec les méthodes neuronales (dont les réseaux de Kohonen), nous nous limiterons à l'aspect exploratoire de la classification probabiliste.

1.3.2 Rappels sur les modèles de mélange

1.3.2.1 Contexte et notation

Dans l'aspect non probabiliste, la ressemblance entre individus peut être mesurée par la distance ou par des indices de dissimilarité ou similarité. Cependant, dans le cadre probabiliste, les données observées $x = (x_1, x_2, ..., x_n)$ sont perçues comme des réalisations indépendantes d'une même distribution de probabilité. Ainsi, la ressemblance entre individus d'une même classe peut être traduite par le fait qu'ils proviennent d'une même loi de probabilité car la population est supposée être structurée en K classes et chaque classe est modélisée par une distribution spécifique. L'accumulation des lois associées à chaque classe produit alors une loi de mélange.
Par la suite, nous allons considérer que nos données sont constituées de couple (x, z) avec $x = (x_1, x_2, ..., x_n)$ l'ensemble des n individus et $z = (z_1, z_2, ..., z_n)$ une partition de ces individus en K classes $G_1, G_2, ..., G_K$, où $z_i = (z_{i1}, z_{i2}, ..., z_{iK})$ telle que $z_{ik} = 1$ si l'individu i appartient à la $k^{ième}$ classe et $z_{ik} = 0$ sinon, $1 \leq k \leq K$.
Les couples (x_1, z_1), (x_2, z_2),..., (x_n, z_n) sont des réalisations indépendantes et identiquement distribuées des n couples de variables aléatoires (X_1, Z_1), (X_2, Z_2), ..., (X_n, Z_n)

1.3.2.2 Définition de mélange de lois

Une loi de mélange p sur un espace X est une loi de probabilités s'exprimant comme une combinaison linéaire de plusieurs lois de probabilités (p_1,p_2,...,p_K). Autrement dit, il existe K coefficients π_1,π_2,..,π_K tels que pour tout x_i réalisation de X_i, $i = 1, 2, ..., n$:

$$P(x_i) = \sum_{k=1}^{K} \pi_k p_k(x_i)$$

avec :

- $\pi_k = P(z_{ik} = 1)$: probabilité a priori que l'individu i provient de la classe G_k

- $p_k(x_i) = P(X = x_i/z_{ik} = 1)$: désigne la loi des observations $(X = x_i/z_{ik} = 1)$ lorsque l'on connait l'appartenance à la classe.

Le modèle de mélange de lois peut être donc défini par :

$$m = \{p_k(x, \theta), \theta \in \Theta\},$$

où Θ représente l'ensemble des paramètres du modèle m avec $\theta = (\pi_k, \alpha_k), \alpha_k$ constituant le ou les paramètres spécifiques de la loi $(X = x_i/z_{ik} = 1)$.

1.3.2.3 Cas gaussien

Les modèles de mélange peuvent être également traduits par une fonction de densité g déterminant la distribution de données (x, z) à l'aide d'une combinaison linéaire de fonctions de densité élémentaires f_k, $k = 1, ..., K$.

L'approche la plus connue est le modèle de mélange gaussien où les densités élémentaires suivent des lois normales multivariées.

En effet, considérons toujours $x = (x_1, x_2, ..., x_n)$ un échantillon composé de n individus caractérisés par p variables continues et z une partition de x en K groupes $(G_1, G_2, ..., G_K)$. Dans le cadre des modèles de mélange gaussiens, on considère que chaque groupe G_k suit une loi normale de moyenne μ_k et de matrice de variance-covariance Σ_k. Par ailleurs, en notant $(\pi_1, \pi_2, ..., \pi_K)$ les proportions des différents groupes dans le mélange, $\alpha_k = (\mu_k, \Sigma_k)$ le paramètre de chaque loi normale et $\theta = (\pi_1, \pi_2, ..., \pi_K, \alpha_1, \alpha_2, ..., \alpha_K)$ le paramètre global du mélange. La loi de mélanges gaussiens que suit les données $X = (X_1, X_2, ..., X_n)$ peut s'écrire sous la forme :

$$g(x, \theta) = \sum_{k=1}^{K} \pi_k f(x, \alpha_k)$$

où $f(x, \alpha_k) = f(x_i, \alpha_k)$ pour $i = 1, ..., n$ avec $f(x_i, \alpha_k)$ la densité de la loi normale multivariée de paramètre $\alpha_k = (\mu_k, \Sigma_k)$ s'exprimant comme suit :

$$f(x_i, \alpha_k) = \frac{1}{(2\pi)^{d/2} \mid \Sigma_k \mid} exp[-\frac{1}{2}(x_i - \mu_k)^T \Sigma_k^{-1}(x_i - \mu_k)]$$

où $\mid \Sigma_k \mid$ est le déterminant de Σ_k

Contrairement aux méthodes de classification par l'approche non probabiliste, les modèles de mélange gaussiens permettent une interprétation géométrique des classes. Ainsi, Banfield et Raftery (1993)[7] ont développé un cadre basé sur un modèle pour la classification par la paramétrisation de la matrice de covariances en termes de sa décomposition en valeurs propres : c'est la décomposition spectrale de la matrice de covariances.

Chaque matrice de covariances des composantes du mélange peut s'écrire sous la forme :

$$\Sigma_k = \lambda_k D_k A_k D_k^T$$

Où

- λ_k représente le volume de la classe k
- D_k la matrice orthogonale des vecteurs propres de Σ_k qui s'interprète comme l'orientation de cette classe
- A_k est la matrice diagonale des valeurs propres de Σ_k et qui s'interprète comme la forme de la classe k

Les modèles gaussiens autorisent certaines fonctionnalités à varier entre les classes, comme l'orientation (D_k), le volume (λ_k) et la forme (A_k) tout en contraignant les autres à être les mêmes. A partir de ces manipulations sur la composition de la matrice de covariance, Celeux et Govaert (1993)[8] ont proposé quatorze (14) modèles différents en permettant à tout ou une partie des termes (λ_k, D_k, A_k) de varier ou non entre les classes (voir tableau 1.3).

Σ_k	Modèle (nom dans R)	Distribution	Le volume	Forme	Orientation
λI	EII	Sphérique	Égal	Égal	-
$\lambda_k I$	VII	Sphérique	Variable	Égal	-
λA	EEI	Diagonale	Égal	Égal	-
$\lambda_k A$	VEI	Diagonale	Variable	Égal	-
λA_k	EVI	Diagonale	Égal	Variable	-
$\lambda_k A_k$	VVI	Diagonale	Variable	Variable	-
λDAD^T	EEE	Ellipsoïdale	Égal	Égal	Égal
$\lambda DA_k D^T$	EVE	Ellipsoïdale	Égal	Variable	Égal
$\lambda_k DAD^T$	VEE	Ellipsoïdale	Variable	Égal	Égal
$\lambda_k DA_k D^T$	VVE	Ellipsoïdale	Variable	Variable	Égal
$\lambda D_k AD_k^T$	EEV	Ellipsoïdale	Égal	Égal	Variable
$\lambda_k D_k AD_k^T$	VEV	Ellipsoïdale	Variable	Égal	Variable
$\lambda D_k A_k D_k^T$	EVV	Ellipsoïdale	Égal	Variable	Variable
$\lambda_k D_k A_k D_k^T$	VVV	Ellipsoïdale	Variable	Variable	Variable

Tableau 1.3 – *Caractéristiques géométriques des quatorze (14) modèles de mélange gaussien obtenues à partir de la paramétrisation de la matrice de covariances*

1.3.3 Estimation des paramètres

1.3.3.1 Estimation par le maximum de vraisemblance

Le paramètre θ est généralement inconnu. L'objectif est alors de l'estimer à partir de notre échantillon X.

Pour cela, on peut maximiser la log-vraisemblance donnée par :

$$L(X,\theta) = L(x_1, x_2, ..., x_n, \theta) = \sum_{i=1}^{n} log(\sum_{k=1}^{g} \pi_k p_k(x_i))$$

Dans les faits, on considère la log-vraisemblance observée. car elle est calculée seulement à partir des données observées x.

En résolvant $\frac{dL(X,\theta)}{d\theta} = 0$, Il est possible de trouver le vecteur de paramètres $\widehat{\theta}$ qui maximise la fonction de vraisemblance, mais cette méthode de calcul est difficile à

mettre en œuvre à cause de la complexité des calculs à effectuer. La principale difficulté vient du fait que les classes définissant le mélange ne sont pas connues.
Cependant, pour estimer les paramètres d'un mélange m de façon optimale, on utilise la vraisemblance des données complètes. Autrement dit, on utilise la vraisemblance calculée comme si on connaissait la classe de chaque individu, elle est notée $L(X, Z, \theta)$. Ainsi, la maximisation de cette dernière pour l'estimation des paramètres d'un modèle de mélange sera rendue possible à l'aide de l'algorithme qui sera défini dans la prochaine section.

1.3.3.2 Algorithme EM pour l'estimation des paramètres

L'optimisation de cette vraisemblance complète est mise en oeuvre à l'aide de l'algorithme Expectation-Maximization EM proposé par Dempster, Laird et Rubin (1977)[9]. Il est généralement utilisé lorsque nous disposons de données incomplètes, c'est-à-dire, lorsque les seules données dont on dispose ne permettent pas l'estimation des paramètres.
Cet algorithme est une méthode itérative dont chaque itération comporte deux étapes : Expectation (E) et Maximization (M)
- La phase « Expectation » ou l'étape E, procède à l'estimation des données inconnues, sachant les données observées et la valeur des paramètres déterminée à l'itération précédente.
- La phase « Maximization » ou « étape M », procède donc à la maximisation de la vraisemblance, rendue désormais possible en utilisant l'estimation des données inconnues θ_E effectuée à l'étape E, et met à jour la valeur des paramètres pour la prochaine itération. L'estimateur ainsi obtenu :

$$\widehat{\theta} = argmax(L(X, Z, \theta_E))$$

1.3.3.3 Estimation des partitions

Après avoir estimé le paramètre θ, on en déduit ensuite une estimation des probabilités conditionnelles d'appartenance aux classes, t_{ik} : $t_{ik} = P(z_{ik} = 1/X = x_i; \theta)$. Avec la formule de Bayes

$$t_{ik} = \frac{\pi_k p_k(x_i; \alpha)}{\sum_{k=1}^{K} \pi_k p_k(x_i; \alpha)}$$

$i = 1, ..., n$; $k = 1, ..., K$.

Une partition $\widehat{z}$ est ainsi obtenue par le principe du MAP (Maximum A Posteriori). Ce dernier signifie qu'on affecte chaque individu x_i à la classe pour laquelle il a la probabilité d'appartenance conditionnelle estimée $\widehat{t_{ik}}$ la plus élevée.

1.3.4 Critères de choix de modèle

La reparametrisation de la matrice de covariances, dans le modèle de mélange gaussien multivarié, permet, non seulement, une certaine modération en offrant une possibilité de réduction du nombre de paramètres d'un modèle à estimer, mais aussi, une variété de modèles en posant certaines contraintes sur la forme, l'orientation et le volume des composantes du modèle. En outre, la vraisemblance permet de quantifier la qualité d'ajustement d'un modèle aux données $x = (x_1, x_2, ..., x_n)$, mais elle augmente avec le nombre de paramètres du modèle. Il serait intéressant donc de comparer des modèles en réduisant leur nombre de paramètres.

Ainsi, en guise de choix de modèle adapté, les critères classiques généralement utilisés pour sélectionner un modèle m sont les critères d'Akaike AIC (Akaike Information Criterion) et le BIC (Bayesian Information Criterion). Ces critères s'expriment comme une pénalisation de la log de vraisemblance du modèle :

$$AIC = L(x, \widehat{\theta}_m) - 0.5\nu_m$$

$$BIC = L(x, \widehat{\theta}_m) - 0.5 v_m log(n)$$

avec v_m : le nombre de paramètres du modèle m.

n : le nombre d'observations

$\widehat{\theta}_m$ correspond à l'estimateur du maximum de la vraisemblance sous le modèle m.

Généralement dans le cadre des modèles de mélange, on utilise plutôt le BIC.

Le BIC est la mesure de la log-vraisemblance ajustée pour le nombre de paramètres et la taille de l'échantillon. Le modèle optimal va correspondre à celui qui a la plus grande valeur du BIC.

Cependant le BIC présente des limites, à savoir la perte de l'objectif de la classification qui est de fournir des classes bien séparées. Autrement dit, il peut arriver qu'on sélectionne un modèle qui fournit la plus grande valeur du BIC, mais avec des classes imbriquées.

C'est dans ce cadre que Biernacki et al, 2000[10] ont proposé le critère ICL (Integrated Complete Likelihood) qui respecte l'objectif de la classification en terme de séparation des classes. En effet, ce critère s'interprète comme une pénalisation du critère BIC quant à l'aspect d'imbrication. Il s'exprime sous la forme :

$$ICL = BIC - E(\widehat{z}, \widehat{t})$$

avec $E(\widehat{z}, \widehat{t})$ mesurant l'écart entre la partition z et les probabilités d'appartenance $\widehat{t}_{ik}$.

Ainsi, en situation de classes trop imbriquées, choisir un modèle qui s'adapte mieux à nos données, tout en respectant l'objectif de la classification, revient à sélectionner celui qui maximise le critère d'ICL (Biernacki, 2009)[11].

1.3.5 Application des modèles de mélange gaussiens sous R

Une classification non supervisée par les modèles de mélange gaussiens peut être réalisée grâce à un package sous le logiciel R sous le nom de « **mclust** » (Fraley, et al., 2012)[12]. C'est un package qui contribue pour la classification et l'estimation de la densité d'un modèle basé sur la modélisation de mélange gaussienne. Il fournit également des fonctions pour l'estimation des paramètres via l'algorithme EM pour les modèles de mélange gaussiens avec le critère BIC et le critère ICL pour les stratégies de choix du nombre de classes. De plus, il fournit une variété de structures de la matrice de covariances qui conduit au quatorze (14) modèles établis par Celeux et Govaert (1993) :
De façon directe, la classification, par les méthodes de mélanges gaussiens, d'un ensemble d'individus représenté par les données X est réalisé sous R avec les lignes de commandes suivantes :

```
#Chargement du package « mclust »
library (mclust)
# Application de mclust
Class <- Mclust (X)
```

Où la fonction Mclust donne les résultats de la classification, tels que le modèle gaussien retenu avec le nombre de classes, l'effectif de chaque classe, la valeur de la log-vraisemblance et celle du BIC.
Aussi, la moyenne des variables dans chaque groupe et sa matrice de covariances sont également fournies.
Toutefois, il est également possible de spécifier le critère de sélection de modèle souhaité. Ainsi, la fonction mclustBIC() et mclustICL() servent un mode de classification utilisant respectivement le BIC et l'ICL comme critères de sélection du meilleur modèle :

```
#Mélanges gaussiens avec les paramètres estimés : Méthode BIC
BIC <- mclustBIC(X)
summary(BIC)
#Mélanges gaussiens avec les paramètres estimés : Méthode ICL
ICL<-mclustICL(X)
summary(ICL)
```

En outre, la fonction plot(BIC) (resp. plot(ICL)) fournit un graphique mettant en évidence la mise en compétition des quatorze modèles selon les valeurs du BIC (resp. de l'ICL). Le modèle optimal correspond à celui qui maximise la valeur du BIC (resp. de l'ICL).

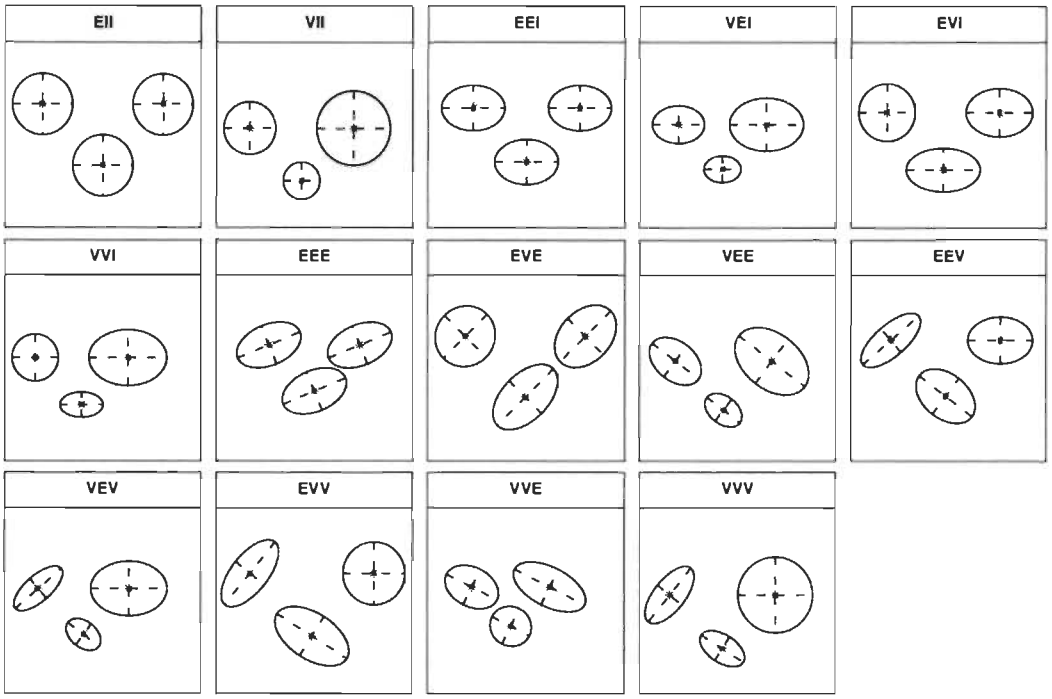
Cependant, Il est important de noter que la fonction « Mclust » utilise par défaut le BIC comme critère de sélection de modèle parmi les quatorze (14) modèles notés comme suit sous R (1.2) : Il s'agit d'une combinaison de trois (3) lettres E (égalité), V (Variation) et I (distribution diagonale).

La première position de la lettre correspond au volume, la deuxième à la forme et la troisième à l'orientation.

Exemple : EVV signifie un modèle avec des classes de même volume mais de forme et d'orientation différentes.

1.4 Choix du nombre de classes optimal en classification

La détermination du nombre de classes optimal dans un jeu de données a été toujours une grande problématique dans le processus de classification. Dans les méthodes précédentes, particulièrement les méthodes non hiérarchiques, il a été soulevé l'incon-



Graphique 1.2 - *Interprétation géométrique de chacun des quatorze modèles gaussiens avec trois classes*

vénient selon lequel la partition finale dépend fortement de la partition initiale dont le nombre de classes est choisi a priori. Par ailleurs, dans les méthodes ascendantes hiérarchiques, pour n individus, il y a n-1 partitions emboîtées possibles.

Ainsi, Milligan & Cooper (1985) confrontent une trentaine de critères d'arrêt sur des données simulées dans un espace euclidien où le nombre de classes, la dimension de l'espace des variables et le nombre d'éléments par classe varient. De plus, les classes sont bien séparées et possèdent une bonne cohésion interne. Quatre méthodes classiques en classification ascendante hiérarchique, soit le saut minimal, le saut maximal, le saut moyen et Ward, ont été considérées en n'utilisant que la distance euclidienne. Les 30 critères ont été classés selon leur performance pour la détermination des classes puisque le nombre de classes est connu a priori. Depuis, de nouveaux critères ont été proposés pour la détermination du nombre de classes. Malheureusement, ces critères n'étaient pas tous programmés et donc non disponibles.

Charrad et al. (2014)[13] ont fait la synthèse de ces principaux critères 1) en reprenant toutes les bases théoriques, 2) en ajoutant les méthodes de classification McQuity, la

médiane et le centroide avec plusieurs mesures de distances et de dissimilarités (Euclidienne, Maximum, Manhattan, Canberra, Binaire, Minkowski) et 3) en les intégrant dans une nouvelle librairie R «NbClust» accessible à tous les utilisateurs spécialistes ou non spécialistes. L'application est faite sur des données réelles et simulées afin d'évaluer la performance de chacun des indices étudiés. L'objectif est de rassembler tous les indices implémentés dans SAS ou dans les libraires R, et d'y ajouter de nouveaux afin de fournir à l'utilisateur une liste de 30 critères de validation lui permettant d'estimer le bon nombre de classes dans un jeu de données.

1.4.1 Présentation du Package NbClust

Il s'agit d'un package implémenté sous R, regroupant trente indices (30) proposés dans la littérature pour la détermination du nombre pertinent de classes dans un ensemble de données en fournissant, non seulement, une fonction pour effectuer la méthode des k-means, mais aussi une classification ascendante hiérarchique avec différentes mesures de ressemblance des individus et des critères d'agrégations des classes. En effet, par le biais d'une seule fonction, **NbClust** offre toute sorte de combinaisons, d'une part, d'indices de validation, de critères d'agrégation et de mesures de ressemblance pour le besoin de la CAH et, d'autre part, de variétés du nombre de classes souhaité pour le besoin des k-means.

Cependant, il convient de noter que **NbClust** fait appel à d'autre packages implémentés sous R tel que **cclust**, **clusterSim**, **clv** et **clValid**. Ces quatre packages renferment chacun des fonctions de calcul de plusieurs indices pour valider les résultats d'une analyse de classification. Ils peuvent être utilisés pour comparer simultanément plusieurs algorithmes de classification afin d'identifier la meilleure approche de classification et le nombre optimal de classes. Toutes les références sont détaillés dans (Charrad et al.,2014)[13]

Dans **NbClust**, le nombre pertinent de classes déterminé par le biais des trente (30) indices, est obtenu, soit par le nombre minimal ou maximal de l'indice, soit une com-

paraison de l'indice par rapport à la valeur critique, soit aussi par des méthodes graphiques. Cette dernière concerne l'indice de **Hubert** et celui de **Dindex**. Ainsi, le nombre de classes est déterminé par le pic le plus significatif en observant la courbe de la seconde différence entre les niveaux de l'indice.

Le choix optimal du nombre de classes sera guidé par les résultats fournis par les trente (30) indices. Autrement dit, le nombre pertinent de classes correspond au nombre fourni par la plupart des indices.

La liste des trente (30) indices utilisés, avec leurs caractéristiques et références, est présentée en annexe (Charrad et al.,2014)[13].

1.4.2 Mise en application

— **Logiciel** : R

— **Package** : NbClust

— **Source et description des données** :

Les données qui seront utilisées pour l'application de « NbClust », sont issues d'un classement académique des universités mondiales (ARWU), fait par des chercheurs de l'université Jiao Tong de Shanghai en Chine(http ://www.shanghairanking.com/).

Les résultats sont publiés chaque année par Shanghai Ranking Consultancy qui est une organisation indépendante de l'enseignement supérieur et n'est juridiquement subordonnée à aucune agence gouvernementale.

Ce classement, connu sous le nom de classement de Shanghai, a été initié en 2003 et constitue une base annuelle. Chaque année, plus de 1200 universités sont classées par l'ARWU et les 500 meilleures sont publiées. Ainsi, les données utilisées ici concernent le classement de ces universités en 2015.

Il s'agit donc de 500 observations mesurées par six (6) variables :

Alumni : Le nombre des anciens étudiants qui ont obtenu le prix Nobel ou médailles Fields. poids=10%

Award : Le nombre de professeurs qui ont obtenu le prix Nobel ou médailles Fields. poids=20%

HiCi : Le nombre de chercheurs les plus cités dans leur discipline. poids=20%

N&S : Le nombre de publications dans les revues scientifiques Nature et Science. poids=20%

PUB : Le nombre de dissertations répertoriées dans le Science Citation Index-Expanded (SCIE) et le Social Science Citation Index (SSCI). poids=20%

PCP : Le nombre moyen des 5 indicateurs derniers. poids=10%

- Script

Les lignes de commandes suivantes, appliquées sur le jeu de données décrit ci-haut, décrivent la procédure **Nbclust** qui détermine le nombre pertinent de classes compris entre 2 et 8 (pas nécessairement le nombre maximum) dont la mesure de distance entre individus est la distance euclidienne et le critère d'agrégation, celui de WARD.

```
library(NbClust)
setwd("C:/Users/NDIOUGA/Desktop/Maitrise/Analyse de données")
universite <- read.table("testuni.txt", dec=".", header=TRUE, row.names=1,
sep="\t",fill=TRUE)
###NbClust
res <- NbClust(universite, distance = "euclidean", min.nc = 2, max.nc = 8,
method = "ward.D2",index = "all")
res$Best.nc
res$All.CriticalValues
res$Best.partition
```

Cependant, la mise en compétition des trente (30) indices est données par l'option «

index=alllong », cette option demande beaucoup de temps pour sortir les résultats à cause de certains calculs lourds dus à l'indice de **Gamma**, **Tau**, **Gap** et **Gplus**.
Ici, l'option « all » est utilisée, elle contient tous les indices sauf les quatre (4) indices cités précedemment. Elle contient donc 26 indices et nécessite moins de temps d'exécution .

- Quelques résultats

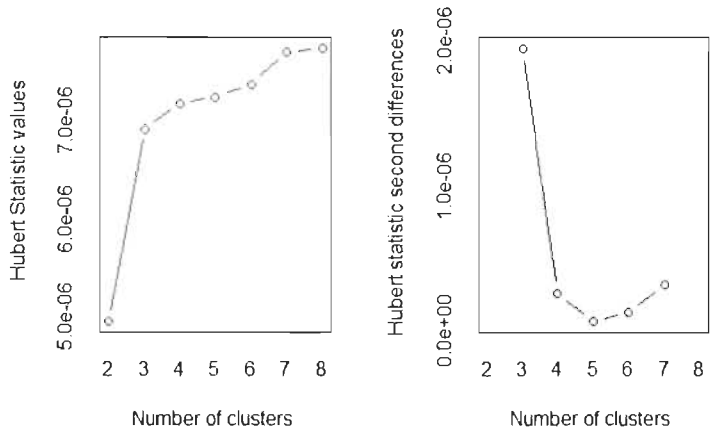
Sur les vingt-six(26) indices mis en compétition, quatorze (14) ont proposé trois (3) classes comme étant le nombre pertinent issu de la classification des 500 meilleures universités mondiales. Les résultats s'affichent comme suit :

```
*******************************************************************
* Among all indices:
* 4 proposed 2 as the best number of clusters
* 14 proposed 3 as the best number of clusters
* 2 proposed 4 as the best number of clusters
* 2 proposed 5 as the best number of clusters
* 1 proposed 8 as the best number of clusters

***** Conclusion *****

* According to the majority rule, the best number of clusters is  3


*******************************************************************
```

- Indice de Hubert :

Celui-ci est une méthode graphique(figure 1.3) permettant de déterminer le bon nombre de classes. Ce dernier est égal à 3, il correspond au pic le plus significatif

de la courbe représentant la dérivée seconde de la valeur de l'indice.



Graphique 1.3 – *Variation de l'indice de Hubert selon le nombre de classes*

Les résultats révèlent l'existence de trois (3) classes. La première classe renferme 12 universités qui sont toutes des universités américaines et britanniques, parmi lesquelles on peut citer :

Harvard, Stanford, Californie, Oxford etc. La deuxième classe a pour effectif 90 universités majoritairement américaines. La classe ayant le plus grand effectif est celle de la troisième avec 398 universités d'origine diverses, mais majoritairement asiatiques et européennes.

On s'attendait à ces résultats suite à l'application de **NbClust**. Ainsi, les universités de la première classe sont prestigieuses dans la mesure où elles disposent des meilleures dotations publiques, meilleurs étudiants, les nombres importants des Prix Nobel remportés par les chercheurs de ces universités.

Cependant, cette classification ne signifie pas que les universités de la deuxième et troisième classes ne sont pas performantes, mais c'est simplement que le monde académique américain est organisé différemment du monde académique asiatique et européen.

Chapitre 2

Méthodes de classification neuronales

2.1 Généralités sur les réseaux de neurones artificiels

2.1.1 Principe

Un neurone artificiel est un objet dont la modélisation s'inspire du fonctionnement du cerveau humain et du système nerveux. Un tel réseau de neurones est ainsi obtenu en inter-connectant plusieurs neurones. Il a pour principe de retourner, à partir des informations en entrée, une information en sortie. Nous allons noter $x_1, ..., x_n$ les informations en entrée affectées chacune d'un poids w_i, $i \in 1, ..., n$ et nous notons par w_0 un poids connu sous le nom de biais et associé à une donnée $x_0 = -1$. Ainsi, le

neurone traitera la combinaison linéaire ou moyenne pondérée :

$$X = \sum_{i=1}^{n} x_i w_i - w_0$$

En effet, après avoir traité l'information X par un apprentissage, le neurone donne une réponse Y compris entre 0 et 1 grâce à une fonction appelée fonction d'activation f. La sortie renvoyée alors par le neurone est définie par :

$$Y = f(\sum_{i=1}^{n} x_i w_i - w_0)$$

Plusieurs types de fonctions d'activation existent, entre autres celles ci-dessous, les plus couramment utilisées en pratique. :

- La fonction **seuil** ou de **Heaviside** définie par

$$f(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ 0 & \text{sinon} \end{cases}$$

- La fonction **sigmoïde** :

$$\forall \quad x \in R \quad f(x) = \frac{1}{1 + \exp(-x)}$$

- La fonction **linéaire** :

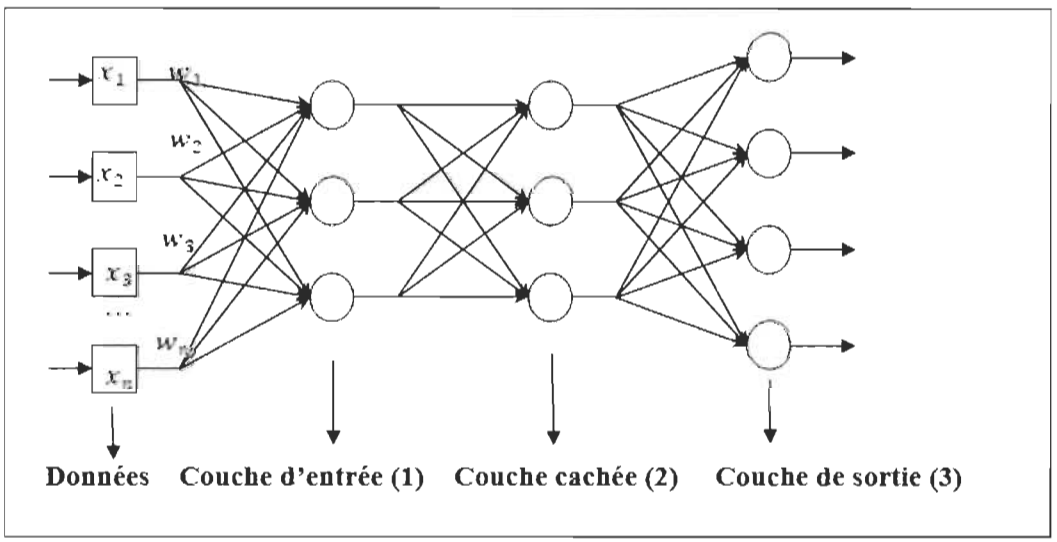
$$\forall \quad x \in R \quad f(x) = x$$

- La fonction **ReLU (Rectified linear unit)** :

$$f(x) = \begin{cases} x & \text{si } x \geq 0 \\ 0 & \text{si } x \leq 0 \end{cases}$$

2.1.2 Architecture des réseaux de neurones artificiels

L'architecture des réseaux de neurones consiste en une structuration des neurones en couches successives. Chacune de ces couches comporte plusieurs neurones. Ainsi, la première couche représente la couche d'entrée permettant de lire les données, la dernière étant la couche de sortie et les couches intermédiaires, non visibles, sont les couches cachées du réseau de neurones.



Graphique 2.1 – *Architecture de réseau de neurones*

Dans la figure 2.1, les cercles représentent les neurones disposés sous formes de couches. Ce réseau a trois couches, la couche d'entrée qui reçoit les informations $x_1, ..., x_n$ avec trois neurones par l'intermédiaire de poids w_i. La couche 3 représente

la couche de sortie comportant quatre neurones et donnant le résultats des calculs internes. Entre les couches 1 et 3, on trouve la couche 2 représentant la couche cachée. Dans les sections suivantes, deux grandes catégories de réseaux de neurones seront décrites : D'une part, les réseaux à apprentissage supervisé dans lesquels le système apprend à partir d'un échantillon d'apprentissage sur la base d'exemples liés aux informations fournies par l'utilisateur ; et d'autre part, les réseaux à apprentissage non supervisé qui sont utilisés lorsque les classes ne sont pas connues. Dans ce cas, les neurones s'entraînent sans aide externe c'est-a-dire qu'il sera inutile de signifier au réseau comment il doit se comporter.

2.2 Apprentissage supervisé

L'apprentissage supervisé est une branche de l'intelligence artificielle qui a pour but d'apprendre des phénomènes pour en faire une modélisation à des fins prédictives. En effet, le système apprend tout seul le modèle en fonction des données qui lui ont été fournies en exemple. Deux types de réseaux de neurones artificiels seront décrits pour le besoin d'apprentissage supervisé : le perceptron simple et le perceptron multi-couche que nous allons décrire très succinctement.

2.2.1 Perceptron

En 1958, Frank Rosenblatt (Rosenbalt, 58)[14] a créé un modèle de réseau de neurones avec un algorithme d'apprentissage supervisé connu sous le nom de perceptron. Ce dernier est un outil de classification linéaire disposant d'une seule sortie à laquelle toutes les entrées, pondérées par des poids, sont connectées. Le perceptron, en tant que réseau de neurones artificiel le plus simple, est un algorithme d'apprentissage supervisé de classifieurs binaires dans la mesure où il ne peut y avoir que deux (2)

classes en sortie. En outre, le perceptron fait une prédiction par le biais de la fonction d'activation qui transforme la somme pondérée $X = \sum_{i=1}^{n} x_i w_i - w_0$ reçue en entrée. Ainsi, les fonctions d'activation utilisées avec ce type de réseau de neurones artificiel sont les fonctions **seuil** et **linéaire**.

Cependant, bien qu'il fut à l'époque une grande avancée dans le domaine de l'intelligence artificielle, il présente des limites pour cause qu'un perceptron à une seule couche peut uniquement séparer les classes si elles sont linéairement séparables. Ce qui s'apparente en statistique à l'analyse discriminante linéaire.

2.2.2 Perceptron multicouche

Un type de réseau de neurones artificiel, en mode d'apprentissage supervisé et capable de traiter des données non linéairement séparables, est le perceptron multicouche (PMC) (Minsky M. et Papert S., 1969) [15]. Ce qui s'apparente à une analyse discriminante non linéaire (quadratique, fléxible...). Ce dernier est un type de réseau neuronal composé de plusieurs couches comme indiqué dans la figure 2.1.
Dans l'environnement du PMC, un calcul est effectué au niveau de chaque neurone, mais les entrées passent par plusieurs couches de calcul avant de fournir un résultat. A cet effet, les sorties de la première couche de neurones sont les entrées de la couche suivante et ainsi de suite.
Aussi, il est possible de déterminer, souvent par essai d'erreur, les différents paramètres du réseau tel que le nombre de couches, le nombre de neurones contenus dans chaque couche, les poids etc.

2.3 Apprentissage non supervisé

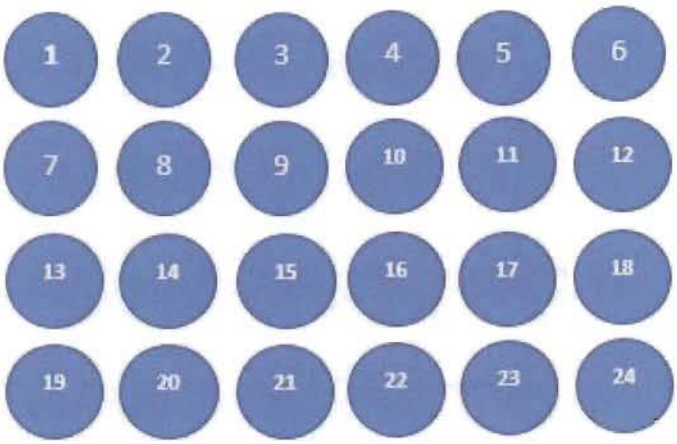
Dans le cadre de la classification non supervisée, la carte de Kohonen ou Self Organizing Map (SOM) (Kohonen, 1982)[16], assimilées aux méthodes neuronales, font l'objet dans cette section.
Plus précisément, SOM, communément appelée réseau de Kohonen, est une méthode automatique d'analyse de données traitant des problèmes liés au regroupement et à l'exploration des données de grandes dimensions. Ce type de réseau est constitué d'un ensemble de neurones reliés par une forme de voisinage.

2.3.1 Caractéristiques de la carte auto-organisatrice

Une carte auto-organisatrice est composée d'une grille de neurones représentée dans un espace de faible dimension.
Le nombre de neurones voisins est lié à la dimensionnalité de la grille. En effet, quand la grille est unidimensionnelle, chaque neurone a deux voisins. Si la grille est de dimension 2, la disposition des neurones est de topologie rectangulaire ou carré et chaque neurone a quatre voisins ; ou de topologie hexagonale où chaque neurone possède six voisins.
Quelle que soit la typologie, chaque neurone est reconnu à l'aide de son numéro et son emplacement r au niveau de la grille. En guise d'exemple, une carte auto-organisatrice rectangulaire, comportant 4 lignes et 6 colonnes, est représentée par la figure 2.2. Ainsi, le neurone 7 est désigné par le vecteur $r_7 = (2, 1)$.

2.3.2 Principe

Le réseau de Kohonen est un type de réseau de neurones artificiels constitué d'un ensemble de neurones disposé sur une grille à faible dimension où chaque neurone



Graphique 2.2 - *Carte auto-organisatrice rectangulaire de 24 neurones*

correspond à un vecteur prototype appartenant à l'espace des observations (dimension p=nombre de variables), ce qui pourrait s'apparenter aux représentants des classes dans la méthode des k-means.

Dans le but de projeter les observations sur la carte, l'apprentissage de ce réseau se fait de manière non supervisée.

Dans la première étape de l'algorithme de Kohonen, une observation $x(t)$ de l'ensemble des données est choisie au hasard et elle est comparée à tous les vecteurs prototypes dans le but de déterminer son neurone vainqueur (ou classe gagnante). Le neurone vainqueur d'une observation est le neurone dont le vecteur prototype est le plus proche de l'observation au sens d'une distance donnée (ex : distance euclidienne). Notons par w_k le vecteur prototype du neurone k, v le neurone vainqueur de l'observation $x(t)$ et K le nombre total de neurones. v est alors déterminé comme suit :

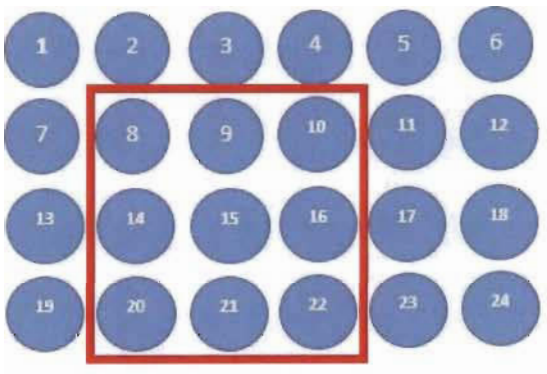
$$d(w_v(t), x(t)) = min_{k\in\{1,2,...,K\}} \{d(w_k(t), x(t))\}$$

Ensuite, le neurone vainqueur est activé dans la deuxième étape et son vecteur prototype est mis à jour pour se rapprocher de l'observation $x(t)$ présentée au réseau. Cette mise à jour concerne aussi les vecteurs des neurones voisins du neurone vainqueur. Autrement dit, les vecteurs de ces neurones voisins vont s'ajuster vers l'observation $x(t)$.

Cet ajustement a une amplitude déterminée par la valeur d'un pas d'apprentissage $\alpha(t)$ et la valeur d'une fonction de voisinage $h(t)$. La formule de mise à jour des vecteurs prototypes (Ritter et Schulten, 1986) [17] est la suivante :

$$w_k(t+1) = w_k(t) + \alpha(t)h_{vk}(t)[x(t) - w_k(t)]$$

Où $h_{vk}(t)$ est la fonction de voisinage qui définit la proximité entre les neurones v et k sur la carte. Cette fonction est égale à 1 si le neurone k se trouve à l'intérieur du carré (figure 2.3) centré sur le neurone v et 0 sinon. Le rayon de ce carré est appelé rayon de voisinage.

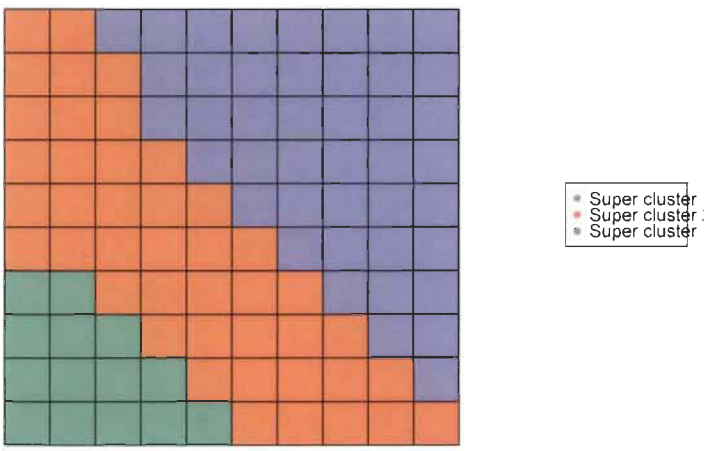


Graphique 2.3 – *Carte auto-organisatrice 4 par 6 neurones (Voisins du neurone 15)*

Au terme de cet apprentissage, le résultat est la projection non linéaire de l'ensemble des observations sur la carte. Ainsi, chaque observation est attribuée à son neurone vainqueur (ou classe gagnante). Plus important, la topologie des données est préservée grâce à la fonction de voisinage. Autrement dit, deux neurones voisins sur la carte auto-organisatrice signifient des classes voisines dans l'espace de données. Le nombre de neurones correspond donc au nombre de classes formées.

2.3.3 Regroupement et super-classe

Après convergence de l'algorithme de Kohonen, toutes les observations sont classifiées en k classes selon une distance choisie au préalable, k étant le nombre de neurones constituant la carte auto-organisatrice.
Le choix du nombre de classes est arbitraire et souvent le nombre de neurones sur la carte est élevé, ce qui rend difficile l'interprétation des classes obtenues. Pour faciliter l'analyse et la description des classes, on peut réduire le nombre de classes, en appliquant une classification ascendante hiérarchique sur les vecteurs prototypes.
Ces nouvelles classes issues de la classification hiérarchique sont appelées super-classes. Ces dernières ne regroupent normalement que des classes contiguës, ce qui s'explique par la propriété de respect de la topologie de l'algorithme de Kohonen. Il est possible de colorier, par le biais du logiciel R, les classes nouvellement regroupées pour les rendre visibles. La figure 2.4 illustre un exemple de carte auto-organisatrice avec la mise en évidence des super-classes.



Graphique 2.4 – *Mise en évidence des super-classes sur une carte auto-organisatrice 10 par 10 neurones*

Cette double classification donne l'avantage d'analyser les données à un niveau beaucoup plus simple faisant apparaitre les caractéristiques principales tout en facilitant l'interprétation.

2.4 Application des méthodes de classification par les réseaux de Kohonen sous R

Il est possible, sous R, de réaliser une classification non supervisée par la carte auto-organisatrice de Kohonen grâce à « **class** », « **som** », « **popsom** », « **Kohonen** », « **yasomi** » et «**SOMbrero**». Ces packages implémentent des versions stochastiques des algorithmes de Kohonen pour les données numériques, données de dissimilarité et pour les données décrites par des tableaux de contingence. Dans ce document, nous utiliserons le package « **SOMbrero** » (Vialaneix et al , 2019)[18] en raison de sa flexibilité, de ses nombreuses fonctionnalités de traçage et le fait qu'il propose le plus grand nombre de diagnostics (graphiques, super-classes etc.) afin de faciliter l'interprétation des résultats.

2.4.1 Construction de la carte auto-organisatrice

La fonction permettant de réaliser une carte auto-organisatrice est connue sous le nom de « trainSOM ». Elle est utilisée comme suit :

```
###############Algorithme SOM ##########
somRes  <- trainSOM(x)
```

Avec x l'ensemble des données. Il est possible de rajouter d'autres arguments tels que le nombre de sauvegarde intermédiaire durant l'apprentissage (« **nb.save** ») ou encore de limiter ou non les sortie (« **verbose=TRUE or FALSE** »).
Cette fonction retourne un objet de résultats « somRes » qui contient les informations suivantes :

— « **clustering** » : La classification des individus sur la carte auto-organisatrice

- « **Prototypes** » : les coordonnées des vecteurs prototypes
- « **Backup** » : une liste contenant des sauvegardes intermédiaires des coordonnées des vecteurs prototypes.

Pour mieux illustrer la mise en œuvre de l'algorithme de Kohonen sur R par le biais de « trainSOM », les lignes de commandes suivantes ont été appliquées sur les Iris de Fisher qui est constitué de 3 espèces de fleurs (setosa, virginica, versicolor) décrites par 4 variables (longueur sépale, largeur sépale, longueur pétale, largeur pétale), chaque espèce contient 50 individus pour un total de 150 Iris.

```
###############Algorithme SOM ##########
##Chargement du package
library(SOMbrero)
### isoler la cinquième variable qui est l'espèce
Iris2<-iris[,1:4]
###############Algorithme SOM ##########
Iris_som <- trainSOM(Iris2)
```

A la suite de l'exécution du programme ci-dessus, toutes les informations concernant la topologie de la carte (carré 5 par 5), le nombre d'itérations, la méthode d'initialisation des vecteurs prototypes (aléatoire) etc., choisis par défaut, sont affichées comme suit :

```
Self-Organizing Map algorithm...

  Parameters of the SOM

    SOM mode                        :  online
    SOM type                        :  numeric
    Affectation type                :  standard
    Grid                            :
      Self-Organizing Map structure

        Features   :
           topology     :  square
           x dimension  :  5
           y dimension  :  5
           distance type:  euclidean

    Number of iterations            :  750
    Number of intermediate backups  :  5
    Initializing prototypes method  :  random
    Data pre-processing type        :  unitvar
    Neighbourhood type              :  gaussian
```

Une analyse de la variance (ANOVA) de chaque variable d'entrée en fonction des classes de la carte peut être effectuée à l'aide de la fonction "**summary()**". Cette fonction teste la significativité de chaque variable dans chacune des classes de la carte. Cette analyse de la variance est utile parce qu'elle permet de voir quelles sont les variables qui participent à la formation des classes.
On voit que toutes les variables ont des moyennes significativement différentes dans les classes (indiqué par *** dans la colonne **significativity**), elles sont donc pertinentes pour la formation de la classification.

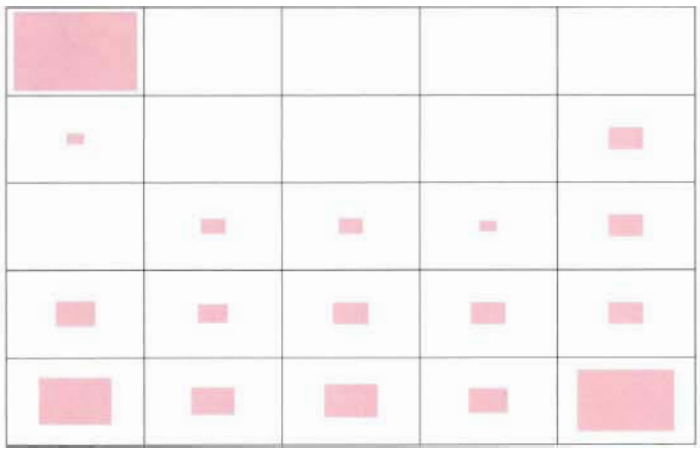
```
ANOVA                :

Degrees of freedom :  16
                F         pvalue     significativity
Sepal.Length  39.105      0              ***
Sepal.Width   16.480      0              ***
Petal.Length 286.743      0              ***
Petal.Width  146.222      0              ***
```

Pour la visualisation de la carte auto-organisatrice, plusieurs types de graphique sont implémentés.

a) Type « Hitmap »

Ce type graphique (figure 2.5) donne la distribution des individus dans chacune des classes de la carte. Il y a plus d'effectif dans les classes où le carré rose est grand. Les cases toutes blanches signifient des classes vides. La classe 5 (côté supérieur gauche) et classe 25 (coin inférieur droite) contient plus d'individus par rapport aux autres classes de la carte.

```
#### graphique hitmap
plot(Iris_som, what="obs", type="hitmap")
```



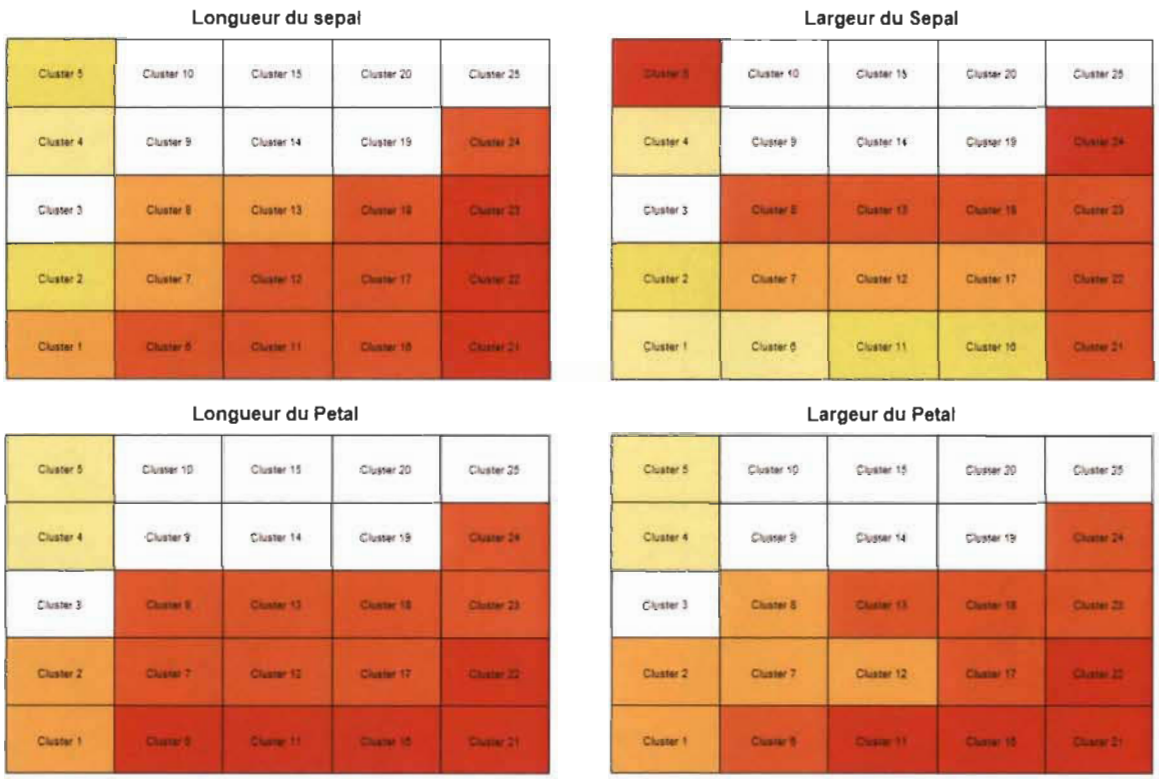
Graphique 2.5 – *Graphique de type "Hitmap"*

b) Type « color »

Ce type de graphique (figure 2.6) peut permettre de caractériser les classes et voir le comportement des composantes de chaque classe vis-à-vis des variables.

```
#Graphique de type Couleur
par(mfrow=c(2,2))
```

```
plot(Iris_som, what="obs", type="color", variable=1, print.title=TRUE,
main="Longueur du sepal")
plot(Iris_som, what="obs", type="color", variable=2, print.title=TRUE,
main="Largeur du Sepal")
plot(Iris_som, what="obs", type="color", variable=3, print.title=TRUE,
main="Longueur du Petal")
plot(Iris_som, what="obs", type="color", variable=4, print.title=TRUE,
main="Largeur du Petal")
```

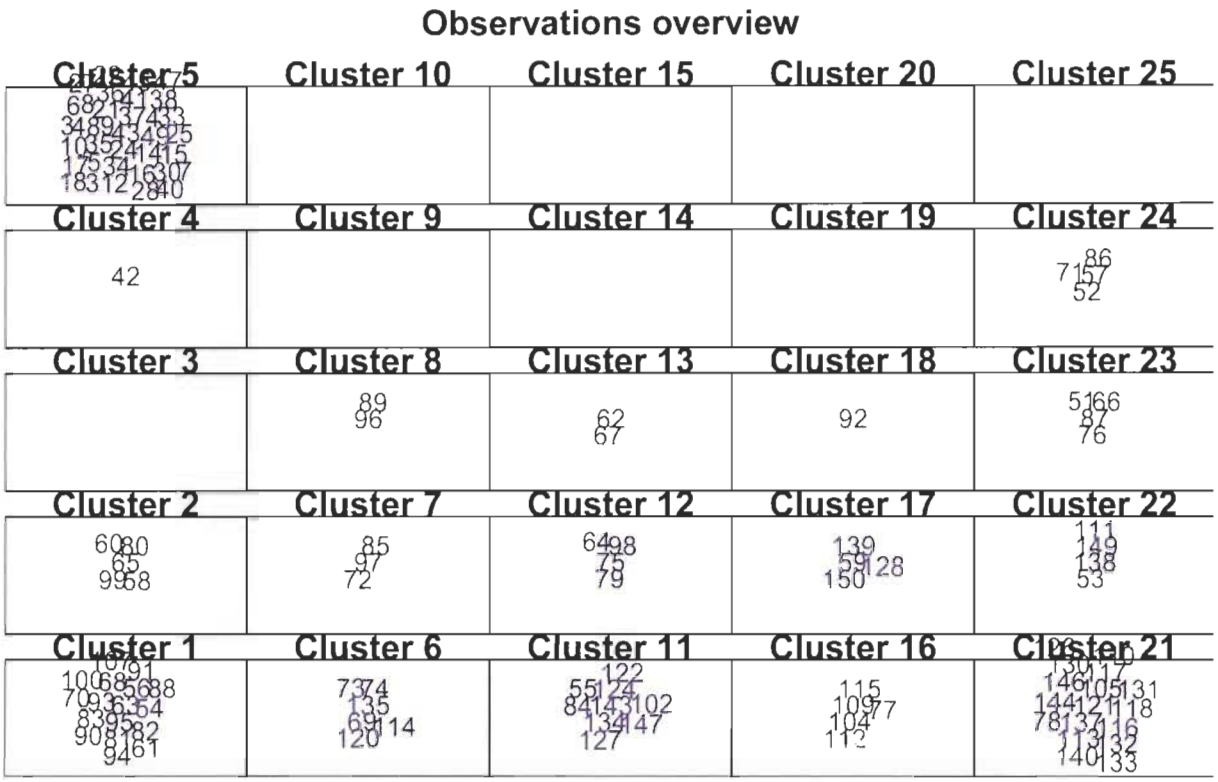


Graphique 2.6 – *Graphique « color »*

c) Type « names » Ce graphique (figure 2.7) affiche la carte avec le contenu de chaque classe. Rappelons que les iris sont numérotés de 1 à 150.

```
#Graphique de type names
```

```
plot(Iris_som, what="obs", type="names", print.title=TRUE, scale=c(0.9,0.5))
```



Graphique 2.7 - *Graphique « names »*

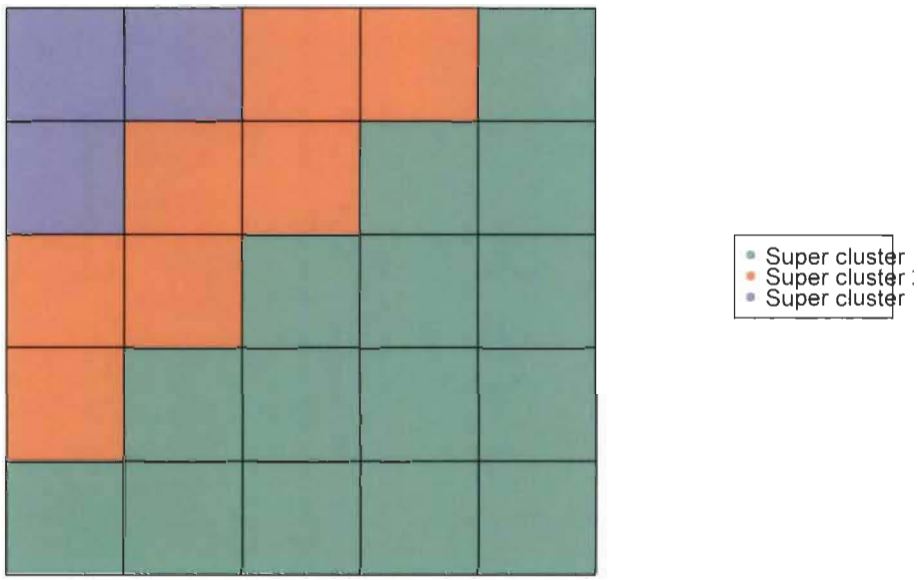
D'autres types de graphique sont implémentés dans le package Sombrero.

2.4.2 Formation des super-classes

Comme souligné dans la section 2.3 du chapitre, le nombre de classe obtenu est sensiblement égale au nombre de neurone de la carte (25). La réduction du nombre de classe par le biais de la classification ascendante hiérarchique est implémentée sous « **SOMbrero** » à l'aide de la fonction « **superClass** ». Le script ci-dessous permet de réaliser une classification ascendante hiérarchique des classes de la carte en 3 classes.

```
##Construction de super-classes à partir du fichier SOM
my.sc <- superClass(Iris_som, k=3)
plot(my.sc, type="grid", plot.legend=TRUE)
```

Avec le graphique de type « **grid** » (figure 2.8), on voit clairement, les super-classes formées ne regroupent que des classes contiguës, ce qui confirme la convergence de l'algorithme de Kohonen et l'homogénéité des classes.



Graphique 2.8 – *Classification finale avec les super-classes*

Chapitre 3

Étude comparative entre les méthodes classiques et neuronale

Ce chapitre est consacré à la comparaison des différentes méthodes classiques et neuronale en classification. Tel que mentionné auparavant, un ensemble de méthode de classification non supervisée se fonde sur une approche géométrique pour classifier les données en utilisant comme principe de base une mesure de ressemblance (distance ou dissimilarité ou similarité) pour comparer deux individus. Ces méthodes à approche géométrique qui seront utilisées pour l'étude comparative sont la classification ascendante hiérarchique (CAH) et la méthode des k-means. Une autre méthode à approche probabiliste concerne les mélanges gaussiens. Cette méthode suppose que les données constituent un échantillon indépendant tiré à partir d'un mélange de plusieurs distributions gaussiennes. Par conséquent, les individus de la même classe sont ceux qui appartiennent probablement à la même distribution. A ces méthodes, s'ajoutent la carte auto-organisatrice de Kohonen qui est un type de réseau de neurones artificiels dont l'apprentissage se fait de manière non supervisée. Cette carte est composée d'un ensemble de neurones disposé sur une grille à faible dimension où chaque neurone correspond à un vecteur prototype qui se spécialise pour représenter une classe des

données selon les caractéristiques qui les rassemblent.
Ce chapitre est organisé en trois sections. La première est consacrée à la démarche adoptée pour mener l'étude comparative entre les méthodes statistiques et neuronale de classification non supervisée. La deuxième décrit les données sur lesquelles seront appliquées les différentes méthodes de classification concernées et la troisième discute des résultats obtenus en utilisant le logiciel libre R.

3.1 Démarche

Les différentes méthodes de classification, décrites auparavant, seront appliquées sur les données présentées à la section suivante. La particularité à préciser est que parmi les trois manières de réaliser une classification ascendante hiérarchique sur R, la classification hiérarchique sur composantes principales est retenue en raison de sa flexibilité et de ses larges fonctionnalités. Cette méthode utilise en amont la distance euclidienne comme mesure de ressemblance des individus et le critère de Ward comme mesure d'agrégations des classes. Pour ce qui est du réseau de Kohonen, une classification ascendante hiérarchique sera aussi appliquée, c'est-à-dire, une classification des vecteurs prototypes représentant chaque neurone, afin de former les super-classes (les classes obtenues après regroupement des premières classes qui découlent de l'algorithme de Kohonen). Pour toutes ces méthodes, le choix optimal du nombre de classes est un enjeu important. Ainsi, la méthode des k-means et celle de Kohonen exigent dès le départ de connaitre le nombre de classes, alors que la CAH ne le nécessite pas. Pour pallier à ce problème, nous allons faire appel au package R **NbClust** (Charrad et al., 2014)[13] qui permet de déterminer le nombre optimal de classes. Ainsi, nous allons évaluer la performance de chaque méthode sur le choix optimal du nombre de classes. Ce nombre nous permettra d'une part, d'apprécier les résultats de la méthode de Ward et, d'autre part, le choix à considérer pour les méthodes de k-means, celle de Kohonen et celle du modèle gaussien. De plus, les différentes méthodes seront compa-

rées sur la base des formes fortes (i.e. ensemble d'éléments classés ensemble à plusieurs reprises), communes à plusieurs partitions issues de ces méthodes.

3.2 Description des données

Pour effectuer l'analyse comparative des différentes méthodes classiques et neuronale de classification, nous allons utiliser le jeu de données « turkiye student evaluation R spécific - Educational data » (Rossi et Ahmed, 2013)[19] . Ce dernier provient d'un formulaire d'évaluation de l'enseignement des cours qui a ont été complétés par des étudiants de l'Université Gazi à Ankara (Turquie). Au total, on dispose de 5820 scores d'évaluation fournis par des étudiants. En clair, nous allons analyser un jeu de données formé de 5820 individus et de 33 variables dont 28 sont des questions spécifiques aux cours, en plus de 5 autres variables. Ces dernières comprennent l'identifiant de l'instructeur, le code du cours ainsi que le profil comportemental de l'étudiant, notamment, le nombre de fois qu'il suit le cours, son niveau d'assiduité et sa perception du niveau de difficulté du cours. Les 28 variables d'évaluation portant spécifiquement sur les cours, sont des scores dont les valeurs prises sont 1, 2, 3, 4, 5.

Les cours concernés sont au nombre de 13, donnés par 3 instructeurs. En effet, l'instructeur 1 donne 3 cours (cours 2, 7 et 10) et l'instructeur 2 donne 4 cours (cours 1, 6, 11, 13). C'est l'instructeur 3 qui donne le plus grand nombre de cours, soit 6 cours. Le score moyen global de l'évaluation, pour toute question confondue, est de 3.18. Le cours 2 (score moyen de 3.60) est le mieux noté et les moins bons scores donnés par les étudiants concernent le cours 4 (score moyen de 2.93).

Notons que ces données ne suivent pas une loi gaussienne. Par conséquent l'hypothèse de normalité sous-adjacente à la méthode de mélange gaussien n'est pas satisfaite.

3.3 Présentation des résultats

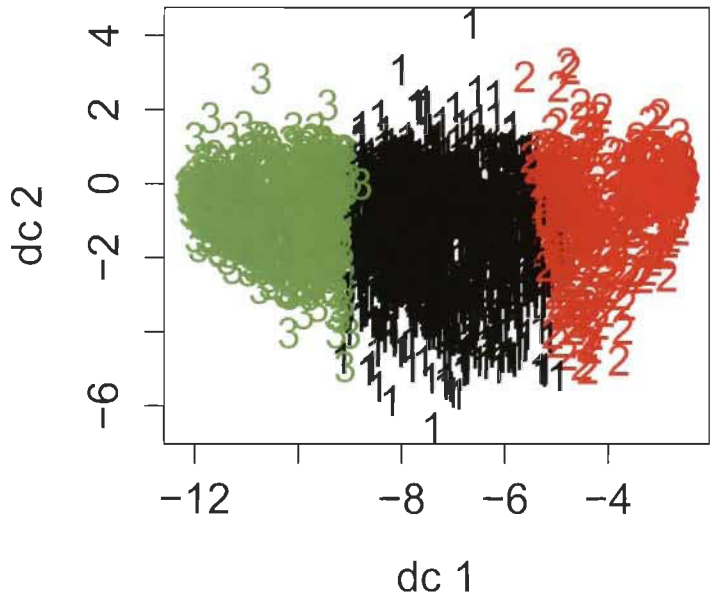
3.3.1 Analyse et description des classes

Le nombre de classes optimal obtenu grâce au package **NbClust** est égal à 3. Avec la méthode des k-means, la première classe de la partition obtenue contient 2358 éléments, soit la classe la plus « peuplée », l'effectif de la deuxième classe est 1239 et la troisième est constituée de 2223 éléments. En mettant l'accent sur le score moyen de chaque question dans chacune des classes et le score global moyen pour toutes les questions confondues (voir Tableau 3.1), il en ressort que la classe 1 caractérise les réponses neutres ou moyennement satisfaites (score moyen de 3), la classe 2 quant à elle, représente la classe des insatisfaits avec un score moyen de 1.4, et enfin la classe 3 caractérise les réponses totalement en accord ou la classe des satisfaits (score moyen de 4.3).

Classes	Taille	Moyenne d'évaluation globale
1	2358	3
2	1239	1.4
3	2223	4.3

Tableau 3.1 – *Composition des classes selon la méthode des k-means*

Les classes peuvent être visualisées à l'aide de la figure 3.1.



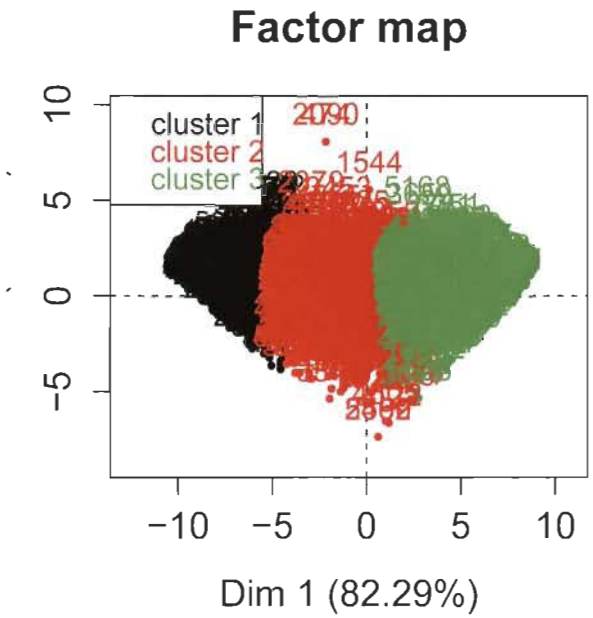
Graphique 3.1 - *Visualisation des classes avec la méthode des k-means*

Les résultats qui découlent de la CAH ne sont pas très loin de ceux des k-means. En effet, le tableau 3.2 renseigne que la classe 1 est la classe de plus petite taille et contient 1228 éléments avec un score moyen de 1.4 (insatisfaits des cours). La classe 2 a l'effectif le plus élevé, elle comptabilise 2357 éléments avec un score moyen de 3 (moyennement satisfaits ou neutres). La classe 3 contient 2235 éléments avec un score moyen d'évaluation de 4.3 (totalement satisfaits des cours).

Classes	Taille	Moyenne d'évaluation globale
1	1228	1.4
2	2357	3
3	2235	4.3

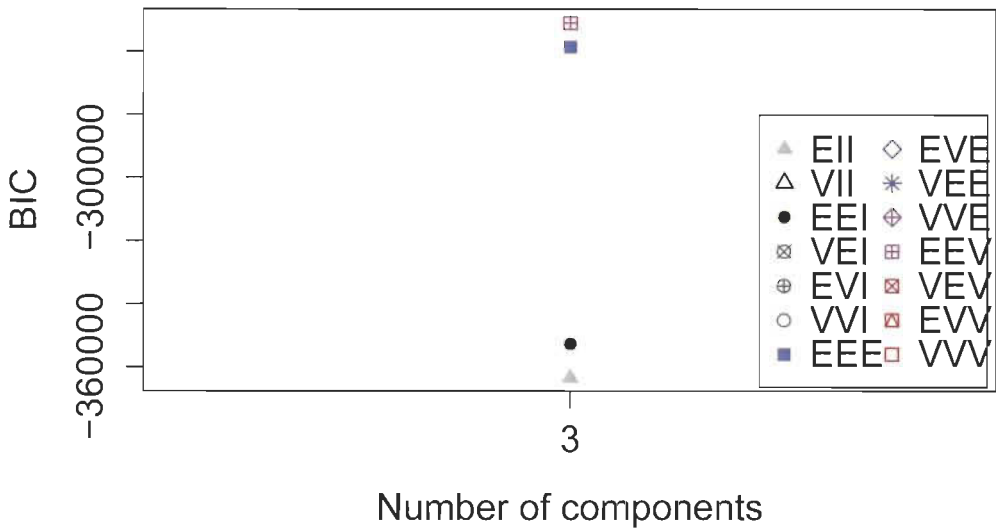
Tableau 3.2 - *Composition des classes selon la méthode CAH*

Ces classes sont mises en évidence dans la figure 3.2.



Graphique 3.2 – *Visualisation des classes avec la CAH*

Concernant la méthode de mélanges gaussiens, tel que mentionnée précédemment, l'hypothèse de normalité n'est pas satisfaite pour ce jeu de données. Néanmoins, nous allons présenter les résultats obtenus pour cette méthode. Ainsi, 14 modèles ont été mis en compétition selon la valeur du BIC. La figure 3.3 laisse apparaitre que, parmi les 14 modèles, un modèle de mélange à trois composants (classes) maximisant la valeur du BIC est le modèle avec des classes de volumes et formes identiques, mais d'orientations différente (EEV).



Graphique 3.3 – *Quatorze modèles de mélange gaussien selon la valeur du BIC*

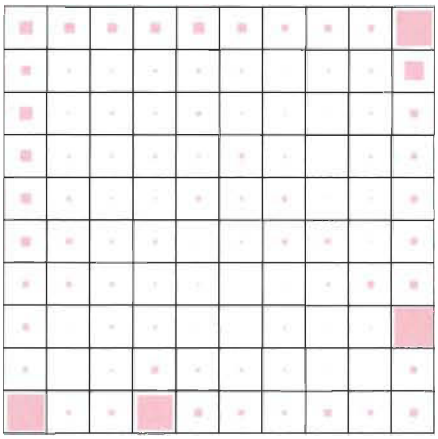
Le modèle gaussien retenu renvoie 3 classes (voir Tableau 3.3) dont la classe 1 qui a un très faible effectif (106) avec un score d'évaluation moyen de 2.82 (peu satisfaits), la classe 2 a un effectif 15 fois supérieur (1677) à celui de la classe 1 et une moyenne d'évaluation de 3.7 (satisfaits). Enfin, la classe 3 qui représente la classe de plus grande taille (4037) avec une moyenne de 3 (neutres ou moyennement satisfaits). Il est important de souligner le score moyen de la classe 1 est très de celui de la classe 3.

Classes	Taille	Moyenne d'évaluation globale
1	106	2.8
2	1677	3.7
3	4037	3

Tableau 3.3 – *Composition des classes selon les modèles de mélange gaussiens*

La carte auto-organisatrice de Kohonen obtenue, est une carte de forme carrée et contient 100 neurones (10 par 10). La figure 3.4 permet une visualisation de la distribution des éléments attirés par chaque neurone (classe).

Plus le carré est dense, plus le neurone correspondant attire les objets à classifier.



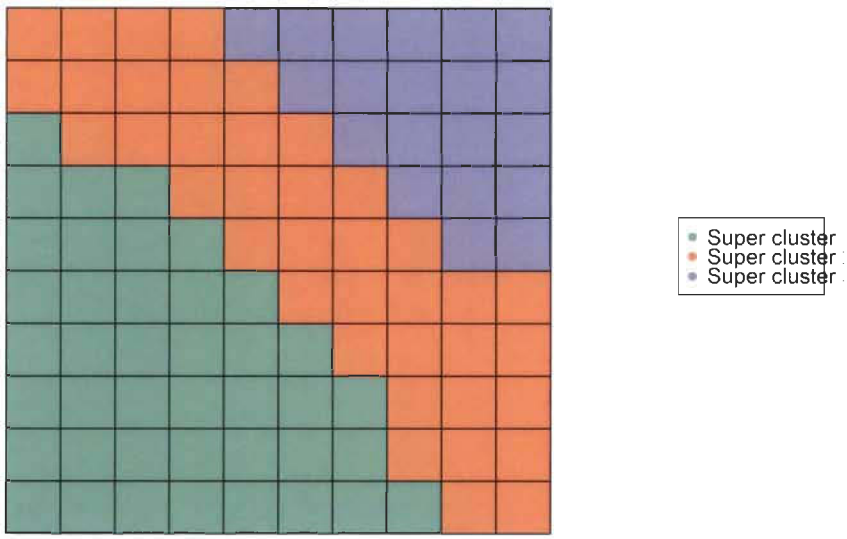
Graphique 3.4 – *Effectif des classes de la carte auto-organisatrice*

Le nombre de classes étant assimilé au nombre de neurones de la carte, donc très élevé. D'où l'idée des super-classes obtenues grâce à un regroupement des 100 classes de la carte par le biais d'une classification ascendante hiérarchique. Ce regroupement est fait donc en 3 classes (nombre optimal). Ces dernières sont matérialisées sur la carte auto-organisatrice après regroupement (figure 3.5).

La situation des classes obtenues après regroupement hiérarchique des classes de la carte de Kohonen est consignée dans le tableau 3.4. La même tendance se dessine par rapport aux résultats précédemment exposés. La classe 1 représente la grande classe des réponses totalement positives (score moyen de 4.2), la classe 2 représente celle des réponses à satisfaction moyenne et la classe 3 qui comptabilise le plus petit effectif caractérise un niveau de satisfaction faible.

Classes	**Taille**	**Moyenne d'évaluation globale**
1	2528	4.2
2	1817	3
3	1475	1.6

Tableau 3.4 – *Composition des classes selon la carte auto-organisatrice de Kohonen*



Graphique 3.5 · *Regroupement des classes de la carte auto-organisatrice en super-classe*

3.3.2 Identification des formes fortes et comparaison

La description des classes permet une concordance de la codification des classes afin de faciliter l'identification des formes fortes. Ainsi, une codification uniforme des classes découlant de chaque méthode s'impose (voir tableau 3.5). Prenons la codification des classes issues de la CAH comme codification de références. En se référant donc à la description des classes, la classe 1 de la CAH peut être assimilée à la classe 2 issus des k-means, à la classe 3 du Kohonen et à la classe 1 du modèle gaussien. La classe 2 de la CAH, quant à elle, peut être assimilée à la classe 1 des k-means, à la classe 3 du modèle de mélange gaussien et à la classe 2 issue du Kohonen. Enfin, la classe 3 de la CAH peut être assimilée à la classe 3 des k-means, à la classe 2 du modèle gaussien et à la classe 1 issue des réseaux de Kohonen.

Pour rappel, les individus qui restent toujours dans la même classe quelle que soit la méthode appliquée sont appelées formes fortes. En prenant en compte toutes les méthodes, les formes fortes identifiées représentent 40.3% de l'ensemble des individus.

CAH	k-means	Méthode gaussienne	Kohonen
1	2	1	3
2	1	3	2
3	3	2	1

Tableau 3.5 – *Correspondance des classes issues des méthodes de classification concernées*

La comparaison des méthodes classiques deux à deux permet de constater que la CAH et la méthode des k-means donnent des résultats très similaires dans la mesure où il en ressort 99.6% de formes fortes, tel qu'illustré dans le tableau 3.6.

	k-means		
CAH	1	2	3
1	**1228**	0	0
2	11	**2346**	0
3	0	12	**2223**

Tableau 3.6 – *Formes fortes résultant de la méthode des k-means et de la CAH*

Tel que spécifié dans le tableau 3.7, le modèle de mélange gaussien comparé, d'une part, à la CAH, et d'autre part, à la méthode des k-means, donne pratiquement des résultats identiques. En effet, les formes fortes représentent 43.1% de l'ensemble des individus en considérant le modèle gaussien et la CAH ou la méthode des k-means.

	MODELE GAUSSIEN		
k-means	1	2	3
1	**6**	1023	210
1	100	**1647**	611
3	0	1367	**856**
CAH	1	2	3
1	**6**	1016	206
1	100	**1645**	612
3	0	1376	**859**

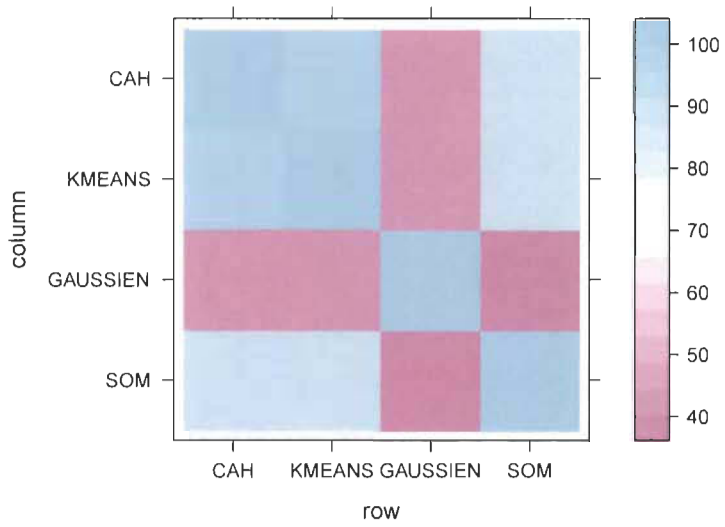
Tableau 3.7 – *Formes fortes résultant de la méthode gaussienne, des k-means et de la CAH*

Pour la comparaison entre les méthodes statistiques classiques et neuronale (voir tableau 3.8 et figure 3.6), il ressort que la méthode neuronale est plus proche des méthodes statistiques à approche géométrique (k-means et CAH) qu'aux méthodes à approche probabiliste (modèle gaussien). En comparant la méthode de Kohonen aux méthodes à approche géométrique (k-means et CAH), le taux de représentation des formes fortes est de 90.5%. Ce taux est à 41% si cette dernière est comparée avec les résultats du modèle gaussien.

	Kohonen		
k-means	1	2	3
1	**1239**	0	0
2	236	**1817**	305
3	0	0	**2223**
CAH	1	2	3
1	**1228**	0	0
2	247	**1817**	293
3	0	0	**2235**
MODELE GAUSSIEN	1	2	3
1	**30**	70	6
2	1112	**1362**	1563
3	333	385	**959**

Tableau 3.8 – *Formes fortes : Comparaison entre la méthode neuronale de Kohonen et les méthodes statistiques classiques*

La figure 3.6 rend plus lisible les résultats du tableau 3.8 en laissant apparaitre une bonne liaison entre les méthodes géométriques (CAH et k-means) et la méthode neuronale (Kohonen). En effet, plus le carré formé par deux méthodes est de couleur bleue, plus ces méthodes ont une bonne représentation des formes fortes. Il apparaît clairement donc que le modèle gaussien est plus susceptible de donner des résultats différents.



Graphique 3.6 – *Comparaison des méthodes statistiques classiques et neuronale selon le taux de représentation des formes fortes*

Cette situation peut être expliquée par la différence des approches de classification. La méthode de Kohonen, tout comme la CAH et la méthode des k-means, utilisent la distance comme mesure de ressemblance entre individus pour classer ceux-ci. Alors que le modèle gaussien stipule qu'au préalable les données suivent des lois normales et que des individus appartenant à la même classe indiquent qu'ils sont tirés de la même distribution. En dehors de la mesure de ressemblance entre individus comme point commun, la méthode de Kohonen et celle des k-means ont des principes quasi-similaires du point de vue algorithmique. En effet, une différence majeure entre ces deux méthodes est qu'avec la méthode des k-means, les nœuds (centres de classes) sont indépendants les uns des autres. Alors qu'avec la méthode de Kohonen, les nœuds (vecteurs prototypes) sont placés sur une grille et chaque vecteur est considéré comme ayant des voisins. En outre, la mise à jour du vecteur gagnant provoque en même temps une modification de ses voisins. Ainsi, la méthode des k-means peut être considérée comme un cas particulier de la méthode de Kohonen si aucun voisin n'est pris en compte lors de la modification des vecteurs prototypes.

Conclusion et perspectives

L'objectif de ce mémoire est d'établir un cadre de comparaison des méthodes statistiques et neuronale en classification non supervisée sur un jeu de données réelles et ce, en utilisant le logiciel libre R. Qu'il s'agit du premier ou du deuxième cas, le but reste le même, à savoir, la détermination d'une partition d'un ensemble d'individus en plusieurs classes homogènes. L'idée globale qui ressort de cette étude est que les méthodes statistiques qui utilisent la distance comme critère de ressemblance des individus, sont plus similaires à la méthode de Kohonen qu'aux méthodes statistiques à approche probabiliste. Cette situation peut être expliquée par la diversité de l'approche d'une méthode à l'autre, d'autant plus que la méthode de Kohonen utilise la distance comme mesure de ressemblance entre individus. Du point de vue algorithmique, la méthode des k-means est similaire à la méthode Kohonen dans la mesure où les vecteurs prototypes de la carte auto-organisatrice se comportent presque de la même manière que les centres mobiles définis dans les initialisations de chaque classe issue des k-means. Il faut souligner que le modèle gaussien, bien qu'il soit performant, suppose déjà que dans chaque classe, la distribution est unique et elle suit une loi normale. Ce qui n'est pas le cas pour un bon nombre d'ensemble de données réelles. Par ailleurs, la méthode des k-means, bien qu'elle soit simple et facile à utiliser, présente aussi quelques inconvénients, à savoir l'obligation de spécifier le nombre de classes dès le début. De plus, elle ne peut analyser que des données numériques. Ces inconvénients ne rendent pas impossible son utilisation, mais peuvent affecter les résultats car ceux-ci dépendent de la configuration initiale de l'algorithme. Un avantage de la CAH par

rapport à la méthode des k-means est qu'on peut choisir une mesure de ressemblance (distance ou dissimilarité ou similarité) adaptée aux données étudiées. Aussi, la lecture du dendrogramme permet de déterminer le nombre optimal de classes, mais sa mise en œuvre est trop couteuse en temps de calcul. Plus la taille des données devient important, moins la CAH est commode. Contrairement à ces méthodes classiques, la méthode de Kohonen a une capacité à s'adapter à des cas complexes grâce à sa capacité d'apprentissage moyennant un calibrage de ses paramètres (configuration de la carte, sa dimension, nombre de neurones etc). Son grand avantage est qu'elle nécessite moins de temps de calcul, ce qui la rend plus simple à utiliser.

Cette étude s'est limitée à la comparaison des méthodes classiques avec la méthode neuronale de Kohonen. Il serait aussi intéressant de se pencher à comparer les méthodes classiques avec d'autres méthodes, qui sont très utilisées telle que le Support vector machine (SVM) par apprentissage non supervisée. Aussi, une autre perspective de ce travail consiste à établir, sur un cadre théorique, les liens entre la méthode de Kohonen et celle des k-means.

Bibliographie

[1] Forgy E. Cluster analysis of multivariate data : efficiency versus interpretability of classifications. *Biometrics*, 21 :768–780, 1965.

[2] MacQueen J. B. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[3] Diday E. Une nouvelle méthode en classification automatique et reconnaissance des formes la méthode des nuées dynamiques. *Revue de Statistique Appliquée*, 19(2) :19–33, 1971.

[4] Kaufman L. and Rousseeuw P.J. *Finding Groups in Data : an introduction to cluster analysis.* Wiley, New York, 1990.

[5] Maechler M., Rousseeuw P., Struyf A., and Hubert M. Cluster analysis basics and extensions. 2005.

[6] Husson F., Josse J., and Pages J. Principal component methods - hierarchical clustering - partitional clustering : why would we need to choose for visualizing data ? 2010.

[7] Banfield J. D. and Raftery A. E. Model-based gaussian and non-gaussian clustering. *Biometrics*, 49 :803–821, 1993.

[8] Celeux G. and Govaert G. Gaussian parsimonious clustering models. Technical Report RR-2028, INRIA, Sep 1993.

[9] Dempster A. P., Laird N. M., and Rubin D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society, series b*, 39(1) :1–38, 1977.

[10] Biernacki C., Celeux G., and Govaert G. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7) :719-725, July 2000.

[11] Biernacki C. Pourquoi les modèles de mélange pour la classification ? *Revue MODULAD*, 20 :1–13, 2009.

[12] Fraley C., Raftery A., Murphy T., and Scrucca L. Mclust version 4 for r : Normal mixture modeling for model-based clustering, classification, and density estimation. *Technical Report No. 597*, 01 2012.

[13] Charrad M., Ghazzali N., Boiteau V., and Niknafs A. NbClust : An R package for determining the relevant number of clusters in a data set. *Journal of Statistical Software*, 61(6) :1–36, 2014.

[14] F. Rosenblatt. The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6) :386–408, 1958.

[15] Minsky M. and Papert S. *Perceptrons*. MIT Press, Cambridge, MA, 1969.

[16] Kohonen T. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43 :59–69, January 1982.

[17] Ritter H. and Schulten K. On the stationary state of kohonen's self-organizing sensory mapping. *Biological Cybernetics*, 54 :99–106, 1986.

[18] Vialaneix N., Mariette J., Olteanu M., Rossi F., Bendhaiba L., and Bolaert J. *SOMbrero : SOM Bound to Realize Euclidean and Relational Outputs*, 2019.

[19] Rossi R.A. and Nesreen K. A. turkiye student evaluation r specific - educational data, 2013.

ANNEXES

Annexe A : Liste des trente (30) indices utilisés dans le package Nbclust

N	**Noms des indices utilisés dans Nbclust**	**Nombre optimal de classe**
1.	"ch" (Calinski and Harabasz 1974)	Valeur maximale de l'indice
2.	"duda" (Duda and Hart 1973)	Le plus petit nombre tel que l'indice> la valeur critique
3.	"pseudot2" (Duda and Hart 1973)	nombre de groupes plus petits tels que l'indice < à la valeur critique
4.	"cindex" (Hubert and Levin 1976)	Valeur minimale de l'indice
5.	"gamma" (Baker and Hubert 1975)	Valeur maximale de l'indice
6.	"beale" (Beale 1969)	Le nombre de groupe tel que la valeur critique> = alpha
7.	"ccc" (Sarle 1983)	Valeur maximale de l'indice
8.	"ptbiserial" (Milligan 1980, 1981)	Valeur maximale de l'indice
9.	"gplus" (Rohlf 1974; Milligan 1981)	Valeur minimale de l'indice
10.	"db" (Davies and Bouldin 1979)	Valeur minimale de l'indice

11.	"frey" (Frey and Van Groenewoud 1972)	Le niveau correspondant à un indice avant valeur de l'indice <1
12.	"hartigan" (Hartigan 1975)	Différence maximale entre les niveaux de la hiérarchie de l'indice
13.	"tau" (Rohlf 1974 ; Milligan 1981)	Valeur maximale de l'indice
14.	"ratkowsky" (Ratkowsky and Lance 1978)	Valeur maximale de l'indice
15.	"scott" (Scott and Symons 1971)	Différence maximale entre les niveaux de la hiérarchie de l'indice
16.	"marriot" (Marriot 1971)	La valeur maximale des différences secondes entre les niveaux de l'indice
17.	"ball" (Ball and Hall 1965)	Différence maximale entre les niveaux de la hiérarchie de l'indice
18.	"trcovw" (Milligan and Cooper 1985)	Différence maximale entre les niveaux de la hiérarchie de l'indice
19.	"tracew" (Milligan and Cooper 1985)	La valeur maximale des différences secondes entre les niveaux de l'indice
20.	"friedman" (Friedman and Rubin 1967)	Différence maximale entre les niveaux de la hiérarchie de l'indice
21.	"mcclain" (McClain and Rao 1975)	Valeur minimale de l'indice
22.	"rubin" (Friedman and Rubin 1967)	Valeur mininale de secondes différences entre les niveaux
23.	"kl" (Krzanowski and Lai 1988)	Valeur maximale de l'indice
24.	"silhouette" (Rousseeuw 1987)	Valeur maximale de l'indice
25.	"gap" (Tibshirani et al. 2001)	Le plus petit nombre tel que la valeur critique $>= 0$

26.	"dindex" (Lebart et al. 2000)	méthode graphique
27.	"dunn" (Dunn 1974)	Valeur maximale de l'indice
28.	"hubert" (Hubert and Arabie 1985)	méthode graphique
29.	"sdindex" (Halkidi et al. 2000)	Valeur minimale de l'indice
30.	"sdbw" (Halkidi and Vazirgiannis 2001)	Valeur minimale de l'indice

Annexe B : Score moyen des questions dans chaque classe

Questions	Classe 1	Classe 2	Classe 3
Q1	1.396287	2.631891	4.100765
Q2	1.403551	2.831637	4.261808
Q3	1.509282	3.003817	4.294647
Q4	1.435835	2.849449	4.247413
Q5	1.390638	2.868533	4.31354
Q6	1.408394	2.898643	4.275753
Q7	1.396287	2.814249	4.264507
Q8	1.379338	2.786684	4.239316
Q9	1.502825	2.966497	4.304543
Q10	1.368846	2.836302	4.320288
Q11	1.467312	3.000848	4.334683
Q12	1.380952	2.784563	4.224022
Q13	1.419693	3.1162	4.393162
Q14	1.448749	3.20017	4.413855
Q15	1.467312	3.188295	4.406658
Q16	1.357546	2.983036	4.377418
Q17	1.622276	3.371077	4.417454
Q18	1.400323	3.080153	4.389114
Q19	1.430993	3.141645	4.409357
Q20	1.424536	3.194656	4.418803
Q21	1.463277	3.223494	4.424202
Q22	1.48184	3.227311	4.436347
Q23	1.393059	3.031807	4.390463
Q24	1.386602	2.97922	4.358075
Q25	1.495561	3.223919	4.419253

Q26	1.430993	3.074215	4.377418
Q27	1.407587	2.982188	4.311741
Q28	1.49314	3.22307	4.409807

ANNEXE C :SCRIPT DE L'ETUDE SOUS R

```
#CHARGEMENT DES PACKAGES NECESSAIRES POUR L'ETUDE
library(mclust)
library(NbClust)
library(stats)
library(cluster)
library(SOMbrero)
library(kohonen)
library(FactoMineR)
library(ggplot2)
library(colorspace)
library(dendextend)
library(cluster)
library(fpc)
library(clust)

#DEFINITION DU REPERTOIRE DE TRAVAIL ET CHARGEMENT DES DONNEES
setwd("C:/Users/HP/Desktop/DATA SET")
mydata<- read.table("evaluation.csv", dec=".", header=TRUE, sep=";")

#CREATION DU NOUVEAU JEU DE DONNEES EN ISOLANT LES 6 PREMIERES
VARIABLES QUALITATIVES
evaluation<-mydata[,7:34]
class_labels<- thyroid[,5]

#création de couleurs arc-en-ciel pour la caractérisation des classes
class_col <- rev(rainbow_hcl(3))[as.numeric(class_labels)]

##VISUALISATION
```

```
# Plot
pairs(thyroid2, col = class_col,
lower.panel = NULL,
cex.labels=2, pch=19, cex = 1.2)

# legend
par(xpd = TRUE)
legend(x = 0.05, y = 0.4, cex = 2,
legend = as.character(levels(class_labels)),
fill = unique(class_col))
par(xpd = NA)



set.seed(20)


######STATISTIQUES DESCRIPTIVES#####
#Nombre de réponse par instructeur
ggplot(mydata, aes(x=instr))+
geom_histogram(color="darkblue", fill="lightblue")

# Nombre de réponses par cours
ggplot(mydata, aes(x=class))+
geom_histogram(color="darkblue", fill="lightblue")

####################METHODES STATISTIQUES DE CLASSIFICATION##############

###### UTILISATION DE NBCLUST POUR LA DETERMINATION DU NOMBRE DE CLASSES####
###nbclust
```

```
res <- NbClust(evaluation, distance = "euclidean", min.nc = 2,
 max.nc = 8, method ="ward.D2", index = "all")
res$Best.nc
res$All.CriticalValues
res$Best.partition

#############CAH

#Méthode 1: Hclust
#Contruction de la matrice de distance
d<-dist(evaluation)
cah<-hclust(d, method="complete")
dend <- as.dendrogram(cah)
# Couleurs des branches en se basant sur les classes
dend <- color_branches(dend, k=3)
#plot
par(mar = c(3,3,3,7))
plot(dend,
main = "Classification évaluation",
horiz =  TRUE,)


rect.hclust(cah,k=3)


#Méthode 2: Agnes
M2 <- agnes(evaluation, metric="euclidean", method = "complete")
M2
plot(M2)
```

```
#Méthode 3: HCPC
#Application d'une méthode d'analyse factorielle (ACP par exemple)
res.pca = PCA(evaluation)
#Classification hierarchique sur les résultats de l'ACP
res.hcpc = HCPC (res.pca)

##Effectif par classe CAH
table(res.hcpc$data.clust$clust)

#creer une colonne pour les classes dans le fichier original
mydata$clustCAH<-res.hcpc$data.clust$clust
write.csv(mydata, file="CAHeva.csv", row.names=TRUE)

### MOyenne de l'appréciation globale par classe
aggregate(mydata[,6:33], list(mydata$clustCAH), mean)

######CLASSIFICATION NON HIERARCHIQUE
######KMEANS
set.seed(100)
evaluation_kmeans<-as.matrix(evaluation)
kms <- kmeans(evaluation_kmeans, 3, nstart = 20)
kms
kms$cluster #indique le numéro de la classe de chaque individu
kms$centers #fournit la moyenne des variables dans chacune des classes
kms$size    #donne les tailles des classes obtenues.


##Visualisation des classes crées
kms$cluster <- as.factor(kms$cluster)
plotcluster(evaluation, kms$cluster)
```

```
#creer une colonne pour les classes dans le fichier original
mydata$clustKms<-kms$cluster
write.csv(mydata, file="KmeansEVA.csv", row.names=TRUE)

################ASPECT PROBABILISTE: METHODE DE MELANGE GAUSSIEN##########
eva_prob<-Mclust(evaluation, G=3)
summary(eva_prob, parameters = TRUE)
plot(eva_prob, what = c("BIC", "classification"))

#Mélanges gaussiens avec les paramètres estimés : Méthode BIC
#BIC <- mclustBIC(Thyroprob)
#summary(BIC)
#Mélanges gaussiens avec les paramètres estimés : Méthode ICL
ICL<-mclustICL(eva_prob)
summary(ICL)

#creer une colonne pour les classes dans le fichier original
mydata$Mclusts<-eva_prob$classification
write.csv(mydata, file="MclustsEVA.csv", row.names=TRUE)

###############TRain som##########
eva_som <- trainSOM(x.data=evaluation, verbose=TRUE, nb.save=5)

#composition classe
eva_som$clustering
###effectif classe. qui peut également être visualisé par un
 graphique hitmap:
table(eva_som$clustering)
```

```
#### graphique hitmap
plot(eva_som, what="obs", type="hitmap")
### information sommaire (Anova)
summary(eva_som)

#### classification par variable
par(mfrow=c(2,2))
plot(eva_som, what="obs", type="color", variable=1, print.title=TRUE,
main="T3resin")
#radar
plot(eva_som, what="obs", type="radar")
## graphique avec les composantes des classes
plot(eva_som, what="obs", type="names", print.title=TRUE, scale=c(0.9,0.5))

### Caractérisation par la 5e variable (nom des classes) étiquettage
#plot(thyrosom, what="add", type="grid", variable=thyroid$Classe)

##Construction de super-classes à partir du fichier SOM
my.sc <- superClass(eva_som, k=3)
summary(my.sc)
### dendrogramme avec classes identifiés
plot(my.sc, plot.var=FALSE)
### graphique couleurs classes
plot(my.sc, type="grid", plot.legend=TRUE)

#creer une colonne pour les classes dans le fichier original
cluster_assign <- my.sc$cluster[eva_som$clustering]
mydata$cluster1 <- cluster_assign

#EXPORTATION DU FICHIER FINAL SOUS UN FORMAT CSV
```

```
write.csv(mydata, file="som_eva.csv", row.names=TRUE)
table(my.sc$cluster)
table(mydata$cluster1)

#SCORE MOYENS DES ETUDIANTS DANS CHAQUE CLASSE
aggregate(mydata[,6:33], list(mydata$cluster1), mean)
```