UNIVERSITÉ DU QUÉBEC

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

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN MATHÉMATIQUES ET INFORMATIQUE APPLIQUÉES

PAR
Abdou Aziz Niakh

CLUSTERING, CO-CLUSTERING ET RÉSEAUX DE NEURONES ARTIFICIELS : APPLICATION EN SCIENCE DE DONNÉES

Avril 2023

Université du Québec à Trois-Rivières

Service de la bibliothèque

Avertissement

L'auteur de ce mémoire, de cette thèse ou de cet essai a autorisé l'Université du Québec à Trois-Rivières à diffuser, à des fins non lucratives, une copie de son mémoire, de sa thèse ou de son essai.

Cette diffusion n'entraîne pas une renonciation de la part de l'auteur à ses droits de propriété intellectuelle, incluant le droit d'auteur, sur ce mémoire, cette thèse ou cet essai. Notamment, la reproduction ou la publication de la totalité ou d'une partie importante de ce mémoire, de cette thèse et de son essai requiert son autorisation.

Résumé

Le partitionnement et la classification s'avèrent très importants dans les traitements des données massives et la prise des décisions en statistique. L'objectif général de ce mémoire est d'effectuer le lien entre trois méthodes de classification : la classification non supervisée (clustering), co-clustering aussi appelé biclustering et les réseaux de neurones artificiels afin de faire une étude comparative. Ces méthodes ont le même objectif, mais leurs approches diffèrent. La première partie consiste à faire un rappel sur les notions de base d'algèbre. Ensuite, on étudie la théorie des méthodes de clustering et co-clustering, suivie d'une illustration avec des jeux de données. Il s'agit de données sur les CVs des professeurs québécoises des universités, les «Iris de Fisher» et l'ensemble des données Wine. Cette partie a permis de mieux comprendre la théorie de co-clustering et ses applications en science de données. La deuxième partie présente la méthode de Kohonen qui s'apparente à une classification non supervisée. Cela nous aidera à comprendre le fonctionnement des réseaux de neurones artificiels en classification, mais aussi le lien formel qu'il a avec le co-clustering. Les applications de ces méthodes sont réalisées par le logiciel R et Python.

Abstract

Partitioning and classification are very important in mass data processing and statistical decision making. The general objective of this paper is to relate three classification methods : unsupervised classification (clustering), co-clustering (also called biclustering) and artificial neural networks in order to make a comparative study. These methods have the same objective, but their approaches differ. The first part consists of a reminder of the basic notions of algebra. Then, the theory of clustering and co-clustering methods is studied, followed by an illustration with data sets. These are data on the CVs of Quebec university professors, the ≪Iris of Fisher≫ and the Wine data set. This part provided a better understanding of co-clustering theory and its applications in data science. The second part presents Kohonen's method which is similar to unsupervised classification. This will help us understand how artificial neural networks work in classification, but also the formal link they have with co-clustering. The applications of these methods are carried out using R and Python.

Avant-propos

Tout d'abord, je tiens à adresser mes sincères remerciements à ma directrice de recherche, Madame Nadia Ghazzali, pour avoir accepté de diriger mon mémoire de recherche. Je voudrais également la remercier pour son appui rigoureux durant tout l'encadrement, ces encouragements et son soutien financier. Merci infiniment pour les efforts mis dans la lecture et les corrections de forme et de fond effectuées, sans lesquelles ce document ne pourrait atteindre ce niveau de précision. Je remercie aussi MEES (Ministre de l'Education et l'Enseignement supérieur) pour leur appui financier.

L'enseignement de qualité dispensé en maîtrise de mathématique et informatique appliquée a également pu nourrir mes réflexions et a représenté une profonde satisfaction intellectuelle, merci donc au corps professoral.

Mes remerciements vont aussi à l'endroit de mes parents qui m'ont toujours accompagnés, orientés et motivés depuis le début de mes études et un grand merci également à mes frères et sœurs. Enfin, je voudrais exprimer ma reconnaissance envers les amis et professeurs qui m'ont apporté leur soutien moral et intellectuel tout au long de ma démarche qui sont près ou loin.

Table des matières

Liste des tableaux

Table des figures

Introduction

L'analyse des données permet de traiter un nombre très important de données. Depuis l'époque de Louis XIV et même avant, on trouve l'existence de tableaux de données et de techniques descriptives. Différentes méthodes simples sont connues depuis lors, jusqu'aux ébauches de la classification nées dans les années 1740.

La classification non supervisée désigne un ensemble de méthodes ayant pour objectif de dresser ou de retrouver une typologie existante caractérisant un ensemble de n observations, à partir de p caractéristiques mesurées sur chacune des observations. Rappelons que les données ne sont pas étiquetées. Le « clustering non supervisé » aussi appelé classification non supervisée, est un processus qui permet de rassembler des données similaires. Il consiste à regrouper un ensemble d'objets tel que les objets d'une même classe (appelé « **cluster**») soient plus proches les uns aux autres qu'à ceux d'autres classes.

Le co-clustering est une méthode de classification qui consiste à effectuer une partition simultanée des lignes et des colonnes d'une matrice en utilisant une partition Z de l'ensemble I de lignes et une partition W de l'ensemble J de colonnes. Dans ce cas, I peut être l'ensemble des individus et J l'ensemble des variables. Pour effectuer cette partie, on s'appuie sur les travaux de G. Govaert and M. Nadif [7]. Pour cette méthode, on doit identifier d'abord les types de données avant de pouvoir faire un partitionnement. Les types de données peuvent être binaires, continues ou des tableaux de contingences.

À ces méthodes, s'ajoute la méthode de Kohonen qui est caractérisée par une carte

auto-organisatrice appelée *Self Organizing Map (SOM)* [16] et basée sur les réseaux de neurones artificiels. L'historique de ces derniers a débuté en 1943 avec Warren McCulloch et Walter Pitts [21] qui ont créé un modèle de réseaux de neurones basé sur des algorithmes afin de modéliser le fonctionnement du neurone biologique. Cependant, les premières étapes de la classification non supervisée avec les réseaux de neurones artificiels, ont débuté avec le réseau de Teuvo Kohonen [16] en 1982. Ce dernier a remarqué que, dans le cerveau des chats, une aire était spécialisée dans le traitement des signaux acoustiques. L'aire spécifique était constituée de neurones ordonnés suivant les fréquences acoustiques auxquelles ils sont sensibles. Ainsi les neurones activés par les hautes fréquences sont localisés à l'opposé des neurones sensibles aux basses fréquences.

L'objectif principal de ce mémoire est de contribuer à faire le lien entre le clustering, le co-clustering qui sont des méthodes statistiques et la méthode neuronale de Kohonen en classification. Ce travail est articulé en quatre chapitres. Après l'introduction, le chapitre 1 présente les modèles de clustering en passant en revue des bases d'algèbre linéaire. Le chapitre 2 est consacré aux méthodes de co-clustering. Le chapitre 3, quant à lui, concerne la méthode de classification neuronale, plus précisément la méthode de Kohonen. Le chapitre 4 est dédié à une étude comparative de ces trois méthodes où seront présentés et discutés les résultats obtenus. Une conclusion clôturera ce mémoire.

Chapitre 1

Méthode de clustering

Le clustering est devenu un outil important dans une variété de domaine scientifique y compris le regroupement et la classification des données. Ce chapitre étudie la classification non supervisée dite clusgtering dont le but est de former des classes homogènes ou classes naturelles en s'assurant que les objets d'une classe sont similaires les uns aux autres. Pour ce faire, on applique différents types de méthodes. Mais avant de voir ces méthodes de clustering, nous allons étudier quelques notions de base d'algébre linéaire qui seront utiles pour les chapitres suivants.

1.1 Rappels d'algèbre

Nos rappels portent sur les notions d'espaces vectoriels, de matrices et de distances.

1.1.1 Espace vectoriel

Soit un ensemble E (par exemple $\mathbb{R}^n$ ou $\mathbb{C}^n$) muni des opérations : $+$: l'addition (dit loi de composition interne), $\times$: la multiplication (dit loi de composition externe) avec un scalaire réel ou complexe (en fonction de l'ensemble de départ).
On dit que l'ensemble $(E, +, \times)$ possède une structure d'espace vectoriel (e v) sur $\mathbb{R}$ (respectivement $\mathbb{C}$) si on peut définir les conditions suivantes :

- $\forall x, y \in E \Longrightarrow x + y \in E$,
- $\forall x \in E, \lambda \in \mathbb{R}\ (ou\ \mathbb{C}) \Longrightarrow \lambda x \in E$,
- $(E, +)$ groupe commutatif (associativité, commutativité, élément neutre et opposé)
- Les propriétés ci-dessous sont celles de la multiplication $\times$

$$\left.\begin{array}{l} \lambda \times (x + y) = \lambda \times x + \lambda \times y \\ (\lambda + \mu) \times x = \lambda \times x + \mu \times x \end{array}\right\} distributivité \tag{1.1}$$

$$\left.\begin{array}{l} (\lambda\mu) \times x = \lambda(\mu \times x) = \mu(\lambda \times x) \end{array}\right\} associativité \tag{1.2}$$

1.1.2 Matrice

Soit $A = (a_{ij})_{\substack{1 \leq i \leq n \\ 1 \leq j \leq d}}$ d'ordre (n, d). On appelle **transposée** de A la matrice d'ordre (d, n) notée A^T telle que : $A^T = a_{ji}$. Une matrice est dite **symétrique** si : $A^T = A$.
Une matrice B est dite inverse de A si $AB = BA = I_n$
I correspond à la matrice identité est **l'élément neutre** de la multiplication matricielle.
L'utilisation de la notion de **déterminant** amènera à l'introduction du polynôme caractéristique qui fait sortir les vecteurs propres. On peut calculer le determinant de

$A = (a_{ij})_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}}$ par :

$$det(A) = \sum_{j} (-1)^{j+1} a_{ij} det(B) \tag{1.3}$$

Ou $B = A/(A_1, A_j)$ est la sous matrice obtenue en supprimant la première ligne et les j-ième colonnes de A.

La **trace** de la matrice A correspond à la somme des éléments de sa diagonale principale, c'est-à-dire :

$$Tr(A) = \sum_{i=1}^{n} a_{ii} \tag{1.4}$$

Considérons A comme une matrice d'une application linéaire f d'un espace vectoriel E de dimension p dans un espace vectoriel F de dimension n, tel que les colonnes de A sont les p vecteurs qui sont engendrés par $f(E)$, image de E. Le **rang** de A, noté $rg(E)$ est alors celui de f et s'écrit :

$$rg(f) = dim(f(E)) \tag{1.5}$$

Rappelons que le rang de la matrice A est le nombre de colonnes de A linéairement indépendantes.

Valeurs propres, vecteurs propres

Soit $A = (a_{ij})$ une matrice carrée d'ordre n. Le polynôme caractéristique permet de calculer les valeurs propres. Il est défini par :

$$p(\lambda) = det(A - \lambda I) = 0 \tag{1.6}$$

Les solutions trouvées λ de cette équation sont **les valeurs propres** de la matrice A.

On appelle **vecteurs propre** de A tout vecteur u non nul de $\mathbb{R}^n$ respectivement ($\mathbb{C}^n$)

tel qu'il existe un scalaire vérifiant :

$$Au = \lambda u \tag{1.7}$$

Ainsi un vecteur $u = (u_1, u_2, ..., u_n)$ associé à la valeur λ est déterminé en résolvant l'équation suivante :

$$(A - \lambda I)u = 0 \tag{1.8}$$

Remarques :

1) Une valeur propre est associée à un seul vecteur propre.

2) Un vecteur propre est défini à une constante près.

Diagonalisation

Soit $A = (a_{ij})$ une matrice carrée d'ordre n. On dit que A est diagonalisable si elle est semblable à une matrice diagonale D. Autrement dit s'il existe une matrice inversible P dite matrice de passage telle que :

$$A = PDP^{-1} \tag{1.9}$$

Diagonaliser une matrice A, revient à trouver une matrice inversible P, telle que $P^{-1}AP$ soit diagonale.

1.1.3 Mesure de ressemblance

En général, le choix d'une mesure de ressemblance dépend essentiellement de la nature des éléments à étudier [1]. Dans ce cas, on étudiera soit la distance, soit la similarité ou la dissimilarité.

Définition 1 : Une distance est une fonction d qui à tout couple (x,y) de $\mathbb{R}^+$, vérifie

les conditions suivantes :

1) $d(x, y) = d(y, x) \geq 0$, (symétrie)

2) $d(x, y) = 0 \Rightarrow x = y$, (séparation)

3) $d(x, z) \leq d(x, y) + d(y, z)$, (inégalité triangulaire)

D'après cette définition, si la condition 3) n'est pas vérifiée, on parlera d'indice de **dissimilarité** (ou mesure de dissimilarité).

Définition 2 : Une similarité est une fonction S qui à tout couple (x, y) associé une valeur dans $\mathbb{R}^+$, et telle que

a) $S(x, y) \geq 0$ (positivité)

b) $S(x, y) = s(y, x)$; (symétrie)

c) $S(x, x) \geq s(x, y)$. (maximalité)

Contrairement à la dissimilarité, plus les unités x et y se ressemblent plus la valeur est élevée.

Il existe plusieurs mesures de distances,les plus utilisées sont notamment :

– **Distance euclidienne :** Elle est définie entre deux vecteurs x et y par

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{1.10}$$

– **Distance maximum :** C'est la distance d entre deux composantes des vecteurs x et y, définie de la manière ci-dessous :

$$d(x, y) = \max_{1 \leq i \leq n} \mid x_i - y_i \mid \tag{1.11}$$

– **Distance de Manhattan :** C'est la distance d entre deux composantes des vecteurs x et y, définie de la manière ci-dessous :

$$d(x,y) = \sum_{i=1}^{n} \mid x_i - y_i \mid \tag{1.12}$$

– **Distance de Minkowski :** Cette distance représente la p-ième racine de la somme des différences entre les composantes des vecteurs x et y à la puissance p. Rappelons que si $p = 1$ on retrouve la distance de Manhattan et si $p = 2$ on retrouve la distance euclidienne. Elle s'écrit sous la forme ci-dessous :

$$d(x,y) = (\sum_{i=1}^{n} \mid x_i - y_i \mid^p)^{\frac{1}{p}} \tag{1.13}$$

Pour illustrer les différences entre ces mesures, on prend un exemple de deux vecteurs qui sont x et y tel que :

$$x = (0, 2, -1) \; et \; y = (7, 6, -4) \tag{1.14}$$

– Calculons d'abord la distance euclidienne :

$$d(x,y) = \sqrt{(0-7)^2 + (2-6)^2 + (-1-(-4))^2} = \sqrt{49+16+9} = \sqrt{74} = 8,60 \tag{1.15}$$

– Calcul de la distance maximum :

$$d(x,y) = \max(\mid 0-7 \mid, \mid 2-6 \mid, \mid -1-(-4) \mid) = \max(7,4,3) = 7 \tag{1.16}$$

– Calcul de la distance de Manhattan :

$$d(x,y) = \mid 0-7 \mid + \mid 2-6 \mid + \mid -1-(-4) \mid = 7+4+3 = 14 \tag{1.17}$$

– Calcul de la distance de Minkowski (si $p = 1$ on retrouve la distance de Man-

hattan et si $p = 2$ on retrouve la distance euclidienne. On teste $p = 3$) :

$$d(x, y) = (\mid 0 - 7 \mid^3 + \mid 2 - 6 \mid^3 + \mid -1 - (-4) \mid^3)^{\frac{1}{3}} = 7,57 \tag{1.18}$$

1.2 Diverses méthodes de clustering

Il existe plusieurs méthodes de clustering pour faire la classification non supervisée. Dans ce chapitre, nous nous intéressons aux méthodes hiérarchiques et la méthode de k-means.

1.2.1 Approche hiérachique

Le but des méthodes hiérarchiques est de construire une séquence de partitions d'un ensemble allant des partitions de singletons à l'ensemble entier. Il existe différents types d'approche hiérarchique, mais au cours de notre travail nous allons aborder la classification hiérarchique ascendante (CAH).

Principe de base de la CAH

– A l'étape initiale, les n individus constituent des classes à eux seuls.
– On calcule les distances deux à deux entre individus, et les deux individus les plus proches sont réunis en une classe.
– La distance entre cette nouvelle classe et les $n - 2$ individus restants est ensuite calculée, et à nouveau les deux éléments (classes ou individus) les plus proches sont réunis.

Cette démarche est réitérée jusqu'à ce qu'il ne reste plus qu'une unique classe constituée

de tous les individus. On constate que la nouveauté ici vient de la nécessité de définir deux distances : celle entre individus et l'autre entre classes. Le choix d'une distance entre individu ayant déjà été discutée, il reste à choisir une autre entre classe, en gardant à l'esprit que l'objectif est de trouver la partition en K classes des observations dont l'inertie intra-classe est minimale.

L'approche ascendante hiérarchique vérifie les propriétés suivantes :
-$\Omega \in H$: au sommet de la hiérarchie, tous les éléments sont dans la même classe ;
-$\forall \omega \in \Omega, \omega \in H$: en bas de la hiérarchie, tous les éléments constituent une classe à eux seuls ;
-$\forall (h, h') \in H^2, h \bigcap h' = \emptyset$ ou $h \subset h'$ ou $h' \subset h$: en considérant deux classes du regroupement, soit elles n'ont pas d'individus en commun, soit l'une est incluse dans l'autre.
Avec H la hiérarchie construite, c'est-à-dire l'ensemble des classes produites à chaque étape de l'algorithme.
Plusieurs critéres d'agrégations des classes existent. On cite celui basé sur la méthode de Ward très utilisée en classification

Méthode de Ward : Considérons l'évolution de l'inertie intra-classe au fur et à mesure de la classification. À l'initialisation, toutes les classes sont composées d'une unique observation. Chaque classe est donc parfaitement homogène, et l'inertie intra-classe est nulle. À la dernière étape de l'algorithme, toutes les observations sont regroupées pour former une unique classe. L'objectif étant d'aboutir à une partition en K classes d'inertie intra-classe minimum, la stratégie va consister à regrouper à chaque étape les deux classes dont la fusion entraine le plus faible gain d'inertie intra-classe (ou de manière équivalente la plus faible perte d'inertie inter-classe). La distance entre deux classes A et B est celle entre leur barycentre au carré pondéré par les effectifs n_A et n_B des deux groupes. Elle pré-suppose l'existence d'une distance euclidienne.

$$\Delta(A, B) = \frac{n_A n_B}{n_A + n_B} d^2(g_A, g_B) \tag{1.19}$$

avec g_A(resp. g_B) le barycentre de la classe A (resp. B) et d^2 leur distance euclidienne. Le résultat est représenté sous forme d'un arbre de classification.

Dendrogramme

Les dendrogrammes correspondent à la représentation graphique de l'arbre hiérarchique généré par différentes fonctions. Pour ce faire, on a fait une brève description théorique d'un dendrogramme avant de l'appliquer sur un jeu de données. Sur le graphique des pourcentages d'inertie expliquée en fonction du nombre de classes, on cherche un décrochement suivi d'une croissance régulière plus réduite. On sélectionne alors le nombre de classe juste avant ce décrochement.
Comme le nom de la méthode (CAH) l'indique, les partitions successivement obtenues sont hiérarchisées : elles sont emboitées les unes dans les autres. De ce fait, il est possible de représenter l'historique des différentes étapes de l'algorithme à l'aide d'une arborescence, aussi appelée dendrogramme. La figure 1.1 montre l'arborescence liée à la CAH. En bas de l'arbre se trouvent les individus, jouant le rôle des feuilles. La fusion de deux éléments est représentée par une branche reliant ces deux éléments, dont la hauteur est proportionnelle à la distance entre les deux éléments fusionnés. Ainsi, la plus petite branche de l'arbre relie les individus, qui furent les premiers fusionnés. Plus l'on remonte sur l'arbre, plus les classes sont hétérogènes et les branches s'allongent. Dans le cas de la figure 1.1 correspondant à l'exemple considéré, on a choisi de représenter l'arbre avec des hauteurs de branches proportionnelles au pourcentage de perte d'inertie I.

$$\frac{I_\omega^{k+1} - I_w^k}{I_T}$$

Avec $K = 1, ..., n-1$ pour n individus, I_w représente l'inertie intra-classe (pour Within), I_B représente l'inertie inter-classe (pour Between) et I_T représente l'inertie totale.

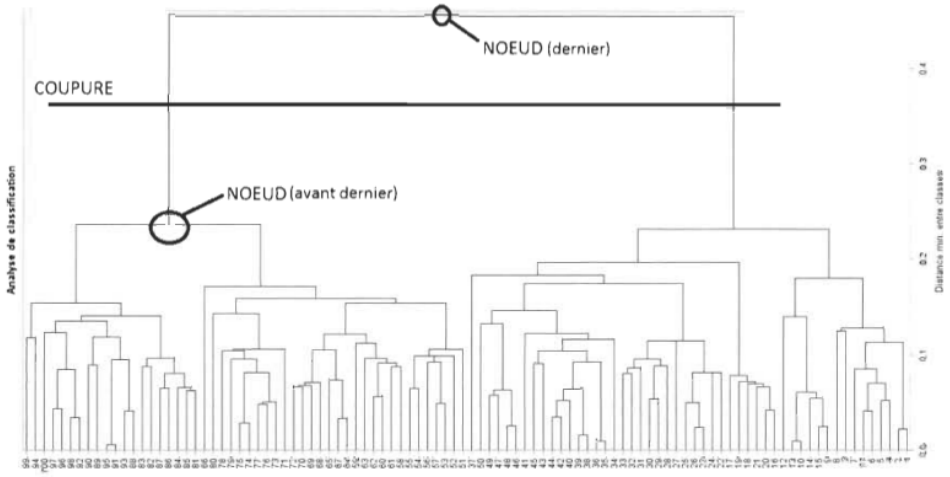

FIGURE 1.1 – *Exemple de dendrogramme*

1.2.2 Application en science de données

Pour illustrer les aspects formels précédemment présentés, on utilise la base de données comportant les CVs de certains professeurs des universités du Québec. Ces données proviennent sur une étude de recherche à l'Institut de Recherche sur Hydrogène (IRH) de UQTR. L'objectif de cette base de données est de faire un partitionnement des professeurs et leurs étudiants aux cycles supérieurs. Ce tableau représente une capture des neuf premières lignes de cette base de données où on a remplacé les noms des professeurs par des lettres A à V1 . Notons que la base de données comporte 45 observations (professeurs) notées en ordre de A, B, W puis A1, B1, V1 jusqu'à la 45ème observation et trois variables (doctorat, maitrise et postdoc). La base de données est l'objet de recherche sur la supervision de certains professeurs avec leurs étudiants aux cycles supérieurs (maitrise, doctorat et postdoc) dans les

universités québécoises. Le tableau 1.1 représente les neuf premières ligne de cette base de données.

	Obsevation	Doctorat	Maitrise	Postdoc
1	A	2	5	0
2	B	9	31	1
3	C	11	2	3
4	D	10	4	1
5	E	1	0	0
6	F	2	0	0
7	G	4	0	2
8	H	8	1	7
9	I	5	26	1

TABLE 1.1 – *Un tableau représentant les neuf premières lignes des données des professeurs*

L'application de cette partie a été réalisée grâce au logiciel R par le biais de la fonction **hclust**. Cette fonction est implémentée dans le package **stats**. Elle permet de réaliser une classification ascendante hiérarchique à partir du tableau de distance des individus. Sous le logiciel R, la commande **dist** de library **stats** permet d'obtenir cette fonction. hclust nous permet de produire les résultats de classification, y compris le dendogramme.

```
library(stats)
```

où df représente les données. On pourra aussi utiliser les mesures de ressemblances **maximum** ou **Minkowski** ou **Manhattan**. Par exemple, on peut utiliser la méthode de **Ward** par la commande ci-dessous, pour former les classes.

```
hc <- hclust(df, method = "ward.D2")
```

Pour faciliter la représentation graphique, on a standardisé la base de données tel

qu'indiqué sur le tableau 1.2.

	Doctorat	Maitrise	Postdoc
[1,]	-0.89032921	-0.2550910	-0.7452678
[2,]	0.30309080	3.2153325	-0.2794754
[3,]	0.64406794	-0.6555244	0.6521093
[4,]	0.47357937	-0.3885688	-0.2794754
[5,]	-1.06081779	-0.9224801	-0.7452678
[6,]	-0.89032921	-0.9224801	-0.7452678
[7,]	-0.54935207	-0.9224801	0.1863169
[8,]	0.13260222	-0.7890023	2.5152788
[9,]	-0.37886350	2.5479434	-0.2794754

TABLE 1.2 – *Tableau des données standardisées des neuf premières lignes*

On a codé les observations (A à W1) par les nombres allant de 1 à 45. Comme par exemple $A = 1, B = 2, C = 3, ..., V1 = 45$.
Après la classification, un dendrogramme a été créé avec une partition de 4 classes (voir le tableau 1.3).

Classes	Observations	Taille
1	16	1
2	8 13 34 45 23 39 28 14 27 21 3 40	12
3	2 26 9 10 24 20 22 17 15 42	10
4	29 33 4 19 43 31 12 41 18 38 7 30 32 1 11 44 37 6 5 25 35 36	22

TABLE 1.3 – Les classes obtenues après la méthode hiérarchique

Le tableau 1.3 représente les quatre classes obtenues après la classification hiérarchique. La première classe de la partition est singleton, l'effectif de la deuxième classe est 12, la troisième contient 10 observations et la quatrième classe est constituée de 22 observations, soit la classe la plus dense.

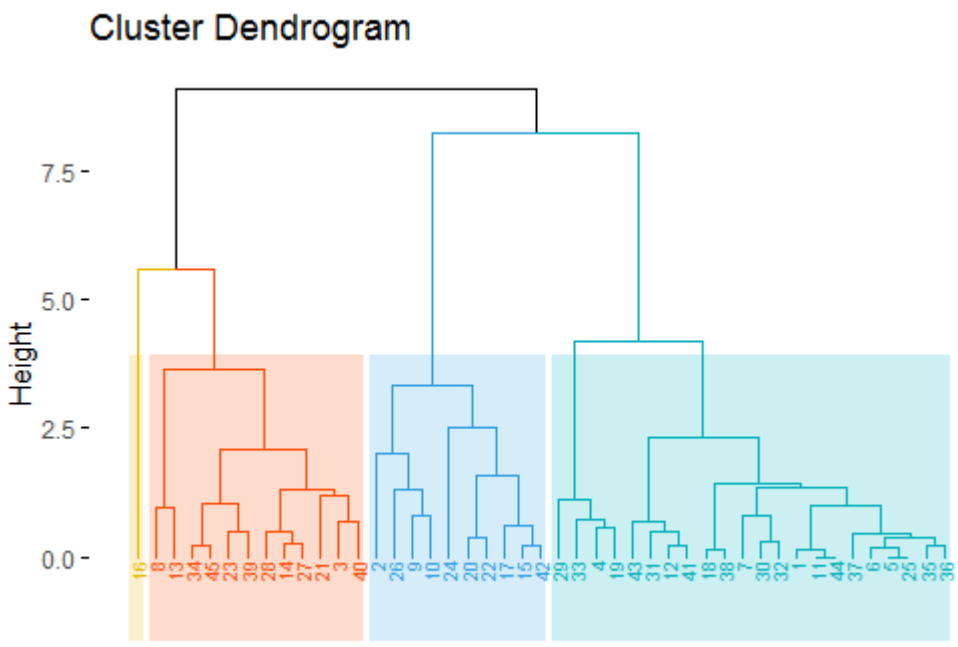

FIGURE 1.2 – *Dendrogramme des études supérieures de certains étudiants avec leurs directeurs de recherche au Québec*

1.2.3 Méthode de k-means

Cet algorithme fut longtemps utilisé sur les grands jeux de données en raison de sa rapidité. L'objectif du clustering partitionnel (ou non hiérarchique) est de définir la partition d'un ensemble d'objets en clusters, que les objets d'un cluster soient plus "similaires" les uns aux autres qu'aux objets d'autres clusters.

Principe

On suppose qu'il existe K classes distinctes. On commence par désigner K centres de classes $\mu_1, ..., \mu_K$ parmi les individus. Ces centres peuvent être soit choisis par l'utilisateur pour leur "représentativité", soit désignés aléatoirement. On réalise ensuite itérativement les deux étapes suivantes :

- Pour chaque individu qui n'est pas un centre de classe, on regarde quel est le centre de classe le plus proche. On définit ainsi K classes $C_1, ..., C_k$ où

$C_i = \{$ensemble des points les plus proches du centre μ_i $\}$.

- Dans chaque nouvelle classe C_i, on définit le nouveau centre de classe μ_i comme étant le barycentre des point de C_i

L'algorithme s'arrête suivant un critère d'arrêt fixé par l'utilisateur qui peut être choisi parmi les critères suivants : soit le nombre limite d'itérations est atteint, soit l'algorithme a convergé [3]

1.2.4 Application en science de données

On a utilisé la même base de données que l'approche hiérarchique avec $k = 4$. On a donc quatre classes. Ces derniers sont représentés sur le tableau 1.4.

Classes	Observations	Taille
1	4 39 3 45 34 14 28 8 13 33 19 29 21 40 27	15
2	16	1
3	20 22 24 2 9 10 26 42 15 17	10
4	23 7 36 12 41 1 11 44 31 37 35 6 43 30 18 32 38 25 5	19

TABLE 1.4 – Les classes obtenues après la méthode de k-means

Les résultats qui composent la méthode des k-means ne sont pas très loin de celle de la CAH. Le tableau 1.4 montre que la classe 1 est composée de 15 observations. La classe 2 est singleton. La troisième classe contient 10 observations et la dernière classe est plus élevée, elle compte 19 observations.

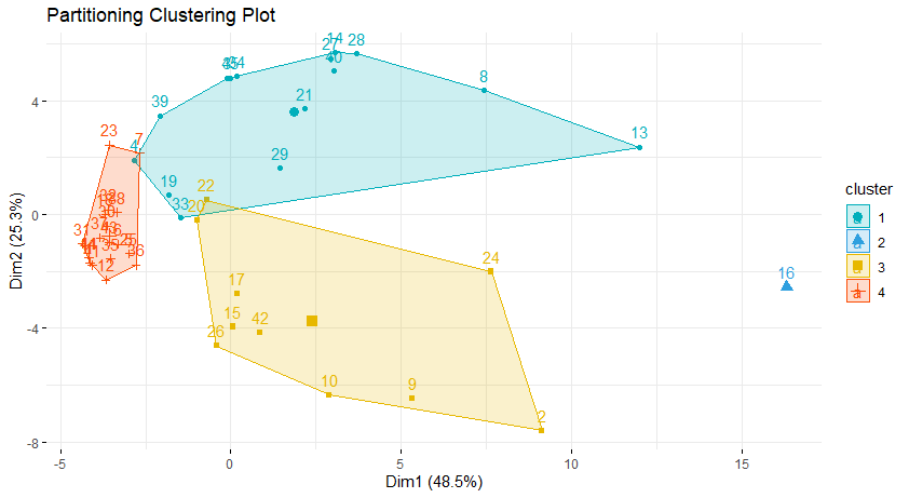

FIGURE 1.3 – *Partitionnement des études supérieures de certain étudiants avec leurs directeurs de recherche au Québec*

Chapitre 2

Méthode de co-clustering et illustrations

L'objectif du co-clustering est de regrouper simultanément les lignes et les colonnes d'une matrice de données d'entrée. Dans ce contexte, il surmonte plusieurs limitations associées aux méthodes de clustering classiques en permettant la découverte automatique de la similarité basée sur un sous-ensemble d'attribut. Cependant, différents modèles de co-clustering produisent souvent des résultats distincts, cela est dû au fait que chaque algorithme a son propre biais lié à l'optimisation des différents critères. Pour cela on va étudier différentes approches sur des données statistiques différentes, dont la méthode de l'approche métrique et le modèle de bloc latent qui sont très utilisés en co-clustering.

Comme exemple d'illustration, on a recueilli différentes bases de données à partir du site *Journaf of software* https ://CRAN.R-project.org/package=blockcluster. Ces bases nous permettent d'appuyer les différents modèles énumérés dans notre travail.

2.1 Modèle de bloc latent (LBM)

2.1.1 Variables latentes et propriétés psychologiques

En étudiant les variables latentes, on a fait une petite introduction sur les objets non-observables en psychologie. La psychologie expérimentale a souvent comme objet d'étude des phénomènes considérés comme non-directement observables (par exemple : intelligence, amour ...) qui génèrent des manifestations observables selon Guyon. H [23]. Pour cela, on a une représentation graphique qui nous montre une structure relationnelle (voir figure 2.1).

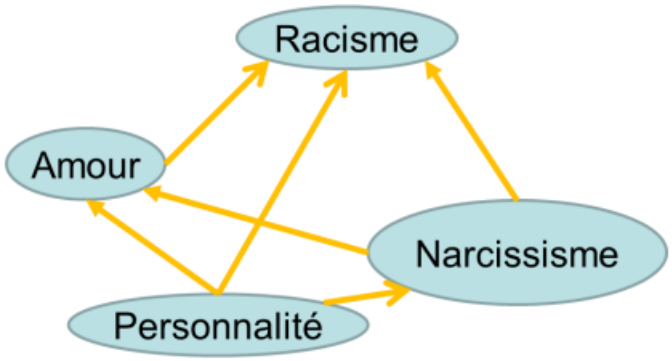

FIGURE 2.1 – *Modèle relationnel (Guyon, H [23])*

Supposons par hypothèse, à partir de cette figure 2.1, une structure relationnelle entre ces propriétés psycologiques : on suppose que la **personnalité** impacte la propension au sentiment **amoureux** et le **narcissisme**, ces deux derniers impactent la propension au **racisme**.
Sans entrer dans le domaine purement psychologique, on considère que ces objets sont réels, en quelque sorte dans le cerveau de la personne. On peut le voir sur l'exemple abstrait de la figure 2.2.

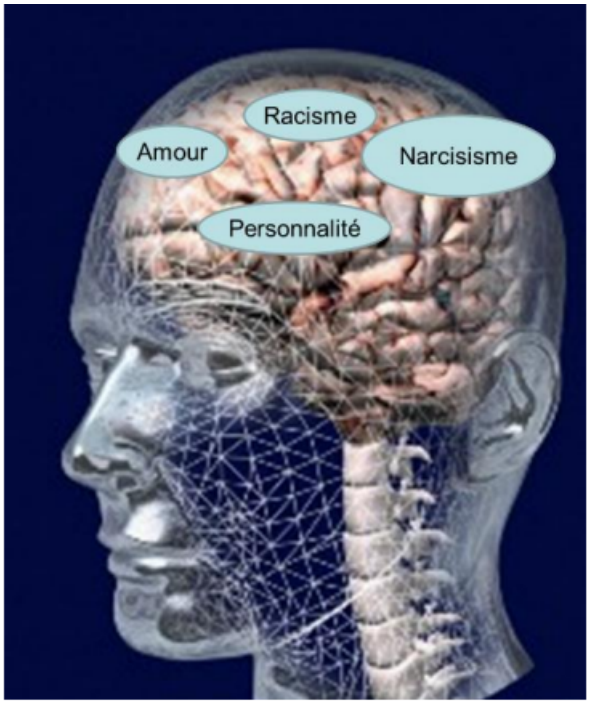

FIGURE 2.2 – *Réalité des objets théoriques (Guyon, H [23])*

Cette petite introduction est pour évoquer que les variables latentes renvoient au réalisme, mêmes certains auteurs les critiquent.

2.1.2 Principe

En statistique, les variables latentes (par opposition aux variables observées) sont des variables qui ne sont pas directement observées, mais qui sont plutôt inférées par une modèle mathématique à partir d'autre variables observées. Dans cette dernière situation, deux ensembles I et J sont considérés comme des échantillons aléatoires, les niveaux de lignes et de colonnes deviennent des variables latentes. L'unité statistique est donc la matrice de données.

Ce modèle est très sollicité dans les travaux avec co-clustering d'après [7]. D'abord, il est basé sous différentes hypothèses telles que d'indépendance conditionnelle. Dans ce

cas, on développe les variables latentes indépendantes pour les modèles de bloc latents. Toujours on continue avec notre matrice de données x de ligne I et de colonne J dont les partitions sont z et w. Les partitions étiquetées $z_1, ..., z_n, w_1, ...w_d$ sont considérées comme des variables latentes et on suppose qu'elles sont indépendantes. La probabilité de partitionnement en z, w est la produit de probabilité de ces dernières, tel que :

$$p(z, w) = p(z)p(w) \quad p(z) = \prod_i p(z_i), p(w) = \prod_i p(w_i) \tag{2.1}$$

D'où pour tout i, la probabilité de la distribution $p(z_i)$ est la distribution $M(\pi_1, ...\pi_g)$ et ne dépend pas de i. De même pour j, $p(w_i)$ est la distribution $M(\rho_1, ...\rho_m)$ et ne dépend pas de j. Rappelons que g et m représentent la taille de la matrice de partitionner $z.w$ c'est-a-dire les clusters (classes).
Enfin la matrice de modèle de bloc latent devient :

$$\theta = (\pi, \rho, \alpha) \tag{2.2}$$

D'où $\pi = (\pi_1, ...\pi_g)$ et $\rho = (\rho_1, ...\rho_m)$ avec $(\pi_k = P(z_{ik}) = 1, k = 1, ...g), (\rho_l = P(w_{jl}) = 1, l = 1, ...m)$ sont des mélanges de proportion et $\alpha = (\alpha_{kl}; k = 1, ...g, l = 1, ...m)$ où α_{kl} est le paramètre de distribution du bloc k, l.
D'après les partitionnements, le modèle peut être représenté par le graphe illustré à la figure 2.3.

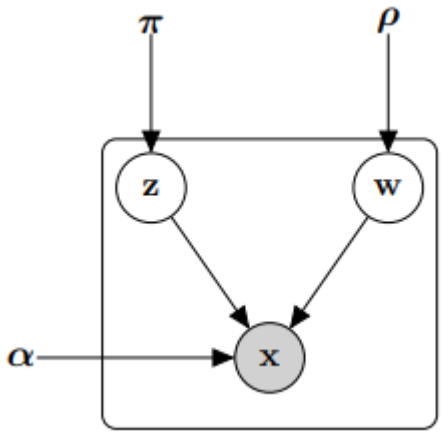

FIGURE 2.3 – *Bloc de modèle latent (Govaert, G. and Nadif, M [7])*

2.2 Données binaires

Les variables binaires (0 ou 1) sont largement utilisées en statistique et jouent un rôle important dans certaines bases de données. Par exemple les bases de données de présence-absence en écologie ou en administration, les pixels noirs et blancs en traitement de l'image, et aussi dans le cas des données obtenues lors du recodage d'un tableau qualitatif.
Les données sont sous forme d'un échantillon $x = (x_1, ...x_n)$ où x_i est un vecteur $(x_{i1}, ...x_{id})$ de valeur x_{ij} appartenant à l'ensemble $\{0, 1\}$.

2.2.1 Modèle d'approche métrique

Comme dans plusieurs disciplines de la statistique, la classification des données nécessite un mécanisme de mesure et de l'évolution. Cela nous permet d'obtenir des blocs homogènes (ou biclusters), dont la méthode de l'approche métrique est sollicitée. Pour obtenir un tel bloc homogène, on a besoin d'une quantité qui représente la

différence entre les données initiales et les données récapitulatives. Cela consiste à minimiser le nombre de fois que les valeurs x_{ij} par rapport à α_{kl} associe les couples de clusters (k, l) qui appartiennent à x_{ij}. Rappelons que α_{kl} est la paramètre de distribution du bloc l, k. Le problème de cette minimisation nous permet de trouver la fonction objectif. Rappelons que la fonction objectif permet ici de minimiser les critères pour obtenir le bloc homogène.

Elle s'exprime comme dans l'équation 2.3 :

$$W(z, w, \alpha) = \sum_{i,j,k,l} z_{ik} w_{jl} |x_{ij} - \alpha_{kl}| \tag{2.3}$$

où $\alpha_{kl} \in \{0, 1\}$

L'objectif de cette minimisation de ces critères est d'obtenir le bloc homogène de 1 ou 0 à partir de la réorganisation des lignes et des colonnes.

On peut illustrer un exemple simple. Considérons une matrice de données X dont les lignes correspondent à un l'ensemble de 8 véhicules de marque différentes et les colonnes ensemble de 8 propriétaires, ces véhicules peuvent avoir (valeur 1) ou ne peuvent pas avoir (valeur 0) (voir tableau 2.1). Ainsi, pour une marque donnée, le propriétaire peut avoir la valeur 1 ou ne pas avoir valeur 0. Avec cet exemple les partitionnements z et w sont respectivement ((α, β, μ), (δ, η, λ), (ξ, γ)) et ((1, 3, 5, 8), (2, 4, 6, 7)), on obtient une matrice de réorganisation X, en considérant simultanément les lignes et les colonnes. Cette réorganisation de la matrice binaire avec ces deux partitionnements peut être résumée par la matrice binaire α, lorsqu'on croise les deux partitionnements. Pour se faire, on choisit au hasard une ligne horizontale h_i et on la classe simultanément avec les colonnes verticales v_i des entrées identiques. Par exemple dans la base de données choisissons une ligne horizontale au hasard $h_i = h_1$. Dans ce cas on va regrouper simultanément les colonnes et observations qui sont majoritairement 1. Ce qui correspond au premier classement (α, β, μ) et (1, 3, 5, 8) (voir tableau 2.2).

	1	2	3	4	5	6	7	8
α	1	0	1	0	1	0	0	1
δ	0	1	0	1	0	1	1	0
ξ	1	0	0	0	0	1	0	1
β	1	0	1	0	1	0	0	0
η	0	1	0	1	0	1	1	0
μ	1	0	1	0	1	0	1	1
γ	1	0	0	0	0	0	1	0
λ	0	0	1	0	0	1	0	1

TABLE 2.1 – Données binaires X

	1	3	5	8	2	4	6	7
α	1	1	1	1	0	0	0	0
β	1	1	1	0	0	0	0	0
μ	1	1	1	1	0	0	0	1
δ	0	0	0	0	1	1	1	1
η	0	0	0	0	1	1	1	1
λ	0	1	0	1	0	0	1	0
ξ	1	0	0	1	0	0	1	0
γ	1	0	0	0	0	0	0	1

TABLE 2.2 – Réorganisation de la matrice X

D'après la matrice de réorganisation (voir tableau 2.2) on a six regroupements sous forme de deux colonnes (noté ϕ et τ) et de trois lignes notées A, B et C. Comme on l'a énoncé dans les données binaires la fonction, c'est la majorité des valeurs. Par exemple, dans un bloc si les données sont majoritairement 1, la sortie va être 1.
On a continué cette démarche pour avoir la somme de la matrice finale que l'on nomme matrice α de partitionnement (voir tableau 2.3). On a résumé ainsi le tableau de 64 valeurs à un tableau de 6. C'est le même processus pour les variables continues et les

tableaux de contingence

	ϕ	τ
A	1	0
B	0	1
C	0	0

TABLE 2.3 – Matrice α

Pour obtenir un résumé des données initiales, nous verrons que les algorithmes proposés se basent, à chaque étape, sur des matrices intermédiaires X^z, X^w et X^{zw} de taille réduite définies dans le tableau 2.4.

Matrice	Taille	Definition
$X^z = (x^z_{kj})$	$g \times d$	$x^z_{kj} = \sum_i z_{ik} x_{ij}$
$X^w = (x^w_{il})$	$n \times m$	$x^w_{il} = \sum_j w_{jl} x_{ij}$
$X^{zw} = (x^{zw}_{kl})$	$g \times m$	$x^{zw}_{kl} = \sum_{i,j} z_{ik} w_{jl} x_{ij}$

TABLE 2.4 – Matrices réduites, tailles et définitions X^z, X^w et X^{zw}

L'obtention de ces partitions intermédiaires (X^z, X^w et X^{zw}) se calcule à partir de la matrice de réorganisation (voir tableau 2.2), dont on a trois classes avec 6 blocs. Soit X^z de taille $g \times d$, on a les premiers éléments qui sont : 1+1+1=3, 1+1+1=3, 1+1+1=3, 1+0+1=2. Soit X^w de taille $n \times m$, on a les premiers éléments qui sont calculés comme suit : 1+1+1+1=4, 1+1+1+0=3, 1+1+1+1=4. Soit X^{zw} qui résume ces deux dernières 1+1+1+1+1+1+1+0+1+1+1+1=11, 0+0+0+0+0+0+0+0+0+0+0+1=1.

$$X^z = \begin{pmatrix} 3 & 3 & 3 & 2 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 2 & 2 & 3 & 2 \\ 2 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

$$X^w = \begin{pmatrix} 4 & 0 \\ 3 & 0 \\ 4 & 1 \\ 0 & 4 \\ 0 & 4 \\ 1 & 1 \\ 2 & 1 \\ 1 & 1 \end{pmatrix} \text{ et } X^{zw} = \begin{pmatrix} 11 & 1 \\ 2 & 9 \\ 3 & 2 \end{pmatrix}$$

La spécificité de cette méthode est que le co-clustering regroupe les éléments de façon simultanée. Pour l'appliquer sur le logiciel R, on fait intervenir la fonction : **cocluster**

Il s'agit d'une fonction implémentée dans le package **blockcluster**. Elle permet de réaliser un regroupement sous forme de classe en fonction du type de données qu'on a eu en format matrice ou numérique. La fonction cocluster permettra alors de produire des résultats de la classification, y compris le regroupement en classe ou en bloc. Selon les différents types de données qu'ils soient des données binaires, des variables continues ou des tableaux de contingences.

Exemple de package blockcluster et la fonction cocluster.

```
library(blockcluster)
cocluster(data, datatype, model =,  nbcocluster,
 strategy = coclusterStrategy(),  nbCore = 1)
```

Par exemple, on a simulé aléatoirement une base binaire composée de 1000 lignes et 100 colonnes, dont on a représenté une partie dans le tableau 2.5.

	V1	V2	V3	V4	V5	V6	V7	V8
1	1	1	1	0	1	1	1	1
2	0	0	1	0	0	0	0	0
3	0	1	1	0	1	1	1	1
4	0	0	0	0	1	0	1	1
5	0	0	0	1	1	1	1	1
6	1	1	1	1	0	0	1	1
7	0	0	0	1	0	0	0	1

TABLE 2.5 – *Exemple de base de données binaires*

La figure 2.5 représente la classification simultanée par co-clutering dont 1 représente la couleur blanche et 0 la couleur noire (voir la légende). On y énumère deux classes et six blocs bien identifiés par des lignes bleu.

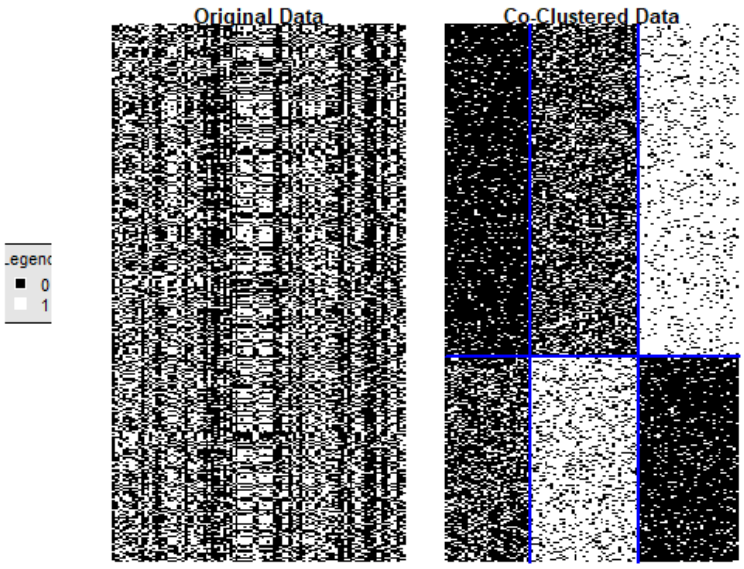

FIGURE 2.4 – *Exemple d'une matrice initiale et une matrice de réorganisation par co-clustering*

2.2.2 Modèle à bloc latent (LBM) de Bernoulli

Comme on l'a introduit, chaque type de données statistiques en co-clustering s'appliquent avec des méthodes de probabilités différentes. Avec les données binaires en modèle de bloc latent, on travaille avec la loi de Bernoulli.
On s'intéresse à l'application de la loi Bernoulli avec le LBM. La morphologie des données binaires nous permet de supposer que pour chaque bloc de couple kl, les valeurs de x_{ij} sont distribuées selon la loi Bernoulli tel que $\beta(a_{kl})$ implique $a_{kl} \in R$ et $0 < a_{kl} < 1$.
Ce modèle s'exprime comme comme dans l'équation 2.4, f représente la distribution de Bernoulli avec α le paramètre de distribution du bloc l, k.

$$f(x_{ij}; a_{kl}) = (a_{kl})^{x_{ij}}(1 - a_{kl})^{1-x_{ij}} \tag{2.4}$$

Rappelons que $\theta = (\pi, \rho, \alpha)$
D'où $\pi = (\pi_1, ...\pi_g)$, $\rho = (\rho_1, ...\rho_m)$ et $a = (a_{11}, ...a_{gm})$. Ce ci nous permet d'obtenir la formule suivante (équation 2.5).

$$f(x; \theta) = \sum_{z,w} \prod_{i,k} \pi_k^{z_{ik}} \prod_{j,l} \rho_l^{w_{j,l}} \times \prod_{i,j,k,l} ((a_{kl})^{x_{ij}}(1 - a_{kl})^{1-x_{ij}})^{z_{ik}w_{jl}} \tag{2.5}$$

D'où $f(x; \theta)$ représente la distribution de Bernoulli par le modèle à bloc latent.

Pour amplifier les résultats des données, la méthode complète de vraisemblance peut être employée [7]. Mais on ne va pas développer ce modèle dans notre partie de travail.
Par exemple pour illustrer cette partie, on a continué avec les mêmes données binaires. Ces données sont classées à partir du modèle latent de la distribution de Bernoulli (voir figure 2.5). Cette distribution nous a permis d'identifier chaque élément dans son bloc. La figure 2.5 montre après la classification, on a deux classes regoupées en

six blocs. Les bloc (1,1)), (1,2), (2,2) et (2,3) sont majoritairement de 0 et le reste des blocs (1,3) et (2,1) sont majoritairement de 1.

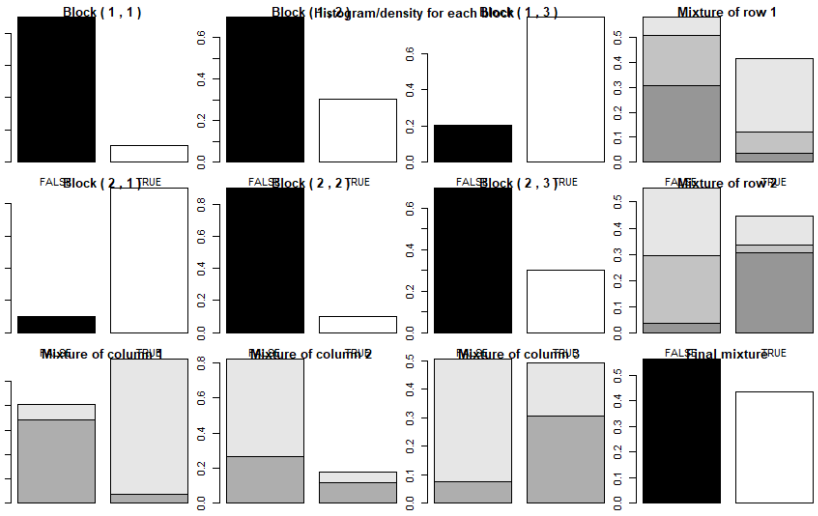

FIGURE 2.5 – *Exemple de classification par la distribution de Bernoulli*

2.3 Co-clustering des données continues

Les données continues sont sans aucun doute les types de données les plus courants et peuvent se retrouver dans tous les domaines. La structure peut être un tableau rationnel où les d colonnes sont des variables $x_{ij} \in \mathbb{R}$. Cependant, elles peuvent être positives ou négatives avec différentes unités ou variabilités.

Dans cette partie de notre étude, on s'intéresse au co-clustering d'une matrice de données $n \times d$. Soit $x = (x_{ij})$ une matrice de données dont I correspond à l'ensemble des n individus et J correspond à l'ensemble des d variables. Tout d'abord, on va étudier la représentation géométrique. En ce sens, on considère que chaque individu i a un poids p_i et chaque colonne j a un poids q_j. Cela nous permet de calculer la distance dont la plus utilisée est celle euclidienne. Dans cette première distance

on a les p_i qui représentent les poids des individus d'un ensemble n de point $\mathbb{R}^n$ et q_j peuvent être utilisés pour définir le métrique euclidienne entre deux individus i et i'.

$$d^2(i,i') = \sum_{j=1}^{d} q_j (x_{ij} - x_{i'j})^2 \tag{2.6}$$

On utilise la même méthode pour la représentation géométrique des variables q_j par un ensemble d points de $\mathbb{R}^n$. Mais sauf que les q_j et les p_i peuvent être utilisés pour définir le métrique euclidienne entre deux variables j et j'.

$$d^2(j,j') = \sum_{i=1}^{n} p_i (x_{ij} - x_{ij'})^2 \tag{2.7}$$

Dans le but de simplifier la notation, on a supposé que $p_i = \dfrac{1}{n}$ pour tout $i (i = 1, ..., n)$ et $q_j = 1$ pour tout $j (j = 1, ..., d)$.

Le co-clustering avec des données continues permet d'obtenir la matrice récapitulative à partir de la moyenne, représentée dans la figure 2.6. Cela est obtenu comme suit, la matrice continue des données est résumée par une matrice continue plus petite en associant chaque bloc à sa moyenne.

En utilisant la partition z des individus I et la partition w des variables J, la matrice de données continue sera la somme des deux ensembles de poids $p^z = (p_1^z, ...p_g^z)$, $q^w = (q_1^w, ...p_m^w)$ et la matrice $g \times m$ décrit $x^{zw} = (x_{kl}^{zw})$.

On peut les définir de la façon suivant :

$$p_k^z = \frac{\sum_i z_{ik}}{n} = \frac{z_{\cdot k}}{n}, \qquad q_l^w = \sum_j w_{jl} = w_{\cdot l} \tag{2.8}$$

et

$$x_{kl}^{zw} = \frac{\sum_{i,j} z_{ik} w_{lj} p_i q_j x_{ij}}{\sum_{i,j} z_{ik} w_{lj} p_i q_j} = \frac{\sum_{i,j} z_{ik} w_{jl} x_{ij}}{z_{\cdot k} w_{\cdot l}} \tag{2.9}$$

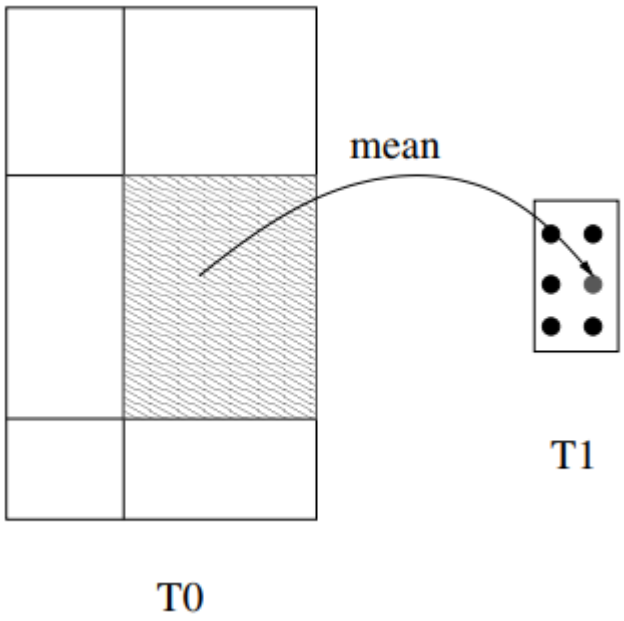

FIGURE 2.6 – *Représentation d'une matrice de données continues à l'aide de co-clustering*

La figure 2.6 montre la création de $T1$ à partir de $T0$ par la moyenne des données les plus proches.
Cela nous permet d'associer la partitionnement de z et w a un nouveau triplet (x^{zw}, p^k, q^w) ayant la même structure de données que le triplet initial (x, z, w). Ainsi on peut définir la mesure de l'information décrite dans l'équation (2.10). Cette mesure de l'information nous facilite le calcul de la fonction objectif.

$$I(x^{zw}, p^z, q^w) = \sum_{k,l} p_k^z q_l^w (x_{lk}^{zw})^2 = \frac{1}{n} \sum_{k,l} z_{.k} w_{.l} (x_{lk}^{zw})^2 \tag{2.10}$$

Voici un exemple qui nous permet d'illustrer la situation théorique.

$$x = \begin{pmatrix} 1 & 2 & 8 \\ 2 & 1 & 7 \\ 2 & 4 & 7 \\ 4 & 4 & 6 \end{pmatrix}$$

Ce exemple est appliqué pour illustrer figure 2.6. On remplace p et q par leur valeurs. Or $p_i = \dfrac{1}{n}$ où n=4, c'est qui équivaut à $p = (1/4, 1/4, 1/4, 1/4)$ et $q_j = 1$ où $q = (1, 1, 1)$. Soit la partition $z = (1, 1, 2, 2)$ et celle $w = (1, 1, 2)$. Cela montre que la partition de z est composée de deux partitionnements dont chaqu'un est composé de deux lignes d'où $i = 2$ qui équivaut à $\sum_i z_{ik} = z_{\cdot k} = 2$. Même démarche pour le deuxième partitionnement de ligne. Cependant, on peut obtenir le poids $p_k^z = \dfrac{z_{\cdot k}}{n} = (2/4, 2/4) = (1/2, 1/2)$ (voir tableau 2.6 (1)). La partition de w est composée de deux partitionnements dont le premier comporte deux colonnes et l'autre est singleton d'où $j = 2$ pour le premier qui équivaut à $\sum_j w_{jl} = w_{\cdot l} = 2$ et $j = 1$ pour le deuxième $w_{\cdot l} = 1$ (voir tableau 2.6 (2)). Ainsi on peut obtenir $q_l^w = w_{\cdot l} = (2, 1)$. Ces partitions nous permettent d'obtenir la somme de x^{zw}.

1)

$\frac{1}{2}$	1	2	8
	2	1	7
$\frac{1}{2}$	2	4	7
	4	4	6

2)

	2	1
1	2	8
2	1	7
2	4	7
4	4	6

3)

		2	1
$\frac{1}{2}$	1	2	8
	2	1	7
$\frac{1}{2}$	2	4	7
	4	4	6

TABLE 2.6 – Partitionnement des matrices intermédiaires

Rappelons que les matrices intermédiaires $x^z = (x_{kj}^z)$ de taille $(g \times d)$ qui équivaut à (2.3) et $x^w = (x_{il}^w)$ de taille $(n \times m)$ qui équivaut à (2.4) peuvent être définis :

$$x_{kj}^z = \frac{\sum_{i,k} z_{ik} p_i x_{ij}}{\sum_{i,k} p_i z_{ik}} = \frac{\sum_{i,k} z_{ik} x_{ij}}{z_{\cdot k}} \tag{2.11}$$

et

$$x_{il}^{w} = \frac{\sum_{j,l} w_{jl} q_j x_{ij}}{\sum_{j,l} q_j w_{jl}} = \frac{\sum_{j,l} w_{jl} x_{ij}}{w_{.l}} \tag{2.12}$$

Avec le partitionnement du tableau 2.6, les matrices intermédiaires deviennent :

La première ligne de x^z est obtenue en appliquant l'équation (2.11), ce qui donne $\frac{1+2}{2} = 1,5$, $\frac{2+1}{2} = 1,5$, $\frac{8+7}{2} = 7,5$

Celle de x^w, on utilise la même démarche en appliquant l'équation (2.12), $\frac{1+2}{2} = 1,5$, 8 car pour le deuxième $w_{.l} = 1$

Pour la partition x^{zw} on remplace $z_{.k}w_{.l} = 2 \times 2 = 4$ pour le premier bloc et $z_{.k}w_{.l} = 2 \times 1 = 2$ pour le deuxième. La première ligne en appliquant l'équation (2.9) donne $\frac{1+2+2+1}{4} = 1,5$, $\frac{8+7}{2} = 7,5$ (voir tableau 2.6 (3)).

$$x^z = \begin{pmatrix} 1,5 & 1,5 & 7,5 \\ 3 & 4 & 6,5 \end{pmatrix} \qquad x^w = \begin{pmatrix} 1,5 & 8 \\ 1,5 & 7 \\ 3 & 7 \\ 4 & 6 \end{pmatrix} \qquad \text{et } x^{zw} = \begin{pmatrix} 1,5 & 7,5 \\ 3,5 & 6,5 \end{pmatrix}$$

En particulier quand la partition z est une partition singleton, nous noterons (x^w, p, q^w) le triplet obtenu et quand w est une partition singleton, nous le noterons (x^z, p^z, q). Par conséquent, nous obtenons la mesure d'association associée comme suit :

$$I(x^z, p^z, q) = \frac{1}{n} \sum_{k,j} z_{.k} (x_{kj}^z)^2 \tag{2.13}$$

et

$$I(x^w, p, q^w) = \frac{1}{n} \sum_{i,l} w_{.l} (x_{il}^w)^2 \tag{2.14}$$

Pour l'application des données, on a fait une simulation des données continues. Cette dernière consiste à générer des données continuées sous forme aléatoire. La base de

données est composée de 1000 lignes et de 50 colonnes.

Le tableau 2.7 représente les premières lignes et colonnes de cette base de données.

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
1	-9.318100	12.82700	-16.50500	14.7320	-8.330300	-12.99000	11.25100	13.79400	-14.5650	3.561300	-2.73690	-14.11900	14.71100
2	-6.119900	9.22690	-8.46600	11.6710	-0.020574	-10.04900	13.74400	9.48890	-8.5349	-2.496300	1.04930	-7.60270	5.84460
3	17.335000	-18.09700	7.58990	-10.9870	-2.619300	4.54310	-11.51800	-3.79910	10.5160	-4.167100	3.17210	13.92800	-9.98950
4	2.862100	-7.24950	17.39100	-15.0350	-0.535990	10.05200	-9.23540	-16.89600	7.4427	-2.242700	-0.61528	11.97100	-11.50600
5	-9.183900	9.43560	-12.85500	8.3094	2.207300	-9.99870	8.50060	7.74190	-8.5541	1.610800	3.12940	-8.96550	6.81540
6	3.519400	-16.62800	10.53400	-9.6032	4.794800	12.40000	-17.55600	-17.76100	12.7790	1.320400	0.92170	11.78500	-7.97480
7	-11.512000	12.36600	-1.83900	8.4888	-1.823400	-12.63500	14.62400	10.77600	-9.9732	2.131500	-3.89050	-14.37300	11.01900
8	7.675800	-6.41420	9.02760	-2.9457	-1.677000	5.07090	-8.78300	-12.57700	19.3450	-2.416700	5.42060	12.56200	-10.79000
9	-6.453300	9.31550	-14.44500	5.7800	4.199700	-5.51000	13.31300	7.28590	-10.1780	9.654000	5.73820	-11.15900	9.47370
10	10.167000	-21.98600	1.82860	-6.4602	0.477410	4.73720	-7.44030	-13.42000	8.0823	-3.438000	0.78431	12.93400	-10.87600
11	7.424300	-12.69900	10.65800	-10.5660	3.616800	5.30160	-11.51400	-9.84680	17.7830	4.549000	-0.56364	18.30400	-11.47100
12	-11.279000	5.29480	-4.01310	12.1600	-1.612800	-13.94800	9.23130	8.33710	-1.3410	-3.718100	-1.77570	-12.53900	10.54900
13	9.020100	-2.60020	2.79720	-9.0356	-7.996300	19.06200	-16.57700	-11.27800	12.0530	6.866300	-1.65450	12.04400	-8.28710
14	9.281700	-9.79370	12.24400	-9.2818	-0.033964	10.40800	-6.51980	-11.93200	7.5489	-5.077200	-5.40460	21.39000	-6.00140
15	7.139900	-12.13200	9.38550	-10.0870	-2.727800	15.08000	-4.90030	-10.48000	15.9950	4.374600	-0.69333	9.37710	-9.74540
16	-14.848000	11.09200	-11.33200	11.1720	-5.001600	-11.62800	9.38900	13.28400	-12.6710	-2.560400	0.31696	-8.38890	7.75140
17	-15.086000	10.50000	-15.58200	8.4341	-3.619700	-10.34800	5.08180	5.61970	-5.3170	-11.076000	-0.68842	-8.25550	12.90300
18	3.367600	-12.11500	7.67480	-7.3999	3.952900	9.28680	-9.54160	-6.36410	8.4338	-0.704220	3.94200	7.61120	-12.55500

TABLE 2.7 – *Exemple de base de données des variables continues*

On a réorganisé cette base de données pour classer chaque individu dans son bloc. La figure 2.7 est composée de deux classes avec six blocs délimités par les lignes bleues.

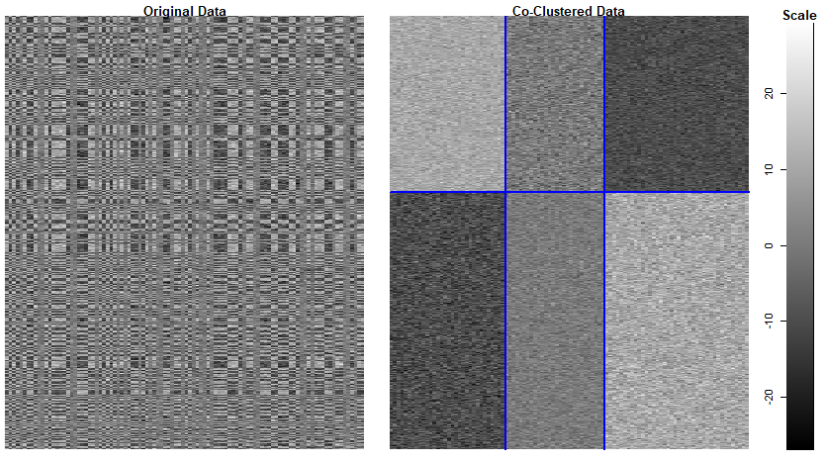

FIGURE 2.7 – *Matrice initiale et matrice réorganisée de classification*

Voici l'exemple de code R.

```
h <- cocluster( gaussiandata,
 datatype = "continuous" , # visualiser les type de données
  nbcocluster = c(2,3),
   # nombre de regroupement de lignes et de colonnes de clustre
  model = "pik_rhol_sigma2kl", # Modele de proportion et de dispertion
  strategy = coclusterStrategy(
    nbtry = 5, # Nombres d'essaient
    nbxem = 10
  )
)
# Reorganisation de nos données (classer les données simultanement en bloc)
plot(x = h, asp = FALSE, type = "cocluster")
```

2.3.1 Modèle à bloc latent (LBM) de la loi gaussienne

Le modèle de bloc latent, très sollicité pour les regroupements simultanés, est étudié sur les données continues avec la loi gaussienne. Cette loi facilitera la visualisation des données latentes continues. Dans cette optique, plusieurs méthodes peuvent s'y appliquer. Par exemple, on peut travailler sur les variables minimum pour expliquer le phénomène en général.

Premier modèle

On a démontré la méthode générale du modèle latent au début du chapitre dont l'objectif est de s'appuyer sur les variables latentes (les variables cachées). Dans notre partie de travail, on va essayer de montrer l'application de ce modèle avec les variables continues sous l'intervention de la loi gaussienne.
Après réorganisation de la matrice, chaque bloc kl des valeurs x_{ij} est distribué selon

la loi gaussienne tel que :

$N(\mu_{kl}, \sigma^2_{kl})$ avec $\mu_{kl} \in \mathbb{R}$ et $\sigma^2_{kl} \in \mathbb{R}^+$.

Cela nous permet d'obtenir le modèle de bloc latent gaussien avec la fonction de densité de la probabilité suivante.

$f(x, \theta)$ s'exprime sous la forme :

$$f(x,\theta) = \sum_{(z,w)\in Z,W} \prod_{i,k} \pi_k^{z_{ik}} \prod_{j,l} \rho_l^{w_{jl}} \times (\frac{1}{\sqrt{2\Pi\sigma^2_{kl}}} exp - \{\frac{1}{2\sigma_{kl}}(x_{ij} - \mu_{kl})^2\})^{z_{ik}w_{jl}} \quad (2.15)$$

Avec $\theta = (\pi, \rho, \alpha)$ ou $\pi = (\pi_1, ..., \pi_g)$, $\rho = (\rho_1, ..., \rho_m)$ et $\alpha = (\mu_{11}, ..., \mu_{gm}, \sigma^2_{11}, ..., \sigma^2_{gm})$

Avec ces mêmes variables continues, on a classé les données selon la distribution gaussienne qui nous a permis d'obtenir la figure 2.8. Cette dernière est composée de deux classes sous forme de six blocs. On a, par exemple, le bloc (1,1) est caractérisé par des élément de 0 à 20, le bloc (2,2) par des éléments de -10 à 10.

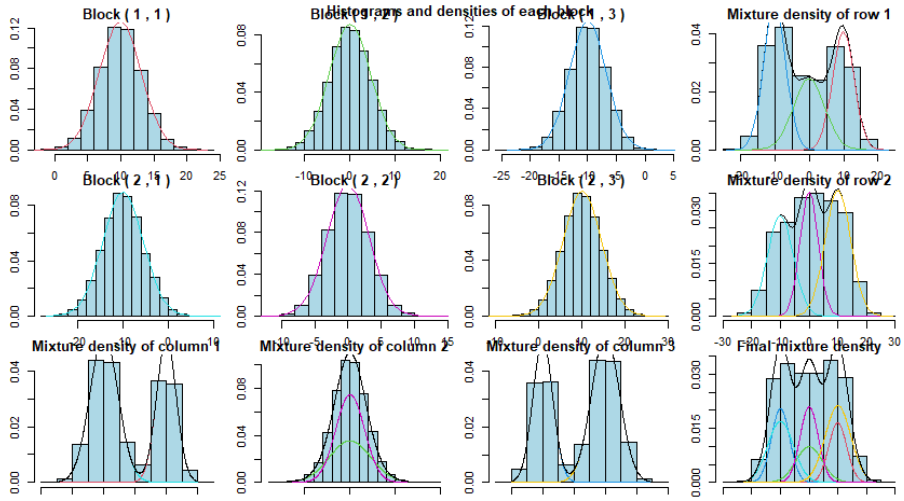

FIGURE 2.8 – *Classification par le bloc gaussien*

Voici le code R ci-dessous.

```
library(blockcluster)
```

```
library(blockmodels)

# Classification par simulation du bloc
#latent par la distribution gaussiennne
plot(x = h, type = "distribution")
```

Le package **blockmodels** permet d'executer les fonctions de distributions des variables latentes.

Restriction du modèle de bloc latent gaussien

La restriction ou parcimonie d'un modèle peut être définie par le fait qu'il impose des contraintes sur les variances. Dans ce cas, on obtient le modèle $[\sigma]$ lorsque la variance ne dépend pas de cluster ligne ni de cluster colonne. Le modèle de partitionnement $[\sigma_k]$ est obtenu par les variances provenant que des clusters lignes. La même chose pour le modèle $[\sigma_l]$ obtenu par les variances des clusters colonnes.
Cette paramétrisation restrictive peut nous permettre de comprendre la relation entre les classifications CROEUC (classification croisée optimisant un critère basé sur la distance euclidienne) et le modèle de bloc latent.
Si on prend le cas le plus simple, dans le modèle $[\sigma]$, étant donné des proportions identiques ($\pi_k = \frac{1}{g}$ et $\rho_l = \frac{1}{m}$) de x_{ij}. Pour faire cette classification, on utilisera la méthode de vraisemblance. C'est une méthode statistique utilisée pour inférer les paramètres de la loi de probabilité d'un échantillon donné en cherchant les valeurs des paramètres maximisant la fonction de vraisemblance. L'intégralité de cette méthode n'est pas développée dans notre travail.
Cependant, nous allons utiliser ces outils pour exprimer notre modèle parcimonieux des données complètes qui prend la forme :

$$L(z, w, \alpha) = -\frac{nd}{2} \log \sigma^2 - \frac{1}{2\sigma^2} \sum_{i,j,k,l} z_{ik} w_{jl} (x_{ij} - \mu_{kl})^2 - n \log g - d \log m \qquad (2.16)$$

Cela nous permet facilement de voir la maximisation de L. Cette maximisation équivaut à la minimisation de la fonction :

$$W(z, w) = \sum_{i,j,k,l} z_{ik} w_{jl} (x_{ij} - x_{kl}^{zw}) \qquad (2.17)$$

2.4 Co-clustering des tableaux de contingences

Les méthodes de co-clusrtering ont une importance pratique dans une grande variété d'application, telle que dans le regroupement des documents ou souvent les données sont sous forme d'un tableau de contingence. Dans cette situation, les ensembles I et J sont souvent des variables catégorielles. C'est ainsi que les lignes et les colonnes correspondent aux différentes catégories des deux variables de la matrice de données. Les ensembles I et J peuvent également être deux ensembles quelconques avec une matrice de données définissant une relation binaire sur IJ. Cette partie de notre travail présente un cadre cohérent pour expliquer certains critères existants, l'algorithme de l'analyse de tableau de contingence et de proposer de nouveaux tableaux. Nous nous intéressons à l'approche basée sur la minimisation de la fonction objectif qui est basée sur la mesure d'association, plus particulièrement la méthode de khi-deux sera développée.

La figure 2.9 représente un exemple de tableau de contingence et sa distribution empirique.

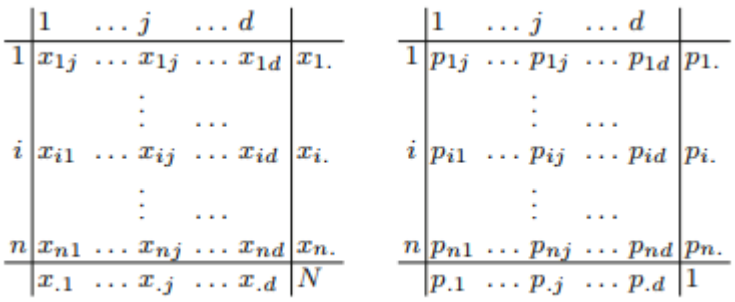

FIGURE 2.9 – *Exemple d'un tableau de contingence et distribution empirique (Govaert, G. et Nadif, M [4])*

2.4.1 Mesure de l'association

Le tableau de contingence caractérise les liens de dépendance entre deux ensembles I et J en mesurant la force de l'association. C'est une longue tradition en statistique, remontant au moins aux travaux de Pearson. On étudie en particulier l'une des mesures d'association : coefficient de khi-deux.

Coefficient de khi-deux

Plusieurs mesures d'association découlent de la statistique khi-deux, généralement basée sur un test indépendant.

$$\chi^2(x) = \sum_{i,j} \frac{(x_{ij} - x_{i.}x_{.j})^2}{x_{i.}x_j} \tag{2.18}$$

Où $x_{i.}$ représente les profiles-lignes et $x_{.j}$ représente les profiles-colonnes et N le nombre total d'observations.

Le mesure χ^2 souvent fournit une preuve statistique significative ou dépendance entre lignes et colonnes dans un tableau. Le coefficient de phi-carré de Pearson de la moyenne

carrée du tableau de contingence est basé sur χ^2 et s'écrit de la manière suivante.

$$\Phi^2(P_{IJ}) = \frac{\chi^2(x)}{N} = \sum_{ij} \frac{(p_{ij} - p_i.p_j)^2}{p_i.p_j} = \sum_{i,j} \frac{p_{ij}^2}{p_i p_j} - 1 = \sum_{i,j} p_i.p_j(\frac{p_{ij}}{p_i.p_j} - 1)^2 \quad (2.19)$$

L'équation (2.19) montre l'écart du modèle d'indépendance en considérant $x_{ij} = \frac{p_{ij}}{p_i p_j} - 1$ comme terme général de x. De plus, le coefficient khi-deux peut être vu comme l'estimation de l'écart entre les deux probabilités $\xi_i \times \xi_j$ et probabilité ξ_{ij} que nous aurions si deux variables aléatoires catégorielles étaient indépendantes. Le coefficient de phi-carré permet de corriger la sensibilité du khi-deux à la taille de l'échantillon en la divisant par le nombre d'observations N.

On a un exemple de tableau de contingence voir (tableau 2.8) dont la démarche est de chercher d'abord le nombre de degré de liberté. Comme dans le cas de tableau 2.8 on a 6 modalités de i et 5 modalités de j d'où $(6-1) \times (5-1) = 20$ degrés de liberté. Ensuite, calcule la distance de khi-deux pour chaque modalité en utilisant l'équation (2.18).

	1	2	3	4	5	
1	5	4	6	1	0	16
2	6	5	4	0	1	16
3	1	0	1	7	5	14
4	1	1	0	6	5	13
5	4	5	3	4	5	21
6	5	4	4	3	4	20
	22	19	18	21	20	100

	1	2	3	4	5	
1	0.05	0.04	0.06	0.01	0.00	0.16
2	0.06	0.05	0.04	0.00	0.01	0.16
3	0.01	0.00	0.01	0.07	0.05	0.14
4	0.01	0.01	0.00	0.06	0.05	0.13
5	0.04	0.05	0.03	0.04	0.05	0.21
6	0.05	0.04	0.04	0.03	0.04	0.20
	0.22	0.19	0.18	0.21	0.20	1.00

TABLE 2.8 – *Exemple de tableau de contingence et distribution conjointe associée*

La figure 2.10 représente une partitionnement de T1 à partir de T0 qui constitue les données initiales.

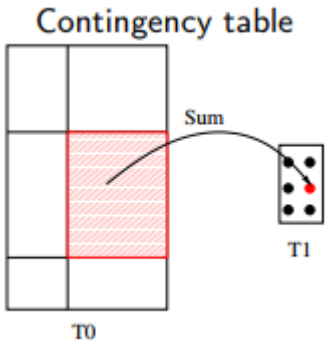

FIGURE 2.10 – Exemple d'illustration d'un résumé de tableau de contingence

En se basant sur la figure 2.10, on va illustrer notre théorie qui évoque le partitionnement des tableaux de contingence, il s'agit de la somme. Soit un tableau de contingence à deux entrées $g \times m$ tel que $y^{zw} := \{y_{kl}^{zw}, k = 1, ...g; l = 1, ...m\}$. Ceci peut être obtenu à partir du tableau initial en calculant la somme des partitionnements de z et w et cela donne

$$y_{kl}^{zw} = \sum_{i,j} z_{ik} w_{jl} x_{ij} \tag{2.20}$$

A partir de ce résultat, on peut calculer la distribution associée z et w. Elle est définie par $P^{zw} = P_{kl}^{zw} = \frac{y_{kl}^{zw}}{N}$. Soit les partitions $z = (1, 1, 2, 2, 3, 3)$ et $w = (1, 1, 1, 2, 2)$ qui sont équivaux à l'agrégation des lignes et des colonnes, (1,1) signifie le premier partitionnement des deux lignes et (1,1,1) celui des trois colonnes du tableau 2.9. Le tableau 2.10 est obtenu à partir de ce dernier en utilisant l'équation (2.20). Par exemple, ligne 1 et colonne 1 du tableau 2.10 est calculée par $a_{11} = 5+4+6+6+5+4 = 30.0$ et ainsi de suite pour les autres lignes et colonnes.

	1	2	3	4	5	
1	5	4	6	1	0	16
2	6	5	4	0	1	16
3	1	0	1	7	5	14
4	1	1	0	6	5	13
5	4	5	3	4	5	21
6	5	4	4	3	4	20
	22	19	18	21	20	100

	1	2	3	4	5	
1	0.05	0.04	0.06	0.01	0.00	0.16
2	0.06	0.05	0.04	0.00	0.01	0.16
3	0.01	0.00	0.01	0.07	0.05	0.14
4	0.01	0.01	0.00	0.06	0.05	0.13
5	0.04	0.05	0.03	0.04	0.05	0.21
6	0.05	0.04	0.04	0.03	0.04	0.20
	0.22	0.19	0.18	0.21	0.20	1.00

TABLE 2.9 – *Le tableau de contingence des partitionnements z et w*

	1	2	
1	30.0	2.00	32.0
2	4.00	23.0	27.0
3	25.0	16.0	41.0
	59.0	41.0	100

	1	2	
1	0.30	0.02	0.32
2	0.04	0.23	0.27
3	0.25	0.16	0.41
	0.59	0.41	1.00

TABLE 2.10 – *Le tableau de contingence y^{zw} à gauche et la distribution P^{zw} à droite*

Pour la partie pratique, on a travaillé avec un tableau de contingence montrant la population des femmes actives employées dans la Région métropolitaine de Montréal, Zone de résidence et de profession. Ensuite, on a renommé les variables colonnes par les lettres A, B, C, D et E.

	Professions					
Zone de résidence	Directeurs, gérants, administrateurs et assimilés	Professionnels, enseignants et cols blancs spécialisés	Employés de bureau et travailleurs dans la vente	Ouvriers	Travailleurs spécialisés dans les services, personnel d'exploitation des transports, etc.	**TOTAL toutes professions**
Femmes						
Ville de Montréal	24 025	58 204	76 450	24 385	28 825	**211 889**
Reste de la CUM	22 575	42 207	70 003	14 065	17 435	**166 285**
Couronne Nord	16 785	31 699	63 491	11 975	18 630	**142 580**
Couronne Sud	18 365	35 674	65 290	10 485	19 380	**149 194**
Hors RMR	3 265	7 535	11 089	3 190	3 565	**28 644**
Total Femmes	**85 015**	**175 319**	**286 323**	**64 100**	**87 835**	**698 592**

TABLE 2.11 – *Un exemple d'un tableau de contingence des femmes employées dans la Région métropolitaine de Montréal (Lemelin, A) [13]*

Tableau 2.11 représente une base de données étudiée en 1991 portant sur des professions : (Directeurs, gérants, administrateurs et assimilés), (Professionnels, enseignants et cols blancs spécialisés), (Employés de bureau et travailleurs dans la vente), (Ouvriers) et (Travailleurs spécialisés dans les services, personne d'exploitation des transports, etc) dans la zone résidence : Ville de Montréal, Reste de la CUM (Communauté Urbaine de Montréal), Couronne Nord, Couronne Sud et Hors RMR (Région Métropolitaine de Recensement).

La figure 2.11 représente la matrice initiale et la matrice de classification. Après la classification, on a deux classes avec six blocs délimités par des lignes bleues.

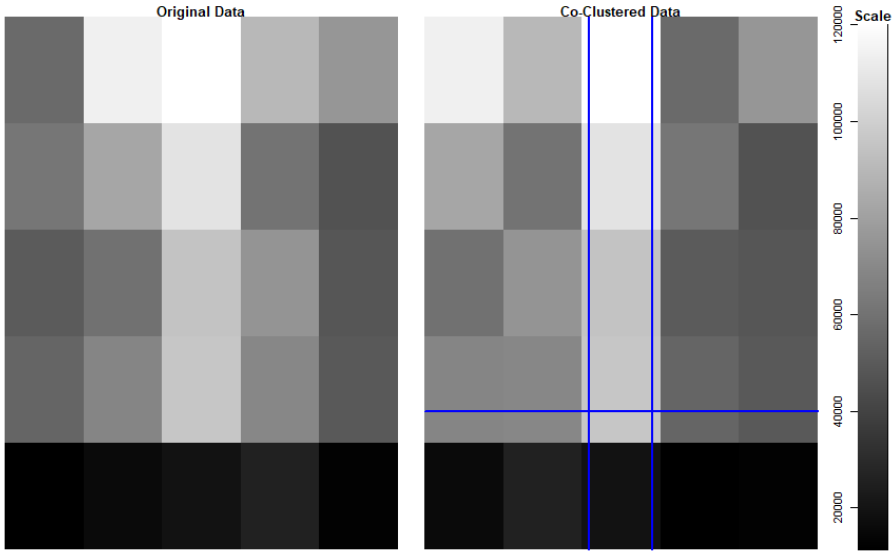

FIGURE 2.11 – *Matrice initiale et la matrice obtenue par la méthode de co-clustering*

Voici une exemple simple avec le code R ci-dessous. On utilise les mêmes packages qu'avec les données continues.

```
df<-matrix(c(56361, 113249, 119996, 89725, 75675, 61721, 82127,
             107822, 60238, 46184, 50072, 59259, 94661,
              74827, 47959, 54371, 68138, 95890, 69263, 49101,
            11535, 16125, 19359, 25495, 12664), byrow = TRUE, ncol=5)

M<-cocluster( df, datatype = "contingency" , nbcocluster = c(2,3),
              model = "pik_rhol_unknown", strategy = coclusterStrategy()
)

# Réorganisation de nos données (classer les données simultanément en bloc)
plot( x = M)
```

2.4.2 Modèle de bloc latent d'un tableau de contingence par la loi de Poisson

Comme dans les données binaires et les variables continues, le modèle de bloc latent est très sollicité pour modéliser les tableaux de contingence. Cependant, à partir du bloc de latent proposé en [4], le bloc latent de Poisson a été utilisé dans les tableaux de contingence pour faciliter le calcul des variables latentes souvent très utilisées dans les grandes bases de données. Ce modèle considère les partitions z et w comme des variables aléatoires latentes (c'est-à-dire non observables). Supposons que chaque bloc $\sigma_k \times \sigma_b$ de x_{ij} est distribué par la loi de poisson $P(\lambda_{ij})$, où les paramètres λ_{ij} s'expriment sous la forme suivante :

$$\lambda_{ij} = \mu_i \nu_j \sum_{k,l} z_{ik} w_{jl} \gamma_{kl} \tag{2.21}$$

Dans ce cas les lignes représentent μ_i, les colonnes représentent ν_j et le partitionnement γ_{kl} sont nos blocs.
Ces paramètres nous permettent de calculer le bloc de modèle latent de Poisson qui s'exprime sous la forme suivante :

$$f(x, \theta) = \sum_{z,w \in Z \times W} \prod_{ik} \pi_k^{z_{ik}} \prod_{j,l} \rho_l^{w_{j,l}} \prod_{i,j} \frac{e^{-\lambda_{ij}} (\lambda_{ij})^{x_{ij}}}{x_{ij}!} \tag{2.22}$$

Cette dernière est paramétrée par $\theta = (\pi, \rho, \mu, \nu, \gamma)$ avec $\pi = (\pi_1, ..., \pi g)$, $\rho = (\rho_1, ..., \rho_m)$, $\mu = (\mu_1, ..., \mu_n)$, $\nu = (\nu_1, ..., \nu_d)$ et $\gamma = (\gamma_{11}, ..., \gamma_{gm})$.

Ce modèle nous aide d'un part, dans la réorganisation et la classification de nos données. D'autre part, cela nous permet de trouver notre matrice de confusion et de faire des précisions sur nos données avec des algorithmes précises. Comme avec les modèles de blocs précédents, ce modèle nécessite des contraintes pour s'identifier.

Supposons que M est une constante arbitraire, voici les contraintes [10] :

$$\mu_k = \sum_i z_{ik}\mu_i \tag{2.23}$$

et

$$\nu_l = \sum_j w_{jl}\nu_j \tag{2.24}$$

Cela nous permet d'exprimer les contraintes suivantes choisies.

$$\sum_i \mu_i = \sum_j \nu_j = \frac{1}{\sum_k \pi_k \gamma_{kl}} = \frac{1}{\sum_l \rho_l \gamma_{kl}} = M \qquad \forall \quad k, l \tag{2.25}$$

Avec ces contraintes on peut conclure que les paramètres à estimer sont réduits en π, ρ et γ puisqu'on peut montrer que $E(x_i) = \mu_i$ et $E(x_j) = \nu_j$. Ainsi μ_i et ν_j peuvent être remplacés par les marges x_i et x_j.

Par exemple, avec la même base de données (tableau de contingence), on a fait une classification avec la loi de distribution de Poisson dont les résultats sont dans la figure 2.12 en deux classe et six blocs.

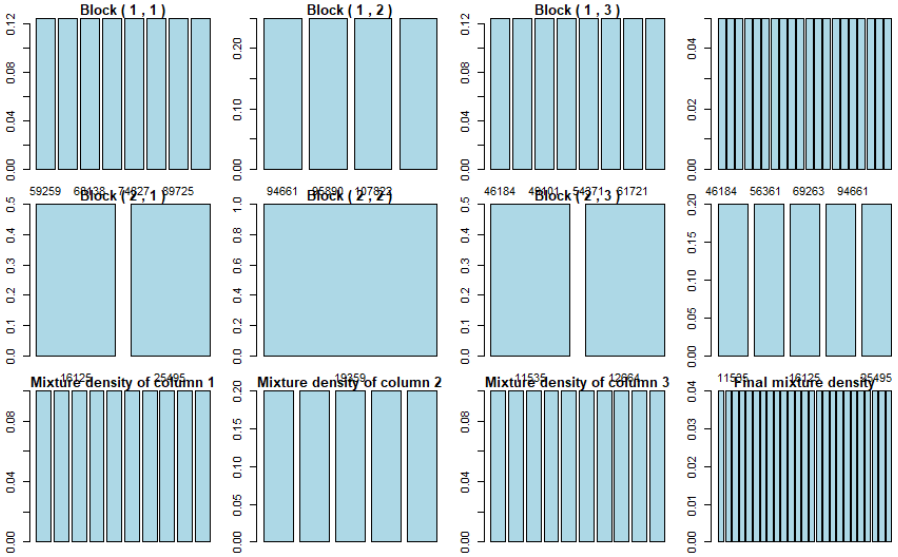

FIGURE 2.12 – *Classification par la distribution de Poisson*

On peut voir le code R ci-dessous.

```
# Classification par simulation
# de bloc latent par la distribution de poisson
plot(x = M,asp = TRUE,type = "distribution")
```

Ainsi ce tableau de contingence à deux dimensions (la zone de résidence et la profession), nous a permis d'obtenir deux classes et six blocs après la classification des variables latentes par la loi de Poisson.
La figure 2.12 représente les deux classes et les six blocs après la classification. Par exemple le bloc (1,1) représente des éléments de (59259,68438,74827 et 89725), le bloc (2,2) est composé des éléments (94661, 95890 et 107822) et le bloc (3,3) est formé des éléments (46184, 49101 et 61721) de la classe 1.

Chapitre 3

Méthode des réseaux de neurones artificiels

3.1 Généralités sur les réseaux de neurones artificiels

3.1.1 Définition d'un réseau de neurones artificiels

De nos jours, plusieurs termes sont évoqués dans la littérature pour désigner l'étendue des réseaux de neurones artificiels comme connexionnisme ou neuromimétique. Cependant, on peut définir un réseau de neurone artificiel ou réseau neuronal artificiel comme un concept schématique dont l'idée est inspirée du fonctionnement des neurones biologiques.

Les neurones biologiques sont des cellules vivantes. Ils assurent la transmission d'un signal bioélectrique appelé influx nerveux. C'est ainsi qu'ils ont deux propriétés : excitabilité c'est-à-dire la capacité de répondre aux messages d'excitation et de convertir

ces messages en influx nerveux et la connectivité, cette étape permet de transmettre les influx nerveux vers le système nerveux. Ces derniers sont liés entre eux grâce à des liaisons appelées **axones**. Les axones conduisent les influx nerveux de la sortie d'un neurone vers l'entré d'un autre neurone. A l'entrée de l'autre neurone se trouve la synapse. Une **synapse** est la région d'interaction entre deux cellules nerveuses, elle permet le passage d'un signal. Elle permet aussi de communiquer l'information sous forme de poids synaptique. Le corps cellulaire représente la partie centrale du neurone où se trouve le noyau de la cellule nerveuse.

3.1.2 Structure

Comme on l'a défini, les réseaux de neurones artificiels sont inspirés des réseaux de neurones biologiques, la figure 3.1 [22] montre une structure de ces derniers. Dans ce cas, chaque neurone artificiel est un processeur élémentaire, c'est-à-dire il reçoit un nombre de variables entrées en provenance de neurone source. De cette façon, chacune de ces entrées est associée à un poids w qui représente la force de la connexion. Dans cette optique, chaque élément est attribué d'une sortie unique, qui se subdivise ensuite pour alimenter un nombre de variables neuronales en avals.

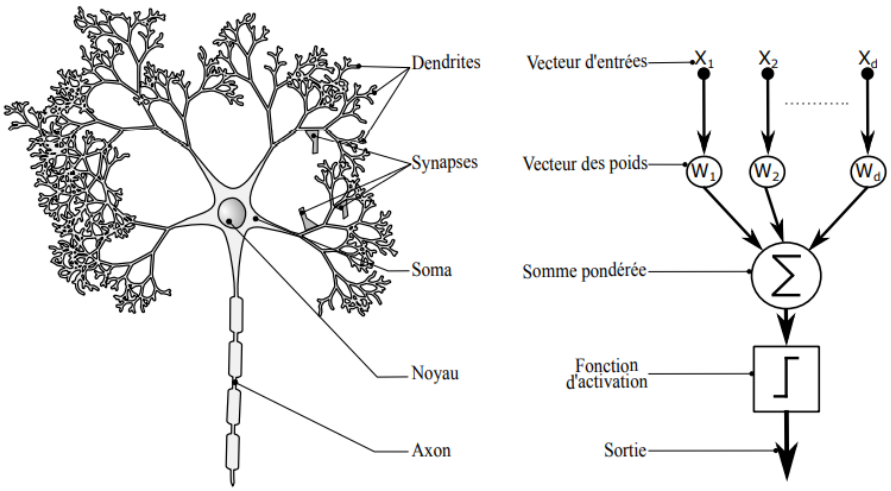

FIGURE 3.1 – Exemple de réseau de neurones artificiels inspiré du réseau de neurones biologiques (Abadi, M [22])

Le tableau 3.1 représente les composantes des neurones biologiques et des neurones artificiels.

Neurone biologique	**Neurone artificiel**
Synapse	Poids de Connexion
Axones	Signal de Sortie
Dendrite	Signal d'Entrée
Corps cellulaire	Fonction d'Activation

TABLE 3.1 – Anotation entre le neurone biologique et le neurone artificiel

- Le poids d'activation : représente dans les réseaux neuronaux l'efficacité d'une connexion. C'est ainsi qu'il favorise différents types d'entrés.

- Les entrées : peuvent prendre des **booléennes**, **binaires (0,1) ou bipolaires (-1,1)** et des nombres **réelles**. Cela nécessite une fonction d'activation pour provoquer une sortie.

Les réseaux neuronaux artificiels ont besoin des fonctions d'activation pour réaliser leurs activités. Pour cela, considérons quelques unes des plus couramment utilisées en pratique mais il en existe d'autres.

- La fonction seuil ou la fonction Heaviside définie par :

$$f(x) = \begin{cases} 1 & si \quad x \geq 0 \\ 0 & si \quad non \end{cases} \tag{3.1}$$

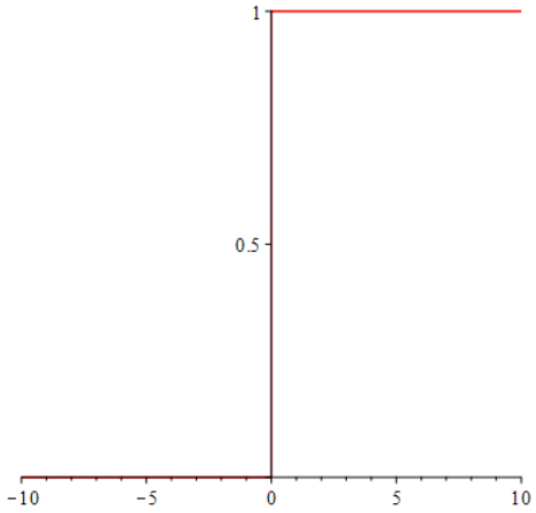

FIGURE 3.2 – *Fonction Heaviside*

- La fonction ReLU (Unité de Rectification Linéaire) definie par :

$$f(x) = max(0, x) \tag{3.2}$$

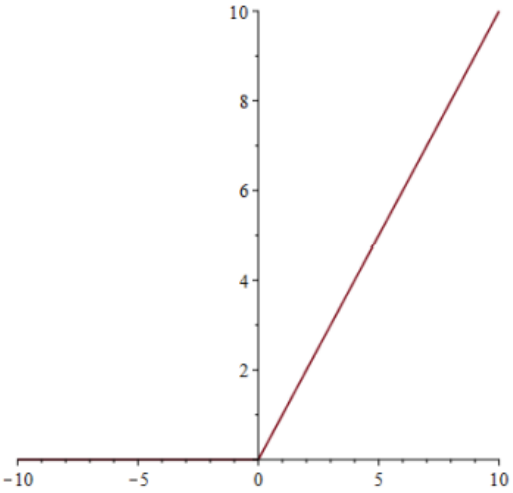

FIGURE 3.3 – *Fonction ReLU*

- La fonction sigmoïde definie par :

$$\forall \quad x \in \mathbb{R} \quad f(x) = \frac{1}{1+e^{-x}} \tag{3.3}$$

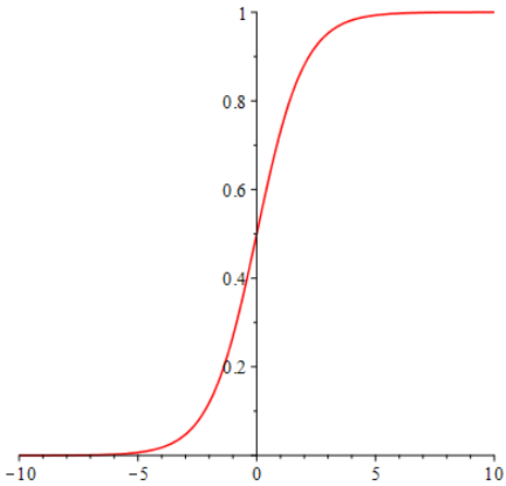

FIGURE 3.4 – *Fonction sigmoïde*

- Fonction linéaire : C'est l'une des fonctions d'activations les plus simples, elle s'exprime par :

$$f(x) = x \tag{3.4}$$

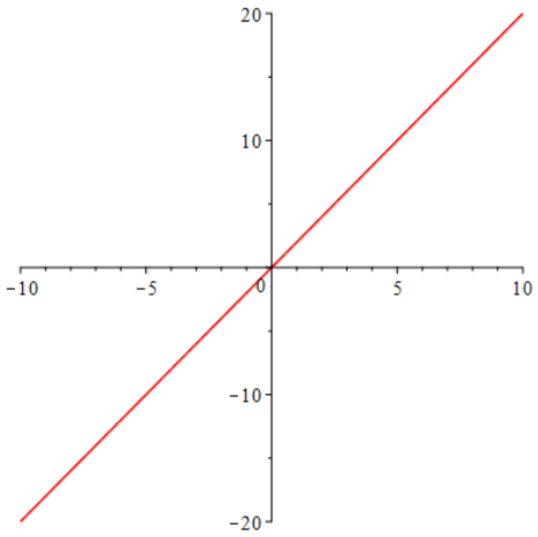

FIGURE 3.5 – *Fonction linéaire*

-Fonction de sortie : Elle permet de calculer la sortie d'un neurone en fonction de son état d'activation.

3.1.3 Principe du modèle mathématique

Considérons le cas général d'un neurone artificiel à n entrées. On va noter les éléments $x_1, ..., x_n$ comme des informations en entrée ou des signaux. C'est ainsi qu'ils vont être associés à un poids synaptique w_i tel que $i \in \{1, ..., n\}$. Notons b le biais du neurone des entrées, on peut l'associer à une donnée $x_0 = -1$. Ainsi, la première opération par le neurone traitera la somme pondérée de ces entrées. Le vecteur d'entrée X de la somme pondérée des informations en entrée est donnée par

l'équation suivante :

$$X = \sum_{i=1}^{n} x_i w_i - b \tag{3.5}$$

X est le niveau d'activation du neurone et le biais b est le seuil d'activation du neurone. Après le traitement de cette dernière, le neurone donne un résultat Y compris entre 0 et 1 grâce à une fonction d'activation f . La sortie Y issue du neurone peut être exprimée par :

$$Y = f(\sum_{i=1}^{n} x_i w_i - b) \tag{3.6}$$

3.1.4 Architecture de réseau

L'architecture ou la topologie des neurones artificiels est un ensemble de plusieurs unités de base associées en couche ou sous-couches et fonctionne en parallèle. Chacune de ces couches comporte un ou plusieurs types de neurones. A partir de figure 3.6, la première couche représente la couche d'entrée, elle permet de lire les informations. La dernière est la couche de sortie, entre ces deux ils se trouvent une couche de liaisons, non visible, on l'appelle une couche cachée du réseau de neurone.

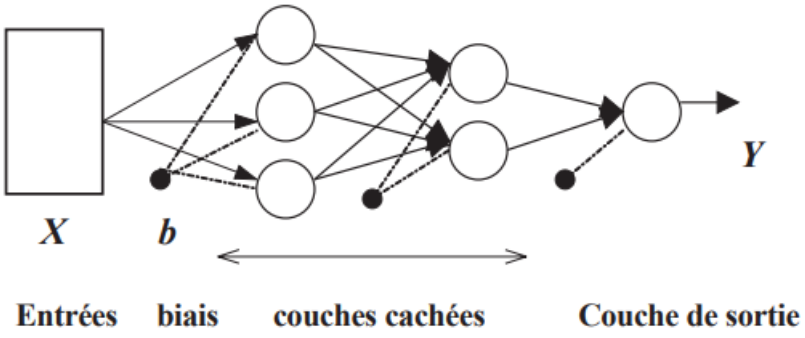

FIGURE 3.6 – *Architecture de réseau de neurones (Coulibaly, P., Anctil, F. et Bernard Bobée, B [19])*

Ici les cercles représentent les neurones disposés sous forme de couches. Notre

réseau comprend trois couches, la couche d'entrée X qui prend les éléments de $x_1, ..., x_n$. La couche de sortie représente la sortie Y et comporte un seul neurone donnant le résultat des calculs internes. On trouve entre ces deux couches, deux couches cachés (hiden en anglais). Il existe différents types d'apprentissage pour les réseaux de neurones artificiels, mais on ne s'intéresse qu'à l'apprentissage non supervisé au cours de notre travail.

3.2 Méthodes neuronales pour la classification non supervisée : Réseau de Kohonen

Teuvo Kohonen [16] a découvert le principe des cartes associatives par l'observation du comportement des êtres vivants. Les cartes auto-organisatrices de Kohonen (SOM, "self-organizing-maps") sont très utilisées pour l'analyse de données et la catégorisation. Elles permettent de cartographier en deux dimensions en général et de distinguer des groupes dans des ensembles de données. Une grande partie de cette section est basée sur les travaux de M. Parizeau 2004 [14].

Le principe des cartes auto-organisatrices est de trouver une projection entre deux espaces : l'espace des données (pleine dimension) et l'espace des représentations (dimension réduite). Mais en cela, la projection doit conserver la topologie des données. La carte de Kohonen est en général à deux dimensions. Comme présenté dans la figure 3.7, chaque neurone de la carte est relié à un neurone de la couche des entrées ; il s'agit ainsi d'un réseau à connexion totale.

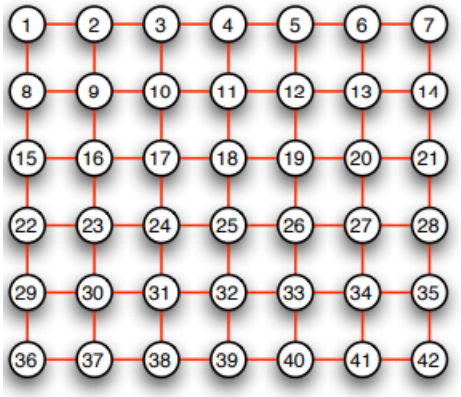

FIGURE 3.7 – *La carte auto-organisatrice de Kohonen (Parizeau, M [14])*

La méthode d'apprentissage de ce type de réseau est dite compétitive. Il s'agit en effet de mettre en compétition entre eux les neurones du réseau et en faire ressortir un vainqueur, ou parfois des neurones voisins du vainqueur. On définit les voisins d'un neurone comme les neurones liés à celui-ci par un chemin de la carte d'au plus k arêtes, k fixé.

L'apprentissage compétitif est caractérisé par les éléments suivants :
- Un ensemble de neurones identiques à l'exception de leur poids,
- Une limite à la force d'un neurone,
- Un mécanisme mettant en compétition les neurones et garantissant qu'un seul neurone de sortie soit actif.

Souvent, un neurone vainqueur modifie son poids w en le rapprochant d'un stimulus d'entrée p lorsqu'il a battu les autres neurones au temps t :

$$\Delta_i\omega(t) = \begin{cases} \eta(t)[p(t) - w_i(t)] & \text{si le neurone est vainqueur} \\ 0 & \text{sinon} \end{cases} \tag{3.7}$$

Avec $\eta(t) \in [0,1]$ représente le taux d'apprentissage. On a le $p(t)$ qui représente le stimulus au temps t. Lorsqu'un voisinage gagne autour du vainqueur, on applique une règle semblable aux voisins, avec cette fois un taux différent :

$$\Delta_i\omega = \begin{cases} \eta_1(t)[p(t) - w_i(t)] & \text{si le neurone est vainqueur} \\ \eta_2(t)[p(t) - w_i(t)] & \text{si c'est un voisin du vainqueur} \\ 0 & \text{sinon} \end{cases} \tag{3.8}$$

où $\eta_1 > \eta_2$.
Le neurone gagnant de la compétition est déterminé comme celui pour lequel la distance avec le stimulus est minimale :

$$g(p) = \arg\min_i \|p - w_i\|, i = 1, 2, ..., S \tag{3.9}$$

où S est le nombre total de neurones du réseau. On met à jour les poids selon l'équation précédente. L'opérateur arg min représente la valeur de la variable pour laquelle la valeur de la fonction concernée atteint son minimum.
A partir de notre première équation, on peut remarquer que le taux d'apprentissage et le neurone gagnant dépendent du temps. Au départ d'un grand taux d'apprentissage ainsi qu'un grand voisinage, on emploie la réduction au fur et à mesure que le temps progresse. Dans ce cas, on utilise souvent une décroissance linéaire :

$$\eta(t) = \begin{cases} \eta_0 - \left(\dfrac{\eta_0 - \eta_\tau}{\tau}\right) t & si \quad t < \tau \\ \eta_\tau & \text{sinon} \end{cases} \tag{3.10}$$

Où η_0 représente le taux d'apprentissage initial, η_τ le taux d'apprentissage final et le τ est le paramètre qui délimite la frontière entre deux phases. En effet, les neurones proches des données initiales sont favorisés et dirigés vers la direction du stimulus (voir figure 3.8).

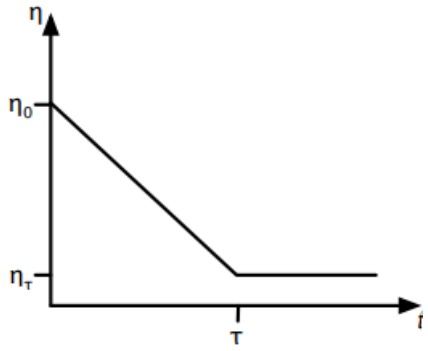

FIGURE 3.8 – *Exemple de décroissance du taux d'apprentissage en fonction du temps (Parizeau, M 2004 [14])*

On peut utiliser plusieurs fonctions pour le voisinage autant qu'elles respectent la condition de non-croissance.
Dans cette étape, les vecteurs poids initiaux sont choisis aléatoirement. L'apprentissage non supervisé est une règle d'apprentissage généralement utilisée avec des réseaux autonomes appelés auto-organisateurs. Les algorithmes à base de cette règle d'apprentissage sont alimentés, au cours de la phase d'apprentissage, par une base d'apprentissage constituée d'un ensemble de stimulus avec une certaine redondance.

Il est possible d'utiliser trois types d'initialisations des neurones [17] : l'initialisation aléatoire, l'initialisation à partir d'un échantillon et l'initialisation linéaire. L'initialisation aléatoire attribue à chaque neurone des poids au hasard. L'initialisation à partir d'un échantillon de l'ensemble d'apprentissage utilise des éléments de l'ensemble d'apprentissage pour déterminer les poids. Ceci a l'avantage que les vecteurs poids des neurones pointent déjà sur les mêmes régions que l'ensemble d'apprentissage. Quant à l'initialisation linéaire, elle tire profit de l'analyse en composantes principales pour l'analyse des données. Les vecteurs poids des neurones sont donc initialisés de telle sorte qu'ils se trouvent dans le même plan que celui formé par les deux vecteurs propres se trouvant dans les directions de plus grande inertie.

3.2.1 Application et interprétation au jeu de données « Iris de Fisher »

Notre application porte sur le jeu de données des Iris de Fisher (voir tableau 3.2). Cette base de données est composée de 150 fleurs qui appartiennent à trois espèces (Setosa, Virginica et Versicolor). L'ensemble des données se compose de 50 échantillons de chacune des espèces. Ces dernières sont reparties en quatre variables quantitatives et désignent respectivement la longueur des sépales, la largeur des sépales, la longueur des pétales et la largeur des pétales. Elles sont toutes mesurées en cm. Les observations sont notées de 0 à 149, le but est de faire une classification des Iris. Pour ce faire, on a utilisé les librairies de python **pandas** et **numpy**. Le tableau 3.2 représente les cinq premières lignes de la base de données.

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

TABLE 3.2 – *Tableau des cinq premières lignes d'Iris de Fisher*

La classification est faite sur les données standarisées (voir le tableau 3.3). Pour faciliter notre représentation graphique, on a nommé les quatre variables par des chiffre de 0 à 3.

	0	1	2	3
0	-0.900681	1.019004	-1.340227	-1.315444
1	-1.143017	-0.131979	-1.340227	-1.315444
2	-1.385353	0.328414	-1.397064	-1.315444
3	-1.506521	0.098217	-1.283389	-1.315444
4	-1.021849	1.249201	-1.340227	-1.315444

TABLE 3.3 – *Tableau standardisé des cinq premières lignes d'Iris de Fisher*

Le graphique illustré à la figure 3.9 est obtenu en fixant le nombre d'époques (epoch en anglais) ainsi que le nombre d'intervalles pour enregistrer les erreurs. Dans ce cas, l'opération progresse de manière décroissante jusqu'à atteindre son minimum de convergence. En effet, nous avons utilisé la librairie **som** qui contient la fonction **SOM**. Un exemple permettant d'importer la fonction SOM : ***from som import SOM*** se trouve à l'annexe C .

Nous avons utilisé un réseau carré de taille 5×5. La figure 3.9 montre la progression dans le temps par diminution progressive jusqu'à un certain nombre afin de former un plateau.

FIGURE 3.9 – *Graphique d'apprentissage avec 10000 époques*

La librairie **som** nous permet de visualiser le nombre d'échantillons qui sont classés en groupe, sur la carte. Le graphique illustré à la figure 3.10 est obtenu par le calcul de la distance des éléments les plus proches et permet de les visualiser.
Il représente les trois classes obtenues après la classification, numérotées de 0, 1 et 2 (voir la légende).
La première classe représente les espèces de setosa qui sont bien identifiées par rapport au deux autres espèces. La deuxième classe ceux de virginica et la troisième les versicolor, sont aussi bien identifiées mais il existe quelques fleures de virginica mal placer, qu'elle associe à versicolor.

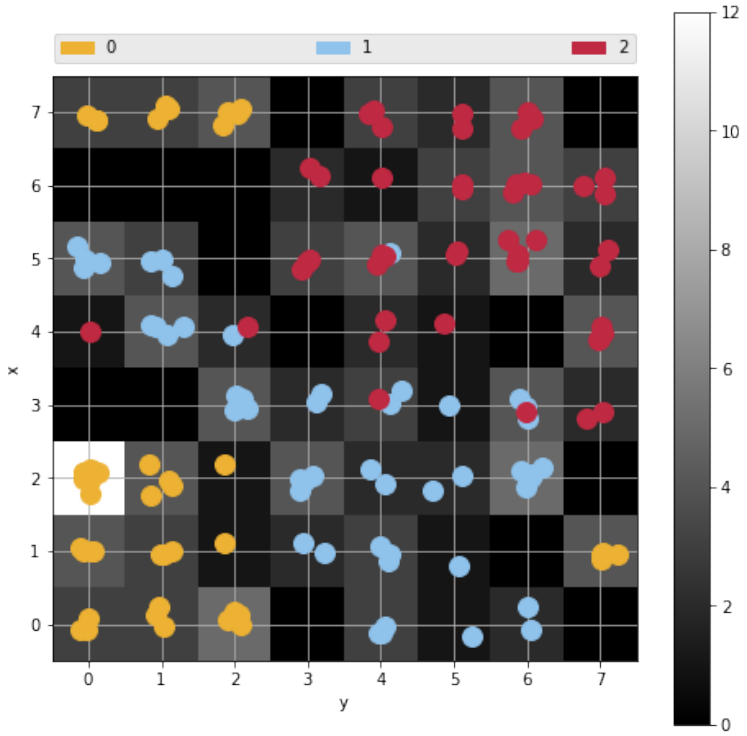

FIGURE 3.10 – *Classification non supervisée du jeu de données d'Iris de Fisher*

Chapitre 4

Étude comparative clustering, co-clustering et réseau de neurones artificiels

Ce dernier chapitre présente une analyse comparative des différentes méthodes : clustering, co-clustering et réseau de neurone artificiel à l'aide des données "Wine". Ce dernier porte sur trois catégories d'observations de zones différentes. En premier lieu, on présentera le jeu de données puis on décrira la démarche utilisée. Par la suite, on fera la comparaison à l'aide des résultats obtenus pour chaque méthode par le logiciel R.

4.1 Presentation des données Wine

Les données concernent le vin blanc cultivé dans la même région Nord du Portugal (https ://archive.ics.uci.edu/ml/datasets/wine+quality). Ces données sont le résultat

des tests physico-chimiques de 12 constituants trouvés dans chacun des trois types de vins. Cet ensemble de données était utilisé par Digitas Advanced Analytics dans l'objectif de comprendre les relations entre certaines propriétés physico-chimiques et la qualité obtenue des vins afin de prendre des décisions plus concrètes lors de la production. Le jeu de données est composé de 4898 observations et de 12 variables. Soit 11 variables physico-chimiques et une variable de sortie (qualité). Toutes les variables physico-chimiques sont quantitatives. Les variables sont composées de :

1 **fixed acidity** qui correspond à des acides impliqués dans le vin, fixes ou non volatils.

2 **volatile acidity** c'est la quantité d'acide acétique dans le vin, qui, à des niveaux trop élevés, peut entraîner un goût de vinaigre désagréable.

3 **citric acid** c'est la variable qui peut ajouter la fraîcheur et la saveur aux vins.

4 **residual sugar** correspond à la quantité de sucre restant après l'arrêt de la fermentation.

5 **chlorides** c'est la quantité de sel dans le vin.

6 **free sulfur dioxide** indique la forme libre de SO2 (dioxyde de soufre) qui existe en équilibre entre SO2 moléculaire et l'ion bisulfite.

7 **total sulfur dioxide** correspond à la quantité de formes libres SO2.

8 **density** représente la densité de l'eau.

9 **PH** décrit le degré d'acidité ou de basique d'un vin sur une échelle de 0 (très acide) à 14 (très basique).

10 **sulphates** indique l'additif du vin qui peut contribuer aux niveaux de gaz sulfureux (S02).

11 **alcohol** qui correspond à la variable alcool.

12 **qualité** c'est la variable de sortie

Dans la pratique, on a décidé d'enlever les variables dioxyde de soufre libre et qualité. En effet, le dioxyde de soufre libre n'est pas retenu, car c'est une composante de dioxyde de soufre total. D'où ces variables sont fortement corrélées. C'est ce qui

nous permet à retenir l'une d'entre elles. Dans cette optique, pour ne pas aberrer la comparabilité de nos méthodes, il convient de ne pas tenir en compte la qualité du vin. D'où la variable qualité ne sera pas utilisée. Enfin, on se retrouve avec 10 variables quantitatives.

Le tableau 4.1 représente les premières lignes de la base de données Wine avec les dix premières variables.

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
1	7.0	0.270	0.36	20.70	0.045	45.0	170.0	1.0010	3.00	0.45
2	6.3	0.300	0.34	1.60	0.049	14.0	132.0	0.9940	3.30	0.49
3	8.1	0.280	0.40	6.90	0.050	30.0	97.0	0.9951	3.26	0.44
4	7.2	0.230	0.32	8.50	0.058	47.0	186.0	0.9956	3.19	0.40
5	7.2	0.230	0.32	8.50	0.058	47.0	186.0	0.9956	3.19	0.40
6	8.1	0.280	0.40	6.90	0.050	30.0	97.0	0.9951	3.26	0.44
7	6.2	0.320	0.16	7.00	0.045	30.0	136.0	0.9949	3.18	0.47
8	7.0	0.270	0.36	20.70	0.045	45.0	170.0	1.0010	3.00	0.45
9	6.3	0.300	0.34	1.60	0.049	14.0	132.0	0.9940	3.30	0.49
10	8.1	0.220	0.43	1.50	0.044	28.0	129.0	0.9938	3.22	0.45
11	8.1	0.270	0.41	1.45	0.033	11.0	63.0	0.9908	2.99	0.56
12	8.6	0.230	0.40	4.20	0.035	17.0	109.0	0.9947	3.14	0.53
13	7.9	0.180	0.37	1.20	0.040	16.0	75.0	0.9920	3.18	0.63
14	6.6	0.160	0.40	1.50	0.044	48.0	143.0	0.9912	3.54	0.52
15	8.3	0.420	0.62	19.25	0.040	41.0	172.0	1.0002	2.98	0.67
16	6.6	0.170	0.38	1.50	0.032	28.0	112.0	0.9914	3.25	0.55
17	6.3	0.480	0.04	1.10	0.046	30.0	99.0	0.9928	3.24	0.36

TABLE 4.1 – *Tableau des données Wine*

4.2 Démarche utilisée

Dans cette section, les trois méthodes décrites auparavant seront appliquées sur les données Wine. La particularité à souligner est que parmi les trois méthodes de co-clustering, seule celle sur les données continues est appliquée étant donné le type des données. Elle utilise la distance euclidienne pour la représentation géométrique et le modèle de probabilité notamment la loi gaussienne pour former les blocs des éléments similaires. Le k-means par sa simplicité sera utilisé pour classer les observations en

K classes (groupe) en fonction de leur similarité. Cette classification sera effectuée à l'aide de package R **NbClust** [18] afin de déterminer le choix optimal du nombre de classe. En ce qui concerne le réseau de Kohonen, une classification des vecteurs prototypes représentée par chaque neurone sera effectué pour former les super-classes. Pour ces méthodes de classification, le choix de nombre de classes est important. Ainsi, ces trois méthodes (k-means, co-clustering des données continues et Kohonen) exigent dès le départ de connaitre le nombre de classes de sortie. La démarche utilisée normalise les données avant d'appliquer l'une de nos méthodes de classification.

Le tableau 4.2 indique les données standardisées, ce qui nous facilitera la classification des méthodes décrites auparavant. Cette normalisation s'exprime de la manière suivante :

$$n(x) = \frac{x_i - \bar{x}}{\sigma_x} \tag{4.1}$$

Avec $n(x)$ représente la norme calculée, x_i les valeurs des vriables, $\bar{x}$ la moyenne de chaque variable et σ_x l'écart-type.

fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide
0.17207939	-0.08176155	0.21325843	2.821061144	-0.03535139	0.56987339
-0.65743400	0.21587359	0.04799622	-0.944668824	0.14773200	-1.25289074
1.47560044	0.01745016	0.54378284	0.100271952	0.19350284	-0.31210925
0.40908322	-0.47860841	-0.11726599	0.415725772	0.55966962	0.68747108
0.40908322	-0.47860841	-0.11726599	0.415725772	0.55966962	0.68747108
1.47560044	0.01745016	0.54378284	0.100271952	0.19350284	-0.31210925
-0.77593592	0.41429702	-1.43936365	0.119987816	-0.03535139	-0.31210925
0.17207939	-0.08176155	0.21325843	2.821061144	-0.03535139	0.56987339
-0.65743400	0.21587359	0.04799622	-0.944668824	0.14773200	-1.25289074
1.47560044	-0.57782013	0.79167615	-0.964384688	-0.08112224	-0.42970694
1.47560044	-0.08176155	0.62641395	-0.974242620	-0.58460155	-1.42928727
2.06811001	-0.47860841	0.54378284	-0.432056368	-0.49305986	-1.07649421
1.23859662	-0.97466699	0.29588953	-1.023532279	-0.26420562	-1.13529306
-0.30192826	-1.17309042	0.54378284	-0.964384688	-0.08112224	0.74626992
1.71260427	1.40641417	2.36166713	2.535181121	-0.26420562	0.33467802
-0.30192826	-1.07387870	0.37852064	-0.964384688	-0.63037240	-0.42970694
-0.65743400	2.00168446	-2.43093689	-1.043248143	0.01041946	-0.31210925
-0.77593592	3.78749533	1.20483167	-1.023532279	-0.76768494	-0.37090810
0.64608705	0.61272045	0.70904505	-1.043248143	-0.58460155	-1.07649421
-0.42043018	0.31508531	-1.60462585	0.218567135	-0.08112224	-0.07691388
-0.77593592	3.78749533	1.20483167	-1.023532279	-0.76768494	-0.37090810
-0.53893209	0.31508531	0.37852064	-0.688362596	-0.35574732	-0.95889653
-0.06492444	-0.18097327	0.70904505	-0.924952961	0.14773200	0.33467802
0.88309087	3.88670704	-1.60462585	-0.964384688	1.29200317	-0.60610347
-0.30192826	-0.08176155	0.62641395	-1.003816415	0.28504454	-1.13529306
0.17207939	-0.28018498	-0.11726599	0.514305090	0.01041946	1.21666067
0.05357748	-0.37939670	0.13062732	-1.062964006	0.28504454	-0.01811504
0.17207939	0.01745016	0.46115174	0.455157499	0.23927369	-0.19451157

TABLE 4.2 – *Tableau standardisé*

4.3 Présentation des résultats

Les résultats obtenus nous permettront de comparer les performances des clustering, co-clustering et réseau de Kohonen (Voir script R en Annexe D).

Analyse par k-means

Avec la méthode de k-means, le nombre de classe optimal obtenu grâce au package de **NbClust** est égale à trois. La première classe de la partition contient 1125 observations, soit la classe la moins dense. La deuxième est composée de 1977 observations et la troisième contient 1796 observations. Ces deux dernières classes sont plus denses. Le tableau 4.3 résume la partition des classes retenue après la classification.

Classes	Taille
1	1125
2	1977
3	1796

TABLE 4.3 – Résultat des classes selon la méthode de k-means

La figure 4.1 illustre les données en trois classes par la méthode de k-means.

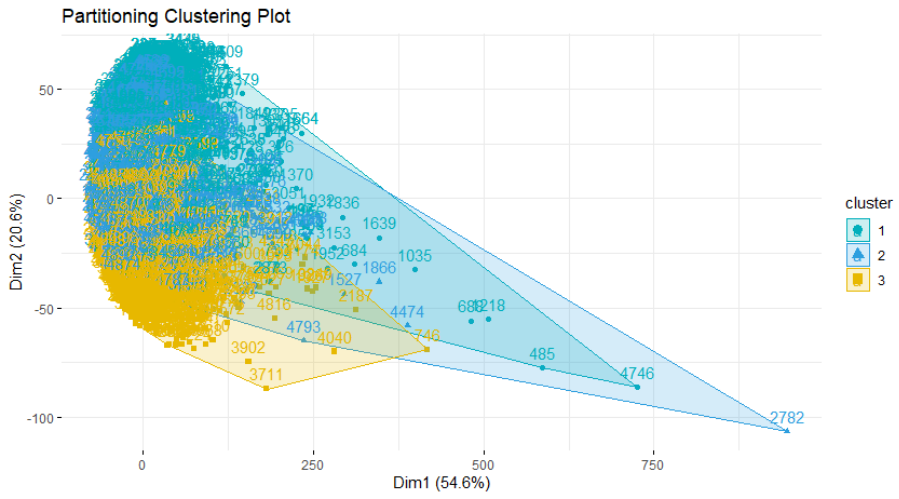

FIGURE 4.1 – *Partitionnement des classes avec la méthode de k-means*

Analyse par le co-clustering

Le co-clustering utilisé est la méthode des données continues, on conserve la même base de données matricielle des données Wine voir (tableau 4.1).

Une première classification en bloc de trois lignes et de trois colonnes a permis de classer chaque élément dans le bloc auquel il appartient. On obtient, par la suite, trois classes avec neuf blocs séparé en ligne bleue illustrés à la figure 4.2.

Le résultat obtenu après la classification par la méthode des variables continues est représenté par le tableau 4.5. On remarque que la deuxième classe est composée de 2415 éléments, soit la classe la plus dense.

Classes	**Taille**
1	1792
2	2415
3	691

TABLE 4.4 – Résultat des classes par la méthode de co-clustering

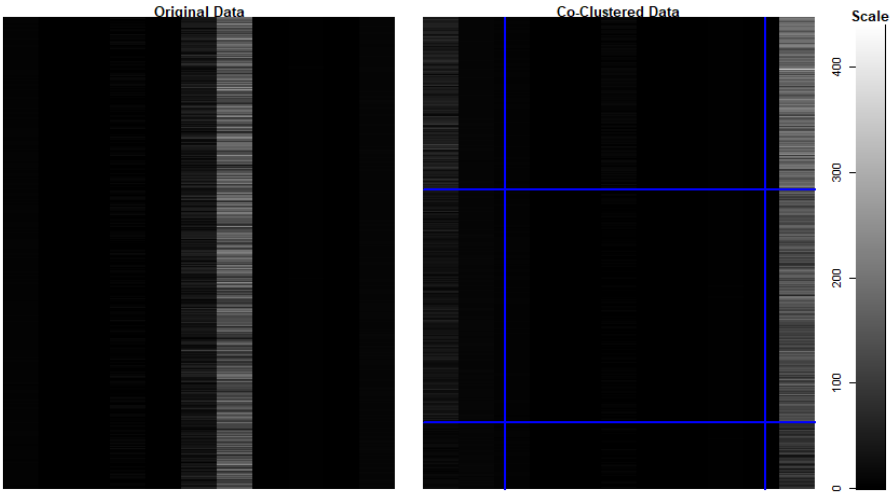

FIGURE 4.2 – *Classification par la méthode de co-clustering*

Analyse par réseau de neurone Kohonen

La carte auto-organisatrice de Kohonen obtenue après la classification est de forme

carrée et dispose 100 neurones (10 par 10). Elle est illustrée à la figure 4.3 qui permet de visualiser des composants attirés par chaque classe de neurone. On peut voir que plus le cercle est dense, plus le neurone correspondant attire les objets à classifier.

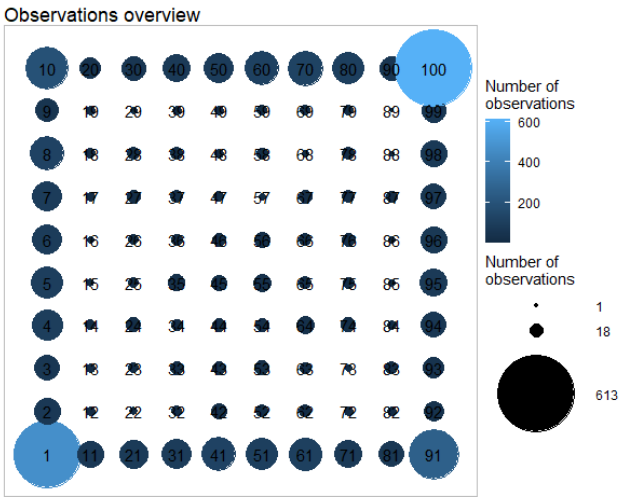

FIGURE 4.3 – *Les éléments des classes de la carte auto-organisatrice*

On peut dire que le nombre de classes correspond au nombre de neurones de la carte. La création des super-classes à un regroupement de 100 classes de la carte est effectuée par le bias de la classification ascendante hiérarchique. Ce regroupement sera fait en trois classes, nombre optimal. Ces classes sont représentées sur carte auto-organisatrice après la classification (voir figure 4.4).

Le résultat des classes obtenues après ce regroupement est représenté au tableau 4.5. On peut dire qu'on a peu près la même distribution que k-means. La première classe est composée de 1840 éléments, soit la classe la plus dense. Les classes 2 et 3 sont moins denses et représentent moins d'éléments.

Classes	Taille
1	1840
2	1534
3	1524

TABLE 4.5 – Résultat des classes selon la méthode de kohonen

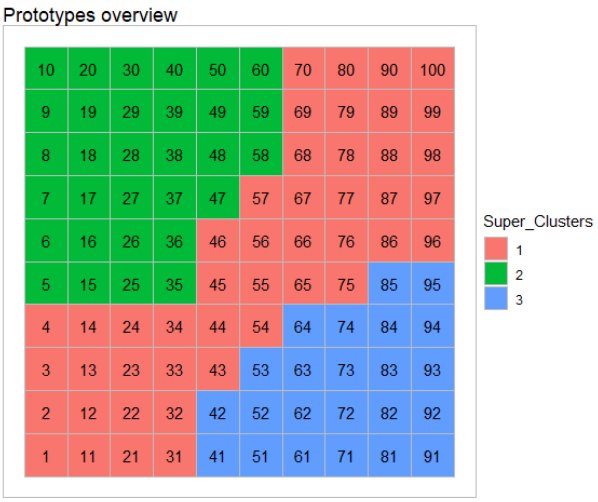

FIGURE 4.4 – *Répresentation des super-classe de la carte auto-organisatrice*

4.4 Identification des formes fortes et comparaison

La concordance des classes facilite l'identification des formes fortes. Ces dernières indiquent l'ensemble des individus stables qui restent toujours dans la même classe quelle que soit la méthode appliquée.

La comparaison des k-means avec le co-clutering permet d'analyser les observations qui se retrouvent dans les mêmes classes malgré la différence qui existe entre les deux algorithmes. Il en résulte 71% de formes fortes, tel que représente dans le tableau 4.6. Ainsi on peut constater que la classe 1 des k-means ressemble plus à la classe 3 de co-clustering. La classe 2 des k-means correspond à la classe 2 de co-clustering (voir

tableau 4.6).

	co-clustering		
k-means	1	2	3
1	103	601	**1273**
2	1	**1113**	11
3	699	5	**1092**

TABLE 4.6 – Formes fortes résultant de la méthode des k-means et de la méthode de co-clustering

La comparaison de ces deux dernières (k-means, co-clustering) avec l'algorithme de Kohonen permet de constater deux formes fortes distincts, k-means et Kohonen donnent 58 % de formes fortes (voir tableau 4.7) et co-clustering et Kohonen ressortent 50%.

On peut remarquer qu' à partir du tableau 4.7, la classe 1 des k-means correspond à celle de Kohonen. La classe 2 ressemble à classe 3 de Kohonen et la classe 3 des k-means ressemble à la classe 2 de Kohonen. Cette disposition de classe est pareille pour le co-clustering et Kohonen voir tableau 4.8.

	Kohonen		
k-means	1	2	3
1	**1105**	470	402
2	404	37	**684**
3	678	**1079**	39

TABLE 4.7 – Formes fortes résultant de la méthode des k-means et de la méthode de Kohonen

	Kohonen		
co-clustering	1	2	3
1	**1041**	990	168
2	833	116	**954**
3	313	**480**	3

TABLE 4.8 – Formes fortes résultant de la méthode de co-clustering et de la méthode de Kohonen

	k-means	co-clustering	Kohonen
k-means	100 %		
co-clustering	71 %	100 %	
Kohonen	58 %	50 %	100 %

TABLE 4.9 – Tableau récapitulatif des formes fortes

En se basant sur les résultats obtenus, nous remarquons que les approches statistiques classiques (k-means et co-clustering) ont de meilleurs résultats concernant l'identification des formes fortes (71%) par rapport à la méthode de Kohonen et k-means (58 %) (voir tableau 4.9). On peut voir aussi que les k-means et la méthode de Kohonen (58 %) sont plus proche que co-clustering et Kohonen (50 %) , cela peut être expliquer qu'ils (k-means et Kohonen) utilisent la distance comme mesure de ressemblance entre les observations. Alors que le co-clustering des données continues après le regroupement des données avec la distance comme mesure, utilise le modèle gaussien des variables latentes. En effet, la différence majeure de ces trois méthodes est qu'avec les k-means, les nœuds sont indépendants les unes des autres. Avec la méthode de Kohonen, les nœuds sont situés sur une grille et chaque vecteur est considéré comme ayant des voisins. En plus, la mise à jour du vecteur gagnant peut provoquer une modification de ces voisins. Quant au co-clustering, la dispersion

dépend des types de données en sommant tous ces blocs pour former un seul bloc distinct de classe.

Conclusion et perspectives

Le but de ce mémoire est d'établir un cadre de comparaison des méthodes statistiques et neuronale en matière de classification non supervisée dont le but principal est de regrouper les individus en classes homogènes. Il s'agit aussi d'étudier le lien entre trois méthodes de classification à savoir le k-means qui est une méthode classique, le co-clustering qui est une méthode simultanée et Kohonen qui est une méthode neuronale.

Nous avons presenté les trois méthodes avant de les comparer. Ensuite, nous les avons comparées en les appliquant sur la base de données **Wine**. Il en ressort que les méthodes statistiques, qui utilisent la distance comme mesure de ressemblance, sont plus similaires à la méthode de Kohonen qu'aux méthodes de co-clutering à l'approche probabiliste, c'est-à-dire les variables latentes utilisées après le regroupement par les lois de probabilité. On peut en déduire que sur le plan algorithmique, la méthode de k-means est similaire à la méthode de Kohonen dans le cas où les vecteurs prototypes de la carte auto-organisatrice se manifestent presque de la même manière que les centres définis dans les initialisations de classes issues de k-means. Il faut noter que les modèles de co-clustering, bien qu'ils soient performants, supposent que dans chaque classe, la distribution est unique et suit la loi de probabilité spécifique au type de données. Par ailleurs, on peut dire que la méthode de Kohonen nécessite une taille importante de données pour effectuer l'apprentissage.

Comme perpective, il serait intéressant d'élargir notre étude comparative à d'autres jeux de données. Il serait important de mener une étude approfondie des méthodes de co-clutering et envisager éventuellement de considérer le co-clustering dans les réseaux de neurones artificiels.

Bibliographie

[1] Ghazzali, N. Méthodes d'analyse des données. *Université du Québec à Trois-Rivières*, 2020.

[2] Berry, M. W. Survey of Text Mining Clustering, Classification, and Retrieval Scanned by Velocity. *Springer-Verlag New York Berlin Heidelberg*, 2003.

[3] Lebarbier, E. et Mary-Huard, T. Classification non supervisée. *AgroParis-Tech*, 2016.

[4] Govaert, G. et Nadif, M. Mutual information, phi-squared and model-based co-clustering for contingency tables. *Springer-Verlag Berlin Heidelberg*, janvier 2016.

[5] Salah, A. and Nadif, M. Directional co-clustering. *Springer-Verlag GmbH Germany* pages 2-30, Avril 2018.

[6] Huang, S., Wang, H., Li, D., Yang, Y. and Li, T. Spectral co-clustering ensemble. *Elsevier*, 2015.

[7] Govaert, G and Nadif, M. Co-Clustering. *ISTE Ltd and John Wiley & Sons, Inc.*, pages 101-204, 2014.

[8] Parent, E et Bernier, J. Le raisonnement bayésien : Modélisation et inférence. *Springer-Verlag France, Paris*, pages 170-181, 2007.

[9] Ficcara, E., De Micheli, G., Yoon, S., Benini. L and Macii E. Joint co-clustering : Co-clustering of genomic and clinical bioimaging data. *Elsevier*, 2008.

[10] Brault, V and Mariadassou M. Co-clustering through Latent Bloc Model : a Review. *Journal de la Société Française de Statistique*, Vol. 156 2015.

[11] Mariadassou, M., Brault, V. et Keribin, C. Normalité asymptotique de l'estimateur du maximum de vraisemblance dans le modèle de blocs latents. *HAL Id : hal-01440084*, 27 Jan 2017.

[12] Bhatia, P.S., Iovleff. S. et Govaert, G. blockcluster : An R Package for Model-Based Co-Clustering. *Journal of Statistical Software*, V 76, Février 2017.

[13] Lemelin, A. L'analyse des tableaux de contingence. *UQAM* 2004.

[14] Parizeau, M. Réseaux de neurones. *Université de Laval*. Automne 2004.

[15] Bessai, F.Z, Krelifaoui, M. and Guergab, M. tilisation des cartes auto-organisatrices de Kohonen dans la recherche documentaire. *Centre de Recherche sur l'Information Scientifique et Technique (CERIST)*, 2002.

[16] Kohonen, T. Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics*, 43, 59-69 1982.

[17] Ralia, A.T. Developpement stratégique des PME manufacturiéres : une approche typologique par les réseaux de neurones. *Université du Québec à Trois-Rivières Service de la bibliothèque*, Novembre 2005.

[18] Charrad, M. et al. An R Package for Determining the Relevant Number of Clusters in a Data Set. *Journal of Statistical Software*, Vol. 61, October 2014.

[19] Coulibaly, P., Anctil, F. et Bernard Bobée, B. Prévision hydrologique par réseaux de neurones artificiels : état de l'art. *CNRC Canada*, Vol. 26, 1999.

[20] Belanger, H. Reseau de kohonen pour la detection des contours d'object dans une image a niveau gris. *Ecole technologie superieure université du Québec*, 28 Fevrier 1998.

[21] McCulloch, W.S. and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Vol. 5, 1943.

[22] Abadi, M. Réalisation d'un réseau de neurones "SOM" sur une architecture matérielle adaptable et extensible à base de réseaux sur puce "NoC". *Université de Lorraine (Ecole Doctorale IAEM Lorraine)*, 7 Juillet 2018.

[23] Guyon, H. Les modèles d'équations structurelles avec R et JASP (version 4). *Université de Bretagne Occidentale*, janvier 2020

Annexe A

Application de clustering sur la base de données des CVs avec Rstudio

```
################## Classification par culustering ###############
library(stats)
library(readxl)
library(factoextra)


## Importation de la base de données

Tab1 <- read_excel("Master Automne/Recherche/Tab1.xlsx")
#View(Tab1)
def <- Tab1
def
scd <- scale(def[,-1,2])
```

```
scd
## spectacle de decomposition
df<-princomp(scd)
df

## dendogramme des niveaux d'études superieurs IRH

df <- dist(scale(def[,-1,2]), method = "euclidean")

hc <- hclust(df, method = "ward.D2")
fviz_dend(hc, k = 4,
          cex = 0.5,
          k_colors  <- c("#E7B800", "#FC4E07", "#2E9FDF","#00AFBB"),
          colors_labels_by_k = TRUE,
          rect = TRUE,
          rect_border = c("#E7B800", "#FC4E07","#2E9FDF","#00AFBB"),
          rect_fill = TRUE)

## Partitionnement clustering des données étudiants et niveau d'études IRH

km.res<- kmeans(def[,-1,2], 4, nstart = 25 )

fviz_cluster(km.res, data = df,
             palette = c("#00AFBB","#2E9FDF", "#E7B800", "#FC4E07"),
             ggtheme = theme_minimal(),
             main = "Partitioning Clustering Plot")
```

Annexe B

Application de co-clustreing sur différents jeux de données

```
library(blockcluster)
library(blockmodels)
library(NbClust)
library(class)
library(cluster)
library(questionr)

######### Cette partie consiste a simulée les données binaires

u <- coclusterBinary(
  data = binarydata,
  nbcocluster = c(2,3), # Nombres de cluster
  model = "pik_rhol_epsilonkl", # Model binaire permetant
  # de donnée la proportion et la dispersion des variable
```

```
  a = 6, # les paramétres de controle
  b = 8,
  strategy = coclusterStrategy(
    nbtry = 4, # Nombre d'essaie
    nbxem = 9 # Nombre maximum d'itineration
  )
)
# Visualisons la methode latente

summary(object = u)

# Reorganisation de nos données (classer les données
# simultanement en bloque)

plot(
  x = u,
  asp = FALSE,
  type = "cocluster"
)

# Recherche pour bien comprendre la fonction cocluste
?cocluster

# Classification de la bloque latente par la distribution de Bernoulli
plot(
  x = u,
  asp = TRUE,
  type = "distribution"
)
# Visualiser les classes des lignes
```

```
table(u@rowclass)

# Visualiser les classes des colonnes
table(u@colclass)

##### Exemple simple pour les tableaux de contégenges
# On utilise les méme library ci-dessus

# Population active employée dans la Région métropolitaine de Montréal
# Zone de résidence selon la profession, 1991

df<-matrix(c(56361, 113249, 119996, 89725, 75675, 61721,
           82127, 107822, 60238, 46184,  50072,
           59259, 94661, 74827, 47959, 54371,
           68138, 95890, 69263, 49101, 11535,
           16125, 19359, 25495, 12664), byrow = TRUE, ncol=5)
df

colnames(df)<-c("A","B","c","D", "E")
rownames(df)<-c("Ville de Montréal ", "Reste de la CUM","Couronne Nord",
 "Couronne Sud","Hors RMR ")
df

summary(df)

M<-cocluster(
   df,
  datatype = "contingency" , # visualiser les type de données
  nbcocluster = c(2,3), # nombre de regroupement de lignes et de colonnes
  model = "pik_rhol_unknown", # Modele de proportion et de dispertion
```

```
  strategy = coclusterStrategy(
    nbtry = 5, # Nombres d'éessaient
    nbxem = 1 # Nombres d'essaient
  )
)

# Reorganisation simultaner des  données
plot( x = M, asp = FALSE)
# Classification par la distribution de Poisson
plot(x = M,asp = TRUE,type = "distribution")

### Classification simultanée par LBM de Gaussien des variables continues
# Modelisons les données (simulation)

h <- cocluster( gaussiandata,
datatype = "continuous", # visualiser les type de données
  nbcocluster = c(2,3),
 # nombre de regroupement de lignes et de colonnes de clustre
 model = "pik_rhol_sigma2kl", # Modele de proportion et de dispertion
  strategy = coclusterStrategy(
    nbtry = 5, # Nombres d'essaient
    nbxem = 10
  )
)

##### Reorganisation de nos données
plot(x = h, asp = FALSE, type = "cocluster")
# Classification  par la distribution gaussiennne
plot( x = h, asp = TRUE, type = "distribution")
# Visualisation des obsevation de chaque classe
```

```
 h@rowclass
# Visualiser les classes
table(h@rowclass)

# Resume des informations effectuées

summary(object = h)

# Visualiser les classes des colonnes

table(h@colclass)
```

Annexe C

Application de réseau de neurones artificiels sur les Iris de Fisher avec Python

```
####### Importons les bibliothéques
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn import datasets
import numpy as np
import pandas as pd

# chargeons le jeu de données de l'iris
iris = datasets.load_iris()
# convertir en dataframe
iris_dataset = pd.DataFrame(iris.data)
# charger les étiquettes d'iris
iris_labels = iris.target
```

```
iris_dataset.head()

# utilisons un scalier standard sur les données
from sklearn.preprocessing import StandardScaler

standard = StandardScaler()
iris_dataset_standar = pd.DataFrame(standard.fit_transform(iris_dataset))
iris_dataset_standar.head()

# importation du bibliothéque
#from sklearn_som.som import SOM
from som import SOM

# Initialison SOM
df = SOM(5, 5)
# Adapton le SOM pour 10000 époques, enregistrer l'erreur toutes les 300 étapes
df.fit(iris_dataset_standar.to_numpy(),10000, save_e=True, interval=300)

# trace l'historique des erreurs d'entraînement
df.plot_error_history()

#
df.plot_distance_map()
```

Annexe D

Clustering, co-clustering et réseau de neurones artifieciels : Données ≪Wine≫

```
library(cluster)
library(fpc)
library(dendextend)
library(mclust)
library(kohonen)
library(SOMbrero)
library(colorspace)
library(NbClust)
library(keras)
library(blockcluster)
library(blockmodels)
```

```
 library(readr)

Vin <- read_delim("Recherche/Doc de recherche/Vin.csv")



View(Vin)

#library(readr)
#vin <- read_csv("Master Automne/Recherche/R/wine.csv")
#View(wine)

# Netoyer les données manquant
sum(is.na(Vin))

D<-Vin[1:11]
D
vin.stand <- scale(D[,-6])
vin.stand

#hf<-princomp(scd)
#hf
hf <- dist(scale(D[,-6]), method = "euclidean")

Result <- kmeans(D[,-6], 3, nstart = 25 )
attributes(Result)
Result$centers # Calcule la moyene des variable dans chacune des classes
Result$cluster # Donne le numero de classe de chaque individu
Result$size # Donne le taille des classe obtenu
Result$betweenss
Result$withinss
```

```
### Representation graphique
table(Result$cluster)

fviz_cluster(Result, data = hf,
             palette = c("#00AFBB","#2E9FDF", "#E7B800", "#FC4E07"),
             ggtheme = theme_minimal(),
             main = "Partitioning Clustering Plot")

# Taille des classes
table (Result$cluster)



# Nombre optimume des cluster
fviz_nbclust(wine.stand, kmeans, method = "wss") +
geom_vline(xintercept = 3, linetype = 2)

#Result$cluster<-as.factor(Result$cluster)
#lot(Vin, Result$cluster)



########## Co-clustering des données continues

coclust<-Vin
# Transformer les données en matrice
coclust<-as.matrix(D[,-6])
coclust

Q <- cocluster(coclust, datatype = "continuous" , nbcocluster = c(3,3),
               model = "pik_rhol_sigma2kl",
               strategy = coclusterStrategy(
```

```
                )
)
# Reorganisation de nos données (classer les données simultanement en bloc)
plot( x = Q, asp = F)
# Classification par simulation de la bloc latente par la distribution de poisson
plot(x = Q, asp = TRUE,type = "distribution", legend=TRUE)

# Resume des informations effectuées
summary(Q)

# Donne le numero de classe de chaque observation
Q@rowclass

# Taille des classes

table(Q@rowclass)

# Formes fortes entre k-means et co-clustering
table(Result$cluster, Q@rowclass)



######## Classification de kohonen
## utilisation de la fonction SOM

def_som<-trainSOM(x.data = D[,-6], verbose = TRUE, nb.save = 5)

# Composition des classes

def_som$clustering
```

```
# Effectiffe des classe

table(def_som$clustering)
def_som$clustering

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

summary(def_som)

#### classification par variable

par(mfrow=c(2,2))

plot (def_som, what="obs", type="color", variable=1, print.title=TRUE,
      main="T3resin")



# classification en superclasse

mc <- superClass(def_som, k=3)
mc

attributes(mc)

mc$cluster

table(Result$cluster,def_som$clustering)
# Representation graphyque
```

```
plot(mc, plot.var = FALSE)

### Plot sup

plot(mc, type="grid")
mc$som

summary(mc)

VT <- mc$cluster[def_som$clustering]



Vin$cluster1 <- VT

Vin$cluster1
# Formes fortes entre k-means et Kohonen

table(Result$cluster, Vin$cluster1)

# Formes fortes entre  co-clustering et Kohonen

table(Q@rowclass, Vin$cluster1)



# Formes fortes entre ces trois

table(Result$cluster, Q@rowclass, Vin$cluster1)

table(Result$cluster, Vin$cluster1, Q@rowclass)
table(Vin$cluster1)
```

```
mc$cluster

table(mc$cluster)

clust <- mc$cluster[def_som$clustering]

wine$cluster1<- clust

table(wine$cluster1)
```