STA211 : Cartes de Kohonen

Vincent Audigier, Ndeye Niang

14 mars, 2025

1 Introduction

Les cartes de Kohonen (aussi appelées cartes auto-organisatrices ou *Self-Organizing Maps* en anglais) ont été introduites par T. Kohonen en 1981. Elles constituent une méthode de classification non-supervisée permettant de partitionner un ensemble de n individus en groupes d'individus homogènes, selon une méthode assez similaire à la méthode des k-moyennes. L'intérêt premier de cette approche par rapport aux k-moyennes est que cette approche fournit automatiquement une visualisation des proximités entre les classes via une carte en dimension 1, 2 ou 3.

2 Principe général

2.1 Eléments de vocabulaire

Les centres des classes sont ici appelés *prototypes* ou *vecteurs référents*. Ce sont des éléments de l'espace des individus (noté ici V et inclu dans $\mathbb{R}^p$). Comme pour la méthode des k-moyennes, une classe sera identifiée à son prototype (ou son vecteur référent). Par ailleurs, à chaque prototype on associe un *neurone* dans une grille. Ici, une classe sera donc aussi identifiée à un neurone. La grille est également appelée *carte* et donne son nom à la méthode. Un exemple de carte est donné en Figure 2.1.

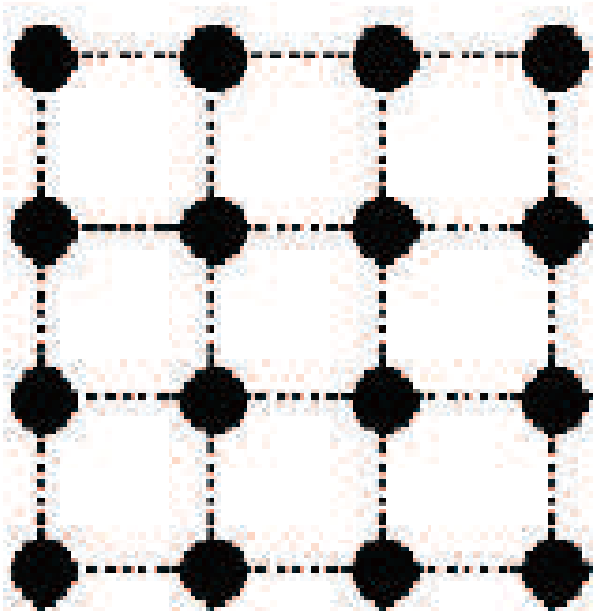


Figure 2.1: Exemple de carte : grille rectangulaire

Un des intérêts majeurs des cartes de Kohonen est que la grille va être munie d'une distance, au même titre que l'espace des individus. Ceci va permettre de définir la notion de voisinage entre neurones et ainsi une topologie sur celle-ci. De ce fait, la carte fournira une visualisation (ici en dimension 2) des proximités entre les classes.

2.2 Algorithme

Comme pour les k-moyennes (au besoin, on pourra consulter Dreyfus et al. (2002), chapitre 7), il existe deux versions de l'algorithme des cartes de Kohonen : une version *on-line*, dans laquelle les prototypes sont mis à jour à chaque prise en compte d'un nouvel individu (tiré au hasard sans remise) et une version *batch*, où les prototypes sont mis à jour uniquement après avoir passé en revue l'intégralité des individus. Nous présentons dans cette section la version on-line qui correspond à la version originelle de l'algorithme SOM (*Self-Organizing Maps*).

Algorithme SOM, version on-line

---

1. Initialisation
   - définir une grille A rectangulaire de K neurones
   - définir une distance et un voisinage entre neurones
   - définir une distance sur l'espace des observations
   - définir une séquence pour la température $[r_{max}, \ldots, r_{min}]$ de longueur T
   - définir une séquence pour le taux d'apprentissage $[\alpha_{max}, \ldots, \alpha_{min}]$ de longueur T
   - initialiser les K prototypes
2. Itération : pour $t \in 1, \ldots, T$
   - tirer une observation x_i au hasard sans remise
   - trouver le prototype w_l le plus proche de x_i (au sens de la distance sur V)
   - déplacer tous les prototypes w_k ($1 \leq k \leq K$) voisins de w_l, i.e. dont les représentants dans la grille sont à une distance inférieure à $r(t)$ (au sens de la distance sur la grille) vers x_i selon

$$w_k^{(t+1)} \leftarrow w_k^{(t)} + \underbrace{\alpha(t) * \left(x_i - w_k^{(t)}\right)}_{\Delta w_k^{(t)}}$$

- fin pour

On considère n individus dans V. K prototypes $(w_k)_{1 \leq k \leq K}$ sont d'abord disposés sur un plan de cet espace, généralement obtenu par une analyse en composantes principales. A chacun de ces prototypes dans V, on associe un représentant dans une grille A, i.e. un couple de coordonnées $s_k \in \mathcal{S}_1 \times \mathcal{S}_2$ avec $\mathcal{S}_1 = \{1, 2, \ldots, s_1^{max}\}$ $\mathcal{S}_2 = \{1, 2, \ldots, s_2^{max}\}$. Cette grille est généralement rectangulaire, mais d'autres types de grilles, tels que les grilles hexagonales, peuvent être considérés. Pour un individu x_i pris au hasard dans V, on cherche dans un premier temps le prototype w_l ($1 \leq l \leq K$) le plus proche au sens d'une certaine distance **dans** $\mathbb{R}^p$ (généralement euclidienne). On déplace alors tous les prototypes les *plus proches* de w_l en direction de x_i selon un pas Δw_k. La notion de voisinage entre prototypes dépend d'une distance dans **l'espace des coordonnées de la grille**. Deux prototypes sont voisins, si leurs couples de coordonnées **dans la grille** sont proches (et non pas s'ils sont proches dans l'espace des individus V) . Toute la spécificité de la méthode repose sur ce dernier point : les prototypes sont ainsi "attirés" par les données, mais contraints à rester rattachés aux autres via une grille en deux dimensions.

Plus précisément, les prototypes sont déplacés selon

$$w_k^{(t+1)} \leftarrow w_k^{(t)} + \underbrace{\alpha(t) * \left(x_i - w_k^{(t)}\right)}_{\Delta w_k^{(t)}}$$

où $\alpha(t)$ est un scalaire appelé *learning rate*. Si $\alpha(t)$ est petit (proche de 0), les voisins sont peu déplacés, et si α est grand (proche de 1), alors le déplacement est important. Par ailleurs, pour définir à partir de quelle distance deux prototypes ne sont plus voisins, la version la plus simple de l'algorithme SOM consiste à utiliser la distance euclidienne (ou de Manhattan) et à se donner un seuil $r(t)$ spécifiant si un prototype est voisin ou non.

Lors de la mise en oeuvre de l'algorithme, les valeurs de $\alpha(t)$ et $r(t)$ sont diminuées progressivement de telle sorte qu'à la fin de l'algorithme, $\alpha(T) = 0$ ce qui implique que les prototypes ne sont plus déplacés, et $r(T) = 1$, ce qui signifie que le voisinage est réduit à un seul prototype. Plus précisément, on se donne d'abord une valeur minimale et maximale pour $r(t)$ et pour $\alpha(t)$. Généralement, $\alpha(t)$ et $r(t)$ évoluent selon un pas variable entre ces bornes. Selon Dreyfus et al. (2002) (chapitre 7), $\alpha(t)$ doit prendre une forme décroissante du nombre d'itérations t, il peut être constant par morceaux, égal à $1/\sqrt{t}$ ou prendre d'autres formes. Quant à $r(t)$, il est généralement choisi selon

$$r(t) = r_{max} * \left(\frac{r_{min}}{r_{max}}\right)^{\frac{t}{T-1}}$$

(voir Figure 2.2 pour une illustration). Le nombre d'itérations T est généralement de l'ordre de plusieurs milliers. Pour un $r(t)$ et $\alpha(t)$ fixés, on tire un individu unique sans remise et on met à jour les prototypes, on passe alors aux valeurs suivantes de $r(t+1)$ et $\alpha(t+1)$ et ainsi de suite.

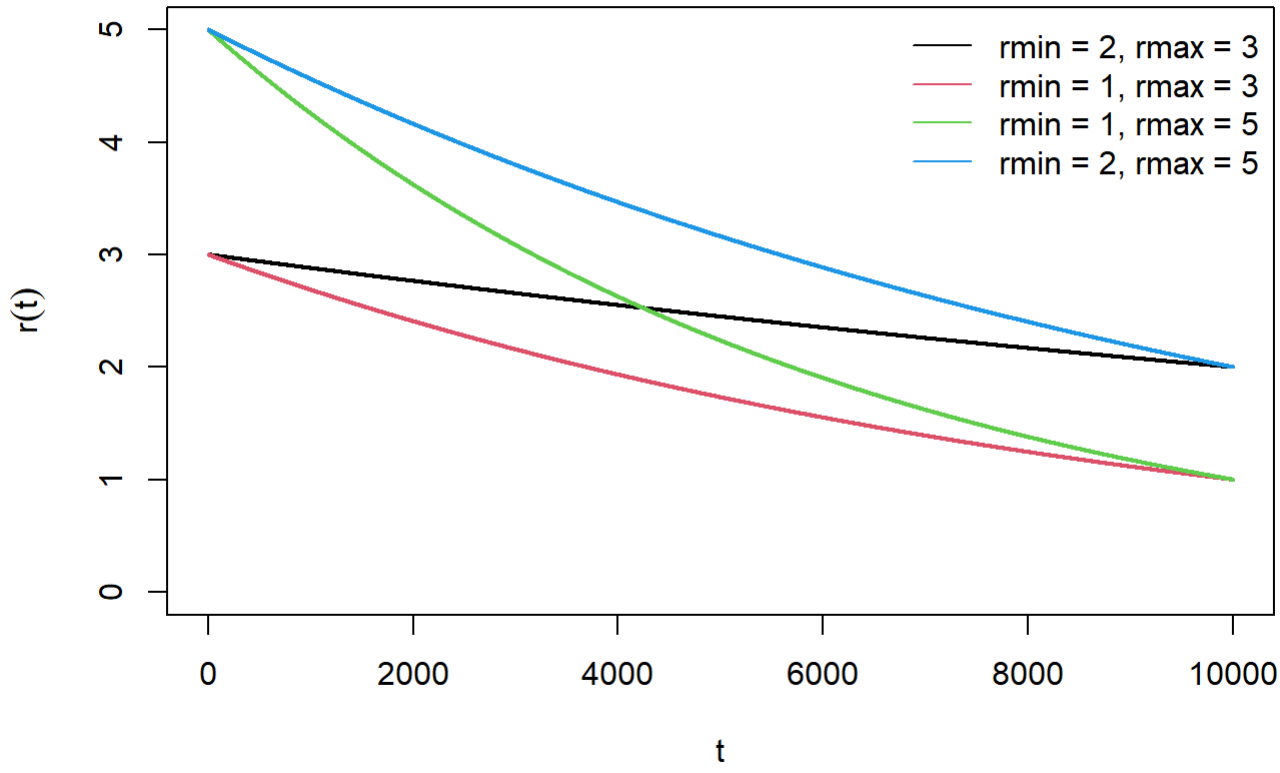


Figure 2.2: Evolution de r en fonction de rmin et rmax pour T = 10000 itérations

3 Exemple

On applique la méthode sur les données iris de Fisher. Ce jeu de données contient les mesures en centimètres des longueurs et largeurs des pétales et sépales pour 150 fleurs appartenant à 3 espèces d'iris (setosa, versicollor et virginica). Les quatre premières variables (continues) sont utilisées pour construire la carte.

On utilise une carte rectangulaire à 5×5 noeuds. La distance sur la carte est la distance euclidienne, idem pour l'espace des individus V. L'algorithme est itéré 500 fois.

Les données sont réparties en 25 classes de la façon suivante :

Table 3.1: Effectifs des classes. Seuls les effectifs des classes non vides sont reportés

Indice de classe	5	6	11	12	13	15	16	17	18	19	21	22	23	24	25
effectif	49	1	7	2	2	1	5	4	2	3	34	6	10	5	19

On voit que certains prototypes n'ont aucun individu rattaché (`1, 2, 3, 4, 7, 8, 9, 10, 14, 20`), tandis que d'autres ne sont rattachés qu'à un seul individu (`6, 15`). Ceci n'est pas surprenant, car le choix du nombre de noeuds a été choisi a priori, et n'est pas problématique dans un premier objectif de visualisation des structures de groupes.

La carte obtenue peut être visualisée de différentes façons. Tout d'abord, on peut représenter les effectifs afin d'identifier les zones de fortes densité (Figure 3.1).

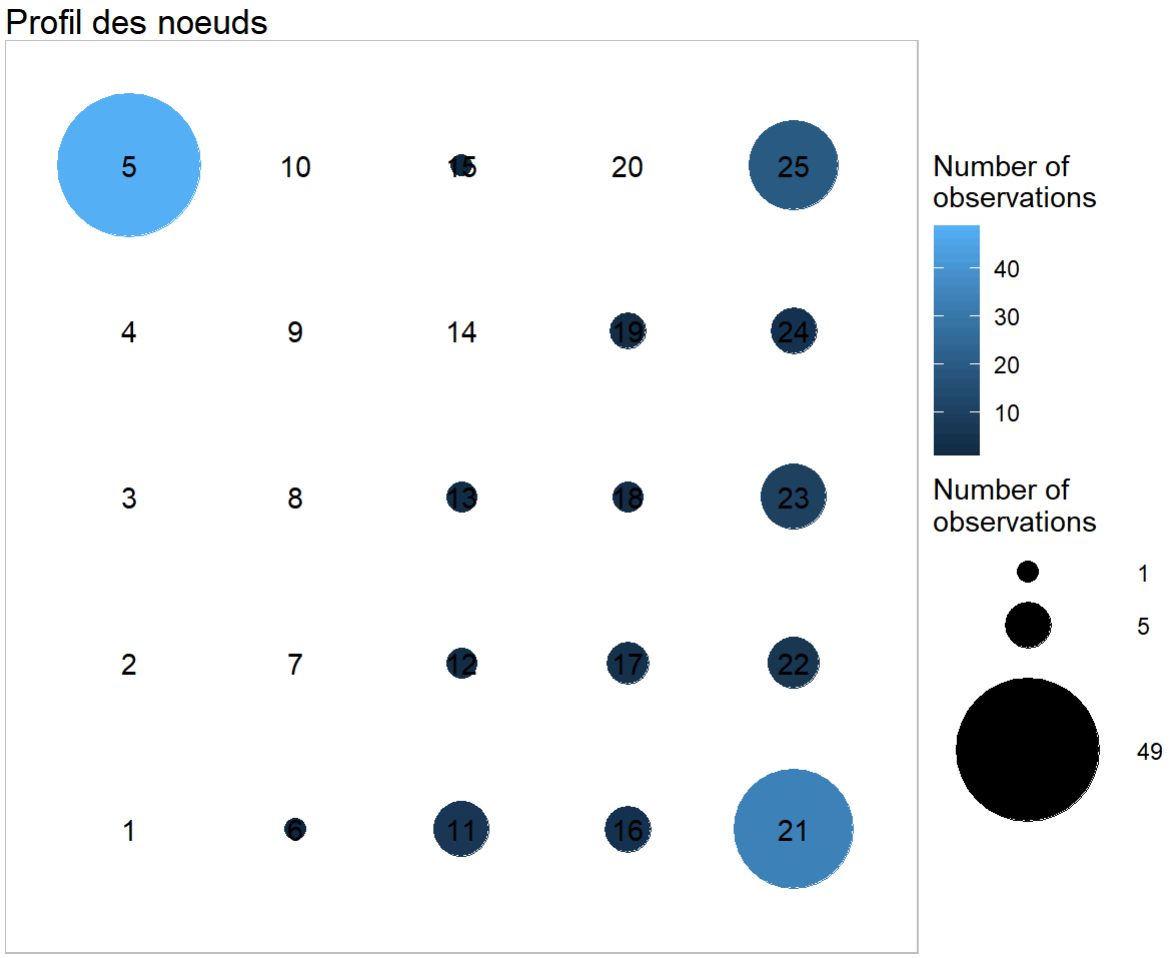


Figure 3.1: Carte de Kohonen pour les données iris : effectifs.

Rappelons que la carte préserve les proximités entre les prototypes, c'est-à-dire que des noeuds proches sur la carte correspondent à des prototypes proches dans l'espace des individus. Ici on peut identifier 3 zones de fortes densités, ce qui correspond au nombre de variétés d'iris.

On peut également la représenter en considérant les distances entre prototypes (Figure 3.2), ce qui permet d'apprécier les proximités entre classes.

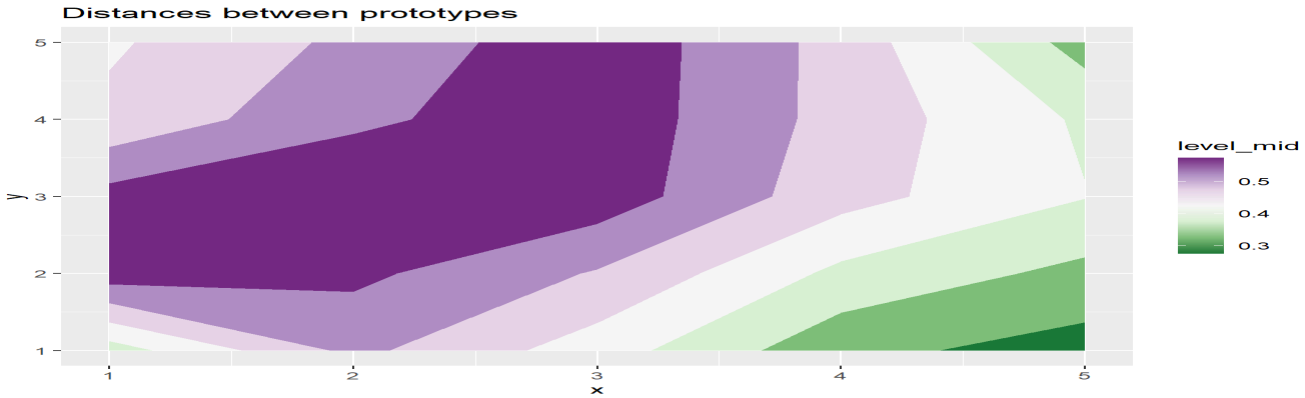


Figure 3.2: Distance entre prototypes : distance moyenne entre un prototype et ses voisins. Les coordonnées d'un point correspondent au couple de coordonnées associé dans la grille. Les valeurs intermédiaires sont obtenues par lissage.

On repère ainsi qu'il y des écarts importants sur la diagonale. Ceci traduit par exemple que les fleurs affectées au groupe de coordonnées (3,5) (groupe 15) sont plutôt différentes de celles de ses groupes voisins comme le groupe de coordonnées (4,5) (groupe 20). En revanche, les groupes de coordonnées (4,1), (5,1) et (5,2) (groupes 16, 21 et 22) sont eux assez proches.

NB : la numérotation des groupes est ici effectuée à partir du coin inférieur gauche, puis en remontant colonne par colonne comme indiqué en Figure 3.1.

Il est également possible de représenter la carte en décrivant les individus rattachés à chaque noeud (3.3).

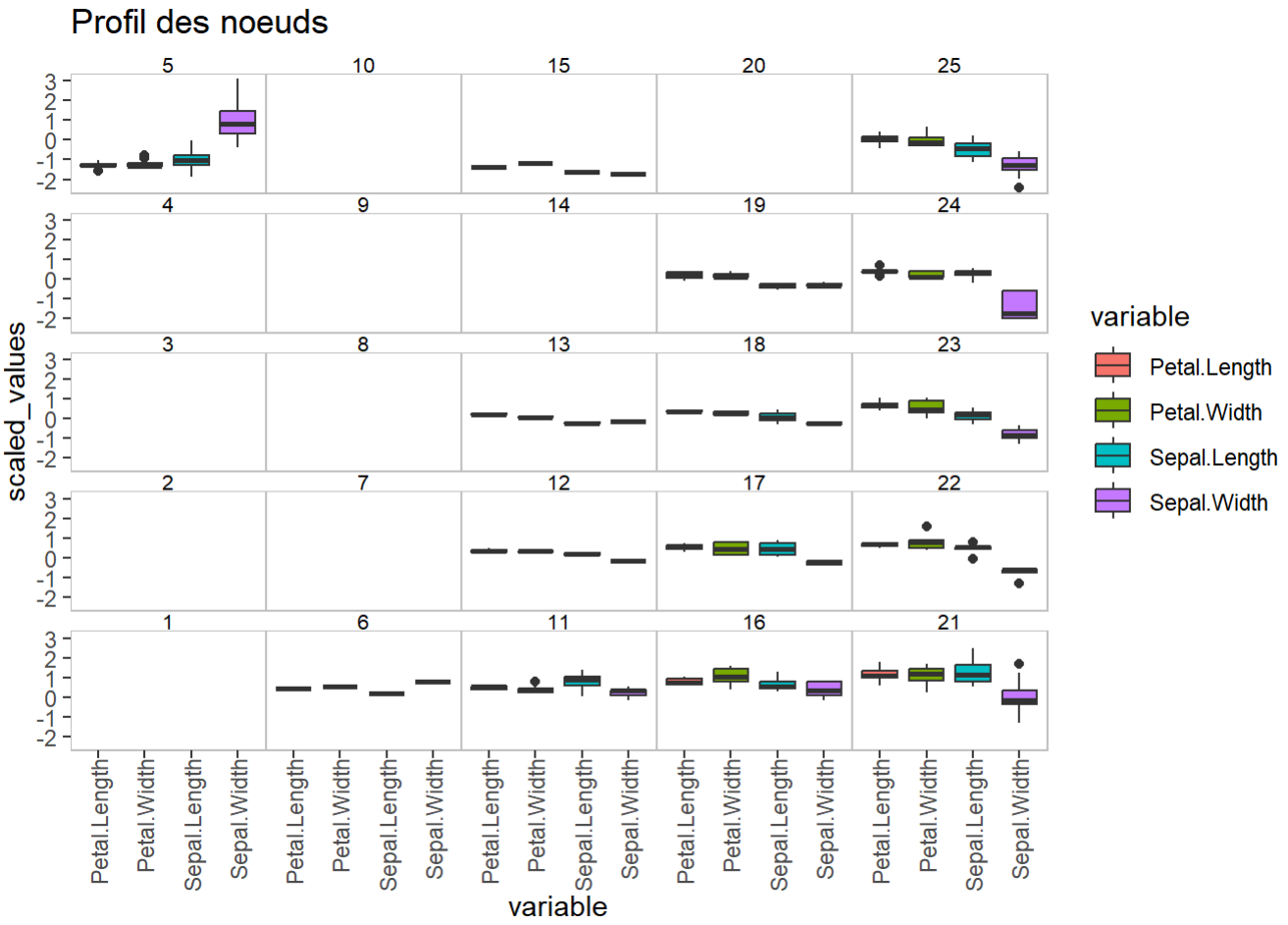


Figure 3.3: Carte de Kohonen pour les données iris : caractérisation.

Dans cette figure, chaque classe est décrite par les boîtes à moustaches (les groupes vides ne sont pas représentés). Par exemple, dans la classe 5 (représentée en haut à gauche), la largeur du sépale est particulièrement élevée. Ce type de représentation est une autre façon de visualiser la carte. Il est clair qu'elle est rapidement limitée quand le nombre de variables est grand. Dans ce cas, on pourra par exemple identifier les variables discriminantes (à l'aide des valeurs-test), puis à ne représenter que ces variables de façon à alléger les graphiques.

4 Variantes

4.1 Fonction de voisinage

La méthode précédente est la version la plus simple des cartes de Kohonen. Une méthode plus sophistiquée (et généralement utilisée) consiste à déplacer les prototypes selon

$$w_k^{(t+1)} \leftarrow w_k^{(t)} + \alpha(t) \times h(\| s_k - s_{k(x_i)} \|_A)(x_i - w_k^{(t)})$$

où $s_{k(x_i)}$ désigne le couple de coordonnées associé au prototype le plus proche de x_i, $\|.\|_A$ la norme sur la grille et h une *fonction de voisinage* donnant plus de poids aux prototypes dont les représentants sur la grille sont les plus proches de $s_{k(x_i)}$. La fonction h est un noyau, c'est-à-dire une fonction symétrique, positive, d'intégrale égale à 1. Les choix les plus classiques pour h sont le noyau Gaussien ou le noyau uniforme (cf Figure 4.1) dont les expressions sont celles des fonctions de densité des lois du même nom.

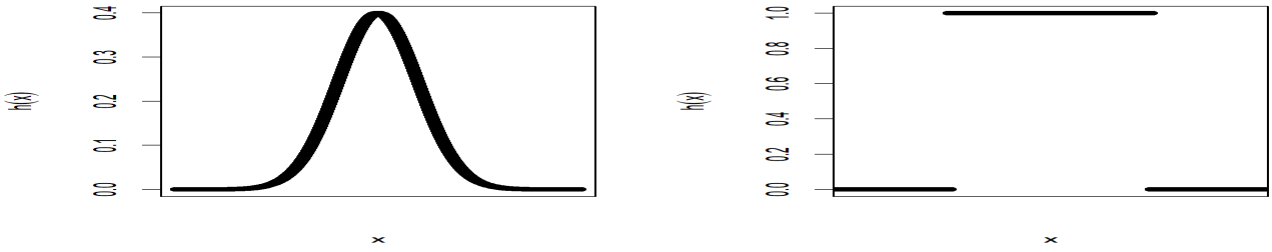


Figure 4.1: Noyau Gaussien et noyau uniforme

Ces noyaux sont paramétrables, ce qui permet de régler la taille du voisinage en fonction des itérations (ainsi on notera h_r plutôt que h), par exemple en modifiant la variance pour le noyau Gaussien, ou le support pour le noyau uniforme. Ainsi, on utilise comme fonction h_r:

- $h_r(d) = \begin{cases} 1 & \text{si } d < r \\ 0 & \text{sinon} \end{cases}$
- $h_r(d) = exp\left(-(d/r)^2\right)$
- $h_r(d) = exp(-|d|/r)$

On notera que la première expression correspond au noyau uniforme utilisée dans la présentation de l'algorithme : si la distance entre deux neurones de la grille est inférieure au seuil r, alors le prototype est déplacé, sinon $h_r(d)$ prend la valeur 0 et il n'y a pas de mise à jour.

4.2 Version batch

La version présentée dans la section 2 est une version on-line où les prototypes sont mis à jour à chaque nouvel individu. La version *batch* consiste elle à mettre à jour les prototypes en identifiant dans un premier temps tous les individus rattachés à un prototype, puis à mettre à jour les prototypes selon le barycentre des individus qui lui sont rattachés. Les poids des individus sont définis selon la fonction de voisinage h_r (Hastie, Tibshirani, and Friedman (2009)).

La version batch permet d'accélérer l'algorithme, tout en donnant des résultats similaires. Elle est néanmoins très sensible à l'initialisation. Contrairement à la version on-line, elle est déterministe, ce qui implique que pour des initialisations identiques l'algorithme donnera toujours la même carte.

Il existe aussi des versions intermédiaires où plusieurs individus sont tirés simultanément.

4.3 Données non-numériques

Des méthodes ont également été proposées pour le cas où les données ne sont pas toutes numériques. L'algorithme de Kohonen a été étendu d'abord au cas de données qualitatives en particulier binaires dans le cadre, par exemple, de l'algorithme Binbatch (Lebbah, Badran, and Thiria (2000)) et à travers les variantes de l'algorithme Kohonen Multiple Correspondence Analysis (KMCA) (Cottrell, Ibbou, and Letrémy (2004)). Ensuite, pour prendre en compte les données mixtes, Lebbah et al. (2005) et Chen and Marques (2005) ont proposé simultanément une version mixte quasi-identique de l'algorithme SOM. Les auteurs utilisent une extension de la distance euclidienne classique combinant une partie continue regroupant l'ensemble des variables quantitatives et une partie constituée de variables qualitatives mises sous forme disjonctive au préalable. Dans l'algorithme NCSOM proposé par Chen and Marques (2005), les auteurs combinent simplement la distance euclidienne classique et la distance de Hamming pour données binaires. Lebbah et al. (2005) propose d'utiliser dans l'algorithme MTM (Mixed Topological Map) un paramètre d'ajustement de l'influence de la partie quantitative des données par rapport à la partie qualitative dans l'évaluation de la similarité entre les observations.

5 En pratique

5.1 Qualité de représentation

Les résultats de l'algorithme SOM dépendent des paramètres d'initialisation pouvant engendrer ainsi une instabilité. Il est habituel d'effectuer plusieurs initialisations puis de retenir comme carte finale la meilleure au sens d'un indice interne de qualité tel que l'erreur de quantification, topologique ou l'erreur de distorsion.

L'**erreur de quantification** est définie selon

$$\frac{1}{n}\sum_{i=1}^{n} \parallel w_{k(x_i)} - x_i\|_V^2$$

où $k(x_i)$ désigne la classe de x_i et $\|.\|_V$ la norme sur V. Cette erreur est la mesure classique d'homogénéité utilisée en classification correspondant au coût à remplacer les données par le représentant de la classe auxquelles elles sont rattachées. Il s'agit du critère optimisé par la méthode des kmeans.

L'erreur topologique correspond à la proportion d'observations pour lesquelles les 2 référents les plus proches ne correspondent pas à des unités adjacentes sur la carte. Elle mesure l'adéquation entre les distances observées sur la carte vis à vis des distances entre les points dans l'espace.

Enfin, la **mesure de distortion** est donnée par

$$distorsion = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} h(\parallel s_k - s_{k(x_i)} \parallel_A) \parallel w_k - x_i\|_V^2$$

Cette erreur est celle qui est minimisée (au moins localement) par l'algorithme de Kohonen. Elle combine un critère de qualité du clustering (i.e. l'homogénéité des classes, mesurée par l'erreur de quantification) et d'organisation correcte (c'est-à-dire que les distance entre neurones reflètent celles des individus associés, ce qui est mesuré par l'erreur topologique).

5.2 Classification par CAH

L'auto-organisation par les cartes de Kohonen, ne permet pas de résoudre directement un problème de classification, elle permet simplement d'affecter une observation à un sous-ensemble d'une partition, représenté par un référent. Il peut être utile de se servir de cette quantification vectorielle comme point de départ d'un clustering.

Puisque chaque sous-ensemble de données est associé à un référent, le problème de classification se résume à celui de l'étiquetage de chaque prototype à l'une des classes. Il faut donc introduire une seconde étape consistant à étiqueter tous les prototypes. On peut par exemple procéder par classification ascendante hiérarchique.

5.3 Choix de la grille

Pour ce qui est du choix du nombre de noeuds sur la grille, celui-ci varie généralement entre une douzaine et une centaine. Il se détermine par essais-erreurs en regardant le résultat pour un nombre de noeuds fixé et en ajustant au besoin, mais dans une optique de classification, celui-ci n'est pas aussi essentiel que dans la méthode des k-moyennes car la carte ne sera qu'une étape préalable à la CAH.

Par ailleurs, puisque l'algorithme SOM est une version contrainte des k-moyennes, il est important de vérifier que ces contraintes sont raisonnables pour les données considérées. Pour cela, on peut comparer l'erreur de quantification à celle obtenue en procédant par K-moyennes (Hastie, Tibshirani, and Friedman (2009)). On s'attend nécessairement à observer une erreur de quantification plus grande pour l'algorithme SOM. Néanmoins, un écart trop important serait suspect et amènerait à éventuellement à modifier la forme de la grille ou la fonction de voisinage.

5.4 Nombre d'itérations

Le nombre d'itérations de l'algorithme permet d'assurer sa convergence. Il est généralement choisi en regardant l'évolution de la distorsion en fonction du nombre d'itérations effectuées.

5.5 Packages R

Il existe différents packages R permettant de mettre de construire des cartes topologiques. On pourra notamment utiliser *SOMbrero*, *som*, ou *kohonen*. On donne ici quelques commandes utiles pour analyser le jeu de données iris via le package *SOMbreo*.

Pour construire une carte de kohonen, on pourra utiliser

```
# charger le package SOMbrero
library(SOMbrero)

# fixer la graine du générateur aléatoire pour un résultat reproductible
set.seed(255)

# Mise en oeuvre de l'algorithme
iris.som <- trainSOM(x.data = iris[,1:4],
                     nb.save = 10 # utilisé pour vérifier ultérieurement le nombre d'itératio
ns
                     )

#calcul des effectifs des classes
table(iris.som$clustering)
```

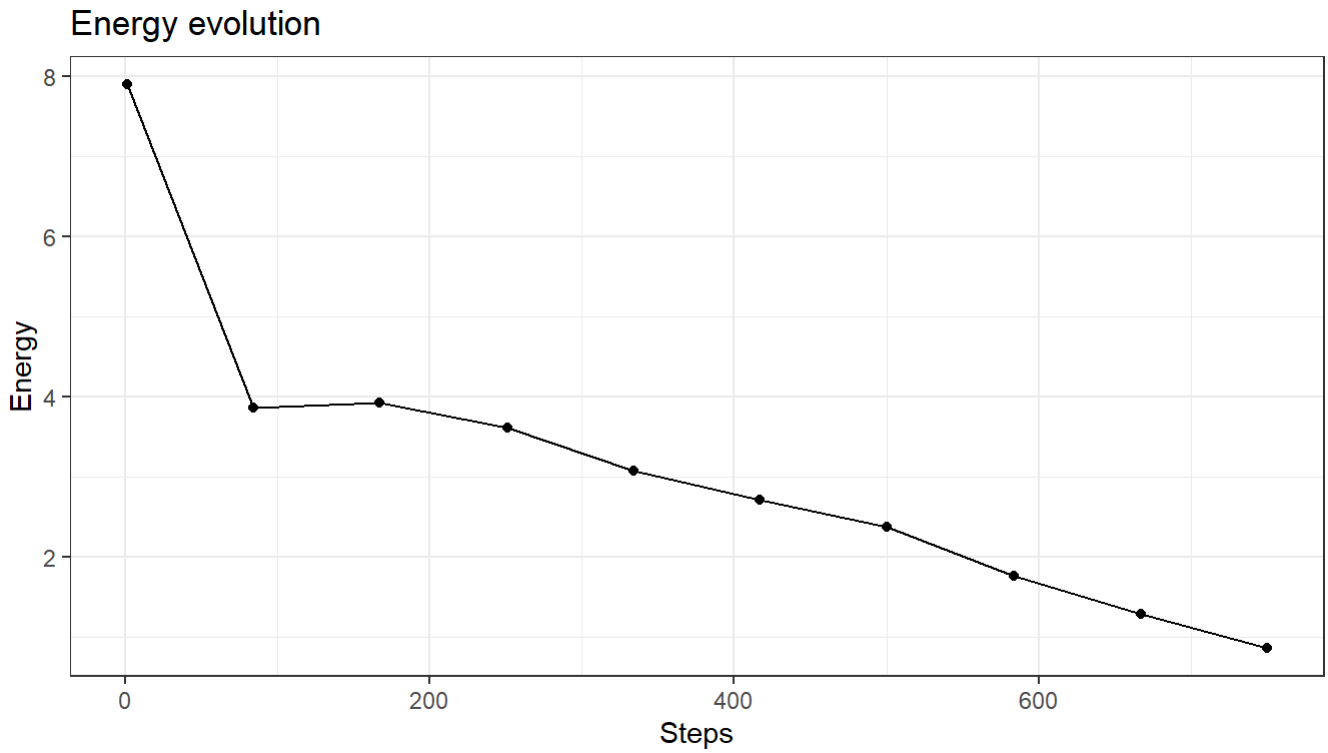
Les arguments par défaut se retrouvent ici :

```
iris.som$parameters
```

```
##
##   Parameters of the SOM
##
##     SOM mode                       :  online
##     SOM type                       :  numeric
##     Affectation type               :  standard
##     Grid                           :
##       Self-Organizing Map structure
##
##         Features   :
##            topology     :  square
##            x dimension  :  5
##            y dimension  :  5
##            distance type:  euclidean
##
##     Number of iterations           :  750
##     Number of intermediate backups :  10
##     Initializing prototypes method :  random
##     Data pre-processing type       :  unitvar
##     Neighbourhood type             :  gaussian
```

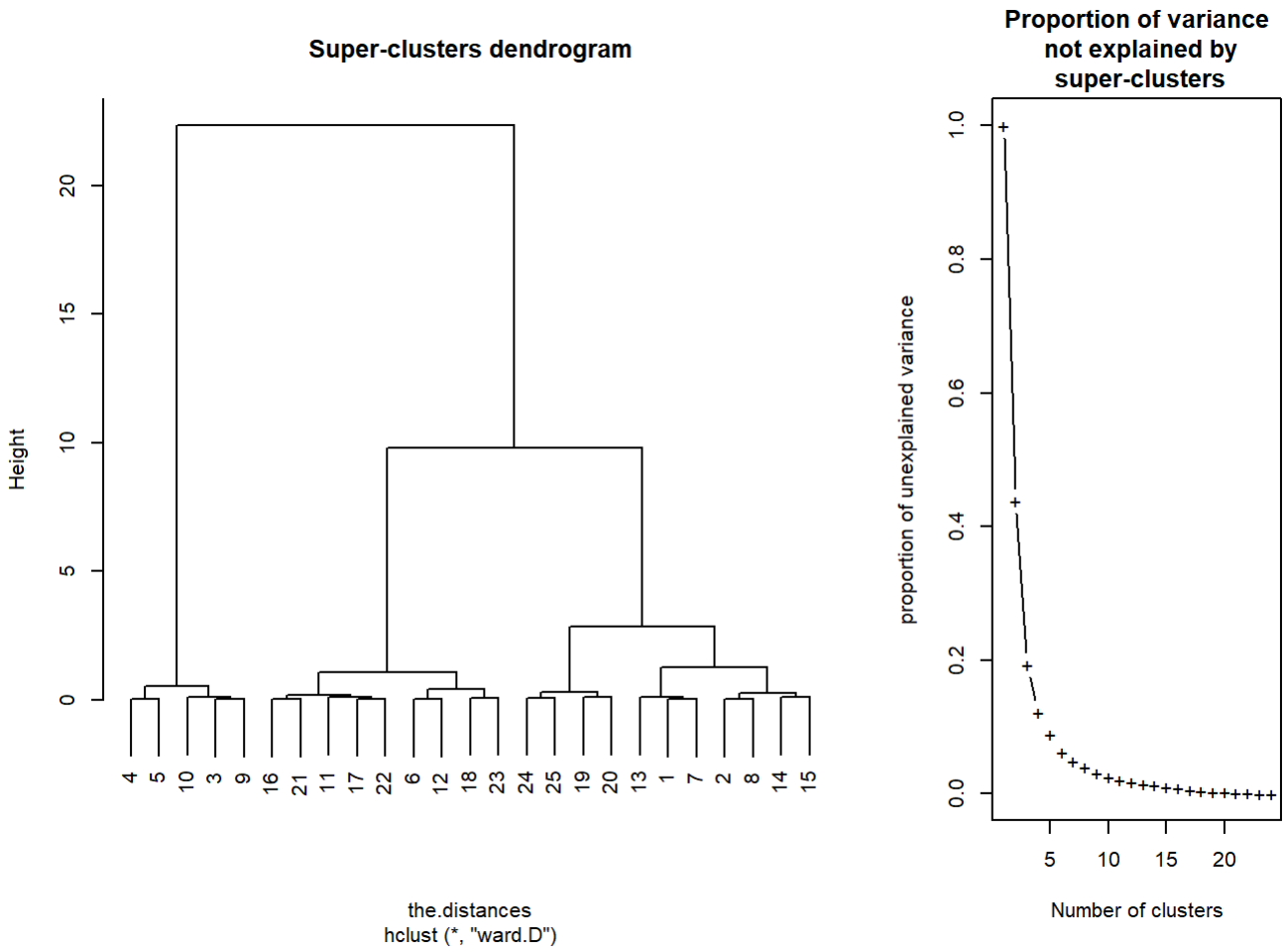
Pour vérifier que le nombre d'itérations est suffisant, on représente la décroissance de la distorsion en fonction du nombre d'itérations.

```
plot(iris.som, what = "energy")
```



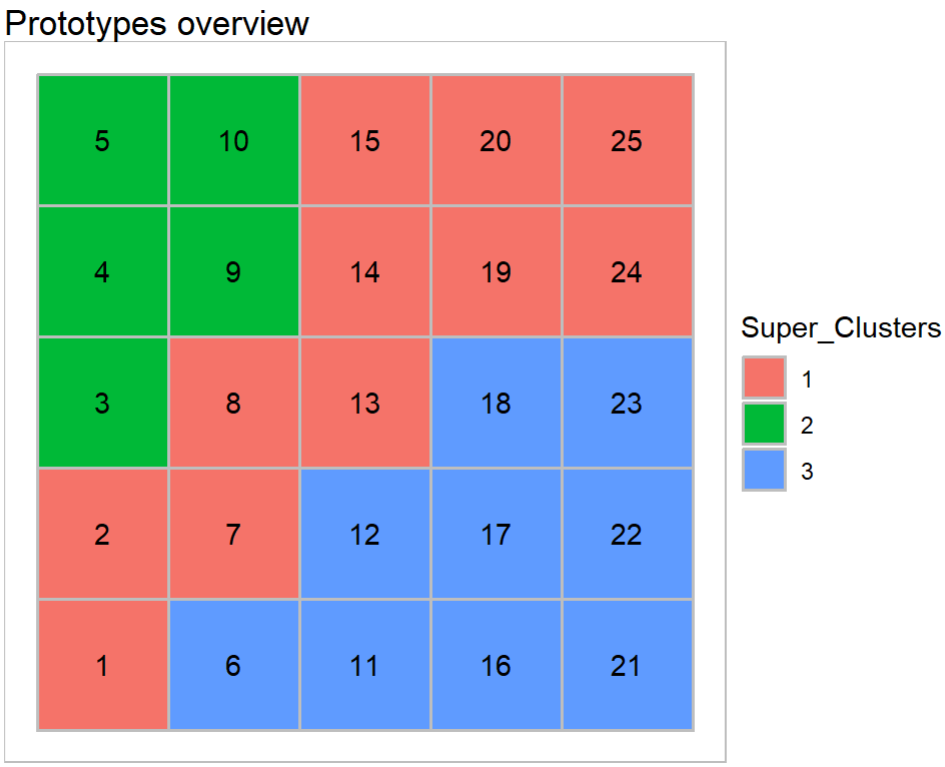
Pour construire la classification ascendante hiérarchique selon la méthode de Ward à partir de la carte, on utilisera

```
plot(superClass(iris.som))
```



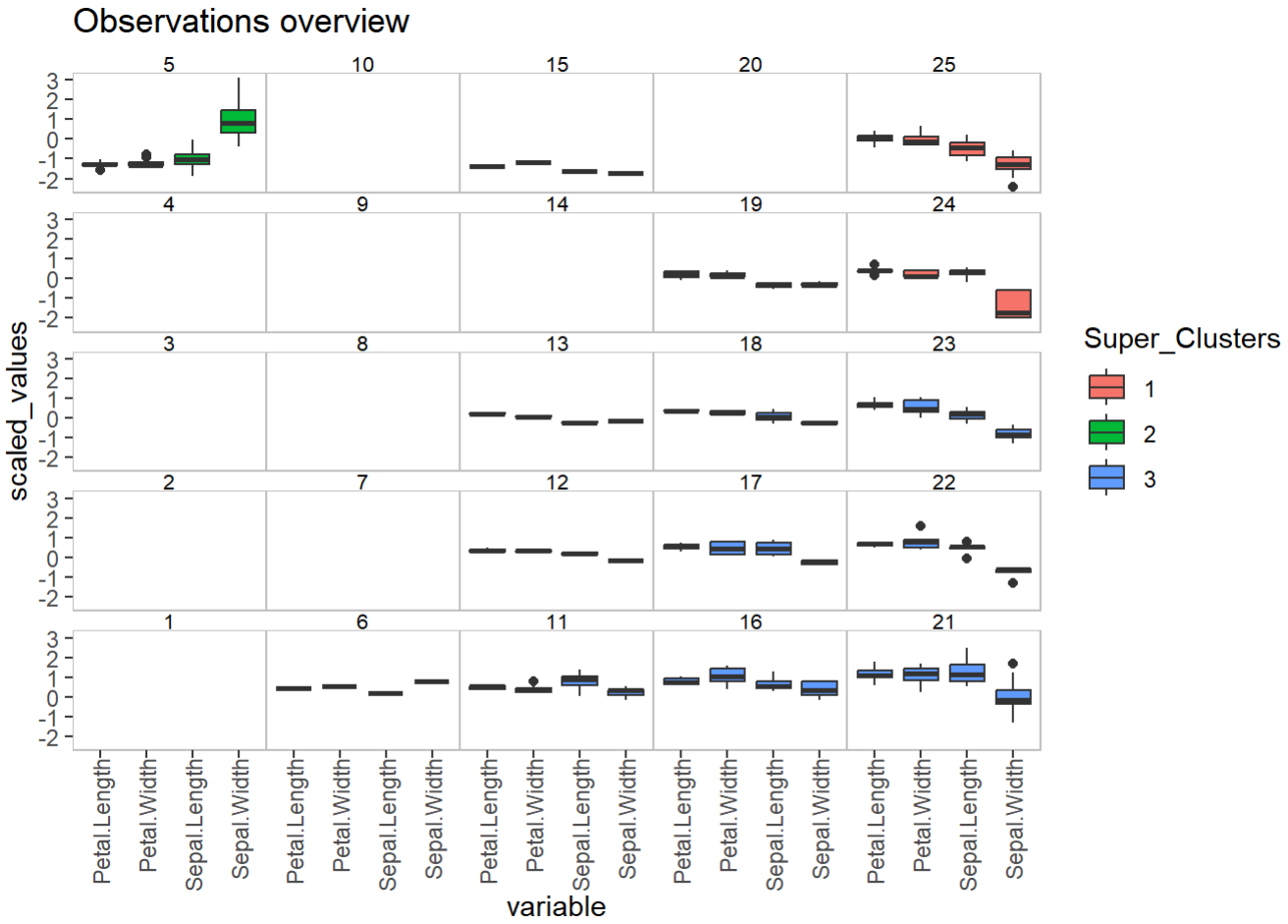
En coupant l'arbre à 3 classes (ce qui correspond par ailleurs au nombre d'espèces d'iris du jeu de données), on obtient la classification suivante

```
my.sc <- superClass(iris.som, k = 3)
plot(my.sc, type = "grid", plot.legend = TRUE)
```



La description des classes peut notamment s'appuyer sur le regroupement des boîtes à moustaches par classe

```
plot(my.sc, type="boxplot", key.loc=c(-0.5,5), mar=c(0,10,2,0))
```



mais plus efficacement en utilisant les valeurs-tests.

```
library(FactoMineR)
don <- cbind.data.frame(iris[,1:4],
                        class = as.factor(my.sc$cluster))
catdes(donnee = don,
       num.var = ncol(don),
       proba = 1)
```

```
##
## Link between the cluster variable and the quantitative variables
## ================================================================
##                       Eta2   P-value
## Sepal.Width  0.0067370200 0.6084464
## Sepal.Length 0.0028842610 0.8087225
## Petal.Width  0.0015796652 0.8902996
## Petal.Length 0.0009454707 0.9328368
##
## Description of each cluster by quantitative variables
## =====================================================
## $`1`
##                   v.test Mean in category Overall mean sd in category Overall sd
## Petal.Width  -0.2494603         1.181818     1.199333      0.7515457  0.7596926
## Petal.Length -0.3380518         3.703030     3.758000      1.7148169  1.7594041
## Sepal.Width  -0.4845580         3.037879     3.057333      0.4528580  0.4344110
## Sepal.Length -0.6078416         5.796970     5.843333      0.7994259  0.8253013
##                 p.value
## Petal.Width  0.8030047
## Petal.Length 0.7353242
## Sepal.Width  0.6279899
## Sepal.Length 0.5432925
##
## $`2`
##                    v.test Mean in category Overall mean sd in category
## Petal.Length  0.29601673         3.843333     3.758000      1.7915884
## Sepal.Length  0.04930141         5.850000     5.843333      0.8289552
## Petal.Width  -0.26244001         1.166667     1.199333      0.7466964
## Sepal.Width  -0.57134829         3.016667     3.057333      0.3688571
##               Overall sd   p.value
## Petal.Length  1.7594041 0.7672173
## Sepal.Length  0.8253013 0.9606791
## Petal.Width   0.7596926 0.7929822
## Sepal.Width   0.4344110 0.5677636
##
## $`3`
##                   v.test Mean in category Overall mean sd in category Overall sd
## Sepal.Width  0.9772241         3.103704     3.057333      0.4409430  0.4344110
## Sepal.Length 0.5875085         5.896296     5.843333      0.8506997  0.8253013
## Petal.Width  0.4766768         1.238889     1.199333      0.7749353  0.7596926
## Petal.Length 0.1029121         3.777778     3.758000      1.7925429  1.7594041
##                 p.value
## Sepal.Width  0.3284582
## Sepal.Length 0.5568623
## Petal.Width  0.6335923
## Petal.Length 0.9180328
```

Les outils graphiques pour analyser les cartes sont riches, nous en avons fourni que les principaux. Pour aller plus loin on pourra par exemple consulter les vignettes (sombrero.clementine.wf/articles/c-doc-numericSOM.html) du package *SOMbrero* ou le tutoriel (https://eric.univ-lyon2.fr/ricco/tanagra/fichiers/fr_Tanagra_Kohonen_SOM_R.pdf) de R. Rakotomalala utilisant le package *kohonen*.

6 Conclusion

Les cartes topologiques peuvent être vues comme une version contrainte des K-moyennes. Il est aussi possible de les présenter comme des réseaux de neurones particulier où chaque noeud de la grille correspond à un neurone. Les cartes ont la propriété de conservation topologique, de sorte que la proximité des neurones sur la carte traduit une proximité dans l'espace initial des données. On montre qu'elles sont plus adaptées à la découverte de classes à structure non-sphérique et à la présence d'observations aberrantes. Elles offrent de plus, des possibilités de visualisation directe (sans passer par un plan factoriel d'ACP par exemple) utiles pour l'interprétation des résultats.

Les cartes peuvent être considérées comme une méthode de classification automatique, mais elles permettent surtout d'effectuer un prétraitement avant d'effectuer une CAH. En effet, contrairement à la CAH, la complexité de l'algorithme SOM est de l'ordre du nombre de données ce qui permet de l'utiliser sur des données où le nombre d'individus ou de variables est grand.

Références

Chen, Ning, and Nuno C. Marques. 2005. "An Extension of Self-Organizing Maps to Categorical Data." In *Progress in Artificial Intelligence*, edited by Carlos Bento, Amílcar Cardoso, and Gaël Dias, 304–13. Berlin, Heidelberg: Springer Berlin Heidelberg.

Cottrell, Marie, Smail Ibbou, and Patrick Letrémy. 2004. "SOM-Based Algorithms for Qualitative Variables." *Neural Networks* 17 (8-9): 1149–67.

Dreyfus, Gérard, J. M. Martinez, M. Samuelides, M. B. Gordon, Fouad Badran, Sylvie Thiria, and L. Herault. 2002. *Réseaux de neurones - Méthodologie et applications*. Edited by Eyrolles. https://hal.science/hal-01125016 (https://hal.science/hal-01125016).

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning*. 2nd ed. Springer Series in Statistics. New York, NY, USA: Springer New York Inc. https://hastie.su.domains/ElemStatLearn/printings/ESLII_print12_toc.pdf.download.html (https://hastie.su.domains/ElemStatLearn/printings/ESLII_print12_toc.pdf.download.html).

Lebbah, Mustapha, Fouad Badran, and Sylvie Thiria. 2000. "Topological Map for Binary Data." In *ESANN*, 267–72.

Lebbah, Mustapha, Aymeric Chazottes, Fouad Badran, and Sylvie Thiria. 2005. "Mixed Topological Map." In *ESANN*, 17:47.