SOMbrero : Cartes auto-organisatrices stochastiques pour l'intégration de données numériques et non numériques

Laura Bendhaïba[1], Madalina Olteanu[1] & Nathalie Villa-Vialaneix[1,2]

1 2

Rencontres R, 28 juin 2013

Plan

Introduction

Cartes auto-organisatrices (aussi appelées “cartes de Kohonen”ou SOM)

- Pour données vectorielles, [Kohonen, 2001]
- Pour données qualitatives, séries temporelles et, de manière plus générale, données non euclidiennes décrites par des tableaux de dissimilarités, [Kohohen and Somervuo, 1998], [Cottrell et al., 2012], [Olteanu and Villa-Vialaneix, 2013]

- Une version SAS existe déjà pour données vectorielles et données qualitatives, mais n’est plus maintenue depuis deux ans :

`http://samos.univ-paris1.fr/Programmes-bases-sur-l-algorithme`

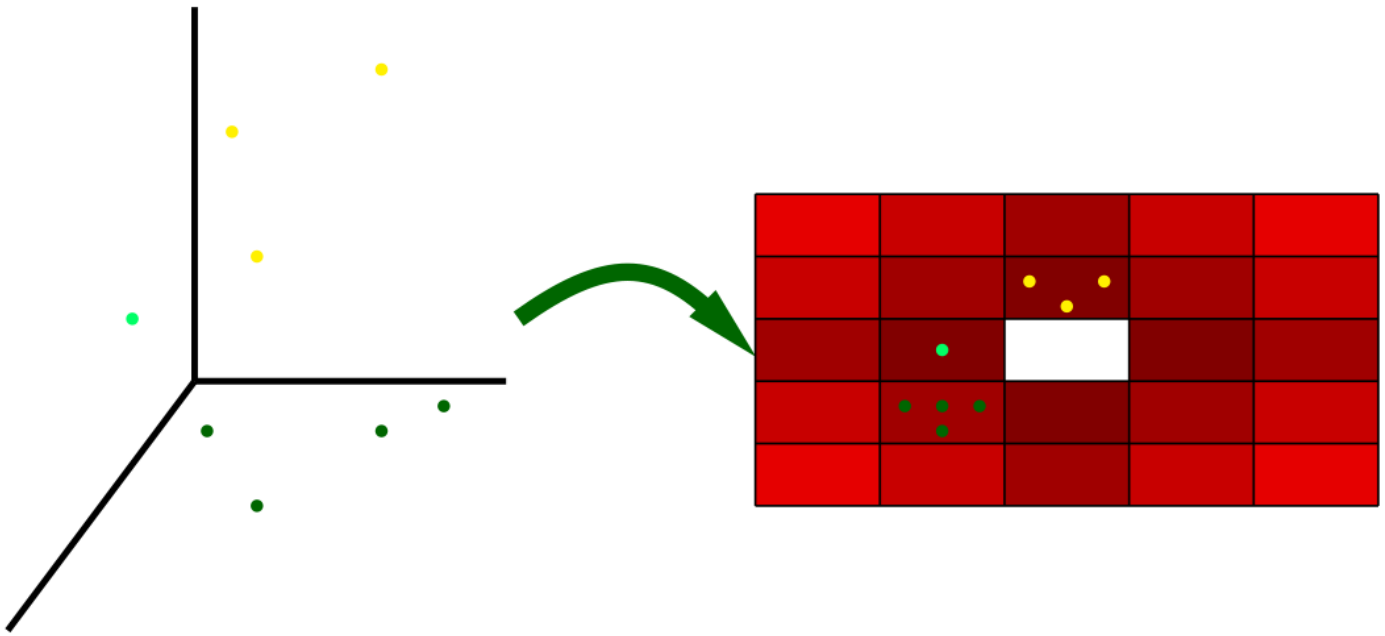
Contexte

Cartes auto-organisatrices : définition

- Apprentissage > Méthodes neuronales > Cartes auto-organisatrices
- Algorithme différent des méthodes de classification ordinaires : respect de la topologie des données d'origine
- Projection nonlinéaire et clustering (réduction de dimension)
- Généralisation des centres mobiles (version stochastique)
- Vocabulaire :
  - Neurones
  - Prototypes (vecteurs-code)

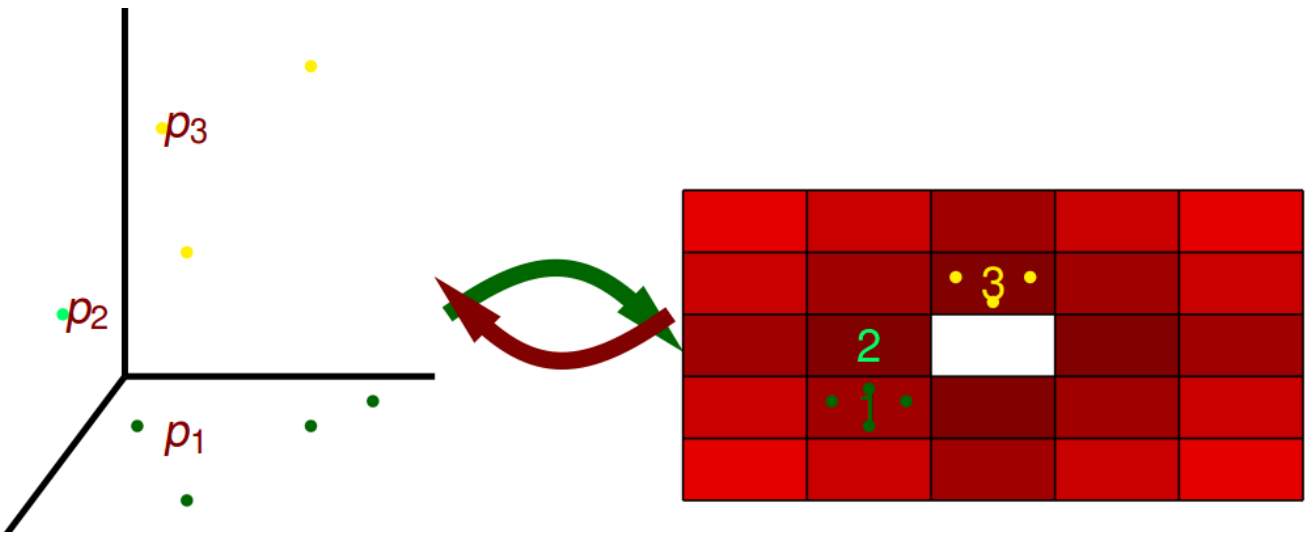
Principe

Cartes auto-organisatrices : définition



Principe

Cartes auto-organisatrices : définition



Notations

- La carte est constituée de U neurones (symbolisés par les nœuds de la grille). A chaque neurone u est associé un prototype p_u (un objet dans le même espace que les données initiales; par exemple, un vecteur numérique).

- La structure de la carte induit une distance naturelle entre les neurones sur la grille : la distance $d(u, u')$ entre les neurones u et u' est définie comme la longueur du plus court chemin entre eux.

- La grille possède également une fonction de voisinage $h^t(d(u, u'))$ telle que $h^t : \mathbf{R}^+ \to \mathbf{R}^+$, $h^t(0) = 1$ et $\lim_{x \to +\infty} h^t(x) = 0$.

- L'algorithme cherche à partitionner les données de manière à minimiser l'énergie (variance intra étendue aux voisins)

$$\mathcal{E} = \sum_{i=1}^{U} \sum_{j=1}^{U} h^t(d(i,j)) \sum_{x \in \Gamma_j} \|x - p_i\|^2 ,$$

où $\Gamma_1, ..., \Gamma_U$ est la partition de Voronoï associée aux prototypes $p_1, ..., p_U$.

Notations

- La carte est constituée de U neurones (symbolisés par les nœuds de la grille). A chaque neurone u est associé un prototype p_u (un objet dans le même espace que les données initiales; par exemple, un vecteur numérique).

- La structure de la carte induit une distance naturelle entre les neurones sur la grille : la distance $d(u, u')$ entre les neurones u et u' est définie comme la longueur du plus court chemin entre eux.

- La grille possède également une fonction de voisinage $h^t(d(u, u'))$ telle que $h^t : \mathbf{R}^+ \to \mathbf{R}^+$, $h^t(0) = 1$ et $\lim_{x \to +\infty} h^t(x) = 0$.

- L'algorithme cherche à partitionner les données de manière à minimiser l'énergie (variance intra étendue aux voisins)

$$\mathcal{E} = \sum_{i=1}^{U} \sum_{j=1}^{U} h^t(d(i,j)) \sum_{x \in \Gamma_j} \|x - p_i\|^2 ,$$

où $\Gamma_1, ..., \Gamma_U$ est la partition de Voronoï associée aux prototypes $p_1, ..., p_U$.

Notations

- La carte est constituée de U neurones (symbolisés par les nœuds de la grille). A chaque neurone u est associé un prototype p_u (un objet dans le même espace que les données initiales; par exemple, un vecteur numérique).
- La structure de la carte induit une distance naturelle entre les neurones sur la grille : la distance $d(u, u')$ entre les neurones u et u' est définie comme la longueur du plus court chemin entre eux.
- La grille possède également une fonction de voisinage $h^t(d(u, u'))$ telle que $h^t : \mathbf{R}^+ \to \mathbf{R}^+$, $h^t(0) = 1$ et $\lim_{x \to +\infty} h^t(x) = 0$.
- L'algorithme cherche à partitionner les données de manière à minimiser l'énergie (variance intra étendue aux voisins)

$$\mathcal{E} = \sum_{i=1}^{U} \sum_{j=1}^{U} h^t(d(i,j)) \sum_{x \in \Gamma_j} \|x - p_i\|^2 ,$$

où $\Gamma_1, ..., \Gamma_U$ est la partition de Voronoï associée aux prototypes $p_1, ..., p_U$.

Notations

- La carte est constituée de U neurones (symbolisés par les nœuds de la grille). A chaque neurone u est associé un prototype p_u (un objet dans le même espace que les données initiales; par exemple, un vecteur numérique).

- La structure de la carte induit une distance naturelle entre les neurones sur la grille : la distance $d(u, u')$ entre les neurones u et u' est définie comme la longueur du plus court chemin entre eux.

- La grille possède également une fonction de voisinage $h^t(d(u, u'))$ telle que $h^t : \mathbf{R}^+ \to \mathbf{R}^+$, $h^t(0) = 1$ et $\lim_{x \to +\infty} h^t(x) = 0$.

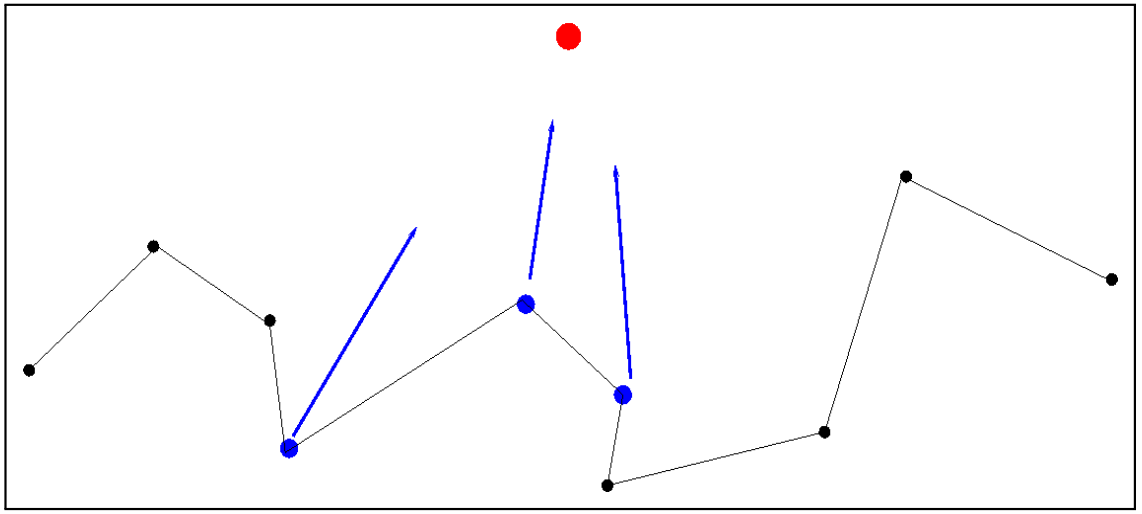
- L'algorithme cherche à partitionner les données de manière à minimiser l'énergie (variance intra étendue aux voisins)

$$\mathcal{E} = \sum_{i=1}^{U} \sum_{j=1}^{U} h^t(d(i,j)) \sum_{x \in \Gamma_j} \|x - p_i\|^2 ,$$

où $\Gamma_1, ..., \Gamma_U$ est la partition de Voronoï associée aux prototypes $p_1, ..., p_U$.

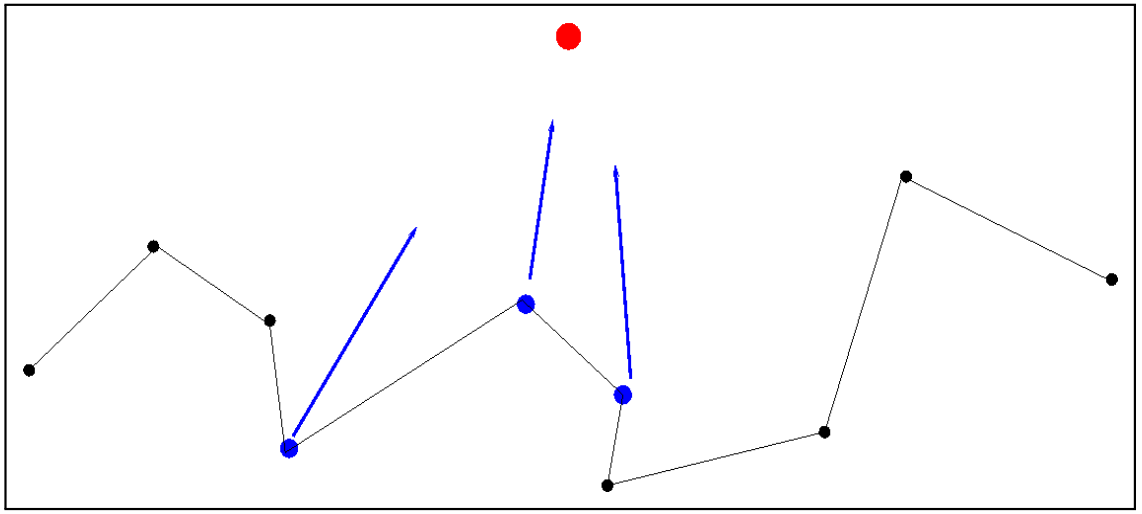
SOM - Comment ça marche?

- Avant

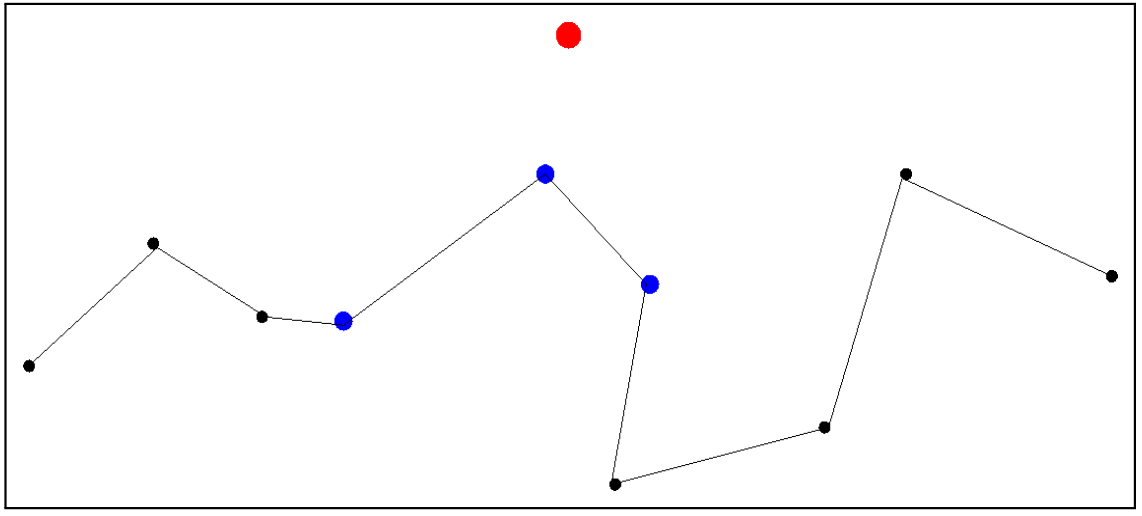


SOM - Comment ça marche?

- Avant



- Après



SOM - l'algorithme de base

Données: $x_1, \ldots, x_n \in \mathbb{R}^d$

1: Initialisation: tirage aléatoire de p_1^0,...,p_U^0 dans $\mathbb{R}^d$
2: **for** $t = 1 \rightarrow T$ **do**
3: Choisir de manière aléatoire $i \in \{1, \ldots, n\}$
4: Affectation

$$f^t(x_i) \leftarrow \arg \min_{u=1,\ldots,U} \|x_i - p_u^{t-1}\|_{\mathbb{R}^d}$$

5: **for all** $u = 1 \rightarrow U$ **do** Mise à jour des prototypes
6: $p_u^t \leftarrow p_u^{t-1} + \mu_t h^t(d(f^t(x_i), u)) \left(x_i - p_u^{t-1}\right)$
7: **end for**
8: **end for**

où h^t décroit la taille du voisinage en fonction de t et μ_t est généralement égal à $1/t$.

Version actuelle

Description

- Version **ONLINE** 0.3 :
  - orientation objet de type “S3”
  - 17 fonctions pour l’utilisateur
  - 3 documentations utilisateur “user-friendly” avec des exemples sur des jeux de données réels
- Implémentation complète du cas *numérique*, *on-line* avec fonctionnalités graphiques, super-classification et critères de qualité
- Implémentation complète du cas *korresp*, *on-line* pour traiter les tables de contingence
- Le package est actuellement disponible sur

`http://tuxette.nathalievilla.org/?p=1099&lang=en`

Données vectorielles

Exemple : données `iris`

4 variables numériques : `Sepal.Length`, `Sepal.Width`, `Petal.Length` et `Petal.Width`

Commande :

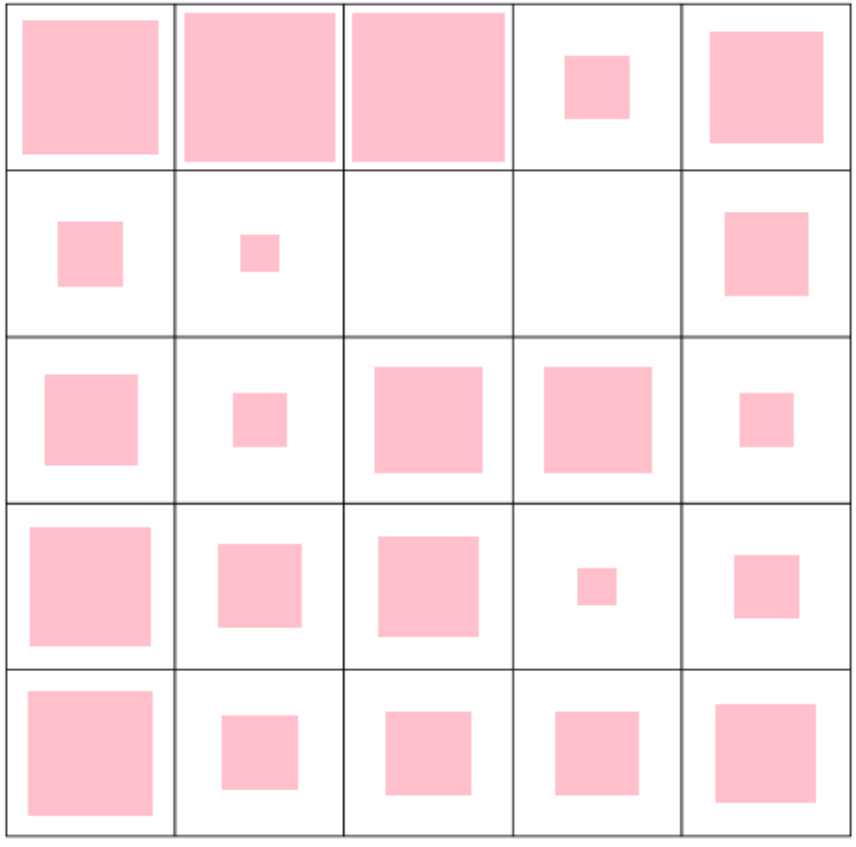
```
iris.som <- trainSOM(x.data=iris[,1:4], nb.save=10)
```

Données vectorielles

Exemple : données `iris`

Hitmap :

```
plot(iris.som, what="obs", type="hitmap")
```

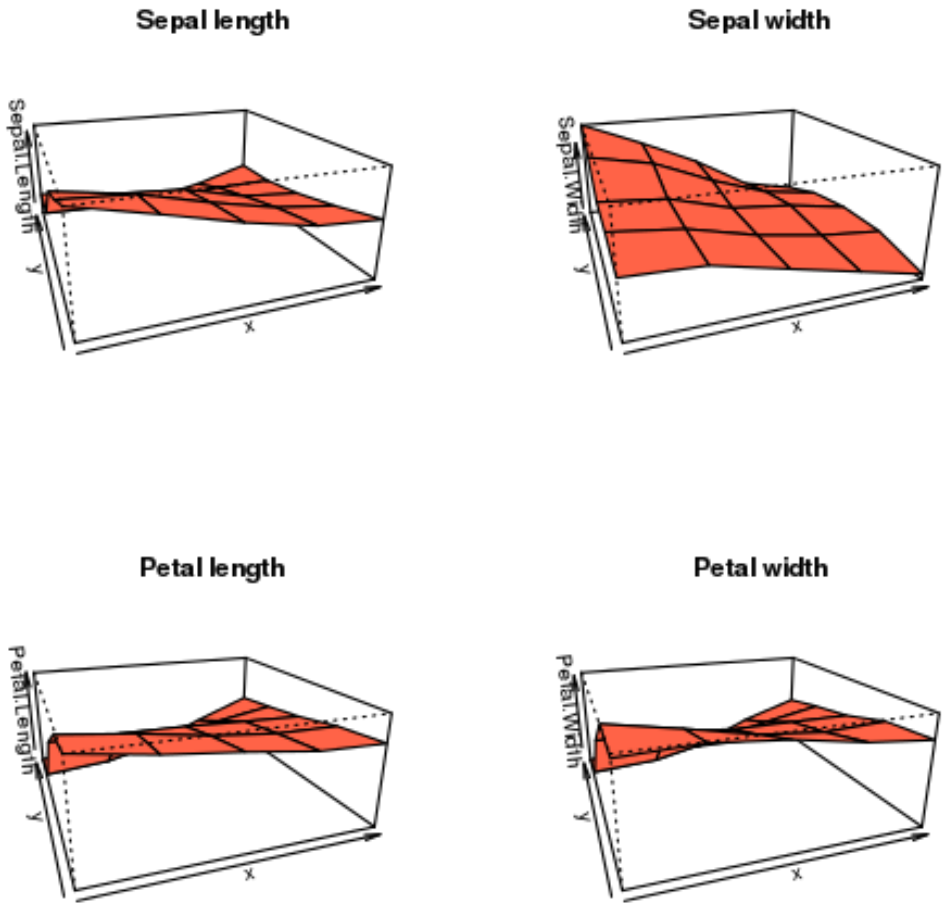


Données vectorielles

Exemple : données `iris`

Niveau de valeur des prototypes pour une variable donnée :

```
plot(iris.som, what="prototypes", type="3d", variable="Sepal.Width", main="Sepal Width")
```

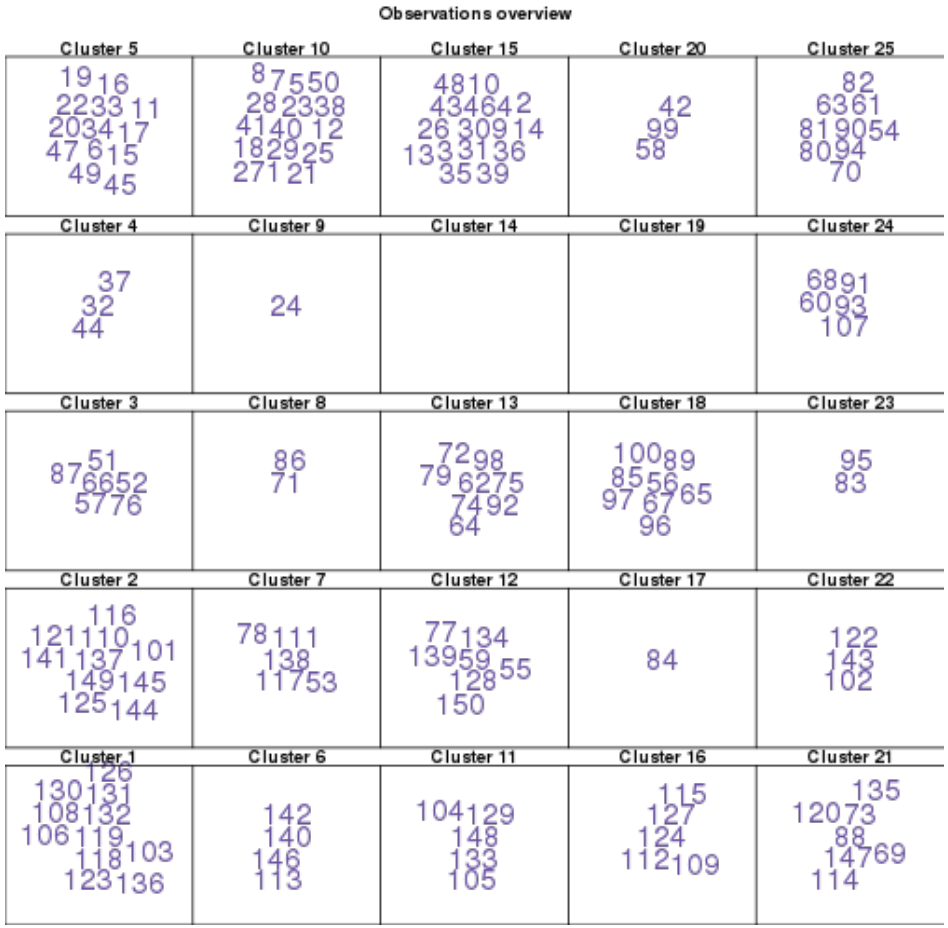


Données vectorielles

Exemple : données `iris`

Répartition des observations sur la carte :

```
plot(iris.som, what="obs", type="names", print.title=TRUE)
```

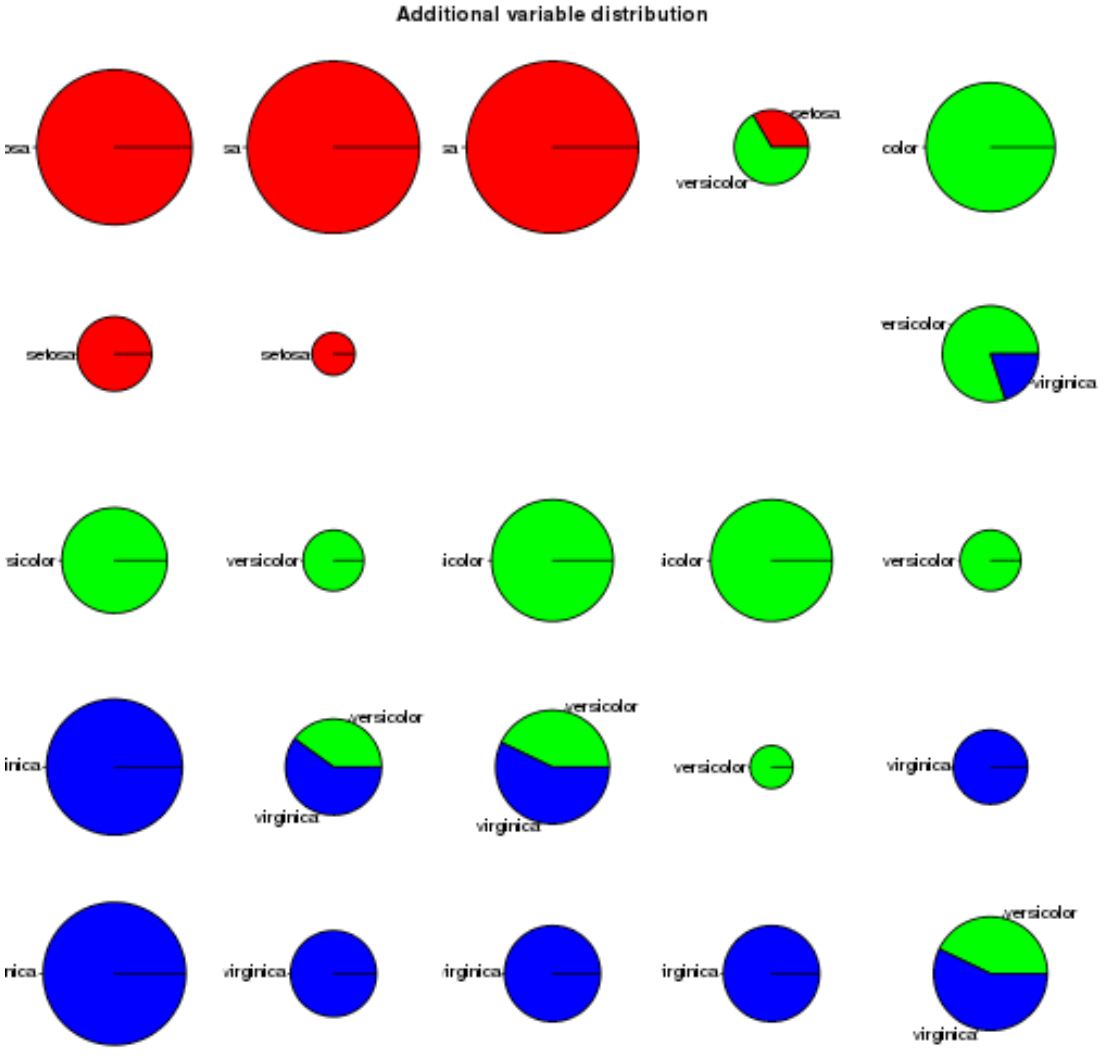


Données vectorielles

Exemple : données `iris`

Graphiques circulaires sur la variable espèce :

```
plot(iris.som, what="add", type="pie", variable=iris$Species)
```

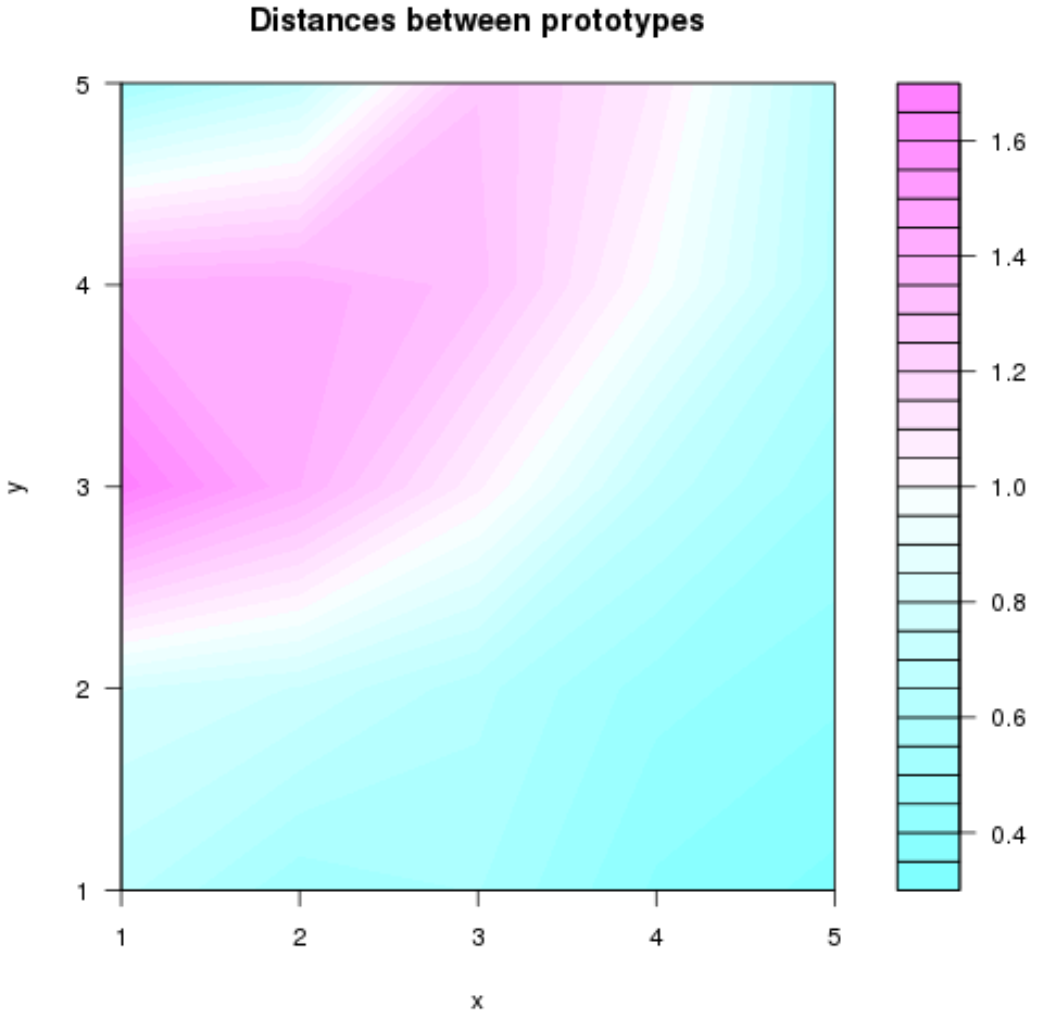


Données vectorielles

Exemple : données `iris`

Distances entre les prototypes :

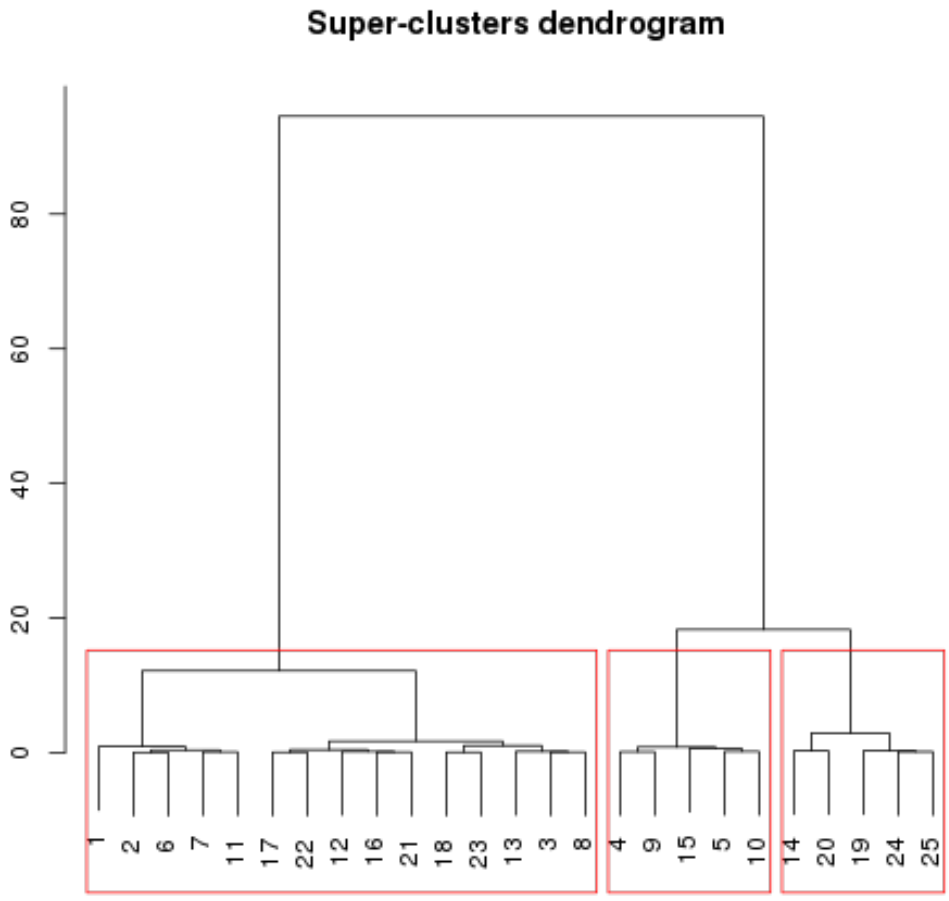
```
plot(iris.som, what="prototypes", type="smooth.dist")
```



Données vectorielles

Exemple : données `iris`

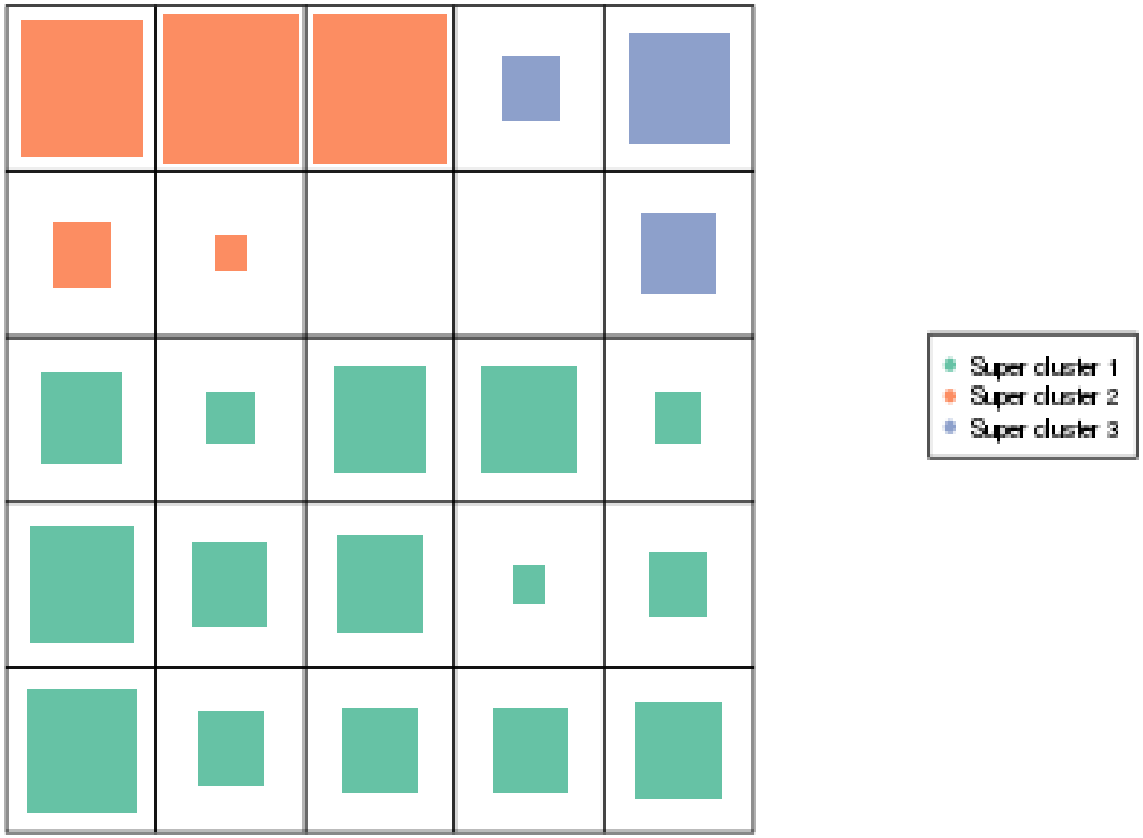
```
sc <- superClass(my.som, k=3)
plot(sc)
```



Données vectorielles

Exemple : données `iris`

```
plot(sc, type="hitmap", plot.legend=TRUE)
```



Tableaux de contingence

Algorithme Korresp

- Données initiales : tableau de contingence $T \in \mathbb{N}^{pxq}$
- Données transformées : $T^{corr} \in \mathbb{R}^{(p+q)x(q+p)}$
  - à chaque profil-ligne $r(i)$ est associé le profil-colonne le plus probable sachant i : $c(j(i))$, $1 \leq i \leq p$;
  - à chaque profil-colonne $c(j)$ est associé le profil-ligne le plus probable sachant j : $r(i(j))$, $1 \leq j \leq q$
- L'algorithme SOM de base est appliqué aux données T^{corr} avec quelques changements, [Cottrell et al., 1999] :
  - la distance de χ^2 au lieu de la distance euclidienne
  - l'observation courante est tirée alternativement parmi les profils-ligne ou les profils-colonne
  - l'unité gagnante est calculée uniquement sur les q premières ou les p dernières composantes

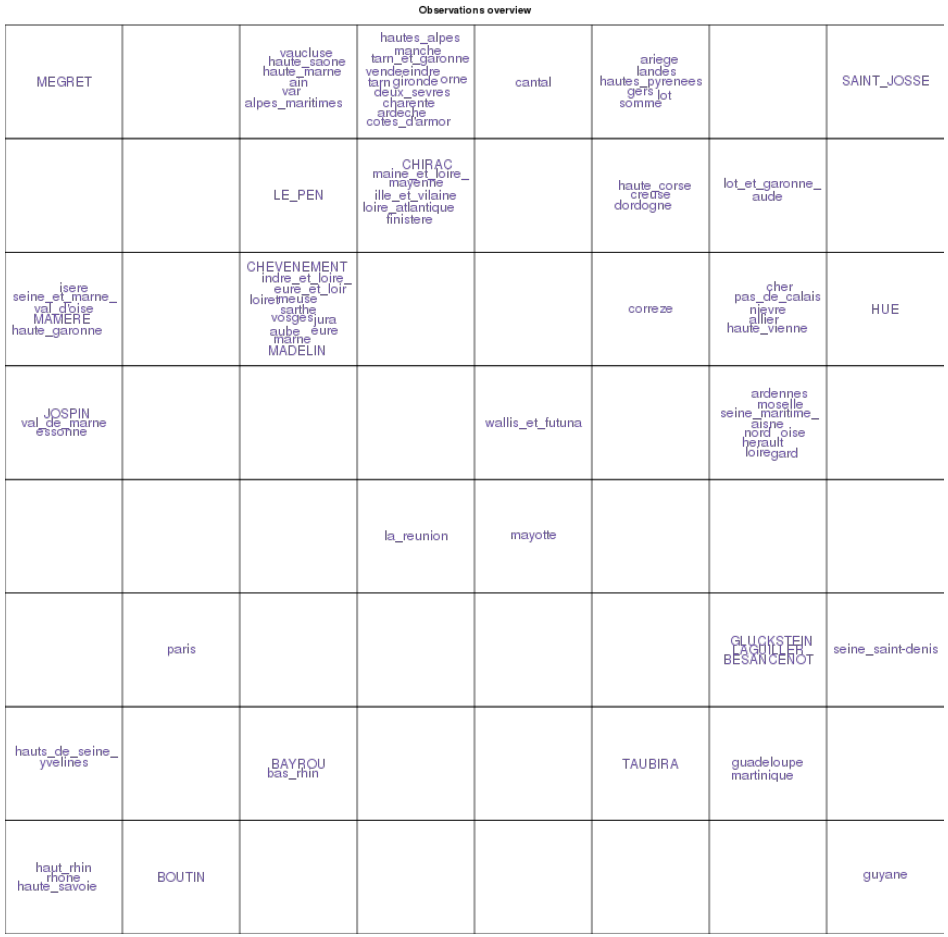
Tableaux de contingence

Exemple : présidentielles 2002

```
presi02 <- trainSOM(presidentielles2002, type="korresp", scaling="chi2")
```

Répartition sur la carte :

```
plot(presi02, what="obs", type="names")
```



Observations overview

MEGRET		vaucluse haute_saone haute_marne ain var alpes_maritimes	hautes_alpes manche tarn_et_garonne vendee indre tarn gironde orne deux_sevres charente ardeche cotes_d'armor	cantal	ariege landes hautes_pyrenees gers lot somme		SAINT_JOSSE
		LE_PEN	CHIRAC maine_et_loire mayenne ille_et_vilaine loire_atlantique finistere		haute_corse creuse dordogne	lot_et_garonne aude	
isere seine_et_marne val_d'oise MAMERE haute_garonne		CHEVENEMENT indre_et_loire eure_et_loir loiret meuse sarthe vosges jura aube eure marne MADELIN			correze	cher pas_de_calais nievre allier haute_vienne	HUE
JOSPIN val_de_marne essonne				wallis_et_futuna		ardennes moselle seine_maritime aisne nord oise herault loiregard	
			la_reunion	mayotte			
	paris					GLUCKSTEIN LAGUILLER BESANCENOT	seine_saint-denis
hauts_de_seine yvelines		BAYROU bas_rhin			TAUBIRA	guadeloupe martinique	
haut_rhin rhone haute_savoie	BOUTIN						guyane

Conclusion et perspectives

Implémentation actuelle :

- Données numériques
- Tableaux de contingence

À venir :

- Données décrites par une matrice de dissimilarités
- Données décrites par des multi-noyaux (apprentissage adaptatif des poids)

Conclusion et perspectives

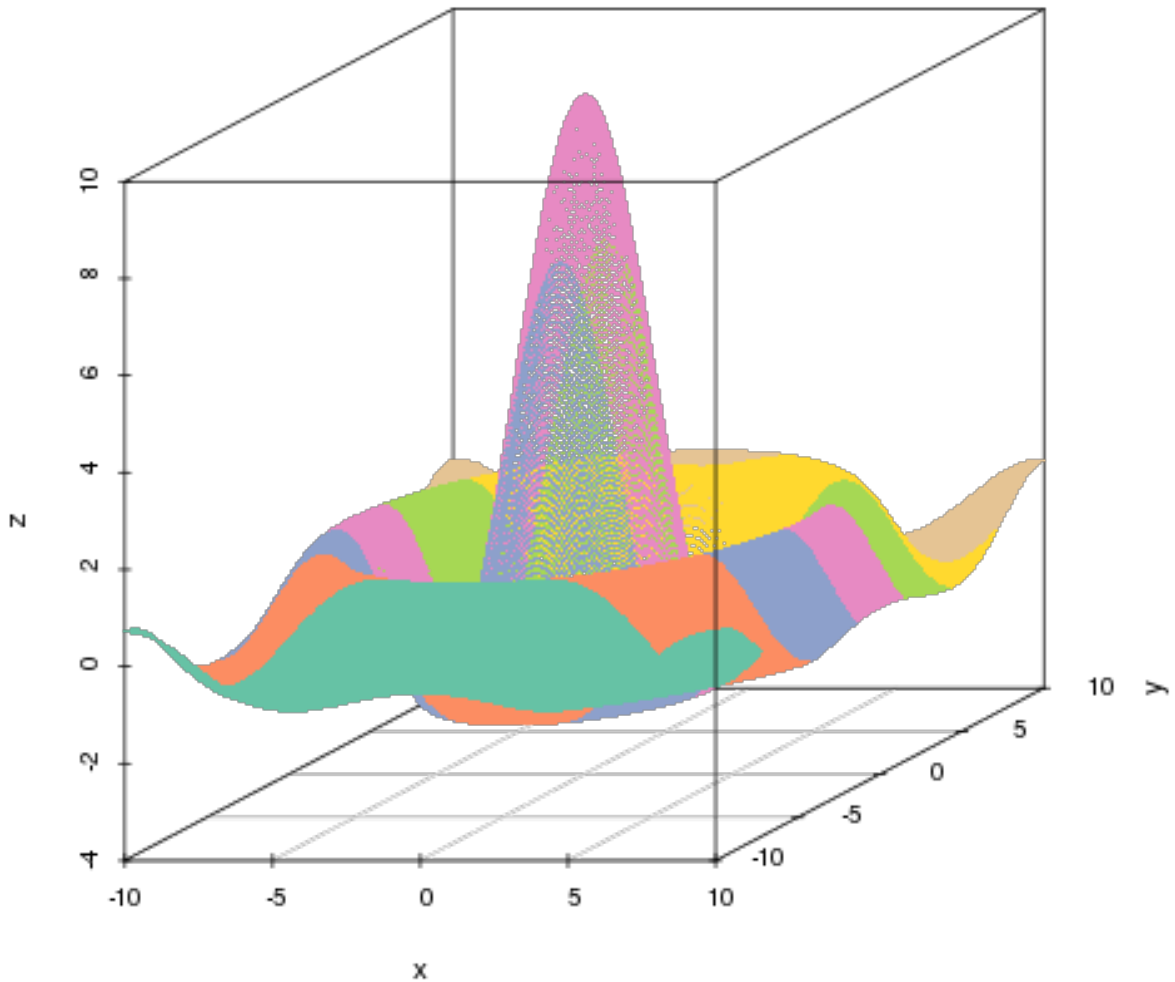
Implémentation actuelle :

- Données numériques
- Tableaux de contingence

À venir :

- Données décrites par une matrice de dissimilarités
- Données décrites par des multi-noyaux (apprentissage adaptatif des poids)

Merci de votre attention



References

Cottrell, M., Gaubert, P., Letremy, P., Rousset, P., and de Paris I: Panthéon-Sorbonne. Maison des sciences économiques, U. (1999).

Analyzing and Representing Multidimensional Quantitative and Qualitative Data: Demographic Study of the Rhone Valley. The Domestic Consumption of the Canadian Families.
Cahiers de la MSE. Universite de Paris I.

Cottrell, M., Olteanu, M., Rossi, F., Rynkiewicz, J., and Villa-Vialaneix, N. (2012).

Neural networks for complex data.
Künstliche Intelligenz, 26(2):1–8.

Kohohen, T. and Somervuo, P. (1998).

Self-organizing maps of symbol strings.
Neurocomputing, 21:19–30.

Kohonen, T. (2001).

Self-Organizing Maps, 3rd Edition, volume 30.
Springer, Berlin, Heidelberg, New York.

Olteanu, M. and Villa-Vialaneix, N. (2013).

Online dissimilarity som.
Submitted for publication.