Classification et visualisation de graphes avec SOMbrero

Madalina Olteanu[1], Nathalie Villa-Vialaneix[2]
madalina.olteanu@univ-paris1.fr,nathalie.villa@toulouse.inra.fr

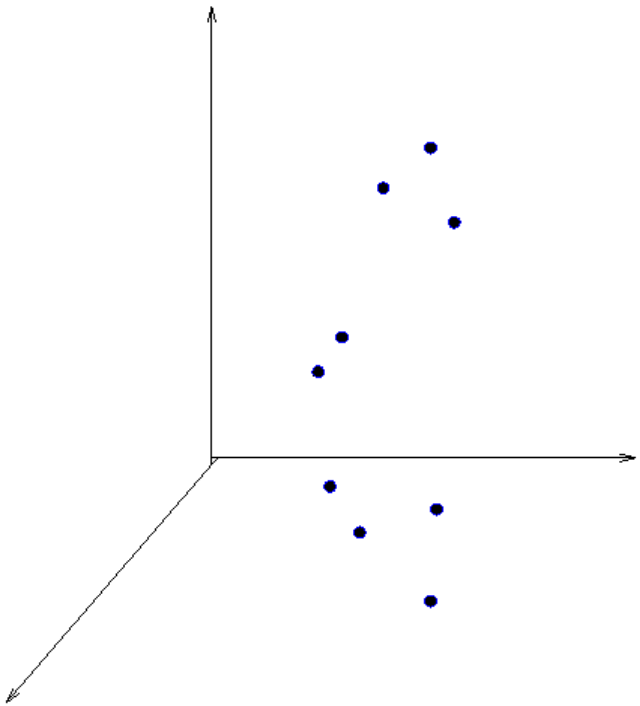
[1] SAMM (Statistique, Analyse et Modélisation Multidisciplinaire)
Université Paris 1 Panthéon-Sorbonne

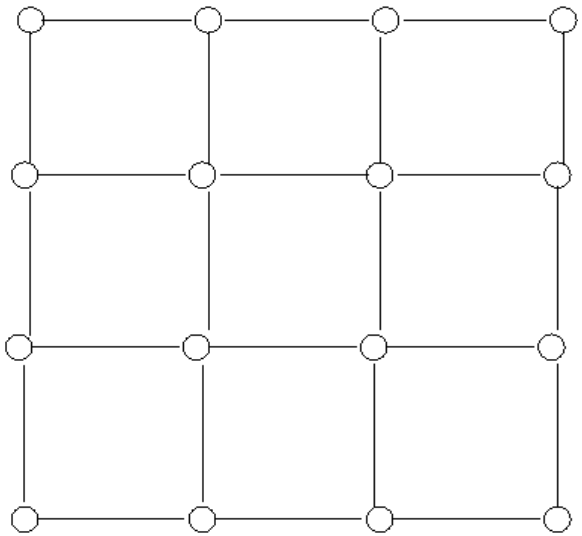
[2] INRA, UR875 MIA-T, Toulouse

Rencontres R, 24-26 juin 2015, Grenoble

Résumé

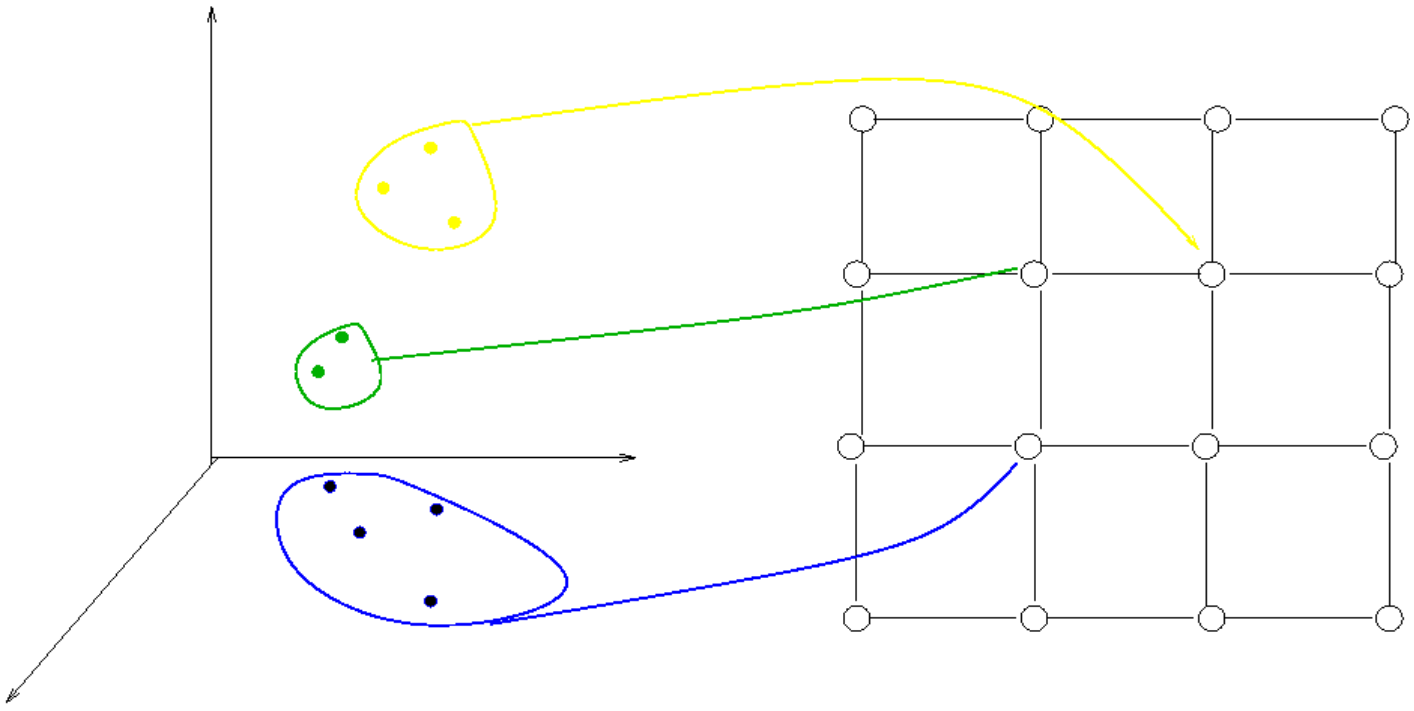
Principe, [Kohonen, 1995]





Projeter les données sur une grille rectangulaire (chaque cellule de la grille représente un cluster)

Principe



Projeter les données sur une grille rectangulaire (chaque cellule de la grille est un cluster) tel que :

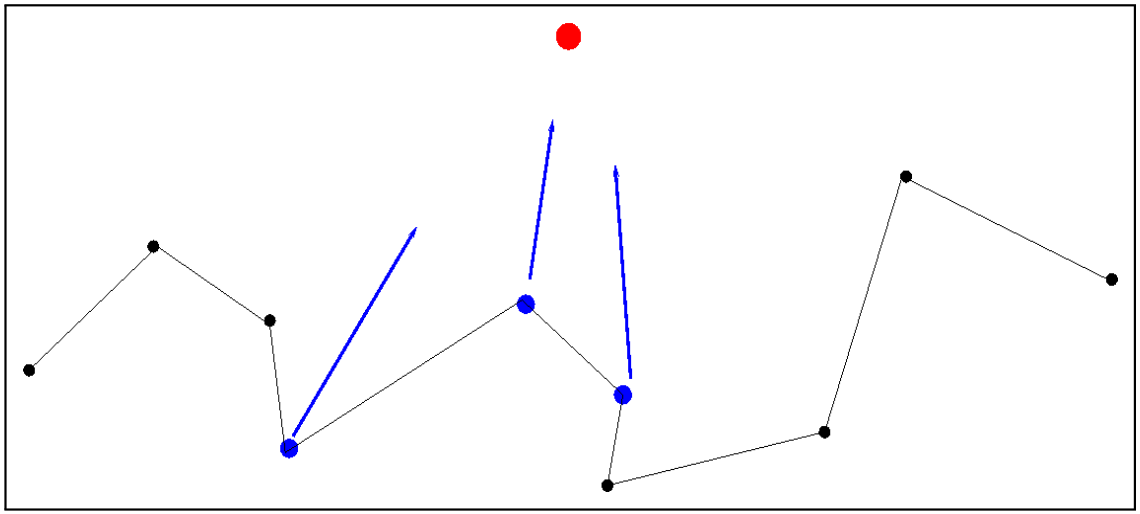
- les données d'un même cluster soient "proches" dans l'espace de départ ;
- les données appartenant à des classes voisines soient "proches" ou relativement "proches" dans l'espace de départ ;
- les données appartenant à des clusters éloignées soient "éloignées" dans l'espace de départ

Plusieurs buts à atteindre

- Extraction de vecteurs-codes ou prototypes : quantification
- Définition et description des classes : clustering et construction d'une typologie
- Projection et visualization de données multidimensionnelles
- Cas très fréquent : prendre en compte des données incomplètes ou manquantes
- Affectation a posteriori d'observations supplémentaires

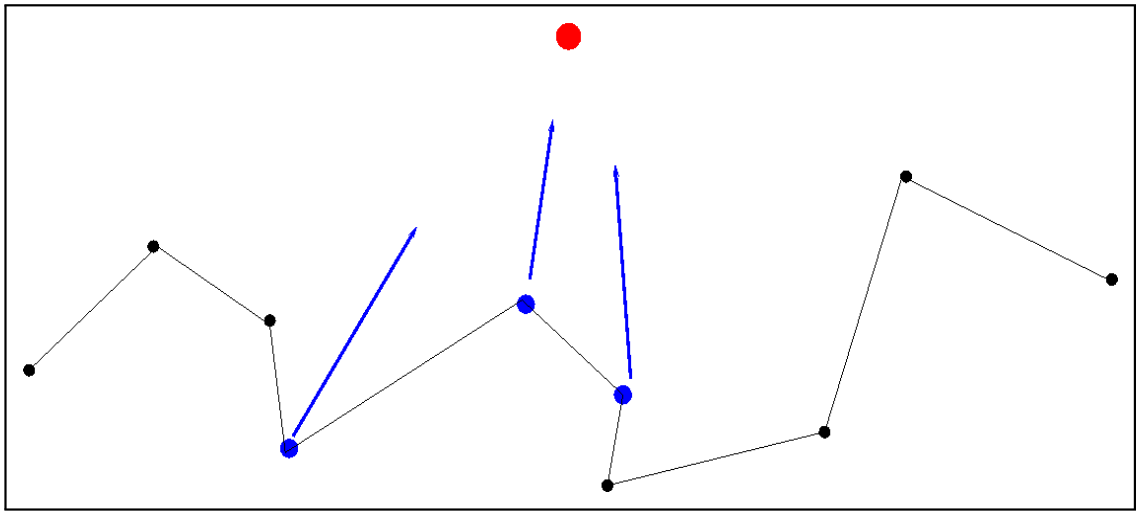
L'algorithme SOM - phase d'apprentissage

- Avant

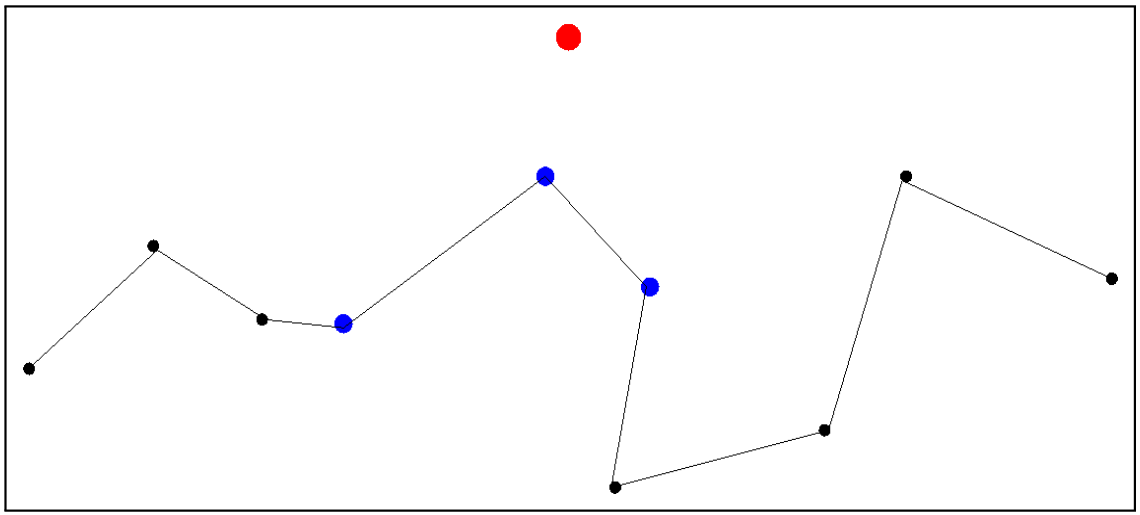


L'algorithme SOM - phase d'apprentissage

- Avant



- Après



SOM pour données vectorielles - notations

- La carte contient U neurones (symbolisés graphiquement par les noeuds de la grille). A chaque noeud u est associé un prototype p_u (un objet dans le même espace que les données de départ)
- La structure de la carte détermine une distance naturelle entre les noeuds sur la grille : la distance $d(u, u')$ entre les noeuds u et u' est définie comme la longueur du plus court chemin entre les deux.
- La carte est également équipée d'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 la meilleure partition des données minimisant l'énergie (variance intra "étendue")

$$\mathcal{E} = \sum_{u=1}^{U} \sum_{i=1}^{n} h^t(d(f(x_i), u)) \|x_i - p_u\|^2 ,$$

SOM pour données vectorielles - l'algorithme

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

1: Initialisation : choisir au hasard p_1^0,...,p_U^0 in $\mathbb{R}^d$
2: **for** $t = 1 \rightarrow T$ **do**
3: Choisir au hasard $i \in \{1, \ldots, n\}$
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
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 μ_t est généralement de l'ordre de $1/t$.

Problèmes avec les données non-vectorielles : comment définir $\|.\|$ et p_u ?

SOM relationnel

Données : $x_i \in \mathcal{G}$ définis par une mesure de dissimilarité $\delta(x_i, x_j)$

- Hypothèse : les prototypes peuvent être écrits symboliquement comme des combinaisons convexes des observations :

$$\forall u = 1, ..., U,\ p_u = \sum_{i=1}^{n} \gamma_{u,i} x_i$$

$$\gamma_{u,i} \geq 0,\ \sum_{i=1}^{n} \gamma_{u,i} = 1$$

- Inspiré de Kernel SOM, [Mac Donald and Fyfe, 2000]

- Si la matrice de dissimilarité est symétrique et définie positive,

$$\begin{aligned} \|x_j - p_u\|^2 &= \langle x_j - p_u, x_j - p_u \rangle \\ &= \langle x_j - \textstyle\sum_{i=1}^{n} \gamma_{u,i} x_i, x_j - \sum_{i=1}^{n} \gamma_{u,i} x_i \rangle \\ &= (\Delta \gamma_u)_j - \tfrac{1}{2} \gamma_u^T \Delta \gamma_u, \end{aligned}$$

où $\Delta = \left(\delta(x_i, x_j)\right)_{i,j=1,...,n}$

SOM relationnel

1: Initialisation : $\gamma_{u,i}^0$ au hasard dans $\mathbb{R}$
2: **for** $t = 1 \rightarrow T$ **do**
3: Choisir au hasard une observation courante, x_i
4: Affectation $f^t(x_i) \leftarrow \arg\min_{u=1,\dots,U} \left(\gamma_u^{t-1}\Delta\right)_i - \frac{1}{2}\gamma_u^{t-1}\Delta(\gamma_u^{t-1})^T$
5: **for all** $u = 1 \rightarrow U$ **do** Mise à jour
6: $\gamma_u^t \leftarrow \gamma_u^{t-1} + \mu_t h^t(d(f^t(x_i), u))\left(\mathbf{1}_i - \gamma_u^{t-1}\right)$ où $\mathbf{1}_i$ est un vecteur avec un seul élément non nul, égal à 1, sur la position$i$
7: **end for**
8: **end for**

[Olteanu and Villa-Vialaneix, 2015]

Equivalence :

- Si la dissimilarité est calculée à partir d'un noyau, alors kernel SOM et SOM relationnel sont équivalents
- Si la dissimilarité est le carré de la distance euclidienne et si les prototypes sont initialisés dans l'enveloppe convexe des observations, alors SOM relationnel et l'algorithme SOM de base sont équivalents

SOM relationnel appliqué aux graphes

Données : $\mathcal{G} = (V, E, W)$, où $V = (x_1, ..., x_n)$ sont les sommets du graphe, E est la matrice d'adjacence et W est une matrice contenant éventuellement des pondérations sur les arrêtes

Dissimilarités possibles :

- La distance du plus court chemin
- Dissimilarités basées sur la décomposition spectrale du Laplacien, [von Luxburg, 2007], [Fouss et al., 2007], [Kondor and Lafferty, 2002]
- Dissimilarités basées sur la décomposition spéctrale de la modularité, [Newman, 2006]

Résumé

Pour quoi un autre package R pour faire du SOM ?

SOM existe dans de nombreuses implémentations

- Matlab : SOM Toolbox (version 2.1, mise à jour en décembre 2012)
  `http://research.ics.aalto.fi/software/somtoolbox`
- SAS : SOM dans Miner et package P. Letrémy (version 9.1.3, mise à jour en décembre 2005 et non-maintenue depuis)
  `http://samos.univ-paris1.fr/Programmes-bases-sur-l-algorithme`
- R : **class, som, popsom, kohonen, yasomi**

L'algorithme SOM relationnel pour données complexes (dissimilarités, noyaux)

- Matlab : SOM Toolbox
- R : **yasomi** (version 0.3, mise à jour en mars 2011, [Rossi, 2012])

mais uniquement en version BATCH

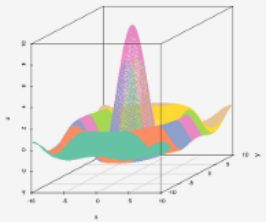
Package **SOMbrero**, version 1.0, disponible depuis mars 2015 sur CRAN :
`http://cran.r-project.org/web/packages/SOMbrero`

Interface graphique

SOMbrero Web User Interface (v1.0)

Select the data type:

Relational



Welcome to SOMbrero, the open-source on-line interface for self-organizing maps (SOM).

This interface trains SOM for numerical data, contingency tables and dissimilarity data using the **R** package SOMbrero (v1.0). Train a map on your data and visualize their topology in three simple steps using the panels on the right.

It is kindly provided by the SAMM team and the MIA-T team under the GPL-2.0 license, and was developed by

Import Data | Self-Organize | Plot Map | Superclasses | Combine with external information | Help

Third step: plot the self-organizing map

In this panel and the next ones you can visualize the computed self-organizing map. This panel contains the standard plots used to analyze the map.

Options

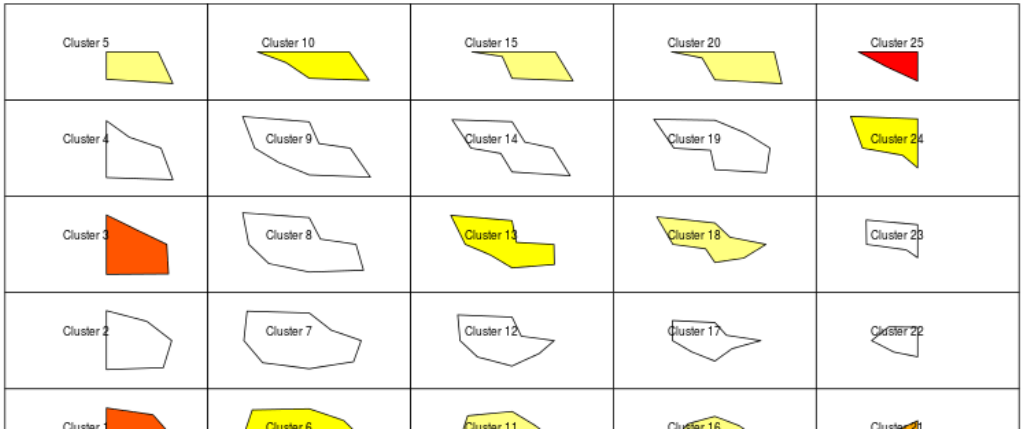
Plot what?

Prototypes

Type of plot:

polygon distances

Show cluster names



Fonction `trainSOM`

Trois algorithmes différents appelés via la même fonction

- `type="numeric"`
- `type="korresp"`
- `type="relational"`

Autres arguments de la fonction :

argument	description	valeur par défaut
`dimension`	dimension de la grille	$\lceil \sqrt{n/10} \rceil$
`dist.type`	d	Distance euclidienne entre les coordonnées des noeuds sur la grille.
`radius.type`	H	Voisinage gaussien, $H(x) = e^{-\beta x}$ avec $\beta > 0$ décroissant avec les itérations
`affectation`	calcul de l'unité gagnante	
`maxit`	Le nombre d'itérations	$5 \times n$.
`init.proto`	comment initialiser les prototypes ?	Les coefficients (γ_{ui}) sont initialisés à 0 à l'exception de $\gamma_{u,i(u)} = 1$ où $i(u)$ est choisi au hasard dans $\{1, \ldots, n\}$ (option "`obs`")
`scaling`	normalisation	Les données de départ
`nb.save`	nombre de sauvegardes intermédiaires	0

Vérifier la qualité des résultats

Deux fonctions

- `summary` : bref résumé des résultats et une ANOVA
- `quality` : deux critères sont calculés, [Polzlbauer, 2004]
  1. l'erreur topographique : quantifie la continuité de la carte par rapport à l'espace de départ
  2. l'erreur de quantification : quantifie la qualité du clustering

$$\frac{1}{n} \sum_{u=1}^{U} \sum_{i:\, f(x_i)=u} \left[(D\gamma_u)_i - \frac{1}{2} \gamma_u^T D\gamma_u \right]$$

Fonction `plot`

Deux arguments pour une très large palette de graphiques

- `what` : les observations, les prototypes, des variables additionnelles
- `type` : par exemple, pour SOM relationnel

type	relational		
	obs	proto	add
color			x
lines		x	x
barplot		x	x
radar		x	x
boxplot			x
poly.dist		x	
umatrix		x	
smooth.dist		x	
mds		x	
grid.dist		x	
hitmap	x		
names	x		x
words			x
pie			x
graph			x

Clustering et projection d'un graphe

Fonction `superClass`

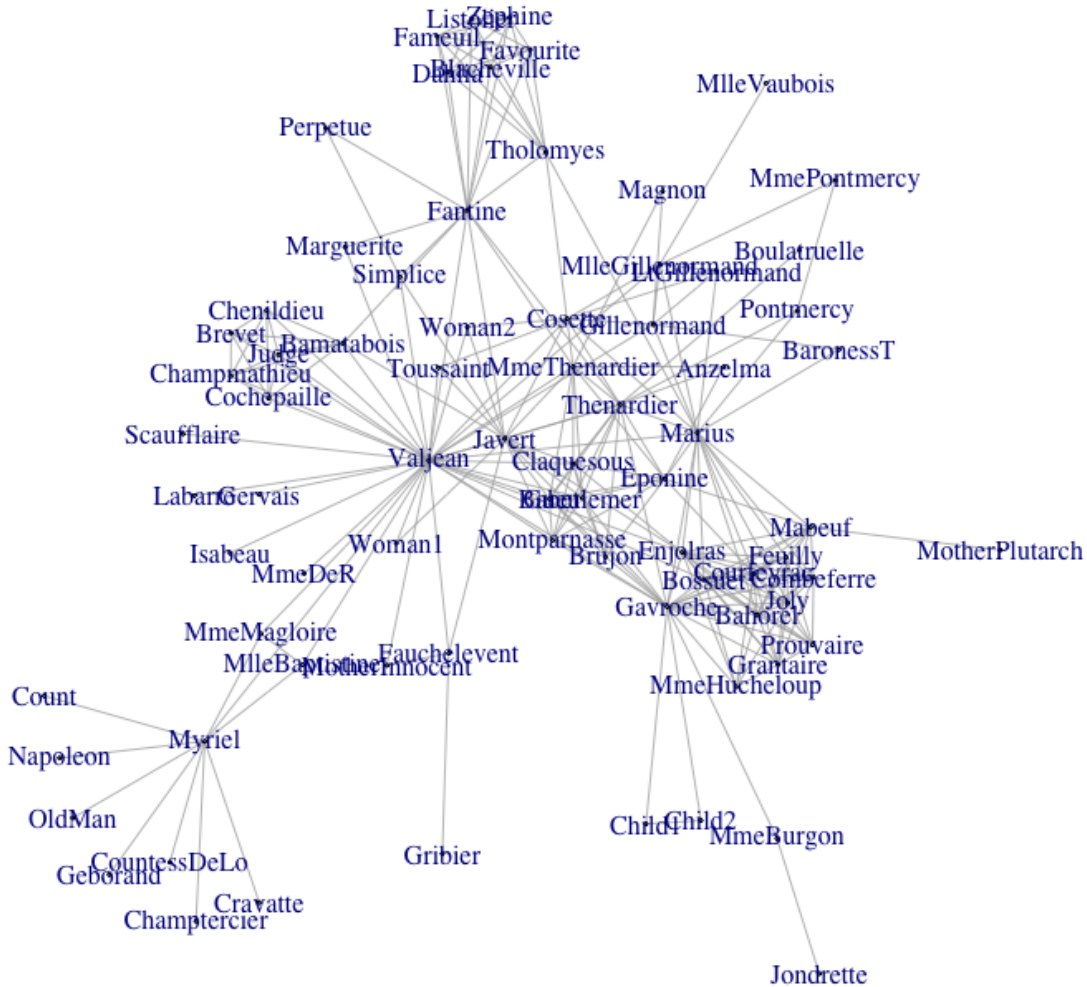
- Classification hiérarchique sur les prototypes de la carte
- Dendrogramme et scree-plot avec `plot.somSC`
- Les résultats peuvent ensuite être combinés avec tous les graphiques implémentés

Fonction `projectIGraph`

- Deux arguments, le résultat de l'algorithme SOM et le graphe de départ comme objet dans `igraph`
- Le résultat est un objet `igraph`
- Version simplifiée du graphe de départ

Résumé

Les Misérables



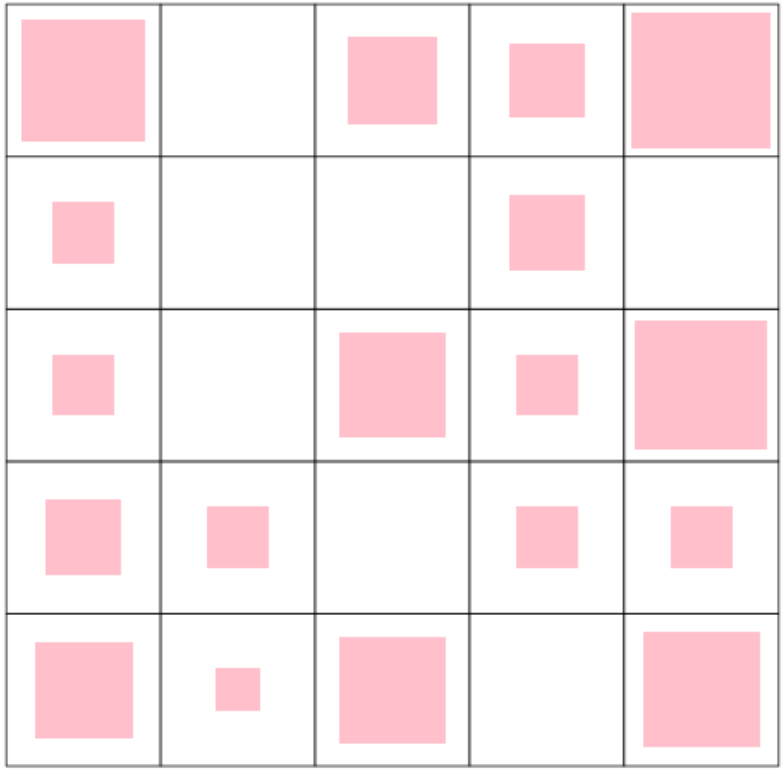
Le code

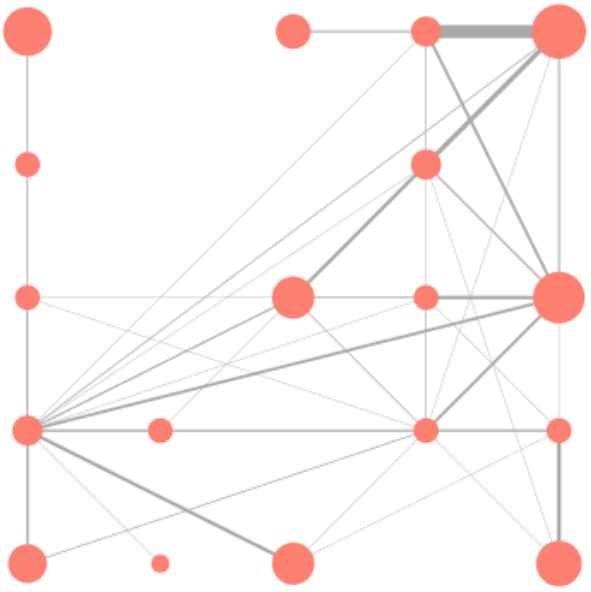
```
data(lesmis)

som.lemis <- trainSOM(dissim.lesmis, init.proto="random", maxit=500,
type="relational", radius.type="letremy")

quality(som.lemis)
$topographic
[1] 0
$quantization
[1] 0.613031
```

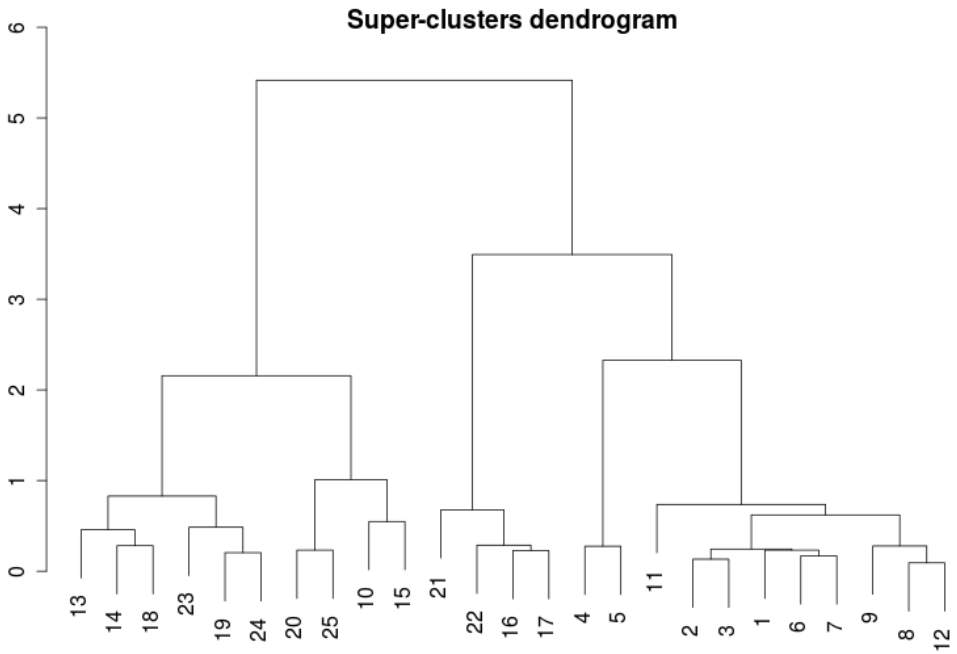
```
plot(som.lesmis, what="obs")
plot(som.lesmis, what="add", type="graph", variable=lesmis)
```

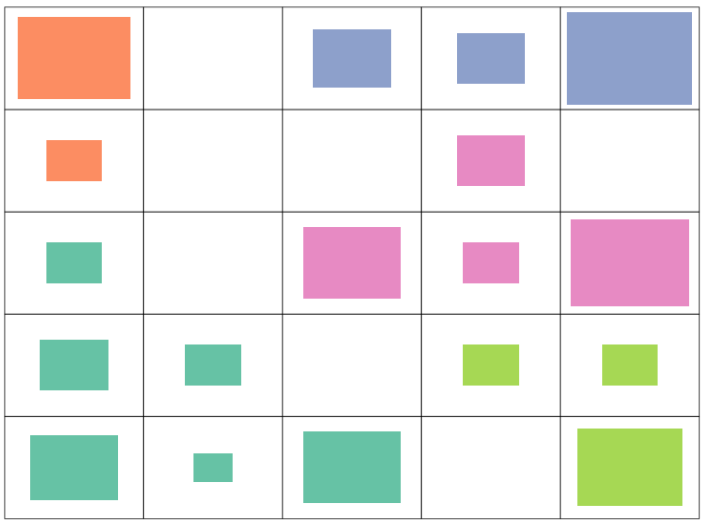




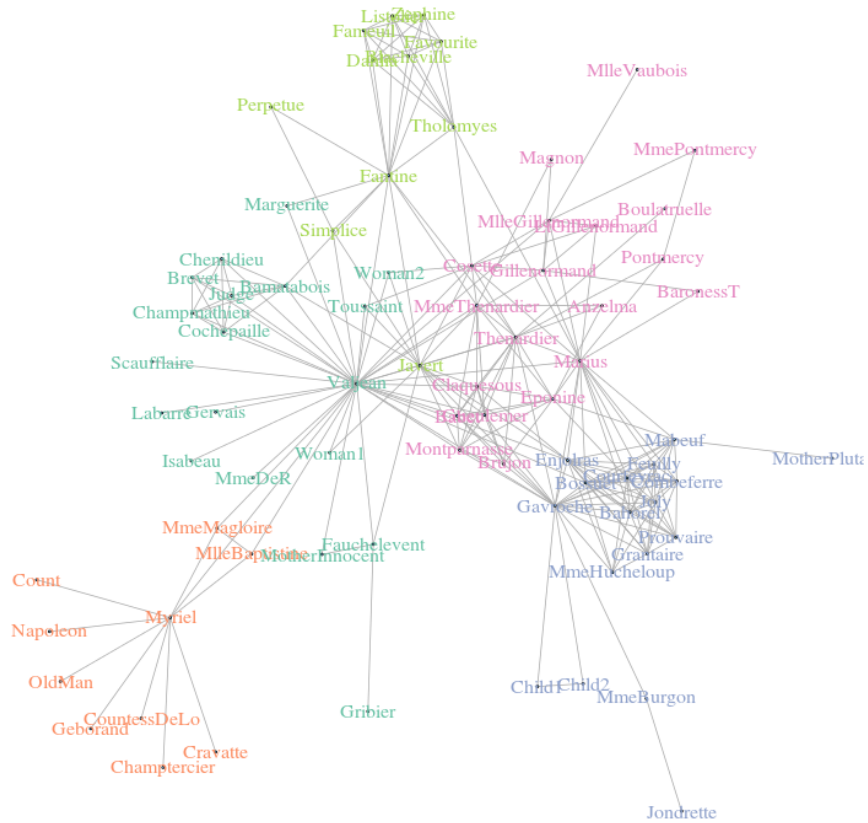
```
plot(som.lesmis, what="obs", type="names", scale=c(0.9,0.5))
```

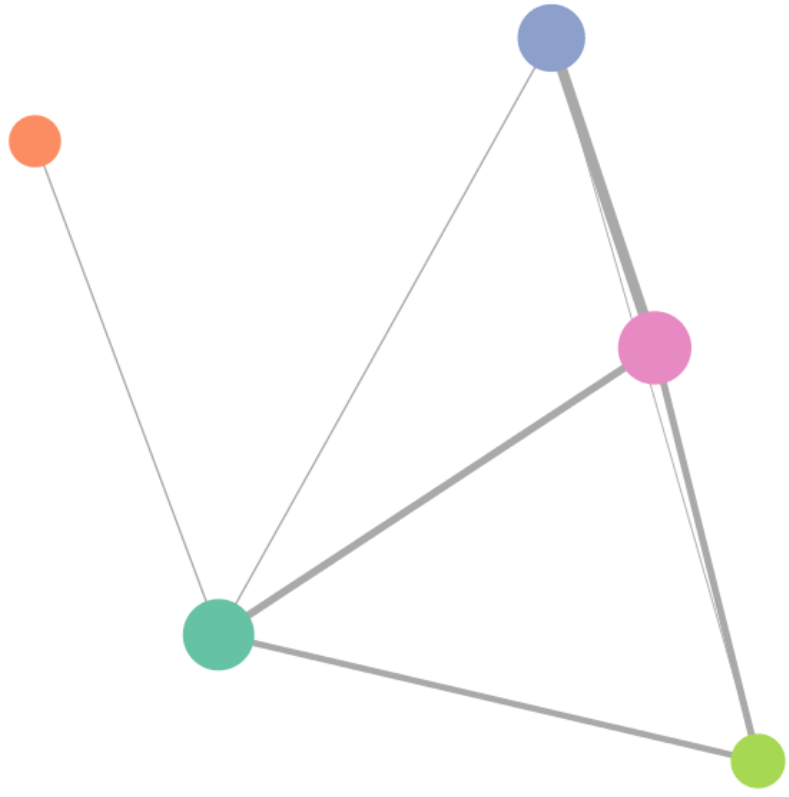
```
 sc.lesmis <- superClass(som.lesmis)
plot(sc.lesmis)
sc.lesmis <- superClass(som.lesmis, k=5)
plot(sc.lesmis, type="hitmap")
```





```
 plot(lesmis, vertex.size=0, vertex.label.color= brewer.pal(5,
"Set2")[sc.lesmis$cluster[som.lesmis$clustering]])
plot(sc.lesmis, type="projgraph", variable=lesmis)
```





Résumé

Conclusion

- Package **SOMbrero**, `http://cran.r-project.org/web/packages/SOMbrero`
- Algorithme SOM relationnel testé sur des graphes
- Représentations simplifiées d'un graphe basées sur le clustering

References

Fouss, F., Pirotte, A., Renders, J., and Saerens, M. (2007).

Random-walk computation of similarities between nodes of a graph, with application to collaborative recommendation.
IEEE Transactions on Knowledge and Data Engineering, 19(3) :355–369.

Kohonen, T. (1995).

Self-Organizing Maps, volume 30 of *Springer Series in Information Science*.

Kondor, R. and Lafferty, J. (2002).

Diffusion kernels on graphs and other discrete structures.
In *Proceedings of the 19th International Conference on Machine Learning*, pages 315–322.

Mac Donald, D. and Fyfe, C. (2000).

The kernel self organising map.
In *Proceedings of 4th International Conference on knowledge-based Intelligence Engineering Systems and Applied Technologies*, pages 317–320.

Newman, M. (2006).

Finding community structure in networks using the eigenvectors of matrices.
Physical Review, E, 74(036104).

Olteanu, M. and Villa-Vialaneix, N. (2015).

On-line relational and multiple relational SOM.
Neurocomputing, 147 :15–30.

Polzlbauer, G. (2004).

Survey and comparison of quality measures for self-organizing maps.
In Paralic, J., Polzlbauer, G., and Rauber, A., editors, *Proceedings of the Fifth Workshop on Data Analysis (WDA'04)*, pages 67–82, Sliezsky dom, Vysoke Tatry, Slovakia. Elfa Academic Press.

Rossi, F. (2012).

yasomi : Yet Another Self Organising Map Implementation.
R package version 0.3/r39.

von Luxburg, U. (2007).

A tutorial on spectral clustering.
Statistics and Computing, 17(4) :395–416.