Journal de la Société Française de Statistique
Vol. 156 No. 3 (2015)

Using SOMbrero for clustering and visualizing graphs

Titre: Utiliser SOMbrero pour la classification et la visualisation de graphes

Madalina Olteanu[1] and Nathalie Villa-Vialaneix[2]

Abstract: Graphs have attracted a burst of attention in the last years, with applications to social science, biology, computer science... In the present paper, we illustrate how self-organizing maps (SOM) can be used to enlighten the structure of the graph, performing clustering of the graph together with visualization of a simplified graph. In particular, we present the R package **SOMbrero** which implements a stochastic version of the so-called relational algorithm: the method is able to process any dissimilarity data and several dissimilarities adapted to graphs are described and compared. The use of the package is illustrated on two real-world datasets: one, included in the package itself, is small enough to allow for a full investigation of the influence of the choice of a dissimilarity to measure the proximity between the vertices on the results. The other example comes from an application in biology and is based on a large bipartite graph of chemical reactions with several thousands vertices.

Résumé : L'analyse de graphes a connu un intérêt croissant dans les dernières années, avec des applications en sciences sociales, biologie, informatique, ... Dans cet article, nous illustrons comment les cartes auto-organisatrices (SOM) peuvent être utilisées pour mettre en lumière la structure d'un graphe en combinant la classification de ses sommets avec une visualisation simplifiée de celui-ci. En particulier, nous présentons le package R **SOMbrero** dans lequel est implémentée une version stochastique de l'approche dite « relationnelle » de l'algorithme de cartes auto-organisatrices. Cette méthode permet d'utiliser les cartes auto-organisatrices avec des données décrites par des mesures de dissimilarité et nous discutons et comparons ici plusieurs types de dissimilarités adaptées aux graphes. L'utilisation du package est illustrée sur deux jeux de données réelles : le premier, inclus dans le package lui-même, est suffisamment petit pour permettre l'analyse complète de l'influence du choix de la mesure de dissimilarité sur les résultats. Le second exemple provient d'une application en biologie et est basé sur un graphe biparti de grande taille, issu de réactions chimiques et qui contient plusieurs milliers de nœuds.

Keywords: graph, R package, self-organizing map, clustering, visualization
Mots-clés : graphe, package R, carte auto-organisatrice, classification, visualisation
AMS 2000 subject classifications: 62-07, 62-09

1. Introduction

Graphs have attracted a burst of attention in the last years, with applications to social sciences, biology, computer science... When wanting to understand the way a graph is structured and how the relations it models organize groups of entities, clustering and visualization have been proven to be useful tools. They are sometimes combined in order to provide the user with a global overview of the graph (Noack, 2007). Several approaches have been developed so far to perform such tasks: (i) using a vertex clustering prior to visualizing the graph induced by the clustering, which is a

[1] SAMM, EA4543, Université Paris 1, 90 bd de Tolbiac, 75013 Paris cedex 13 - France
E-mail: madalina.olteanu@univ-paris1.fr
[2] INRA, UR875 MIA-T, BP 52627, 31326 Castanet Tolosan cedex - France
E-mail: nathalie.villa@toulouse.inra.fr

simplified representation of the graph in which clusters are symbolized by a single vertex; (ii) visualizing the whole graph by adding constraints on the vertex locations (Eades and Huang, 2000) that can be derived from a prior vertex clustering or (iii) combining in one step clustering and visualization by using topology preserving methods such as self-organizing maps (SOM) or derived topological maps (Boulet et al., 2008; Rossi and Villa-Vialaneix, 2010).

In the present article, we shall illustrate the latter approach by showing how a specific self-organizing map designed to handle data described by a dissimilarity matrix can be used to provide a simplified representation and thus an overview of the graph. To do so, we rely on the R package **SOMbrero** which implements several extensions of the stochastic algorithm for self-organizing maps to non vectorial data. The article is organized as follows: Section 2 recalls basic background about self-organizing maps and their extensions to non-vectorial data. A discussion is provided in this section on how these extensions can be used with graph and, in particular, we review different standard dissimilarity measures adapted to quantify the proximity between nodes in a graph. Then, Section 3 reviews SOM programs and R packages and presents **SOMbrero**. The package's main features are described. Finally, Section 4 illustrates its use on two real-life data sets. In particular, the first dataset is used as a complete simple use case example of **SOMbrero** and also as a way to investigate the influence of the chosen dissimilarity on the results. The second dataset comes from a biological application and is discussed regarding external information provided by the application context.

2. Self-organizing Maps and their extension to graphs

2.1. Basics on Self-organizing maps

In its standard (Euclidean) version, the self-organizing map algorithm (SOM, also known as Kohonen's algorithm, Kohonen, 2001) aims at mapping n input data $x_1, \ldots, x_n \in \mathbb{R}^d$ into a low dimensional map (or) composed of U units (or neurons). A prototype p_u, taking values in the same space as the input data, is associated to each unit $u \in \{1,\ldots,U\}$ of the map. The map is equipped with a distance d between units (usually chosen as the length of the shortest path on the map). The algorithm aims at clustering together similar observations while preserving the original topology of the data (*i.e.*, close observations in the input space are clustered together into the same unit or into neighbor units on the map). Several approaches have been proposed to perform the clustering, both stochastic and batch versions. In this article, we focus on the stochastic version, which is known to provide a better topology preservation on the map than the batch version, at the cost of a larger computational time, Fort et al., 2002).

After the prototypes have been initialized in $\mathbb{R}^d$ (usually randomly or using a PCA), the stochastic SOM iterates over two steps:

– an *assignment step* in which one observation, x_i, is picked at random and is affected to the closest prototype:

$$f(x_i) = \arg \min_{u=1,\ldots,U} \|x_i - p_u\|, \tag{1}$$

where $\|\cdot\|$ is the Euclidean distance in $\mathbb{R}^d$;

– a *representation step* in which all prototypes are updated according to the new assignment by mimicking a stochastic gradient descent scheme:

$$\forall u = 1, \dots, U, \qquad p_u^{\text{new}} = p_u^{\text{old}} + \mu H\left(d\left(f(x_i), u\right)\right)\left(x_i - p_u^{\text{old}}\right), \tag{2}$$

where H is the neighborhood function verifying the assumptions $H : \mathbb{R}^+ \to \mathbb{R}^+$, $H(0) = 1$ and $\lim_{x \to +\infty} H(x) = 0$, and μ is a training parameter. Generally, H and μ are supposed to be decreasing with the number of iterations during the training procedure so that H is the indicator function in the last iterations of the SOM algorithm (hence, the last iterations of the algorithm are used to improve the clustering quality of the map, in a way that is similar to on-line k-means algorithm, once the map has been organized during the first iterations).

Contrary to the stochastic version, the batch version of the SOM algorithm (that we will not discuss further in this article) processes all data at each iteration and updates the prototypes with a generalized mean of the observations classified in the neighborhood of the corresponding unit.

2.2. *SOM for kernel and dissimilarity data*

Since the original algorithm for vectorial data was introduced, several variants of self-organizing maps have been proposed to extend its use to more complex data and in particular to graphs. The most popular extensions rely on the computation of a measure of similarity or dissimilarity between the input data. The usual framework is to suppose that the input data are defined in a general space, $x_1, \dots, x_n \in \mathcal{G}$, and one may compute $D = (\delta_{ii'})_{i,i'=1,\dots,n}$ the dissimilarity matrix, generally positive ($\delta_{ii'} \geq 0$) and symmetric ($\delta_{ii'} = \delta_{i'i}$), but not necessarily Euclidean [1].

One approach of extending the SOM algorithm to data described by their dissimilarities consists in using the median principle : replace the standard computation of the prototypes by an approximation in the original data set (Ambroise and Govaert, 1996; Kohohen and Somervuo, 1998; Conan-Guez et al., 2006) and transform the representation step of Equation (2) into a discrete optimization scheme. However, this method, called “median SOM”, increases the computational time, while prototypes remain restricted to the original data and may generate possible sampling or sparsity issues. Another extension is proposed in Graepel and Obermayer (1999) and uses the dissimilarity in a probabilistic framework which is solved with an EM scheme. In this latter case, the optimized cost function is not exactly equivalent to the original cost function of the SOM and the topographic map is not based on prototypes which can help interpret the units.

The so-called “relational SOM” (the reader may refer to Hammer and Hasenfuss (2010) for the batch version and to Olteanu and Villa-Vialaneix (2015) for the stochastic version) overcomes the constrained framework of median SOM by defining the prototypes as (implicit) convex combinations of the original data. The underlying hypothesis for relational SOM is the existence of a pseudo-euclidean embedding of the original dataset, as defined in Goldfarb (1984), (see Hammer and Hasenfuss (2010); Olteanu and Villa-Vialaneix (2015) for further details). The original input data $x_i \in \mathcal{G}$, $i = 1, \dots, n$ may then be represented as vectors in the embedding space, $\Phi(x_i) \in \mathcal{E}$, $i = 1, \dots, n$. The explicit form of Φ is not needed since only the dissimilarities are used :

$$\delta_{ij} = \langle \Phi(x_i) - \Phi(x_j), \Phi(x_i) - \Phi(x_j) \rangle_{\mathcal{E}} \tag{3}$$

[1] see Schoenberg (1935); Young and Householder (1938); Krislock and Wolkowicz (2012) for necessary conditions that make a dissimilarity matrix embeddable in a Euclidean space.

Using the previous representation, prototypes are defined as convex combinations of vectors in the embedding space, $p_u = \sum_{i=1}^n \gamma_{ui}\Phi(x_i) \in \mathscr{E}$, with $\forall u = 1,\dots,U$, $\sum_{i=1}^n \gamma_{ui} = 1$ and $\gamma_{ui} \geq 0$. Defining prototypes as convex combinations of the input data makes sense and does not add supplementary constraints with respect to the original algorithm since this is exactly the case for the original algorithm: in the batch version, prototypes are defined as weighted averages of the input vectors; in the on-line version, this property also holds if the prototypes are initialized in the convex hull of the input data.

One can now define a dissimilarity between input data and prototypes in $\mathscr{E}$:

$$\delta\left(\Phi(x_i), p_u\right) := \|p_u - \Phi(x_i)\|^2_{\mathscr{E}} = (D\gamma_u)_i - \frac{1}{2}\gamma_u^T D\gamma_u$$

where $\gamma_u = (\gamma_{ui})_i \in \mathbb{R}^n$. Furthermore, since the prototypes are constrained to convex combinations of the input data, the update step of the algorithm involves only the weights γ_{ui}. More precisely, after having initialized the prototypes, the two steps of the stochastic version of the relational SOM consist in the following modifications of steps described in Equations (1) and (2) above:

– *assignment step*:

$$f(x_i) = \arg\min_{u=1,\dots,U} (D\gamma_u)_i - \frac{1}{2}\gamma_u^T D\gamma_u \tag{4}$$

which is exactly equivalent to the assignment step in the standard SOM (Equation (1)), if the dissimilarity matrix D is the squared Euclidean distance;

– *representation step*:

$$\gamma_u^{\text{new}} \leftarrow \gamma_u^{\text{old}} + \mu H(d(f(x_i),u))\left(\mathbf{1}_i - \gamma_u^{\text{old}}\right) \tag{5}$$

where $\mathbf{1}_i$ is a vector with a single non null coefficient at the *i-th* position, equal to one. Again, this modification is exactly equivalent to the representation step (Equation (2)) in the standard SOM if the dissimilarity matrix D is the squared Euclidean distance.

In some cases, instead of being described by pairwise dissimilarities, data are given as pairwise similarities, $(s_{ii'})_{i,i'=1,\dots,n}$, sometimes expressed by means of a *kernel*, *i.e.*, a fonction $K : \mathscr{G} \times \mathscr{G} \to \mathbb{R}$ which is symmetric ($K(x,x') = K(x',x)$) and positive ($\forall N \in \mathbb{N}$, $\forall (\alpha_i)_i \subset \mathbb{R}$, $\forall (x_i)_i \subset \mathscr{G}$, $\sum_{i,i'=1}^N \alpha_i\alpha_{i'}K(x_i,x_{i'}) \geq 0$). In this case, an implicit Hilbert space $\mathscr{H}$ exists as well as an embedding function $\Phi : \mathscr{G} \to \mathscr{H}$ such that

$$s_{ii'} = K(x_i,x_{i'}) = \langle \Phi(x_i), \Phi(x_{i'}) \rangle ,$$

(see Aronszajn, 1950). K can thus be interpreted as a dot product in $\mathscr{H}$. Another version of SOM, called *kernel SOM* and very similar to relational SOM, is also available in the literature and was introduced prior to relational SOM (Mac Donald and Fyfe, 2000; Andras, 2002; Boulet et al., 2008). However, kernel SOM was proven (Rossi, 2014; Olteanu and Villa-Vialaneix, 2015) to be exactly equivalent to relational SOM when using a dissimilarity δ which is Euclidean. Actually, the relational algorithm applied to the squared distance induced by K in $\mathscr{H}$, $\delta_{ii'} = s_{ii} + s_{i'i'} - 2s_{ii'}$ is strictly identical to the kernel SOM used with K. In fact, relational SOM generalizes the kernel version of the algorithm thanks to the pseudo-Euclidean embedding but for non-Euclidean dissimilarities/similarities, the two versions differ (because a kernel cannot be defined in these cases) and relational SOM can lead to numerical difficulties caused by saddle points (see Hammer et al., 2014 for further discussions).

Since relational SOM is designed for (dis)similarities and is based on very mild hypotheses, it can be applied to complex data such as graphs as long as a (dis)similarity can be defined. In the following, $\mathscr{G} = (V, E, W)$ denotes a graph with vertices $V = \{x_1, \ldots, x_n\}$, edges E which is a subset of $V \times V$ and possibly a weight matrix W (*i.e.*, an $(n \times n)$-matrix, symmetric, with positive entries and such that $W_{ii} = 0$). When clustering and projecting graphs with relational SOM, vertices represent the input data, while the edges (and the weight matrix) are used for computing the dissimilarity matrix. The clustered vertices are then projected onto the map as a simplified graph as explained in the following.

2.3. *Using the SOM results to display a simplified graph*

As explained in the introduction, it is common practice to represent a clustering of vertices by a simplified graph induced from the original graph and the result of the clustering of its vertices (see, for instance, Herman et al., 2000; Rossi and Villa-Vialaneix, 2011, among others).

In this approach, the vertices of the simplified graph represent the clusters, each cluster being symbolized by a given glyph whose area is proportional to the cluster's size (*i.e.*, to the number of vertices of the original graph that are classified in this cluster). The edges that link two clusters usually have a width proportional to the number of edges (of the sum of the edges' weights) between the pairs of vertices clustered in the two corresponding clusters. When the clustering is a prior step before the visualization, the clustered graph is drawn using modified force-directed placement algorithm (Fruchterman and Reingold, 1991), which can cope with vertices having non-uniform sizes (Harel and Koren, 2002; Tunkelang, 1999; Wang and Miyamoto, 1996). However, when a topographic map is used, this second step is no more necessary because the prior structure of the map provides natural positions for the clusters: as shown in Boulet et al. (2008), standard maps position units in $\mathbb{R}^2$ at coordinates $(1,1)$, $(1,2)$, ..., (p,q) where p and q are the length and width of the map (and thus $pq = U$): the induced graph is thus displayed with vertices positioned at these coordinates. Rossi and Villa-Vialaneix (2010) uses a similar approach to represent graphs, by defining a topographic map algorithm. This method does not rely on a dissimilarity but on a criterion specific to graphs and derived from the modularity (Newman and Girvan, 2004), which is a very standard quality measure for clustering the vertices of a graph.

2.4. *Standard dissimilarities for vertices in a graph*

Let us describe now the standard dissimilarities and kernels that are commonly used in the graph context to measure the dissimilarity between vertices.

One of the most standard dissimilarities for vertices is the length of the shortest path between them. However, this measure does not take into account the number of paths that link two vertices but only the length of the shortest one and it cannot easily include information about the possible weights that can be associated to edges. A very popular method to cluster the vertices of a graph is the so-called spectral clustering (von Luxburg, 2007). This approach is based on a spectral decomposition of the Laplacian of the graph, $L = (L_{ij})_{i,j=1,\ldots,n}$, a matrix encoding the graph structure

$$L_{ij} = \begin{cases} -W_{ij} & \text{if } i \neq j, \\ d_i = \sum_{k \neq i} W_{ij} & \text{otherwise.} \end{cases}$$

More precisely, the method first performs an eigen-decomposition of the Laplacian, $((\lambda_i)_{i=1,\ldots,n}, (v_i)_{i=1,\ldots,n})$ with $\lambda_1 = 0 \leq \lambda_2 \leq \ldots \leq \lambda_n$ the eigenvalues of the Laplacian (in increasing order: the first eigenvalue is always equal to zero) and $v_i \in \mathbb{R}^n$. Then, every vertices x_i are represented by the entries of the eigenvectors associated to the C smallest positive eigenvalues of the Laplacian, where C is the number of clusters that are searched: thus, each vertex is transformed into a vector in $\mathbb{R}^C$, $\mathbf{v}_i = (v_{1i}, \ldots, v_{Ci})$, where v_{ji} is the *i-th* coefficient of the *j-th* eigenvector. This method can be seen as a relaxed version of the normalized cut problem in which the number of edges of the original graph between vertices that belong to two different clusters is minimized while ensuring that the cluster size is large enough. Applied to relational SOM, this gives the idea to use the (squared)-Euclidean distance between $\mathbf{v}_i$ and $\mathbf{v}_{i'}$ to measure the dissimilarity between vertices x_i and $x_{i'}$.

This method can even be refined, using standard kernels for graphs, such as the ones proposed and discussed in Smola and Kondor (2003). These are based on regularized versions of the Laplacian and include, among others, the *commute time kernel*, $K_{\text{CT}} = L^+$ (Fouss et al., 2007), that can be interpreted as the average time a random walk takes to connect two vertices in the graph or the *heat kernel*, $K_{\text{H}} = e^{-\beta L}$ ($\beta > 0$, see Kondor and Lafferty, 2002) can be interpreted in term of a diffusion process on the graph.

As already said, another very standard way to cluster the vertices of a graph is to maximize a quality function called the modularity. For a partition $\mathscr{C}_1, \ldots, \mathscr{C}_C$ of the vertices in a graph, the modularity is equal to

$$\mathscr{Q}(\mathscr{C}_1,\ldots,\mathscr{C}_C) = \frac{1}{2m} \sum_{k=1}^{C} \sum_{x_i, x_j \in \mathscr{C}_k} \left(W_{ij} - \frac{d_i d_j}{2m} \right), \tag{6}$$

with m the total number of edges of the graph (so that $\sum_{ij} D_{ij} = \sum_{ij} W_{ij} = 2m$). Finding a partition that maximizes the modularity is thus equivalent to find a partition in which the observed intra-cluster weights, W_{ij} for x_i and x_j in the same cluster, are larger than those expected in a null model where the weights of the edges would depend only on the degrees of the afferent vertices. In Newman (2006), the modularity maximization problem is addressed using the eigen-decomposition of a matrix called the *modularity matrix* which can be expressed as

$$B = W - D$$

where D is the matrix $D_{ij} = \frac{d_i d_j}{2m}$. Similarly to the spectral clustering, this eigen-decomposition can be used as a mean to compute a dissimilarity measure between vertices in the graph: as suggested in Newman (2006), the eigenvectors of B associated with the positive eigenvalues, $b_1, \ldots, b_p$ (with $p \leq n$), make it possible to provide the following p-dimensional representation for the vertex x_i: $\mathbf{b}_i = (b_{1i}, \ldots, b_{pn})$ and the squared Euclidean distance between those vectors defines a dissimilarity matrix for the graph.

These different dissimilarities are illustrated and compared on a simple example in Section 4.1.

3. SOMbrero

3.1. Self-organizing maps programs and SOMbrero

Several implementations of the SOM algorithm exist in different mathematical/statistical softwares but most of them can only handle (standard) multi-dimensional datasets. Among them are the SOM Toolbox [2] (Matlab library), a function included in SAS Entreprise Miner (current version 12.3), the SAS programs by Patrick Letremy [3] and several R packages (**class**, **som**, **popsom**, **kohonen** and **yasomi**). Among these programs, only **yasomi** [4] and The Matlab SOM Toolbox provide the relational version of the SOM and can handle dissimilarity data. In these two programs, the batch version of the algorithm is implemented.

The R package **SOMbrero** (current version 1.0, released on March, 2015 on CRAN `http://cran.r-project.org/web/packages/SOMbrero`.) proposes two stochastic versions of the SOM algorithm for non vectorial data: the "KORRESP" algorithm by Cottrell et al. (1993), which handles contingency tables by means of a factor analysis [5], and the relational SOM. The implementation is slower than the one in **yasomi** but the package provides additional tools which may help the user for representing and interpreting the results. A special attention was given to making the syntax of the functions as simple as possible and to provide sufficient documentation and vignettes for using **SOMbrero** with different types of data. The package also includes a graphical interface that can be used even by those who do not know R programming language and that makes it suited for teaching also. Finally, several options and methods in the package were thought so as to handle graphs, including, in particular, a function that produces a simplified representation of a given graph given the results of the SOM algorithm or the results of a super-clustering of the prototypes of the SOM. Detailed features of the relational algorithm, which is the version that is to be used with graphs, are described in the next section.

3.2. Features of SOMbrero *for relational data*

3.2.1. Training the SOM: the function `trainSOM`

SOMbrero proposes three different variants of the SOM algorithm with only one function, `trainSOM`. Specifying the argument `type`, the standard multi-dimensional SOM is performed when setting `type="numeric"`, the "KORRESP" algorithm is performed when setting `type="korresp"` and the relational algorithm, that can be used for graphs, is performed when setting `type="relational"`. In the sequel, the description of the package will be restricted to the relational version.

[2] current version 2.1, last updated in December 2012, available at `http://research.ics.aalto.fi/software/somtoolbox` or on Github `https://github.com/ilarinieminen/SOM-Toolbox`

[3] current version 9.1.3, last updated in December 2005, no longer maintained, available at `http://samos.univ-paris1.fr/Programmes-bases-sur-l-algorithme`

[4] Rossi, 2012 (current version 0.3, last updated in March 2011)

[5] The description of this method is out of the scope of this paper since it is not designed to directly deal with graphs. We advice the reader to refer to the original article for further details on the method or Bourgeois et al. (2015) for an example of a way to use of the KORRESP algorithm with graphs.

TABLE 1. *Arguments that can set in the function* `trainSOM` *for the relational algorithm in* **SOMbrero** *1.0*

name	description	value
`dimension`	dimension (width and length) of the map	`dimension` is a vector of size 2. The default value is a square map with width and length approximately equal to $\sqrt{n/10}$. Units are then supposed to be positioned on a rectangular map in $\mathbb{R}^2$, at coordinates $(1,1)$, $(1,2)$, ..., (p,q) where p and q are the length and width of the map (and thus $pq = U$).
`dist.type`	d	The default value is the euclidean distance between the coordinates of the units in $\mathbb{R}^2$. All `method` provided by the function `dist` are also available. Additionally, the value `"letremy"` can be used, which corresponds to the original implementation by Patrick Letremy (more details about this option are provided in Section 4.1.1).
`radius.type`	H	The default value is a Gaussian neighborhood (*i.e.*, $H(x) = e^{-\beta x}$ with $\beta > 0$ decreasing throughout the iterations) but when `dist.type` is set to `"letremy"`, the value `"letremy"` can also be used for `radius.type`, which corresponds to the original implementation by Patrick Letremy (more details about this option are provided in Section 4.1.1).
`affectation`	type of the affectation step described in Equations (1) and (4)	The default value is the standard affectation step described in this article. An alternative is `"heskes"` which corresponds to the modified affectation step described in Heskes (1999).
`maxit`	number of iterations of the algorithm (affectation and representation steps)	The default value is equal to $5 \times n$.
`init.proto`	how to initialize the prototypes?	The coefficients (γ_{ui}) can be initialized randomly in $[0,1]$ and then scaled so that $\sum_i \gamma_{ui} = 1$ (option `"random"`). They can also be initialized to 0 except for one $\gamma_{u,i(u)} = 1$ where $i(u)$ is picked at random in $\{1,\ldots,n\}$ (option `"obs"`, which is the default) or they can be initialized using a MDS (Multi-Dimensional Scaling): in this case, the coordinates (γ_{ui}) are regularly placed along the first two axes of the MDS. Alternatively, the user can pass his/her own initial prototypes with the argument `proto0`.
`scaling`	type of data preprocessing	The default value is to use the original data directly (option `"none"`) but the similarity equivalence for the cosine preprocessing described in Ben-Hur and Weston (2010) is also available with the option `"cosine"` (further details on this preprocessing are given in the package's documentation).
`nb.save`	number of intermediate results to save	The default value is 0 which leads to save only the final map. Alternatively, the user can choose to save intermediate results (clustering and prototypes) during the training process.

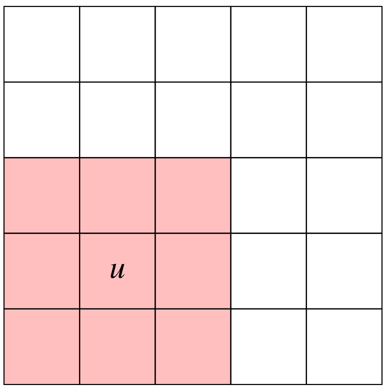
Several other arguments can be passed to the function for tuning the training algorithm. The main arguments that can be used for the relational version of the algorithm are summarized in Table 1.

The result of the function `trainSOM` is an object of class `somRes` which contains, in particular, the values of the prototypes (as a matrix having dimension $U \times n$) which correspond to the γ_{ui} and the clustering. Several methods have been implemented for this class, to help the user interpret the results.

3.2.2. *Checking the SOM's quality: the methods `summary` and `quality`*

Once the map is trained, **SOMbrero** offers several additional functions for checking its quality and for interpreting the results:

- a function `summary` displays a short summary of the results and includes an ANOVA procedure for testing the relevance of the clustering regarding the data. For the relational SOM a dissimilarity ANOVA is performed, as described in Anderson (2001);
- a function `quality` calculates two well-know quality criteria for the map (Polzlbauer, 2004):
  - the topographic error: this error quantifies the continuity of the map with respect to the input space metric by counting the number of times the second best matching unit of a given observation belongs to the direct neighborhood of the best matching unit for this observation. A topographic error equal to 0 means that all second best matching units are in the direct neighborhood of the winner units and thus that the original topology of the data is well preserved on the map. The direct neighborhood of the unit u in the squared map below are the colored units:



  - the quantization error: this error quantifies the quality of the clustering. It is equal to the average distance of the sample vectors to the cluster prototypes by which they are represented. For the relational SOM, this quantity is equal to

$$\frac{1}{n}\sum_{u=1}^{U}\sum_{i:\,f(x_i)=u}\left[(D\gamma_u)_i-\frac{1}{2}\gamma_u^T D\gamma_u\right]; \tag{7}$$

3.2.3. *Displaying the SOM: the method `plot`*

A function `plot` displays a wide range of plots aimed at giving a comprehensive overview of the resulting clusters (colors, bars, pie charts, boxplot, ...). Just two arguments handle the type of the output plot in a handy way: `what` and `type`. The argument `what` is used to plot either the original observations (in the case of the relational plot, only a dissimilarity matrix is passed to the training function and thus this option can not be used except for displaying the names of the observations in every unit of the map), the prototypes (or the distances between pairs of prototypes) or an additional variable that can be combined with the result of the map. This last option was found

TABLE 2. *Summary of plots available in* **SOMbrero** *1.0 for relational data*

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

to be very handy when the dissimilarity matrix corresponds to a graph: in this case, the option `type="graph"` makes it possible to obtain the simplified projected representation of the graph as described in Section 2.3. Table 2 summarizes all available graphics for the relational SOM algorithms.

3.2.4. *Clustering the prototypes: the method* `superClass`

The SOM algorithm often results in a large number of classes, which is not very handy for interpretation. Therefore, a common practice is to run an ascending hierarchical clustering algorithm on the prototypes of the trained map. **SOMbrero** implements this in the function `superClass`. A dendrogram and a scree-plot can be drawn using the function `plot.somSC`, which can guide the user's choice. Results of this super-clustering can also be combined with any of the graphics described in Section 3.2.3.

3.2.5. *Projecting a graph on the SOM: the method* `projectIGraph`

Finally, when the clustered entities are vertices of a graph, the projected graph which is displayed as described in Section 2.3 can easily be retrieved as an `igraph`[6] object using the method `projectIGraph` which takes two parameters: the result of the SOM algorithm and the original graph on the form of an `igraph` object. The resulting `igraph` object contains information about the size of the clusters (one cluster corresponding to one unit in the map) and the number of (or the sum of the weights of the) edges between vertices in two different clusters. An attribute `layout` is attached to the results, which positionned the nodes of the projected graph at the coordinates in $\mathbb{R}^2$ of the corresponding units on the map.

The same feature exists for the super-clusters obtained by the function `superClass`: in this case, the nodes of the projected graph are positioned at the center of gravity of the super-clusters

[6] available from the R package **igraph**, Csardi and Nepusz (2006).

on the map. This feature allows the user to obtain a simplified overview of its graph, as shown in the Section 4.

These graphs can be directly represented using the command line

```
plot(x, what="add", type="graph", ...)
```

with x an object of type somRes or with command line

```
plot(x, type="projgraph", ...)
```

with x the result of the superClass function.

A simple example of the way these functions can be used is provided in Section 4.1 on a toy example available in the package.

3.3. *Getting started with* SOMbrero

SOMbrero also provides five vignettes (HTML documentation files accessible from within the package or on the CRAN webpage) that detail the use of the package for different types of data. Three datasets, provided with **SOMbrero**, are used to illustrate the numeric case, the “KORRESP” case and the relational case. In particular, the dataset provided to illustrate the relational SOM is the graph described and studied in Section 4.1, which is derived from the novel “Les Misérables” by the French author Victor Hugo.

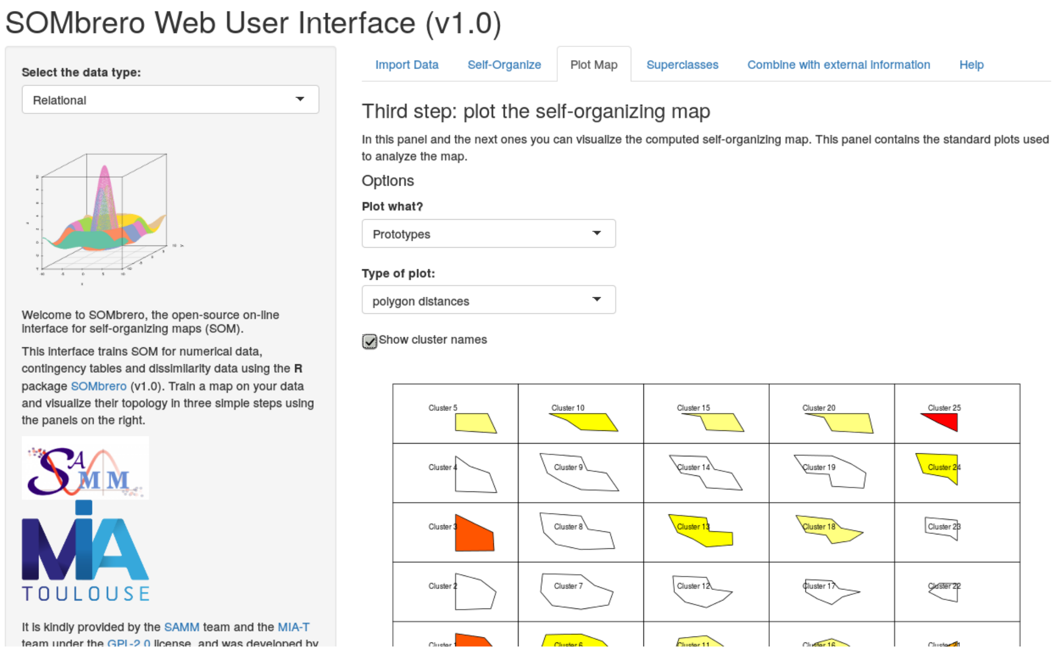
Moreover, **SOMbrero** comes with a user-friendly graphical interface, which makes most of its options available in a few clicks, without resorting to the command line. The interface is programmed using the R package **shiny** RStudio and Inc. (2013). It can be tested using a simple web browser (some of the features may not work with Internet Explorer or Chrome; Firefox must be preferred), and can be accessed on-line at `http://shiny.nathalievilla.org/sombrero`, as shown in Figure 1. The interface was also integrated in the package in order to allow **SOMbrero** users run it locally if preferred (for faster interactivity). Within the package, it may be accessed using the `sombreroGUI()` command.

The interface consists of seven panels: the left hand side panel allows the user to choose the type of SOM and gives general information and references. An “Import Data” panel is used to import a data file in csv or text format, and set formatting options for the importation. If the data are properly imported, a preview table is shown in this panel. The “Self-Organize” panel is used to select the SOM options and train the algorithm. The “Plot Map” panel provides the different graphical outputs implemented in **SOMbrero**. The “Superclasses” panel is used to compute and to display super-classes. The “Combine with external information” panel can be used to import additional data and to display them on the map and thus to combine the results of SOM with external information. Finally, the “Help” panel contains indications about how to use the interface.

4. Application

4.1. *Les Misérables*

The example addressed in this section is described in Knuth (1993) and comes from the novel “Les Misérables” by the French author Victor Hugo. The graph extracted from this novel is a graph



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

FIGURE 1. *Screenshot of the* **SOMbrero** *web user interface*

of co-appearance of the characters of the novel: the vertices of the graph are the 77 characters of the novel and the 254 edges stand for a co-appearance of the corresponding two characters in a same chapter of the novel. The edges are weighted by the number of co-appearances. The graph is available as a `igraph` object in the R package **SOMbrero**. **SOMbrero** also includes the dissimilarity matrix based on the shortest path lengths in the graph, which can be directly used with the relational SOM algorithm included in the package.

This section aims at providing a simple use case example of the R package **SOMbrero** to explain the differents steps that can lead to obtain a simplified and comprehensive representation of the graph. This use case example describes the different command lines and options using the functions in **SOMbrero**, that can be used to obtain typical outputs for a graph. Then, an insight on choices that can be made regarding the dissimilarity to use in a graph is provided. The differences between the different dissimilarities are illustrated in a comparative study.

4.1.1. Use case example

This section presents a use case example of the **SOMbrero** package with the data presented above. The use case is performed from the dissimilarity matrix included in the package, which is the shortest path length. The results shown in the current section are those included in the package's vignette that illustrates the use of the relational SOM. The map was built using a random initialization of the (γ_{ui}) between 0 and 1 (with a scaling so that for every u, γ_{ui} sums to 1) and functions d and H that correspond to the original implementation of the Patrick Letremy's SAS program. More precisely, H is a function of the form $H^t(r) = \begin{cases} 1 & \text{if } r \leq r_t \\ 0 & \text{otherwise.} \end{cases}$, where r_t

takes decreasing successive values from an original value r_0, depending on U and on the number of iterations, to 0. The default d is usually equal to the maximum distance between units u and u' with coordinates (u_x, u_y) and (u'_x, u'_y) so that

$$d(u,u') = \max\left(|u_x - u'_x|, |u_y - u'_y|\right),$$

except when $r_t = 0.5$, at which time d is chosen as the Euclidean distance (every unit has then 4 neighbors, except for the bordering units).

The data are referenced in the package under the name `lesmis`:

```
data(lesmis)
```

which gives access to two variables: `lesmis`, which is the graph itself, provided on the form of an `igraph` object, and the dissimilarity matrix `dissim.lesmis` based on the shortest path length between pairs of vertices. The map is then trained using the command line:

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

Only the dissimilarity matrix and the type of algorithm are mandatory for this function. Default parameters with this dataset would have been to initialize the dimension of the map to 5×5, the number of iterations to 385.

The quality of the map is checked using the function `quality`

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

# $quantization
# [1] 0.613031
```

and various types of graphics can be produced (see results in Figure 2):

```
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))
```

Here, the hitmap is shown, which is the default plot for the observations (hence the argument `type` does not have to be passed to the function `plot`) and displays a rectangle which area is proportional to the number of vertices clustered in every unit. The projected graph, as described in Section 2.3, is also given as well as the result of the clustering where the column names of the dissimilarity matrix (here, the characters' names) are displayed in their corresponding units.

At this step, due to the large number of clusters, the representation of the projected graph is still a bit messy and the clustering quality is not very good (the modularity is equal to 0.389). However, a clustering of the prototypes can be performed to improve it:

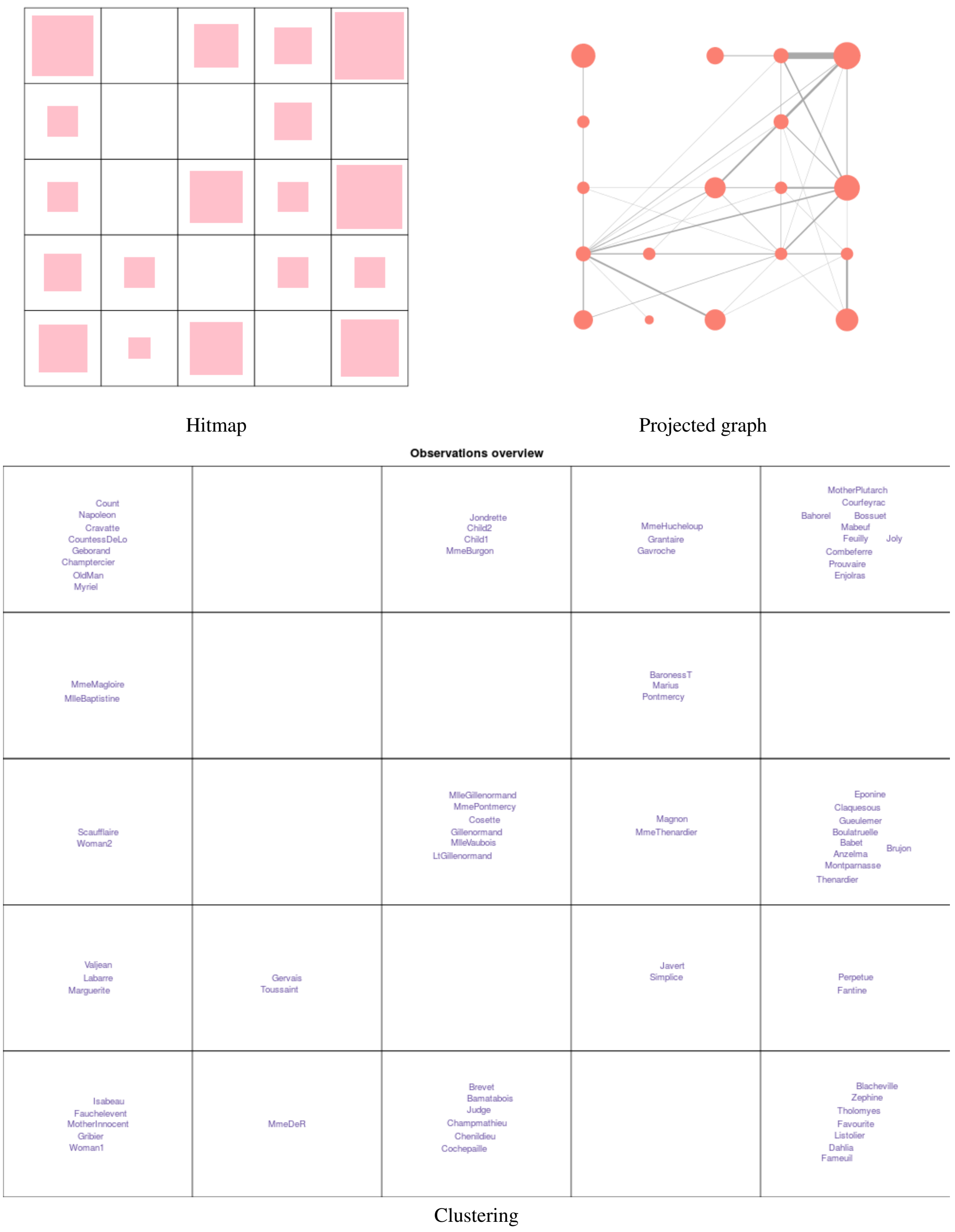


FIGURE 2. *Hitmap, projected graph and clustering on the map*

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

The results are illustrated in Figure 3. In particular, Figure (c) shows that the clusters obtained

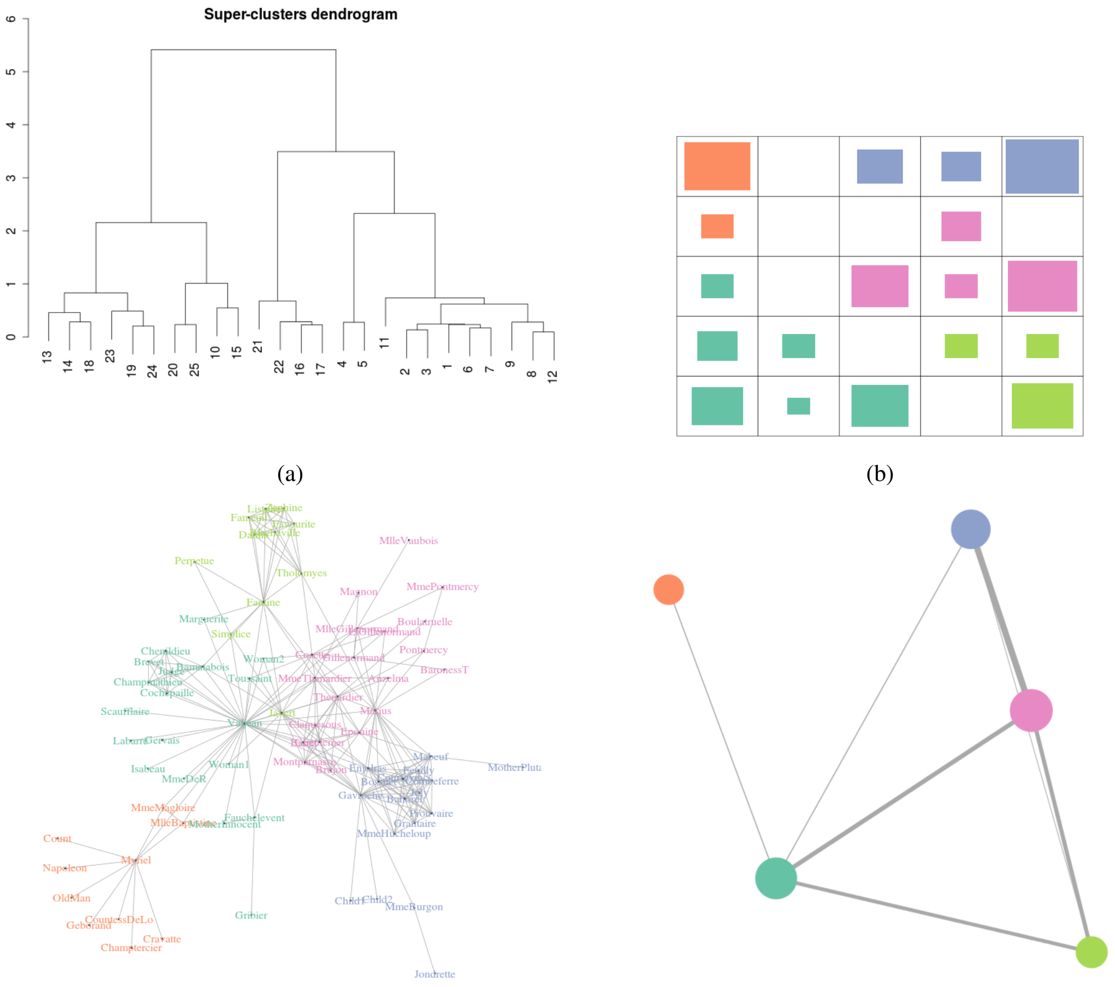


FIGURE 3. *(a) Dendrogram of the clustering of the prototypes: from this figure, we chose to select 6 super-clusters, (b) hitmap colored with the result of the super-clustering, (c) original graph with the result of the super-clustering, (d) projected graph using the super-clustering*

make sense (each one is associated to an important character of the novel : Valjean, Myriel, Fantine, Gavroche, Cosette, Javert and the Thenardier family) and the modularity is improved (0.529). In Figure (d), the super-clustering is used to obtain a final simplified representation of the graph in which all super-clusters are positioned at their gravity center on the map.

4.1.2. *Influence of the dissimilarity measure*

Let us now investigate the influence of the dissimilarity on the SOM performances. To do so, we have used different dissimilarities described in Section 2.4 with the graph of co-appearance from “Les Misérables”. The following dissimilarities were computed on this graph:

(a) the shortest path lengths, computed with the unweighted graph,
(b) the dissimilarity based on the weighted Laplacian eigen-decomposition with $C = 25$ (to be used with a 5×5 map),
(c) the squared distance induced from the commute time kernel which uses also the weighted Laplacian,
(d) the dissimilarity based on the modularity matrix eigen-decomposition, which also uses the weighted graphs.

The different dissimilarities are illustrated on the heatmaps of Figure 4.

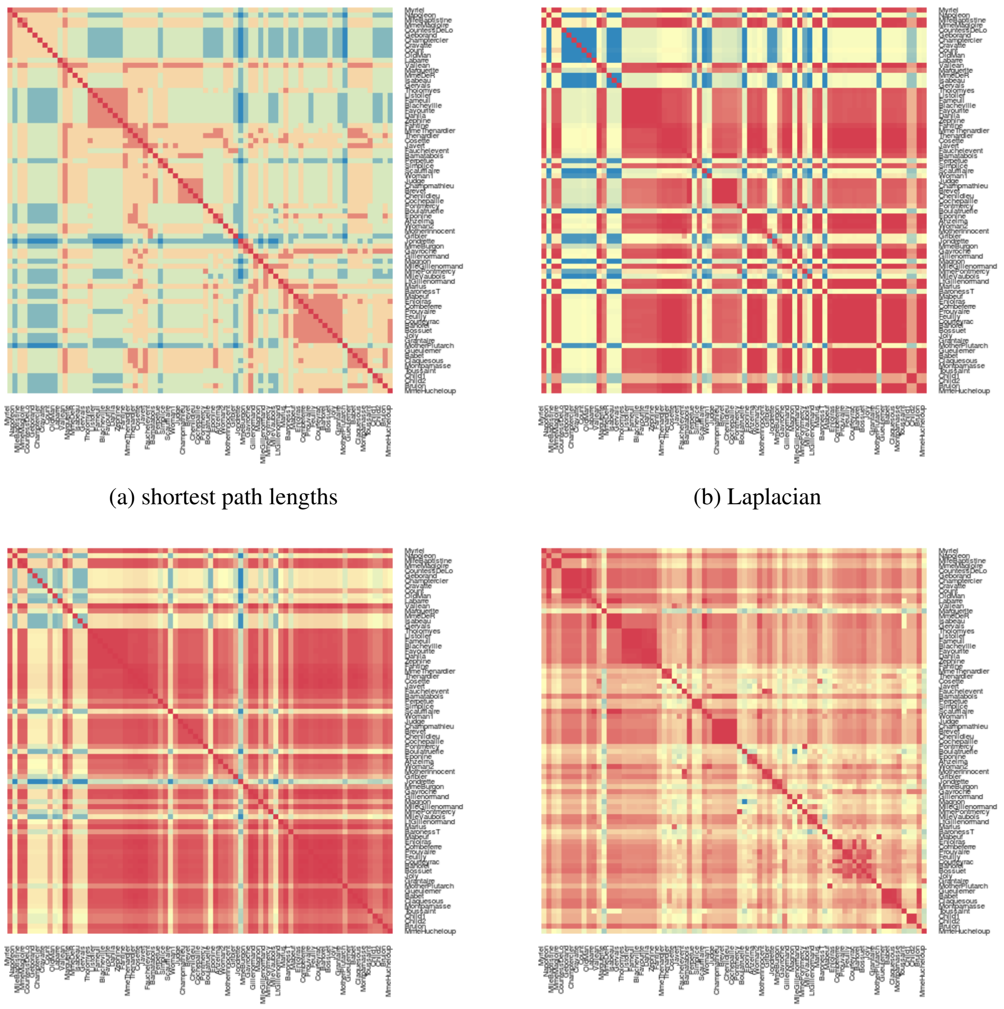
This figure shows that, as expected, the dissimilarity based on the eigen-decomposition of the Laplacian and the one computed from the commute time kernel present very similar patterns, the latter being slightly less contrasted than the former which uses only a part of the eigen-decomposition. The shortest path length (which is based on the unweighted graph, contrary to the other three) exibits rather different patterns and seems to have a larger number of distant vertices than the two previous. Finally, the dissimilarity based on the modularity matrix has exactly the opposite behaviour: very few pairs of vertices are considered as distant. Similar conclusions can be drawn from Figure 5 which illustrates the dissimilarities from Valjean’s point of view [7]. This figure shows that the dissimilarities are quite different except for the dissimilarity based on the eigen-decomposition of the Laplacian and the one computed from the commute time kernel which present similar patterns. In particular, the dissimilarity based on the modularity matrix provides results that are a bit strange to interpret, like the fact that “Old man” (bottom left of the graph) is “closer” (from this dissimilarity point of view) than “Toussain” who is directly connected to Valjean (middle of the graph), which indicates that this dissimilarity is maybe not very relevant for clustering vertices in this graph.

For every dissimilarity matrix, three different initializations of the SOM algorithm were tested, that are the three options available in **SOMbrero** (as described in Section 3.2). Then, for every dissimilarity and every type of initialization, $1,000$ maps were trained using the function `trainSOM` with a 5×5-map, 500 iterations and d and H as in the original implementation of the algorithm by Patrick Letremy. To test if a combination of dissimilarities corresponding to different features can improve the results, two combined dissimilarities were also computed as suggested in Olteanu and Villa-Vialaneix (2015):

$$D^{\text{sp+modularity}} = \frac{D^{\text{sp}}}{\|D^{\text{sp}}\|_F} + \frac{D^{\text{modularity}}}{\|D^{\text{modularity}}\|_F} \quad \text{and} \quad D^{\text{sp+spec. clust}} = \frac{D^{\text{sp}}}{\|D^{\text{sp}}\|_F} + \frac{D^{\text{spec. clust}}}{\|D^{\text{spec. clust}}\|_F} \tag{8}$$

with D^{sp} the dissimilarity matrix based on shortest path length, $D^{\text{modularity}}$ the dissimilarity matrix based on the eigen-decomposition of the modularity matrix, $D^{\text{spec. clust}}$ the dissimilarity matrix based on the eigen-decomposition of the Laplacian and $\|.\|_F$ the Frobenius norm to make the two combined dissimilarities have comparable scales.

[7] Jean Valjean is one of the main (or even the main) character of the novel.

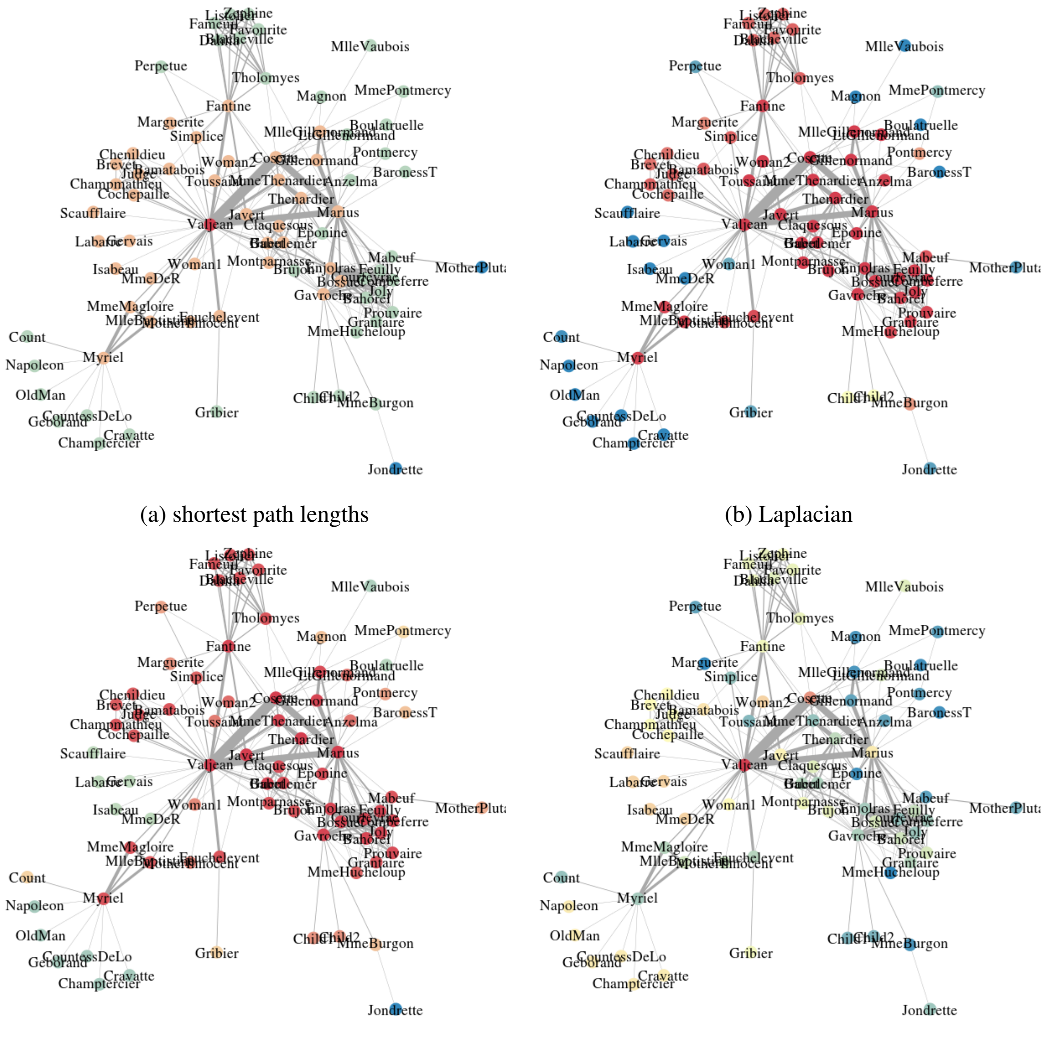


(a) shortest path lengths (b) Laplacian

(c) commute time kernel distance (d) modularity

FIGURE 4. *Heatmap of the different dissimilarities obtained from the graph "Les Misérables". Red corresponds to small (or null) dissimilarities, blue to large ones.*

The different results were compared using different quality measures, two of which being directly computed with the quality function in **SOMbrero**, the topographic error (which provides an insight on the good organization on the map) and the quantization error (which helps quantify the quality of the clustering). Note that these two quality measures are not really a fair criterion for comparing different dissimilarities as they both depend on the dissimilarity itself. However, the topographic error is an error rate and can thus hint which dissimilarities allow for a good topology preservation (*i.e.*, are well projected on the map). The quantization error, whose value is given in Equation (7), was divided by the Frobenius norm of the corresponding dissimilarity matrix to allow for similar scales between the different dissimilarities. Finally, an external clustering quality criterion was used, the modularity (Equation (6)). Two versions of the criterion were computed:


(a) shortest path lengths

(b) Laplacian

(c) commute time kernel distance

(d) modularity

FIGURE 5. *Graph coming from "Les Misérables" with vertex colors corresponding to the distances of the vertices from Valjean. Colors are chosen based on the range of distances from Valjean only: red corresponds to small (or null) dissimilarities, blue to large ones.*

one using the weighted graphs and one the unweighted graph (in this latter case, the weights W_{ij} were then equal to either 0 or 1).

The distribution of the different quality measures other the $1,000$ maps are given by dissimilarity and type of initialization in Figure 6. Additionally, Table 3 gives the average performance for the four quality measures whatever the initialization.

Several conclusions can be drawn from these results: first, the type of initialization does not seem to have a strong impact on the quality of the results, which is expected for a stochastic algorithm. Second, the maps obtained with the shortest path length are the best for the standard quality criteria for SOMs but are not good from the modularity perspective (especially the

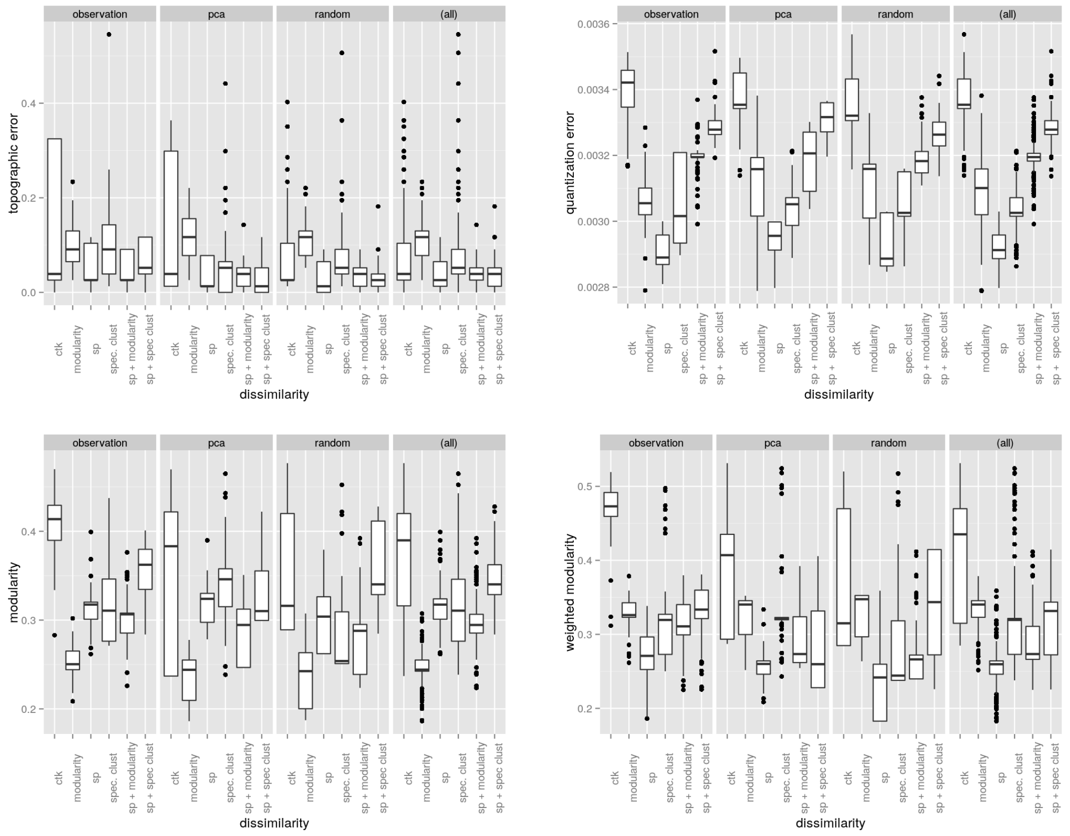


FIGURE 6. *Distribution of the topographic and quantization errors and of the modularity (based on the weighted or unweighted) graph, given by dissimilarity and type of initialization. "ctk" means "commute time kernel", "sp" means "shortest path length" and "spec. clust" stands for the dissimilarity based on the eigen-decomposition of the Laplacian. "observations" means random initialization so that, for every u,* $\gamma_{ui(u)} = 1$ *for a unique* $i(u)$*, "pca" is the initialization based on a PCA-like approach and "random" is the initialization in which* γ_{ui} *are sampled in* $[0,1]$ *and then scaled such that* $\sum_i \gamma_{ui} = 1$. *The last panel of each figure provides the distribution whatever the type of initialization (distribution over* $3,000$ *maps).*

weighted version, which is expected since this dissimilarity does not use the information on weights). On the contrary, the maps based on the dissimilarity computed from the commute time kernel exhibit rather poor topological preservation and quantization error as compared to the other dissimilarities but they have the best performances for the modularity (at the cost of a very large variability). This means that, since they depend on the dissimilarity itself which represents more or less faithfully the graph topology, the standard quality criteria for SOMs are probably not to be trusted too much to compare maps obtained from different dissimilarities. Surprisingly, the dissimilarity which is based on the modularity matrix did not outperform the others for the modularity criteria. This might be due to the fact that the matrix only uses the positive part of the modularity matrix spectrum, which may well be insufficient to grab most of the information

TABLE 3. *Average performance for four quality criterion over all* 3,000 *maps produced for each dissimilarity. "ctk" means "commute time kernel", "modularity" is the dissimilarity based on the modularity matrix eigen-decomposition, "sp" means "shortest path length" and "spec. clust" stands for the dissimilarity based on the eigen-decomposition of the Laplacian. The "+" indicates that the two dissimilarities have been combined as in Equation (8).*

dissimilarity	topographic error	quantization error	modularity	weighted modularity
ctk	0.100	0.00338	**0.369**	**0.406**
modularity	0.106	0.00310	0.242	0.330
sp	**0.037**	**0.00293**	0.309	0.253
spec. clust	0.065	0.00305	0.308	0.301
sp + modularity	0.038	0.00319	0.287	0.288
sp + spec. clust	0.039	0.00329	0.347	0.318

needed to optimize the modularity itself[8]. Also, the dissimilarity based on the modularity matrix had a very poor topographic preservation (which could be the consequence of uninterpretable values, as described above). Finally, the combined dissimilarities provide almost everywhere average performance, except for the quantization error which is among the worst (but this error is probably the most dependent from the values of the dissimilarity themselves and maybe not as reliable as the others).

4.2. Analysis of a bipartite metabolic network

The database used in this section comes from a reconstruction of the human metabolic network from the BiGG database (Schellenberger et al., 2010, `http://bigg.ucsd.edu`). This graph was already used for visualization purposes in Schulz et al. (2008) and its clustering is of interest as demonstrated in Hanisch et al. (2002), which uses similar data combined with expression data to cluster genes and improve their knowledge on their biological functions. We took a snapshot of the data on August 13rd, 2014 and extracted chemical reactions related to *Homo Sapiens* and located in the Cytosol, which corresponded to 957 reactions. A bipartite graph was defined from these data, in which:

- the vertices were the reactions or the elements involved in a reaction;
- the edges meant that one of the corresponding vertices (an chemical element) was involved in the other vertex (reaction).

The largest connected component of this graph contained 1832 vertices (949 reactions and 883 elements) and 4,504 edges. Additionally, contextual information was extracted from the data: the pathway of the chemical reaction (100 different pathways such as Nucleic acid degradation, Extracellular transport, Steroid Metabolism, ...).

The following methodology was used to produce a map from **SOMbrero**: first, the shortest path lengths between every pair of vertices were calculated and used for the dissimilarity matrix input in the `trainSOM` function. The algorithm was trained with a 10×10 map, 5,000 iterations and H and d set as in the original implementation by Patrick Letremy. 10 maps were obtained with different initializations corresponding to the default option $((\gamma_{ui(u)}) = 1$ for a unique $i(u)$ randomly chosen in $\{1,\ldots,n\})$). For every map, the topographic error and the quantization error

[8] Note that Rossi and Villa-Vialaneix (2010) has already observed that using the eigen-decomposition of the modularity matrix is not the best method for defining a kernel which should optimize the modularity.

TABLE 4. *Topographic error and quantization error for the chosen map and average over the 10 trained maps.*

	topographic error	quantization error
chosen map	0.288	1.40
average (10 maps)	0.370	1.40

were calculated. The final selected map was the one with the smallest averaged rank, where the average is taken between the rank on topographic error and the rank on quantization. The final map contained 98 clusters has the quantization and topographic errors given in Table 4, that can be compared to the average topographic and quantization errors over the 10 maps. Note that the expectation for a given unit to be in the neighborhood of another unit on the map is approximately equal to 8.3% which shows that the obtained topographic error fairly correct. The average dissimilarity over the pairs of nodes is equal to 3.9 (with a maximum at 13), which shows also that the quantization error (*i.e.*, the average dissimilarity between the nodes in a given unit and the unit's prototype) is also correct.

Additionally, quality criteria pertaining to the external information available on the chemical reactions were also calculated. More precisely, Table 5 provides the average unit purity and the normalized mutual information (Danon et al., 2005) with respect to the pathway (both quantities were calculated considering only the reactions).

TABLE 5. *Average node purity and normalized mutual information for the pathway for the chosen maps and mean for the 10 trained maps.*

	node purity	nmi
chosen map	53.8%	54%
average	49.9%	53.8%

An enrichment analysis for the pathway was performed for every unit and every pathway in the map. Fisher tests with Bonferroni correction for multiple tests were performed and units found to be significantly enriched in a given pathway were extracted (with risk 5%). 25 units (over 98 non-empty clusters on the map) were found to be significantly enriched in 15 different pathways, which shows a good consistency between the obtained map and the external information provided by the pathway, that was not used to train it.

A super-clustering of the prototypes was then performed leading to the dendrogram shown in Figure 7, from which we chose to keep 8 clusters. These clusters were all tested for pathway enrichment and 5 were found to be significantly enriched in one, two or three different pathways. Finally, Figure 8 shows the projected super-clusters on the map, labeled with enriched pathways. The modularity of this clustering is equal to 0.420 (to be compared with 0.537 and 0.372 which are, respectively the modularity found with the original approach of Newman and Girvan (2004) for 26 clusters and the modularity found using the eigen-decomposition of the modularity matrix as in Newman, 2006 with 5 clusters). However, our approach offers an *a priori* representation of the clusters' positions and should provide useful hints on the graph structure. For instance, Figure 8 shows a very central cluster enriched in "Caruitine shuttle" and "Fatty acid activation" that has strong links with almost all the other clusters. On the contrary, the bottom left cluster enriched in "Tyrosine metabolism" and "Tryptophan metabolism" is rather isolated and not connected with the rest of the graph and must be a very coherent set of reactions associated to these two peculiar functions.

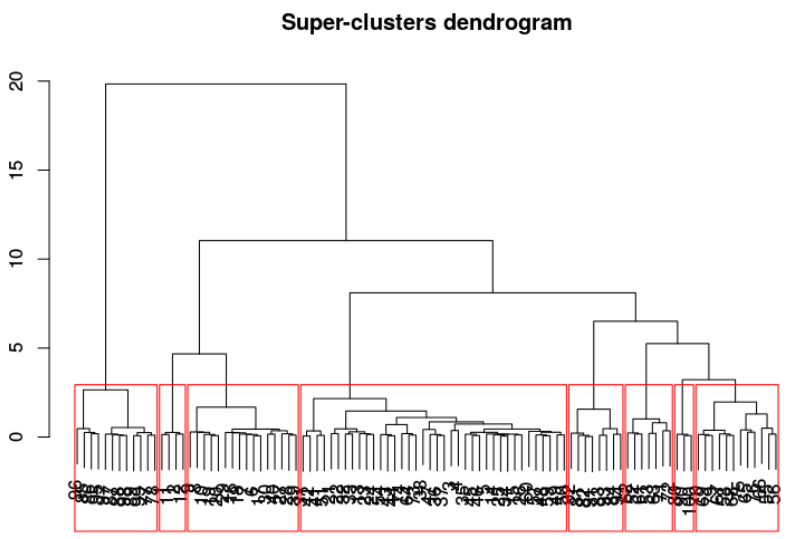


FIGURE 7. *Dendrogram of the super-clustering of prototypes for the chemical reaction bipartite graph. 8 super-clusters were extracted.*

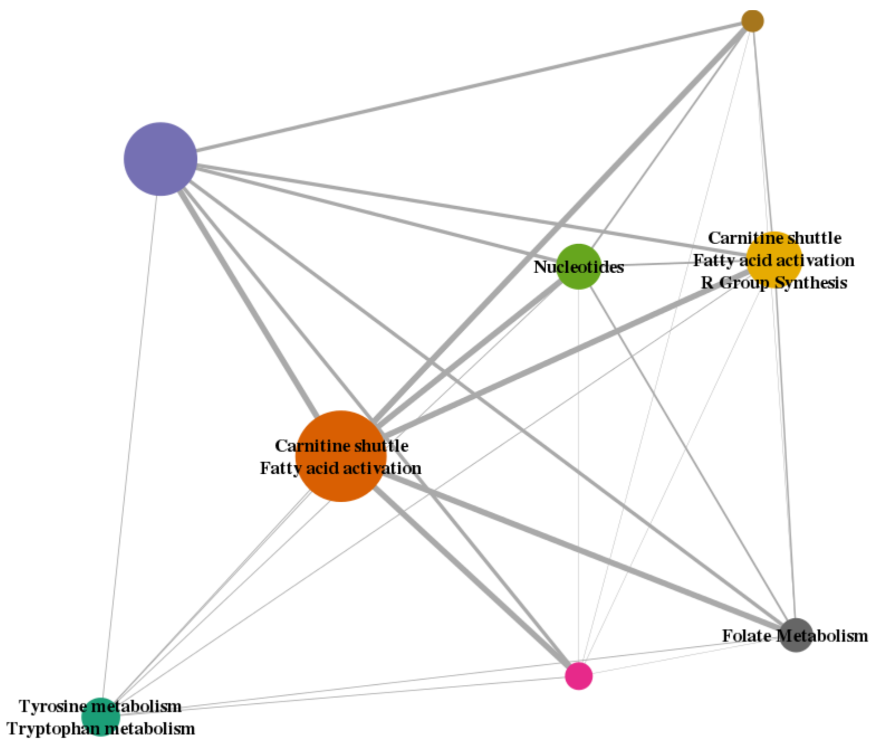


FIGURE 8. *Projected super-clusters at centers of gravity of their clusters labeled with enriched pathways.*

5. Conclusion

In this article, we presented an R package for non-vectorial Self-Organizing Maps, **SOMbrero**. We showed that the relational SOM implemented in the package can be useful to obtain a simplified representation of the graph, based on a clustering. We provided a simple use case of the package, whilst investigating several dissimilarities that can be used for this task. We found that different types of dissimilarities can be used depending on the objectives, the shortest path lengths being the one that seems to give the best performances from the point of view of standard SOM's quality measures. Finally, we have proposed to use super-clustering to provide a simple representation of a clustered graph using the map structure.

References

Ambroise, C. and Govaert, G. (1996). Analyzing dissimilarity matrices via Kohonen maps. In *Proceedings of 5th Conference of the International Federation of Classification Societies (IFCS 1996)*, volume 2, pages 96–99, Kobe (Japan).

Anderson, M. (2001). A new method for non-parametric multivariate analysis of variance. *Austral Ecology*, 26:32–46.

Andras, P. (2002). Kernel-Kohonen networks. *International Journal of Neural Systems*, 12:117–135.

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404.

Ben-Hur, A. and Weston, J. (2010). *Data Mining Techniques for the Life Sciences*, volume 609 of *Methods in Molecular Biology*, chapter A user's guide to support vector machine, pages 223–239. Springer-Verlag.

Boulet, R., Jouve, B., Rossi, F., and Villa, N. (2008). Batch kernel SOM and related Laplacian methods for social network analysis. *Neurocomputing*, 71(7-9):1257–1273.

Bourgeois, N., Cottrell, M., Lamassé, S., and Olteanu, M. (2015). Search for meaning through the study of co-occurrences in texts. Submitted for publication in proceedings of IWANN 2015.

Conan-Guez, B., Rossi, F., and El Golli, A. (2006). Fast algorithm and implementation of dissimilarity self-organizing maps. *Neural Networks*, 19(6-7):855–863.

Cottrell, M., Letrémy, P., and Roy, E. (1993). Analyzing a contingency table with Kohonen maps: a factorial correspondence analysis. In Cabestany, J., Mary, J., and Prieto, A. E., editors, *Proceedings of International Workshop on Artificial Neural Networks (IWANN 93)*, Lecture Notes in Computer Science, pages 305–311. Springer Verlag.

Csardi, G. and Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal*, Complex Systems.

Danon, L., Diaz-Guilera, A., Duch, J., and Arenas, A. (2005). Comparing community structure identification. *Journal of Statistical Mechanics*, page P09008.

Eades, P. and Huang, M. (2000). Navigating clustered graphs using force-directed methods. *Journal of Graph Algorithms and Applications*, 4(3):157–181.

Fort, J., Letremy, P., and Cottrell, M. (2002). Advantages and drawbacks of the batch kohonen algorithm. In Verleysen, M., editor, *Proceedings of 10th European Symposium on Artificial Neural Networks (ESANN 2002)*, pages 223–230, Bruges, Belgium.

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.

Fruchterman, T. and Reingold, B. (1991). Graph drawing by force-directed placement. *Software, Practice and Experience*, 21:1129–1164.

Goldfarb, L. (1984). A unified approach to pattern recognition. *Pattern Recognition*, 17(5):575–582.

Graepel, T. and Obermayer, K. (1999). A stochastic self-organizing map for proximity data. *Neural Computation*, 11(1):139–155.

Hammer, B. and Hasenfuss, A. (2010). Topographic mapping of large dissimilarity data sets. *Neural Computation*, 22(9):2229–2284.

Hammer, B., Hoffman, D., Schleif, F., and Zhu, X. (2014). Learning vector quantization for (dis-)similarities. *Neurocomputing*, 131.

Hanisch, D., Zien, A., Zimmer, R., and Lengauer, T. (2002). Co-clustering of biological networks and gene expression data. *Bioinformatics*, 18(Suppl. 1):S145–S154.

Harel, D. and Koren, Y. (2002). Drawing graphs with non-uniform vertices. In *Proceedings of the Working Conference on Advanced Visualization Interfaces (AVI'02)*, pages 157–166, New York, NY, USA. ACM Press.

Herman, I., Melançon, G., and Scott Marshall, M. (2000). Graph visualization and navigation in information visualisation. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43.

Heskes, T. (1999). Energy functions for self-organizing maps. In Oja, E. and Kaski, S., editors, *Kohonen Maps*, pages 303–315. Elsevier, Amsterdam.

Knuth, D. (1993). *The Stanford GraphBase: A Platform for Combinatorial Computing*. Addison-Wesley, Reading, MA.

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.

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.

Krislock, N. and Wolkowicz, H. (2012). *Handbook on Semidefinite, Conic and Polynomial Optimization*, volume 166 of *International Series in Operations Research & Management Science*, chapter Euclidean distance matrices and applications, pages 879–914. Springer, New York, Dordrecht, Heidelberg, London.

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.

RStudio and Inc. (2013). *shiny: Web Application Framework for R*. R package version 0.6.0.

Newman, M. (2006). Finding community structure in networks using the eigenvectors of matrices. *Physical Review, E*, 74(036104).

Newman, M. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review, E*, 69:026113.

Noack, A. (2007). Energy models for graph clustering. *Journal of Graph Algorithms and Applications*, 11(2):453–480.

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.

Rossi, F. (2014). How many dissimilarity/kernel self organizing map variants do we need? In Villmann, T., Schleif, F., Kaden, M., and Lange, M., editors, *Advances in Self-Organizing Maps and Learning Vector Quantization (Proceedings of WSOM 2014)*, volume 295 of *Advances in Intelligent Systems and Computing*, pages 3–23, Mittweida, Germany. Springer Verlag, Berlin, Heidelberg.

Rossi, F. and Villa-Vialaneix, N. (2010). Optimizing an organized modularity measure for topographic graph clustering: a deterministic annealing approach. *Neurocomputing*, 73(7-9):1142–1163.

Rossi, F. and Villa-Vialaneix, N. (2011). Représentation d'un grand réseau à partir d'une classification hiérarchique de ses sommets. *Journal de la Société Française de Statistique*, 152(3):34–65.

Schellenberger, J., Park, O., Conrad, T., and Palsson, B. (2010). BiGG: a biochemical genetic and genomic knowledgebase of large scale metabolic reconstructions. *BMC Bioinformatics*, 11:213.

Schoenberg, I. (1935). Remarks to Maurice Fréchet's article "Sur la définition axiomatique d'une classe d'espace distanciés vectoriellement applicable sur l'espace de Hilbert". *Annals of Mathematics*, 36:724–732.

Schulz, H., John, M., Unger, A., and Schumann, H. (2008). Visual analysis of bipartite biological networks. In Botha, C., Kindlmann, G., Niessen, W., and Preim, B., editors, *Proceedings Eurographics Workshop on Computing for Biomedicine, VCBM 2008*, Delft, The Nederlands.

Smola, A. and Kondor, R. (2003). Kernels and regularization on graphs. In Warmuth, M. and Schölkopf, B., editors, *Proceedings of the Conference on Learning Theory (COLT) and Kernel Workshop*, Lecture Notes in Computer Science, pages 144–158.

Tunkelang, D. (1999). *A Numerical Optimization Approach to General Graph Drawing*. PhD thesis, School of Computer Science, Carnegie Mellon University. CMU-CS-98-189.

von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416.

Wang, X. and Miyamoto, I. (1996). Generating customized layouts. In Brandenburg, F., editor, *Graph Drawing*, volume 1027 of *Lecture Notes in Computer Science*, pages 504–515. Springer (Berlin/Heidelberg).

Young, G. and Householder, A. (1938). Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3:19–22.