Apprentissage statistique pour l'intégration de données omiques

Jérôme J. Mariette

Université Fédérale
Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *15 décembre 2017* **par :**

Jérôme MARIETTE

Apprentissage statistique pour l'intégration de données omiques

JURY

Christophe AMBROISE	Université Evry Val d'Essonne	Rapporteur
Gwennaele FICHANT	Université Paul Sabatier	Examinatrice
Christine GASPIN	INRA, Toulouse	Codirectrice de thèse
Mahendra MARIADASSOU	INRA, Jouy-en-Josas	Examinateur
Michel VERLEYSEN	Université de Louvain	Rapporteur
Nathalie VILLA-VIALANEIX	INRA, Toulouse	Directrice de thèse

École doctorale et spécialité :
SEVAB : Écologie, biodiversité et évolution
Unité de Recherche :
Unité de Mathématiques et Informatique Appliquées de Toulouse, 875 INRA
Directrices de Thèse :
Nathalie VILLA-VIALANEIX & Christine GASPIN

Remerciements

La tradition exige de commencer les remerciements par ses directeurs-trices de thèse. Mais, si je tiens à vous remercier Nathalie et Christine avant toute autre personne, ce n'est pas pour des raisons d'usage, mais bien pour tout ce que vous m'avez apporté. Je tiens à te remercier Nathalie pour tes qualités pédagogiques et scientifiques, ta franchise, ton énergie et ta patience qui ont été plus que nécessaire pour former un apprenti-chercheur formaté au travail d'ingénieur. J'ai beaucoup appris à tes côtés et je te suis reconnaissant de m'avoir offert l'opportunité de travailler sur des problématiques novatrices et motivantes.

Je ne pourrais jamais te remercier assez Christine de m'avoir encouragé à me lancer dans cette aventure. Merci pour tes conseils et la confiance que tu m'as accordée, aussi bien dans le cadre de cette thèse que durant ces huit années à travailler à tes côtés. Tes qualités humaines qui te poussent à penser au bien être de tes agents, ton implication pour le collectif et ton enthousiasme sont motivants pour bon nombre d'entre nous.

Je veux aussi remercier chaleureusement Christophe Ambroise et Michel Verleysen d'avoir accepté de relire cette thèse et d'en être rapporteur. Merci également aux autres membres du jury : Gwenaelle Fichant et Mahendra Mariadassou, pour avoir accepté de faire partie du jury de soutenance et pour l'intérêt qu'ils ont porté à mon travail. Je tiens aussi à remercier les membres de mon comité de thèse : Marie-Agnès Dillies, Robert Faivre, Marie-Laure Martin-Magnette, Fabrice Rossi et Matthias Zytnicki pour leurs conseils qui ont été pour moi très enrichissants.

Je souhaiterais également exprimer ma gratitude à Frédéric Garcia, ancien directeur du département MIA, Christine Cierco-Ayrolles, des ressources humaines du département MIA, et Christophe Klopp, résponsable opérationnel de Genotoul bioinfo de m'avoir permis de dégager 50% de mon temps ingénieur pour réaliser ce travail doctoral.

Bien sûr, la réalisation de ce travail n'aurait pas été possible sans l'appui de mon équipe et de mon unité, MIA-T. Merci à tous de m'offrir depuis ces huit années un cadre scientifique épanouissant et un environnement bienveillant. Je remercie particulièrement Céline Noirot pour nos discussions et confrontations qui ont sûrement été la clé de notre travail commun, mais aussi Claire Hoede pour son enthousiasme communicatif, Gaelle Lefort dont les qualités de statisticienne n'ont d'égales que celles de ses muffins, Marie-Stephane Trotard pour ses débats militants enflammés et Didier Laborie pour sa bonne humeur sans faille. Je voudrais aussi remercier Fabienne Ayrignac, Natalie Julliand, Alain Perault et notre chef Sylvain Jasson pour leur efficacité dans la résolution des problèmes administratifs.

Je suis également très reconnaissant envers Fabrice Rossi et Madalina Olteanu pour leur accueil lors de ma visite au sein de l'équipe SAMM de l'université Paris 1. Je vous remercie pour cette expérience qui a été une source de rencontres et d'échanges stimulants. Je tiens aussi à remercier Kim-Anh Le Cao de l'Université de Melbourne pour ses conseils avisés mais aussi pour sa recette de granola et de dhal de lentilles corail.

Merci aussi à mes amis grimpeurs et traileurs qui m'ont permis de m'oxygéner durant ces quatre années et à mes plus proches ami-e-s pour leurs soutiens et leurs encouragements. Merci à ma famille de m'avoir appris la persévérance et l'exigence. Et enfin, merci à toi Olatz de m'avoir soutenu, supporté et encouragé. Cette thèse et moi te devons beaucoup. Merci.

Résumé

Les avancées des nouvelles techniques de séquençage ont permis de produire des données hétérogènes, volumineuse, de grande dimension et à différentes échelles du vivant. L'intégration de ces différentes données représente un défi en biologie des systèmes, défi qu'il est critique d'aborder pour tirer le meilleur parti possible de l'accumulation d'informations biologiques pour leur interprétation et leur exploitation dans un but finalisé.

Cette thèse regroupe plusieurs contributions méthodologiques utiles à l'exploration simultanée de plusieurs jeux de données omiques de natures hétérogènes. Pour aborder cette question, les noyaux et les méthodes à noyaux offrent un cadre naturel, car ils permettent de prendre en compte la nature propre de chacun des tableaux de données tout en permettant leur combinaison. Toutefois, lorsque le nombre d'observations à traiter est grand, les méthodes à noyaux souffrent d'un manque d'interprétabilité et d'une grande complexité algorithmique.

Une première partie de mon travail a porté sur l'adaptation de deux méthodes exploratoires à noyaux : l'analyse en composantes principales (K-PCA) et les cartes auto-organisatrices (K-SOM). Les adaptations développées portent d'une part sur le passage à l'échelle du K-SOM et de la K-PCA au domaine des omiques et d'autre part sur l'amélioration de l'interprétabilité des résultats. Dans une seconde partie, je me suis intéressé à l'apprentissage multi-noyaux pour combiner plusieurs jeux de données omiques. L'efficacité des méthodes proposées est illustrée dans le contexte de l'écologie microbienne : huit jeux de données du projet *TARA oceans* ont été intégrés et analysés à l'aide d'une K-PCA.

Abstract

The development of high-throughput sequencing technologies has lead to produce high dimensional heterogeneous datasets at different living scales. To process such data, integrative methods have been shown to be relevant, but still remain challenging.

This thesis gathers methodological contributions useful to simultaneously explore heterogeneous multi-omics datasets. To tackle this problem, kernels and kernel methods represent a natural framework because they allow to handle the own nature of each datasets while permitting their combination. However, when the number of sample to process is high, kernel methods suffer from several drawbacks : their complexity is increased and the interpretability of the model is lost.

A first part of my work is focused on the adaptation of two exploratory kernel methods : the principal component analysis (K-PCA) and the self-organizing map (K-SOM). The proposed adaptations first address the scaling problem of both K-SOM and K-PCA to omics datasets and second improve the interpretability of the models. In a second part, I was interested in multiple kernel learning to combine multiple omics datasets. The proposed methods efficiency is highlighted in the domain of microbial ecology : eight *TARA* oceans datasets are integrated and analysed using a K-PCA.

Organisation de la thèse

Cette thèse est rédigée en français à l'exception des chapitres correspondant à des articles publiés ou en cours de publication.

— Les trois chapitres introductifs sont rédigés en français ;

— Le chapitre 4 a fait l'objet d'une communication orale au $11^{ème}$ workshop WSOM 2016 à Houston aux Etats-Unis, [105] ;

— Le chapitre 5 a fait l'objet d'une communication orale au XXV^{ème} symposium européen ESANN à Bruges en Belgique, [107] ;

— Le chapitre 6 a fait l'objet d'une communication orale au $10^{ème}$ workshop WSOM à Mittweida en Allemagne, [108] ;

— Le chapitre 7 a fait l'objet d'une publication dans le journal *Neurocomputing*, [109] ;

— Le chapitre 8 a fait l'objet d'une publication dans le journal *Bioinformatics*, [106].

Table des matières

I Introduction 16

1 Contexte biologique 18
- 1.1 Séquençage haut-débit en écologie microbienne . . . 18
  - 1.1.1 Séquençage haut-débit et données omiques . . . 18
  - 1.1.2 Métagénomique et métagénétique . . . 18
  - 1.1.3 Analyses bio-informatiques . . . 19
- 1.2 Analyses statistiques en écologie microbienne . . . 21
  - 1.2.1 Normalisation . . . 21
  - 1.2.2 Analyse de la biodiversité . . . 22
  - 1.2.3 Exploration de la structure de populations microbiennes . . . 24

2 Cadre statistique 28
- 2.1 Les noyaux . . . 28
  - 2.1.1 Définition générale . . . 28
  - 2.1.2 L'astuce noyau . . . 28
  - 2.1.3 Intérêts pratiques des noyaux . . . 28
  - 2.1.4 Le cas des données de dissimilarité . . . 29
- 2.2 Analyses exploratoires et noyaux . . . 30
  - 2.2.1 Classification non supervisée à noyau . . . 30
  - 2.2.2 Analyse en composantes principales . . . 30
  - 2.2.3 Cartes auto-organisatrices. . . . 32
- 2.3 Méthodes pour le traitement de données massives . . . 35
- 2.4 Intégration de données omiques . . . 36

3 Contributions 40
- 3.1 SOM, K-SOM et données massives . . . 40
  - 3.1.1 Améliorer la stabilité du SOM numérique . . . 40
  - 3.1.2 Améliorer la complexité du K-SOM . . . 40
  - 3.1.3 Améliorer la complexité et l'interprétabilité du K-SOM . . . 40
- 3.2 Analyse exploratoire multi-omiques . . . 41
  - 3.2.1 Méthodes multi-noyaux non supervisées . . . 42
  - 3.2.2 Améliorer l'interprétabilité de la K-PCA . . . 43

II SOM, K-SOM et données massives 46

4 Aggregating Self-Organizing Maps with Topology Preservation 48
- 4.1 Introduction . . . 48
- 4.2 An overview of aggregation methods for SOMs . . . 48
- 4.3 Description of the optimal transformation method . . . 49
- 4.4 Simulations . . . 51
- 4.5 Conclusion . . . 54

5 Accelerating stochastic kernel SOM **56**
5.1 Introduction . . . 56
5.2 A brief description of the stochastic K-SOM . . . 56
5.3 Reducing the complexity of stochastic K-SOM . . . 57
5.4 The case of dissimilarity data . . . 57
5.5 Application . . . 58
5.6 Conclusion . . . 59

6 Bagged kernel SOM **62**
6.1 Introduction . . . 62
6.2 Method . . . 62
6.2.1 A brief description of the kernel SOM approach . . . 62
6.2.2 Ensemble of SOMs . . . 63
6.2.3 Bagged kernel SOM . . . 64
6.3 Applications . . . 65
6.4 Conclusion . . . 67

7 Efficient interpretable variants of online SOM for large dissimilarity data **70**
7.1 Introduction . . . 70
7.1.1 State-of-the art on SOM for (dis)similarity data . . . 70
7.1.2 Review of methods for large datasets and high dimensional datasets . . . 71
7.1.3 Kernel/relational extensions for large datasets . . . 72
7.1.4 Contributions of the article . . . 72
7.2 Kernel SOM (K-SOM) . . . 73
7.3 A dimension reduction technique for K-SOM . . . 74
7.3.1 Kernel PCA (K-PCA) . . . 74
7.3.2 K-PCA SOM . . . 75
7.3.3 K-PCA from Nyström approximation . . . 76
7.4 Direct sparse K-SOM . . . 77
7.4.1 Description of the approach . . . 78
7.4.2 Variants of the sparse updates . . . 80
7.5 The case of dissimilarity data . . . 80
7.6 Experimental results . . . 81
7.6.1 Methodology . . . 81
7.6.2 Evaluation of the different sparse approaches on various datasets . . . 82
7.6.3 Using K-PCA SOM and sparse K-SOM for mining job trajectories . . . 88
7.7 Conclusion . . . 95
7.8 Appendix : formula for the average intra-cluster intertia . . . 95

III Analyse exploratoire multi-omiques **98**

8 Integrating TARA Oceans datasets using unsupervised multiple kernel learning **100**
8.1 Introduction . . . 100
8.2 Methods . . . 101
8.2.1 Unsupervised multiple kernel learning . . . 101
8.2.2 Kernel PCA (KPCA) and enhanced interpretability . . . 104
8.2.3 Unsupervised multiple kernel and KPCA in **mixOmics** . . . 105
8.3 Implementation on *TARA* Oceans datasets . . . 105
8.3.1 Overview on *TARA* Oceans . . . 105
8.3.2 Selected samples . . . 106
8.3.3 Dissimilarities and kernels for *TARA* Oceans datasets . . . 106

8.4 Results and discussion 108
8.4.1 Similarities between kernels 108
8.4.2 Comparison of the different integration options 109
8.4.3 Proof of concept with a restricted number of datasets 109
8.4.4 Integrating environmental, prokaryotic, eukaryotic and viral datasets 112
8.5 Conclusion 115

9 Conclusion et perspectives **120**

Bibliographie **121**

Introduction

Chapitre 1
Contexte biologique

Les avancées technologiques, faites dans le domaine du séquençage haut-débit, permettent aujourd'hui de produire de grands volumes de données biologiques à moindre coût, et ceci à différentes échelles du vivant. Les données ainsi obtenues, appelées données omiques, sont à la fois de grandes dimensions, de sources multiples (différents omiques) et de natures hétérogènes (numériques, réseaux, facteurs, ...). Dans un tel contexte, les besoins en méthodes intégratives sont de plus en plus pressants afin de considérer un système biologique comme un tout, *i.e.*, comme un ensemble de signatures moléculaires entrelacées contenant gènes, protéines, transcrits et régulateurs, mais aussi des caractéristiques épigénétiques et des corrélations avec le microbiome. Cependant, l'étude des relations et des interactions existant entre différents omiques représente encore aujourd'hui un défi majeur.

Les approches proposées dans le cadre de cette thèse cherchent à prendre en compte l'hétérogénéité des données omiques tout en répondant au besoin en méthodes intégratives dans le domaine. Pour cela, les travaux présentés s'inscrivent dans le contexte des noyaux et des méthodes à noyaux. Toutefois, ces méthodes souffrent d'un manque d'interprétabilité de leurs modèles et d'une forte complexité algorithmique lorsque le nombre d'observations à traiter est grand. Les adaptations développées durant cette thèse se sont concentrées sur deux méthodes exploratoires : l'analyse en composantes principales et les cartes auto-organisatrices et sont illustrées dans le contexte des données omiques et plus particulièrement de l'écologie microbienne.

La première section de ce chapitre présente le séquençage haut-débit et ses applications en écologie microbienne. Le traitement statistique, les mesures de biodiversités ainsi que les analyses exploratoires les plus souvent utilisées en écologie microbienne sont introduits dans la deuxième section.

1.1 Séquençage haut-débit en écologie microbienne

1.1.1 Séquençage haut-débit et données omiques

Les techniques de séquençage, qu'elles soient haut-débit ou non, ont pour objectif de déterminer l'enchaînement en nucléotides d'un fragment d'acide désoxyribonucléique (ADN). Les données produites sont appelées lectures ou encore séquences. Elles sont représentées sous la forme d'une chaîne de caractères dont le dictionnaire est composé des quatre lettres correspondant aux quatre nucléotides formant l'ADN : A pour adénine, C pour cytosine, T pour thymine et G pour guanine.

Depuis la fin du séquençage du premier génome humain en 2003, des progrès considérables ont été réalisés dans ce domaine. L'évolution de ces technologies, retracée dans [64], a permis d'augmenter considérablement la quantité de données générées tout en diminuant les coûts de production. Ainsi, en 2013 la société Illumina proposait de séquencer n'importe quel génome humain pour 1 000 dollars en quelques heures, ce qui avait nécessité deux milliards d'euros et quinze années de travail 25 ans plus tôt. L'apport de ces avancées technologiques ne s'arrête pas là : la diversification des protocoles de séquençage permet aujourd'hui d'étudier un organisme à différentes échelles biologiques. Des protocoles existent pour étudier le génome, ensemble des gènes d'un organisme, le transcriptome, ensemble des ARNs transcrits, mais aussi l'épigénome, ensemble des mécanismes moléculaires qui modulent l'expression du patrimoine génétique. Les données ainsi produites font partie de la famille dite des omiques dans laquelle on retrouve aussi les données décrivant le protéome, ensemble des protéines exprimées, ou encore le métabolome, ensemble des métabolites.

Ces avancées technologiques ont révolutionnés de nombreux domaines d'applications, tels que l'écologie microbienne, en permettant l'étude du contenu génétique d'échantillons issus d'environnements complexes.

1.1.2 Métagénomique et métagénétique

Les micro-organismes regroupent l'ensemble des êtres vivants microscopiques, *i.e.*, non visible à l'œil nu, allant des bactéries aux eucaryotes (champignons, animaux et plantes) tels que le plancton. Ils jouent un rôle vital dans le fonctionnement général de la biosphère en participant, par exemple, au cycle du carbone ou encore au cycle de l'azote. Cependant, leur diversité spécifique et fonctionnelle, les mécanismes régissant leur dispersion ainsi que leur histoire évolutive demeurent encore mal connus. L'essor spectaculaire des techniques de séquençage a permis d'approfondir ces questions en donnant accès à une fraction des micro-organismes restée jusqu'alors inaccessible par les méthodes culturales [76].

La métagénomique et la métagénétique sont deux approches permettant l'étude de métagénomes, c'est à dire de l'information génétique de l'ensemble des micro-organismes vivant au sein d'un environnement spécifique. Cet ensemble d'organismes est appelé microbiote, et le milieu dans lequel ils évoluent le microbiome. Ainsi, contrairement à la génomique qui consiste à étudier un unique génome, la métagénomique, illustrée dans la figure 1.1 (a), extrait, fragmente et séquence l'ADN de l'ensemble des génomes des micro-organismes présents dans un milieu donné. La métagénétique, illustrée dans la figure 1.1 (b), sélectionne tout d'abord un gène cible avant de séquencer une région amplifiée par PCR (Polymerase Chain Reaction). À titre d'exemple, certaines régions spécifiques du gène de l'ARN ribosomique 16S sont classiquement utilisées en bactériologie. L'intérêt que suscite ce gène vient du fait qu'il est présent uniquement chez les bactéries, mais aussi parce qu'il possède des régions constantes, communes entre plusieurs espèces, et des régions variables permettant de différencier les bactéries entre elles.

Bien que la métagénomique offre une vue complète d'un microbiote, elle reste encore aujourd'hui une stratégie coûteuse et difficile à mettre en place de part le volume et la complexité des données

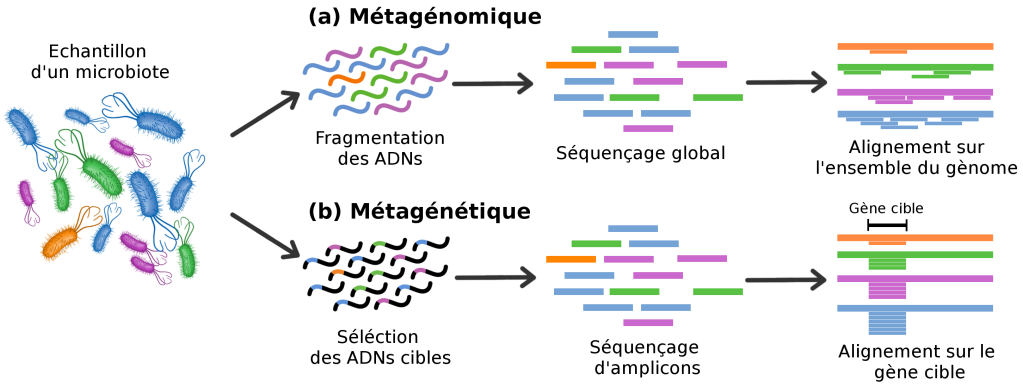

Figure 1.1 **Stratégies de séquençage de métagénomes.** La métagénomique (a) fragmente l'ensemble des ADNs présents dans un échantillon en courts fragments afin de les séquencer. Les séquences ainsi obtenues peuvent provenir de n'importe quel locus du génome. La métagénétique (b) sélectionne des gènes cibles avant de les séquencer. Les séquences représentent seulement les gènes ciblés.

générées. Pour cela, nous nous focaliserons dans la suite de ce chapitre sur la métagénétique, stratégie utilisée en routine en écologie microbienne.

Ces dernières années, de nombreux projets ont eu recours au séquençage massif de centaines de métagénomes. Parmi ceux-ci, on peut citer le projet metaHIT (*METAgenomics of the Human Intestinal Tract*), [49], HMP, (*Human Microbiome Project*), [36] ou encore l'expédition *TARA oceans*, [80, 17]. Les deux premiers projets, metaHIT et HMP, cherchent à explorer le microbiome humain afin de mieux comprendre les relations existant entre le microbiote intestinal et la santé humaine. L'expédition *TARA oceans*, quant à elle, se concentre sur l'étude des micro-organismes vivant dans les différents océans du globe afin d'étudier l'effet des variations environnementales sur ces populations. Pour cela, durant trois ans, les scientifiques du projet ont collecté de nombreux échantillons génomiques tout en effectuant des relevés des conditions physico-chimiques dans lesquelles évoluent les organismes.

1.1.3 Analyses bio-informatiques

Suite au séquençage des différents échantillons, les lectures produites sont pré-traitées pour supprimer les séquences de mauvaise qualité et les séquences chimériques. Celles qui passent ces filtres sont ensuite tronquées afin de conserver uniquement les portions de séquences correspondant à la région d'intérêt. Une fois les séquences ≪ nettoyées ≫, elles peuvent alors être utilisées pour inférer un arbre phylogénétique et pour déterminer la composition en espèces des différents échantillons.

Phylogénétique moléculaire. De nombreuses analyses statistiques ([131, 143, 122, 111, 47]) utilisent l'information d'évolution entre espèces. Pour cela, elles intègrent très souvent dans leur calcul un arbre phylogénétique, présenté dans la figure 1.2. Celui-ci offre une représentation schématique des relations de parenté existant entre différentes espèces ou groupe d'espèces. Un arbre phylogénétique est un graphe composé de nœuds et de branches. Les nœuds correspondent à des unités taxonomiques qui peuvent être hypothétique (HTU), si le nœud est interne, ou opérationnelle (OTU), dans le cas d'une feuille. Les branches, quant à elles, représentent les relations de parentés, *i.e.*, ancêtre ou descendants, entre les unités taxonomiques. L'ensemble des branchements de l'arbre constitue la topologie de l'arbre.

Composition en espèces. Outre l'analyse phylogénétique, les lectures pré-traitées permettent d'explorer la diversité en micro-organismes et la composition taxonomique des différents échantillons. Pour cela, deux stratégies principales, présentées dans la figure 1.3, coexistent : la stratégie *close-reference* et la stratégie *de novo*.

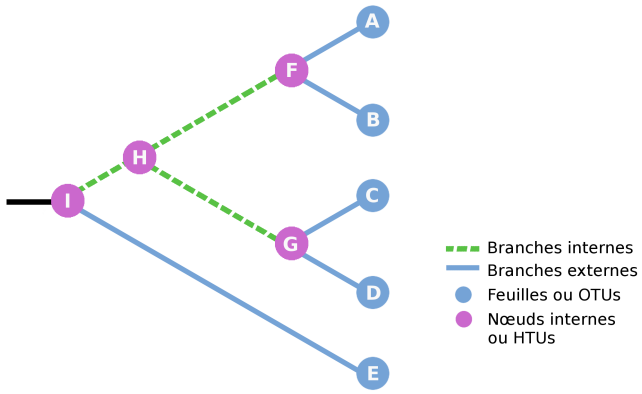

Figure 1.2 Un **arbre phylogénétique** permet de représenter les relations évolutives entre les espèces étudiées. Les nœuds internes F, G, H et I sont les unités taxonomique hypothétiques. Les feuilles A, B, C, D et E sont les unités taxonomique opérationnelles. Le nœud I est la racine de l'arbre.

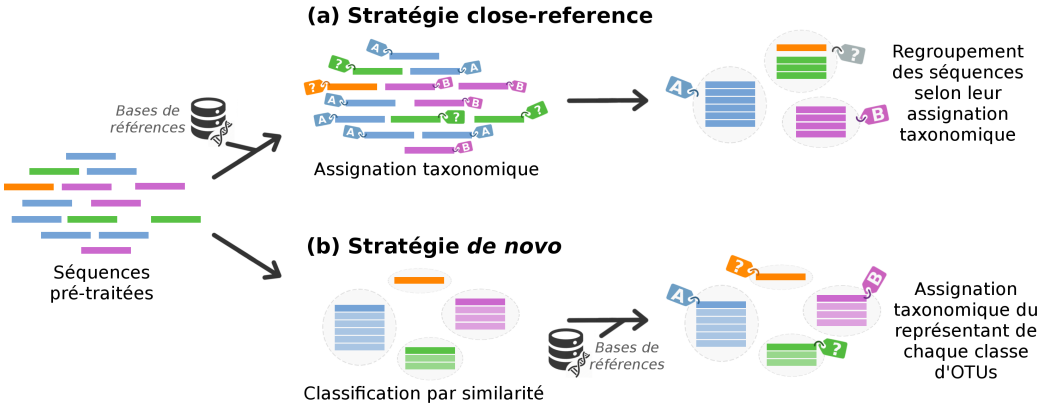

Figure 1.3 **Stratégies bio-informatiques pour l'analyse de données de métagénétique.** La stratégie close-reference (a) attribue une affiliation taxonomique avant de regrouper les séquences en taxons. La stratégie *de novo* (b), quant à elle, regroupe les séquences en OTUs avant de leur attribuer une taxonomie.

Les méthodes reposant sur la stratégie close-reference, tel que Kraken [182] et plus récemment CLARK [127], cherchent en premier lieu à regrouper les séquences à analyser en taxons. Ces derniers représentent des entités conceptuelles permettant de regrouper un ensemble d'organismes vivants selon des critères taxonomiques. À titre d'exemple, l'espèce, le taxon de base, est le niveau taxonomique le plus souvent utilisé car il regroupe des organismes dont le degré de ressemblance est élevé. Pour cela, les séquences sont comparées une à une avec une base de données de référence, tels que Greengenes [45], Silva [140] et RDP [32] dans un contexte bactérien. Cette comparaison est réalisée soit par le calcul d'un score d'alignement soit par classification supervisée en utilisant pour la phase d'apprentissage la composition en k-mers des séquences de la base de référence. Ces méthodes, bien qu'efficaces, ignorent les séquences absentes des bases de référence en leur attribuant l'étiquette ≪ inconnue ≫.

Les chaînes de traitements utilisant la stratégie *de novo*, dont font partis BMP [139], Mothur [156] et QIIME [25], commencent par regrouper les séquences en OTUs. Pour cela, toutes les séquences d'un jeu de données sont comparées entre elles par alignement, avant de les regrouper selon un critère de similarité et ceci indépendamment de leur taxonomie. Des degrés de divergence correspondant à un seuil de similarité de 95%, 97%, ou 99% sont traditionnellement acceptés pour la définition d'une espèce. À noter tout de même que ces seuils, bien qu'ils soient acceptés et utilisés, portent

Table 1.1 **Tableau de comptages** extrait des données du projet *TARA oceans*. Les lignes correspondent à trois identifiants d'échantillons et les colonnes à cinq identifiants représentant cinq OTUs différents.

	EF574663	JN547444	EU800201	EU802831	EF574345	...
T_109_SRF	358	275	256	247	87	...
T_149_MES	1	0	7	0	0	...
T_110_MES	62	61	90	76	0	...

encore aujourd'hui à discussion. En effet, ceux-ci ont été choisis à partir d'organismes isolés en culture, organismes qui ne représentent qu'une minorité de ceux présents dans l'environnement. Pour chaque OTU formé, une séquence, dite représentante, est extraite selon différentes méthodes : séquence consensus, la plus longue ou encore la plus abondante. Cette séquence représentante est alors comparée à une base de données de référence afin de donner une affiliation taxonomique à l'OTU et son représentant. Contrairement à la stratégie *close-reference*, le regroupement en OTUs permet de différencier des séquences d'espèces différentes mais pour lesquelles aucune séquence de référence n'existe.

Ces deux stratégies sont conceptuellement différentes et peuvent conduire à des résultats différents en terme de composition finale. Toutefois, toutes deux permettent de construire un tableau de comptages d'espèces, similaire à celui présenté dans le tableau 1.1, point d'entrée pour toute analyse statistique en écologie microbienne. Dans la suite de ce chapitre, le terme espèce pourra aussi bien désigner le taxon que l'OTU.

1.2 Analyses statistiques en écologie microbienne

Considérons un ensemble d'échantillons, que l'on pourra aussi nommer communautés, $(x_i)_{i=1,\dots,n}$, pour lequel on dispose d'un tableau de comptages (tableau 1.1) représenté par une matrice, notée Y. Cette matrice est de taille $n \times p$, où n correspond au nombre d'échantillons et p au nombre d'espèces observées dans l'ensemble des échantillons. Les entrées de Y sont des valeurs de comptages, notées y_{ij}, correspondant au nombre d'organismes de l'espèce j observés dans la communauté x_i.

1.2.1 Normalisation

Les données de comptage obtenues lors d'un projet de séquençage en écologie microbienne souffrent de biais techniques et de biais de sous-échantillonnage des communautés microbiennes. Ainsi, le nombre de lectures obtenues par séquençage, aussi appelé profondeur de séquençage, peut être très différent entre échantillons. Cette différence empêche la comparaison directe de plusieurs échantillons et impose une normalisation des comptages bruts. La normalisation classiquement utilisée, la normalisation par somme totale, ou *Total Sum Scaling normalisation* (TSS), divise chaque comptage brut par le nombre total d'organismes observés par échantillon :

$$\tilde{y}_{ij} = \frac{y_{ij}}{\sum_{j'=1}^{p} y_{ij'}}$$

Cette normalisation conduit à l'obtention de données d'abondance relative contenues dans un simplex, aussi appelée données de composition. Ce type de données peut être analysé à l'aide de la géométrie d'Aitchison [3] mais les opérations euclidiennes standards, telles que l'addition et la multiplication ne sont plus utilisables. Dans la géométrie d'Aitchison, seuls les opérateurs de

perturbation et de puissance sont disponibles et ceci uniquement pour des données de composition non creuses. Pour pallier ces limitations, une stratégie consiste à projeter les comptages normalisés par la méthode TSS dans un espace euclidien, par une transformation en log-ratio centré, ou *Centered Log Ratio transformation* (CLR) :

$$\tilde{y}'_{ij} = \log \frac{\tilde{y}_{ij}}{\sqrt[p]{\prod_{j'=1}^{p} \tilde{y}_{ij'}}}.$$

Une stratégie alternative, la normalisation par somme cumulée, ou *Cumulative Sum Scaling normalisation* (CSS), a été développée par [129] afin de pallier le biais d'estimation d'abondance différentielle de la normalisation TSS dans le cas de tables de comptages creuses [46]. La normalisation CSS, conçue pour les données de métagénétique, est une extension adaptative de la normalisation par quantiles. Elle consiste à diviser les comptages bruts par la somme cumulée des comptages jusqu'à un centile défini à partir des données.

Une alternative à la normalisation consiste à considérer que les observations, $(x_i)_i$, suivent une loi de probabilité et à en déterminer les paramètres. Dans le contexte de l'écologie microbienne, il semble naturel de considérer que les observations sont distribuées selon une loi de Poisson. Cependant, comme présenté dans [179], les données de comptage issues du séquençage haut-débit sont connues pour être sur-dispersées et lorsque le niveau taxonomique est bas, les comptages bruts peuvent comporter un nombre important de zéros. Ainsi, les distributions Poisson-Gamma, recommandées dans [113], ou Poisson avec sur-abondance de zéros, proposées dans [180], sont souvent envisagées comme alternative.

Une autre méthode efficace permettant d'éviter l'étape de normalisation consiste à utiliser des distances calculées directement sur les comptages bruts. Dans la section suivante, les principales distances utilisées en écologie microbienne sont présentées.

1.2.2 Analyse de la biodiversité

L'analyse de la biodiversité repose sur l'étude des relations entre les variables, telles que l'abondance ou la distance phylogénétique, caractérisant les espèces et les communautés. Pour cela, différents indices ont été mis en place afin d'étudier la biodiversité au sein d'un échantillon unique, c'est la diversité α, et entre échantillons, c'est la diversité β. La diversité γ, quant à elle, correspond à la diversité à l'échelle régionale en mesurant le taux d'addition de nouvelles espèces quand on échantillonne le même habitat en différents endroits.

Diversité α. L'idée la plus simple afin de décrire une communauté microbienne est la richesse, terme définissant le nombre d'espèces observées dans un milieu donné. Dans l'exemple (a) de la figure 1.4, la richesse permet de mettre en évidence la différence en diversité entre les échantillons A et B. Cependant, cette mesure n'est plus suffisante dans l'exemple (b), car celle-ci ne prend pas en compte l'abondance de chaque espèce et donne ainsi trop d'importance aux espèces rares.

L'indice de Shannon, ou entropie de Shannon, introduit par [159], est un exemple de diversité α reflétant aussi bien le nombre d'espèces que leurs abondances. Pour un échantillon x_i donné, cette mesure est définie par $H(x_i) = -\sum_{j=1}^{p} \tilde{y}_{ij} \ln \tilde{y}_{ij}$. Ainsi, si l'on reprend l'exemple (b) de la figure 1.4, l'échantillon A, obtient une valeur de diversité de 1.09, là où l'échantillon B n'obtiendrait que 0.72, résultat plus représentatif de la différence de diversité entre ces deux échantillons. Sans être exhaustif, il existe d'autres estimateurs, poursuivant différents objectifs, tels que la richesse Chao [26], l'indice de Simpson [161] ou encore l'indice de Berger–Parker [12].

Diversité β. Cette mesure de biodiversité définit une dissimilarité entre deux échantillons. Pour cela, certains indices, tels que les indices de Jaccard ([79]) et Sørensen ([164]), traitent les espèces rares et

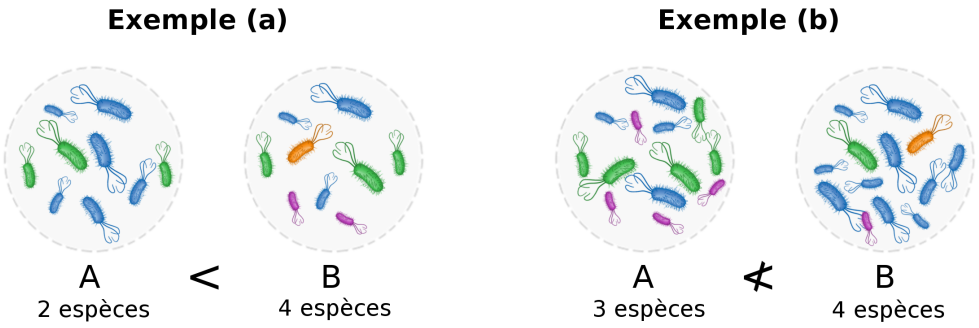

Figure 1.4 Dans l'exemple (a), l'échantillon B est plus diversifié que l'échantillon A car il contient deux fois plus d'espèces. Dans l'exemple (b), bien que l'échantillon B contienne plus d'espèces, il semble moins diversifié que l'échantillon A.

abondantes de façon égale en comparant uniquement le nombre d'espèces partagées et uniques entre les échantillons. L'indice de Jaccard est défini par

$$d_{Jac}(x_i, x_j) = \frac{\sum_{s=1}^{p} \left(\mathbb{I}_{\{y_{is}>0,\, y_{js}=0\}} + \mathbb{I}_{\{y_{js}>0,\, y_{is}=0\}} \right)}{\sum_{s=1}^{p} \mathbb{I}_{\{y_{is}+y_{js}>0\}}}.$$

D'autres indices, telle que la dissimilarité de Bray–Curtis, proposée par [21], complètent ces premières mesures en traitant non pas les espèces mais les organismes de façon identique. Ceci permet ainsi de prendre en compte l'abondance mais rend cette mesure dépendante de la taille d'échantillonnage. Elle est définie par

$$d_{BC}(x_i, x_j) = \frac{\sum_{s=1}^{p} |y_{is} - y_{js}|}{\sum_{s=1}^{p} (y_{is} + y_{js})}.$$

La distance UniFrac, proposée par [100, 101], diffère des mesures de dissimilarités tel que Bray-Curtis par l'utilisation d'informations d'évolution en incluant dans son calcul les distances phylogénétiques entre organismes observés. Elle prend en compte la présence ou l'absence d'organismes entre deux échantillons en calculant la fraction de la longueur totale des branches qui n'est pas commune entre les échantillons. Pour cela, considérons deux échantillons x_i et x_j et un arbre phylogénétique composé de B branches. Pour chaque branche $b \in [1, \ldots, B]$, l_b représente la longueur de la branche b et r_{ib} le ratio de taxa descendants de la branche b et provenant de l'échantillon x_i. La distance UniFrac est alors définie par

$$d_{UF}(x_i, x_j) = \frac{\sum_{b=1}^{B} l_b (\mathbb{I}_{\{r_{ib}>0,\, r_{jb}=0\}} + \mathbb{I}_{\{r_{jb}>0,\, r_{ib}=0\}})}{\sum_{b=1}^{B} l_b \mathbb{I}_{\{r_{ib}+r_{jb}>0\}}}$$

Dans sa version pondérée, cette distance prend en compte l'abondance relative des différents organismes observés. Cette version est très utilisée dans les études de métagénétique car les lectures obtenues par séquençage, de l'ordre du million, apportent une information quantitative. La distance UniFrac pondérée s'écrit sous la forme

$$d_{wUF}(x_i, x_j) = \frac{\sum_{b=1}^{B} l_b |r_{ib} - r_{jb}|}{\sum_{b=1}^{B} (r_{ib} + r_{jb})}.$$

Plus récemment, [28] a proposé une version généralisée de la distance UniFrac et de sa version pondérée avec pour objectif de diminuer la trop grande importance que pouvaient prendre les lignages abondants ou rares. Cette variante de la distance UniFrac est définie par

$$d^{(\alpha)}_{gUF}(x_i, x_j) = \frac{\sum_{b=1}^{B} l_b (r_{ib} + r_{jb})^{\alpha} \left| \frac{r_{ib} - r_{jb}}{r_{ib} + r_{jb}} \right|}{\sum_{b=1}^{B} l_b (r_{ib} + r_{jb})^{\alpha}},$$

où $\alpha \in [0, 1]$ contrôle la contribution des lignages abondants ou rares et $\sum_{b=1}^{B} l_b(r_{ib} + r_{jb})^{\alpha}$ est un facteur de normalisation permettant à la distance d'être comprise entre 0 et 1.

1.2.3 Exploration de la structure de populations microbiennes

La comparaison de plusieurs communautés microbiennes implique l'étude de la biodiversité, mais aussi de la structure existant entre les communautés, *i.e.*, de la façon dont les données sont organisées. Pour cela, les analyses exploratoires sont une première étape importante. On y retrouve l'analyse en composantes principales (ACP), la classification non supervisée mais aussi des outils de visualisation.

Les méthodes d'ordination, telle que l'ACP, permettent l'analyse multivariée de la matrice des comptages, Y. Cependant, le critère de variance utilisée par l'ACP est peu adapté aux données de diversités microbiennes. Des méthodes telles que l'analyse en coordonnées principales (PCoA), introduite par [168], aussi connue sous le nom de positionnement multidimensionnel (*Multidimensional scaling* ou MDS) ou encore la double analyse en coordonnées principales (DPCoA), proposée par [131], sont des extensions de l'ACP adaptées aux métriques liées à la diversité β. À titre d'exemple, la figure 1.5 (a), extraite de [170], montre la structure des communautés planctoniques eucaryotes présentes en surface des océans tempérés et tropicaux.

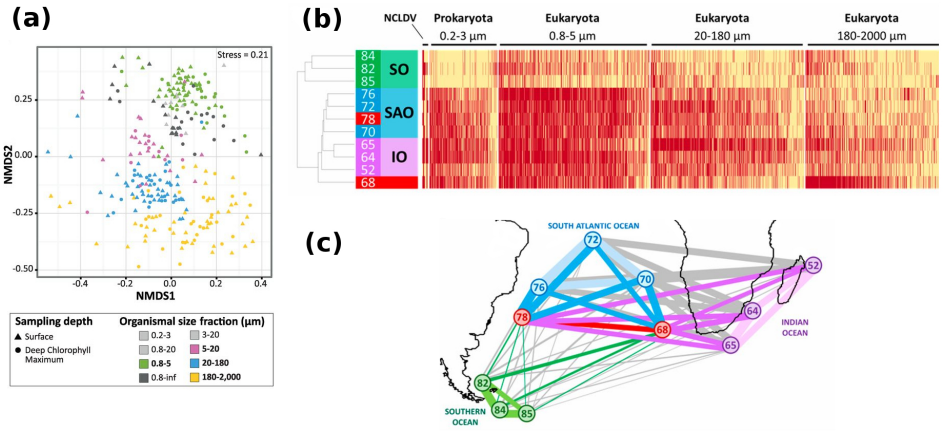

Figure 1.5 **Exemples d'analyses exploratoires.** (a) Représentation des communautés planctoniques eucaryotes à l'aide d'une PCoA (distance de Bray-Curtis), [170]. Chaque symbole représente un échantillon correspondant à une profondeur (la forme) et à la taille des organismes de la fraction (la couleur). (b) Structure des communautés planctoniques de l'océan Indien (IO), l'océan Atlantique sud (SAO), l'océan Austral (SO) et de la zone de l'anneau des Aiguilles (stations 68 et 78, en rouge), [174]. (c) Graphe de co-occurences entre les 11 échantillons, étudiés dans [174]. La largeur de chaque arrête est proportionnelle au nombre d'espèces partagées entre échantillons.

La structure existante entre espèces et échantillons peut être explorée à l'aide de la combinaison d'une *heat map* et d'une classification ascendante hiérarchique (CAH), [178]. Dans la figure 1.5 (b), provenant de [174], la *heat map* permet de visualiser les données de comptages et la CAH révèle l'existence d'une structure entre les espèces et les échantillons. Outre ces méthodes, les graphes de co-occurences, comme celui présenté dans la figure 1.5 (c), sont un outil efficace pour afficher et explorer les interactions entre micro-organismes, [53, 98, 174, 66].

Pour faciliter ces analyses, des outils intégrant différentes visualisations sont mis à disposition des biologistes. Certaines applications comme mg-RAST [117] ou FROGS [52] permettent l'analyse automatique d'un métagénome, d'autres sont dédiées à l'analyse de viromes [152] ou utilisent la structure hiérarchique des données pour faciliter l'exploration des données [176]. Bien que certains de ces outils utilisent l'information phylogénétique, l'ensemble des méthodes exploratoires disponibles

pour étudier la structure de populations microbiennes ne permettent pas l'analyse simultanée de jeux de données omiques hétérogènes.

Chapitre 2
Cadre statistique

Les méthodes statistiques exploratoires sont une famille de méthodes permettant de faciliter la visualisation des données et de révéler leur structure sous-jacente. On y retrouve les méthodes de projection, telle que l'analyse en composantes principales (ACP), et les méthodes de classification non supervisée, telles que les cartes auto-organisatrices ou *Self-Organizing Maps* (SOM).

Ces méthodes sont peu adaptées au traitement des données omiques qui sont très souvent massives et de natures hétérogènes. Plus précisément, comme présenté dans le chapitre 1, en écologie microbienne, les données à analyser peuvent prendre la forme de tableaux de comptage, d'arbres phylogénétiques, de matrices de distance ou encore de graphes de co-occurences. Ces données peuvent aussi provenir de sources différentes et caractériser différents types microbiens : bactéries, virus ou encore eucaryotes. Pour traiter et intégrer ces données, les méthodes à noyaux représentent un cadre naturel, mais nécessitent des adaptations afin de diminuer leurs complexités algorithmiques et d'améliorer l'interprétabilité de leurs modèles. Les contributions proposées dans le cadre de cette thèse, présentées dans le chapitre 3, abordent ces problématiques en s'appuyant sur différents concepts et méthodes présentés dans ce chapitre.

Les deux premières sections introduisent les noyaux, les méthodes exploratoires à noyaux et leurs problématiques, illustrées à l'aide de l'ACP et du SOM. La troisième section aborde différents aspects de la gestion des données massives en algorithmique et la quatrième présente différentes approches d'intégration de données.

2.1 Les noyaux

2.1.1 Définition générale

Considérons $(x_i)_{i=1,\dots,n}$ un ensemble d'observations qui prennent leurs valeurs dans un espace arbitraire $\mathcal{X}$. Par exemple, ces données peuvent être un ensemble d'images ou encore un ensemble de séquences de nucléotides dans un contexte biologique. Un noyau est une fonction $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ qui peut être évaluée pour toute paire d'observations : $K_{ij} = K(x_i, x_j)$. Cette fonction doit être symétrique ($\forall\, x_i, x_j \in \mathcal{X}$, $K_{ij} = K_{ji}$) et positive ($\forall\, n \in \mathbb{N}$, $\forall\, (\alpha_i)_{i=1,\dots,n} \subset \mathbb{R}$, et $\forall\, (x_i)_{i=1,\dots,n} \subset \mathcal{X}$, $\sum_{i,i'=1}^{n} \alpha_i \alpha_{i'} K_{ii'} \geq 0$).

Comme démontré par [7], lorsque ces conditions sont remplies, le noyau définit un produit scalaire dans un espace de Hilbert sous-jacent. Plus précisément, il existe un unique espace de Hilbert $(\mathcal{H}, \langle ., . \rangle_{\mathcal{H}})$, nommé espace image, et une fonction $\phi : \mathcal{X} \to \mathcal{H}$, appelée fonction image, tels que

$$\forall\, x_i, x_j \in \mathcal{X}, \qquad K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}}. \tag{2.1}$$

Dans le cas où $\mathcal{X} = \mathbb{R}^p$, un exemple de noyau, appelé noyau linéaire, est

$$\forall x_i, x_j \in \mathbb{R}^p, \qquad K_{ij} = x_i^T x_j = \langle x_i, x_j \rangle_{\mathbb{R}^p}.$$

D'autres exemples de noyaux adaptés à l'analyse de données biologiques peuvent être cités, comme le noyau spectral [97], le noyau *mismatch* [96], le noyau de Fisher [78] ou encore celui proposé par [172] permettant l'analyse d'arbres phylogénétiques.

2.1.2 L'astuce noyau

L'astuce noyau, ou *kernel trick*, est basée sur le fait que l'on peut calculer le produit scalaire $\langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}}$ directement, sans jamais avoir à calculer explicitement $\phi(x_i)$ et $\phi(x_j)$. Afin d'illustrer ce principe, prenons deux observations $x_i, x_j \in \mathcal{X}$ projetés en deux vecteurs $\phi(x_i)$ et $\phi(x_j)$ dans $\mathcal{H}$. On peut alors définir une distance $d(x_i, x_j)$ entre deux observations par la distance entre leurs images dans l'espace de Hilbert :

$$d(x_i, x_j) = \|\phi(x_i) - \phi(x_j)\|_{\mathcal{H}}. \tag{2.2}$$

L'astuce noyau permet de réécrire la distance, définie dans l'équation (2.2), seulement en terme de valeurs du noyau, sans passer par le calcul de ϕ :

$$d(x_i, x_j) = \sqrt{K_{ii} + K_{jj} - 2K_{ij}}.$$

L'astuce noyau autorise donc des calculs implicites dans l'espace image $\mathcal{H}$ bien que ni celui-ci ni la fonction image ϕ n'aient besoin d'être connus.

2.1.3 Intérêts pratiques des noyaux

L'astuce noyau, présenté dans la section 2.1.2, a des conséquences pratiques importantes. C'est, en premier lieu, une astuce très pratique afin de convertir un algorithme utilisant des données vectorielles en une méthode pouvant utiliser n'importe quel type de données. Cette astuce est cependant restreinte aux nombreux algorithmes faisant intervenir un produit scalaire, un calcul d'une norme ou d'une distance, car il implique de remplacer ce calcul par un noyau plus général défini pour des données plus complexes.

L'utilisation des noyaux permet aussi de transformer un problème initialement non-linéaire en un problème linéaire en projetant les données initiales dans un espace de plus grande dimension, $\mathcal{H}$. Afin d'illustrer cette propriété, considérons le jeu de données non linéairement séparable présenté à gauche de la figure 2.1 avec ses deux cercles concentriques. Une transformation $\phi : \mathbb{R}^2 \to \mathbb{R}^3$, tel que $\phi([x, y]) = [x, y, x^2 + y^2]$, conduit à une représentation linéairement séparable de ces données, comme présenté à droite de la figure 2.1. Ainsi, un jeu de données non linéairement séparable dans $\mathbb{R}^p$ peut

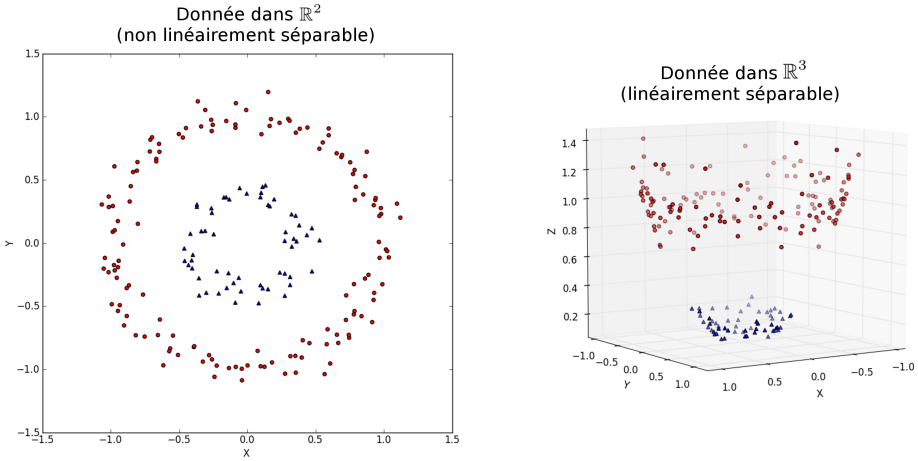

Figure 2.1 À gauche, un jeu de données dans $\mathbb{R}^2$, non linéairement séparable. À droite, le même jeu de données transformé par $[x, y] = [x, y, x^2 + y^2]$, linéairement séparable dans $\mathbb{R}^3$.

être linéairement séparable dans un espace de plus grande dimension $\mathbb{R}^q$ (avec $q \gg p$).

Ces propriétés permettent aux nombreuses méthodes de *Machine Learning* utilisant les noyaux d'être particulièrement pertinentes pour analyser les données de biologie computationnelle [157].

2.1.4 Le cas des données de dissimilarité

Comme pour les noyaux, les dissimilarités décrivent les observations d'un jeu de données deux à deux, conduisant à l'obtention d'une matrice de dissimilarité Δ, dont les entrées $\delta(x_i, x_j)$ représentent une valeur de dissimilarité entre les observations x_i et x_j. On suppose que Δ possède des propriétés basiques : la positivité des entrées ($\delta(x_i, x_j) \geq 0$), la symétrie ($\delta(x_i, x_j) = \delta(x_j, x_i)$)) ainsi qu'une diagonale nulle ($\delta(x_i, x_i) = 0$).

Si la dissimilarité Δ est euclidienne, alors, comme le suggère [95], la similarité définie par

$$K(x_i, x_j) = -\frac{1}{2}\left(\delta(x_i, x_j) - \frac{1}{n}\sum_{k=1}^{n}(\delta(x_i, x_k) + \delta(x_k, x_j)) + \frac{1}{n^2}\sum_{k,\,k'=1}^{n}\delta(x_k, x_{k'})\right), \tag{2.3}$$

est un noyau et les distances entre observations dans l'espace image induit par le noyau sont données par Δ.

Dans le cas d'une dissimilarité non euclidienne, définie dans l'équation (2.3), K n'est pas un noyau. Sa décomposition en valeurs singulières, peut conduire à des valeurs propres négatives. Cette décomposition est donnée par $K = U^T \Lambda U$, où U est une matrice orthogonale, dont les colonnes sont les vecteurs propres de U, et où $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$ avec $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$, est une matrice diagonale dont les coefficients sont les valeurs propres de U. Afin de convertir des similarités en noyaux, plusieurs méthodes ont été proposées par [30], tel que la troncature spectrale, ou *spectrum clip* et l'inversion spectrale, ou *spectrum flip*. La première convertit K en une matrice positive en mettant à zéro toutes les valeurs propres négatives, tel que

$$\Lambda_{clip} = \text{diag}(\max(\lambda_1, 0), \ldots, \max(\lambda_n, 0)).$$

Un noyau peut alors être obtenu par $K_{clip} = K^T \Lambda_{clip} K$. Afin de prendre en compte la partie négative de la décomposition en valeurs singulières de K, l'inversion spectrale inverse les valeurs propres négatives par

$$\Lambda_{flip} = \text{diag}(|\lambda_1|, \ldots, |\lambda_n|),$$

avant de calculer le noyau $K_{flip} = K^T \Lambda_{flip} K$.

2.2 Analyses exploratoires et noyaux

Une approche générale permettant aux méthodes de classification et de fouille de données d'être génériques quant à la nature des données à traiter est de se baser sur les matrices de dissimilarité ou les noyaux, précédemment introduits dans la section 2.1. Dans cette section, nous présenterons, dans un premier temps, des exemples de méthodes de classification non supervisée pour lesquelles une version noyau a été proposée. Puis, nous nous intéresserons aux modifications nécessaires afin de convertir une méthode standard en une version à noyau. Ces modifications seront illustrées sur deux méthodes d'analyse exploratoire : l'ACP et le SOM.

2.2.1 Classification non supervisée à noyau

La classification non supervisée est une approche très commune pour l'exploration de données. Contrairement aux méthodes de classification supervisées, qui ont un objectif de prédiction, les approches non supervisées regroupent les observations similaires dans des classes sans *a priori*, uniquement à partir des données. Pour cela, différentes approches se distinguent.

Une première approche, la classification ascendante hiérarchique (CAH) proposée par [178], est naturellement capable de traiter les données de dissimilarité. Cette méthode procède de manière hiérarchique à l'agrégation de classes, en se basant sur une distance et un critère d'agrégation utilisant cette distance (méthode de Ward, distance minimale, distance maximale, ...). La CAH est dite ascendante car dans son état initial chaque classe est constituée d'une seule observation, avant d'être, au cours des itérations, rassemblée en classes de plus en plus grandes. La hiérarchie produite peut ensuite être visualisée à l'aide d'un dendrogramme qui permet d'obtenir la classification des observations en choisissant une hauteur de troncature.

D'autres types d'approches se basent sur la définition d'un centroïde, correspondant au centre de gravité d'une classe. L'algorithme des k-moyennes partitionne les observations en k classes en minimisant la distance entre les observations d'une classe et le centroïde correspondant. Celui-ci s'étend de manière évidente aux données décrites par un noyau [157] et aux données de dissimilarité grâce à l'algorithme des k-médoïdes [81]. Dans cette dernière version, les centroïdes de chaque classe ne pouvant être directement calculés, ils sont remplacés par la solution optimale recherchée parmi les données d'origine plutôt que dans $\mathcal{X}$. [150] propose une alternative, basée sur le cadre théorique d'espace pseudo-euclidien [7, 62], qui est proche de l'approche noyau en utilisant une combinaison linéaire parcimonieuse des observations pour représenter les centroïdes.

2.2.2 Analyse en composantes principales

L'ACP est une méthode de réduction de dimension cherchant à projeter des données de grande dimension, $(x_i)_{i=1,\ldots,n} \in \mathbb{R}^p$, dans un espace de dimension plus faible, $\mathbb{R}^q$, avec $q \ll p$. Pour cela, les variables observées sont décomposées en un ensemble de variables linéairement décorrélées les unes des autres, appelées composantes principales. Ces dernières peuvent être obtenus à l'aide des

vecteurs propres de la décomposition spectrale, $C\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}$, où C représente la matrice de covariance des variables d'entrée, telle que $C_{ij} = \langle x_i, x_j \rangle$. Les composantes principales correspondent alors aux coordonnées des projections des données par $\mathcal{P}_{\alpha_k}(x_i) = \langle \alpha_k, x_i \rangle$ sur la nouvelle base de représentation des données, définie par les vecteurs propres $(\alpha_k)_{k=1,\dots,n}$. Les composantes ainsi définies peuvent alors servir de base afin de projeter les données. Cette projection autorise non seulement une représentation graphique simplifiée des données, mais permet aussi une réduction de dimension des observations. Dans le cadre de l'ACP, la projection des données est linéaire, mais de nombreuses approches non linéaires existent, comme celles présentées dans [95], ou encore l'ACP à noyau.

ACP à noyau (K-PCA). L'analyse non-linéaire en composante principale, introduite par [158], est une ACP réalisée dans l'espace image $(\mathcal{H}, \langle ., . \rangle)$ induit par le noyau. Pour cela, les données doivent être centrées dans l'espace image, de telle sorte que $\sum_{i=1}^{n} \widetilde{\phi}(x_i) = 0$, par

$$\begin{aligned} \widetilde{K}_{ij} &= \left\langle \phi(x_i) - \frac{1}{n}\sum_{l=1}^{n}\phi(x_l), \phi(x_j) - \frac{1}{n}\sum_{l=1}^{n}\phi(x_l) \right\rangle \\ &= K_{ij} - \frac{1}{n}\sum_{l=1}^{n}(K_{il} + K_{jl}) + \frac{1}{n^2}\sum_{l,l'=1}^{n} K_{ll'}. \end{aligned} \tag{2.4}$$

Ce qui conduit à l'obtention du noyau centré $\widetilde{K} = K - \frac{1}{n}\mathbf{I}_n^T K - \frac{1}{n}K\mathbf{I}_n + \frac{1}{n^2}\mathbf{I}_n^T K \mathbf{I}_n$, dans lequel $\mathbf{I}_n$ est un vecteur avec n valeurs égales à 1.

L'ACP à noyau correspond à la décomposition en valeurs singulières de la matrice C, dont les entrées sont définies par :

$$C_{ij} = \langle \widetilde{\phi}(x_i), \widetilde{\phi}(x_j) \rangle_{\mathcal{H}},$$

correspondant à la matrice de covariance des images centrées des données d'origine par $\widetilde{\phi}$ (la fonction image associée au noyau centré $\widetilde{K}$). Les vecteurs propres, $(\alpha_k)_{k=1,\dots,n} \in \mathbb{R}^n$ et les valeurs propres associées sont obtenus en résolvant le problème de décomposition en valeurs singulières, $C\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}$. Ce qui est équivalent, grâce à l'astuce noyau, à la résolution du problème suivant

$$\widetilde{K}\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}. \tag{2.5}$$

Ces vecteurs propres, $(a_k)_{k=1,\dots,n} \in \mathcal{H}$ se trouvent dans l'espace engendré par $\{\widetilde{\phi}(x_i)\}_{i=1,\dots,n}$ et peuvent s'exprimer sous la forme

$$a_k = \sum_{i=1}^{n} \alpha_{ki}\widetilde{\phi}(x_i)$$

où $\boldsymbol{\alpha}_k = (\alpha_{ki})_{i=1,\dots,n}$. $\boldsymbol{a}_k = (a_{ki})_{i=1,\dots,n}$ sont orthonormées dans $\mathcal{H}$:

$$\forall\, k, k',\ \langle a_k, a_{k'} \rangle = \boldsymbol{\alpha}_k^\top \widetilde{K}\boldsymbol{\alpha}_{k'} = \delta_{kk'} \qquad \text{avec} \qquad \delta_{kk'} = \begin{cases} 0 & \text{si } k \neq k' \\ 1 & \text{sinon} \end{cases}.$$

Les composantes principales sont les coordonnées des projections des images des données d'origine, $(\widetilde{\phi}(x_i))_i$ sur les vecteurs propres $(a_k)_{k\geq 1}$ qui peuvent s'exprimer par

$$\langle a_k, \widetilde{\phi}(x_i) \rangle = \sum_{j=1}^{n} \alpha_{kj}\widetilde{K}_{ji} = \widetilde{K}_{i.}\boldsymbol{\alpha}_k,$$

où $\boldsymbol{\alpha}_k = (\alpha_{ki})_{i=1,\dots,n}$ et $\widetilde{K}_{i.}$ est la i-ème ligne du noyau $\widetilde{K}$. Selon l'équation (2.5), on obtient ainsi $\langle a_k, \widetilde{\phi}(x_i) \rangle = \lambda_k \alpha_{ki}$. Les données de l'espace d'origine peuvent alors être projetées sur les axes avec

$$\mathcal{P}_{a_k}(\widetilde{\phi}(x_i)) = \langle a_k, \widetilde{\phi}(x_i) \rangle a_k = (\lambda_k \alpha_{ki}) a_k.$$

Ces coordonnées sont utiles afin de représenter les échantillons dans un espace de faible dimension et de mieux comprendre les relations qui les lient. Cependant, contrairement à l'ACP standard, la K-PCA ne permet pas de représenter les variables, les échantillons étant décrits par leurs relations,

à travers le noyau, et non pas par des valeurs numériques standards. Les composantes principales sont, quant à elles, plus difficiles à interpréter car celles-ci sont définis par leur similarité à tous les échantillons. De plus, la complexité de la décomposition en valeur singulière, $\mathcal{O}(n^3)$, fait de la K-PCA une analyse mal adaptée aux jeux de données dans lesquels le nombre d'observations, n, est grand.

Autres extensions de la PCA adaptées aux données décrites par une dissimilarité. La PCoA (ou MDS), permet d'explorer les similarités existant entre observations d'un jeu de données. Comme l'ACP, la PCoA cherche à représenter les observations $(x_i)_{i=1,\dots,n}$ dans un espace de dimension q, où $q \ll p$, par q coordonnées principales $(a_i)_{i=1,\dots,q}$. Les données sont initialement décrites par une matrice de distance D, dont les entrées d_{ij} sont définies par une valeur de distance entre les observations x_i et x_j. Cette distance peut être la distance usuelle de $\mathbb{R}^p$, et dans ce cas la PCoA est équivalente à une ACP. Mais elle peut aussi être obtenue à l'aide des distances écologiques présentées dans la section 1.2.2. La PCoA minimise une fonction de coût $S(a_1, \dots, a_n)$, appelée *stress*, permettant de préserver les proximités entre observations :

$$S(a_1, \dots, a_n) = \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} (b_{ij} - \langle a_i, a_j \rangle)^2,$$

avec b_{ij}, les éléments de la matrice B, obtenue par double centrage de D, par $B = (I - \frac{1}{n}J)D^2(I - \frac{1}{n}J)$, où J est une matrice de taille $n \times n$ ne contenant que des 1. Les coordonnées principales, $(a_i)_{i=1,\dots,d}$, peuvent être obtenus à l'aide des q premiers vecteurs propres de la décomposition spectrale, $B\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}$, par $A = \text{diag}(\sqrt{\lambda})\boldsymbol{\alpha}^T$.

[131] propose une extension à la PCoA, la DPCoA, avec l'objectif d'intégrer une matrice décrivant les différences entre espèces (différences phylogénétiques, morphologiques, biologiques, ...) à l'analyse d'une matrice d'abondance. L'astuce de cette méthode réside dans la définition d'un espace euclidien commun, qui permet de prendre en compte aussi bien les informations relatives aux espèces qu'aux échantillons. Pour cela, les échantillons sont positionnés dans l'espace engendré par la PCoA, réalisée sur la matrice des distances entre espèces. Les coordonnées des échantillons correspondent alors à la moyenne pondérée des espèces qu'ils contiennent, pondération réalisée sur leur abondance.

Problématique abordée : Dans le cadre de ma thèse, je me suis intéressé à divers aspects liés à la K-PCA : i) l'interprétabilité des résultats dans le chapitre 8, avec une application sur les données du projet *TARA oceans* ; ii) la complexité cubique de la K-PCA dans le chapitre 7, [109] ; iii) et l'intérêt que représente la K-PCA pour améliorer l'efficacité de la version noyau des cartes auto-organisatrices dans le chapitre 7, [109].

2.2.3 Cartes auto-organisatrices.

Les cartes auto-organisatrices, introduites par [86], sont une méthode de classification non supervisée permettant d'allier projection de données et classification. Initialement inspirées par des principes biologiques, elles font partie de la famille des réseaux de neurones artificiels.

L'algorithme, présenté dans la figure 2.2, projette les données sur une carte qui discrétise l'espace de données en le divisant en U neurones, ou unités, interconnectés. Différentes topologies peuvent être associées à la carte permettant de définir une « distance » entre les unités qui la composent ainsi qu'une forme, souvent régulière et rectangulaire. À chaque neurone r_u $(u = 1, \dots, U)$, est associé un prototype p_u qui prend ses valeurs dans le même espace que les données initiales $(x_i)_{i=1,\dots,n}$. Ce prototype, qui est un représentant du neurone dans l'espace des données, est donc un vecteur de $\mathbb{R}^p$.

Dans le cas où les données sont numériques, la version stochastique du SOM, alterne, à chaque itération t de l'algorithme, une étape d'affectation et une étape de représentation. La première étape cherche à affecter une observation, choisie aléatoirement parmi l'ensemble des données, au neurone dont le prototype est le plus proche par

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

où $f^t(x_i)$ représente l'unité, ou la classe pour reprendre le vocabulaire de la classification non supervisée, à laquelle l'observation x_i a été affectée. Le prototype du neurone vainqueur, ainsi que ceux des neurones voisins, sont ensuite mis à jour selon une étape de pseudo-descente de gradient stochastique lors de la phase de représentation :

$$p_u^t = p_u^{t-1} + \mu_t h^t(f^t(x_i), u)\left(x_i - p_u^{t-1}\right).$$

L'importance de la mise à jour est déterminée par la valeur d'un pas d'apprentissage, μ_t, et par une fonction h^t qui mesure une similarité entre deux neurones u et u' et qui est généralement soit constante par morceaux, soit gaussienne : $h^t(u,u') = \exp\left(-\frac{D(u,u')}{2\sigma^2}\right)$, où D est une distance entre unités qui définit la topologie de la carte. En général, la taille du voisinage est décroissante au cours de l'apprentissage (dans le cas du voisinage gaussien, c'est le paramètre σ qui permet de contrôler la décroissance). L'algorithme complet est fourni dans l'algorithme 1.

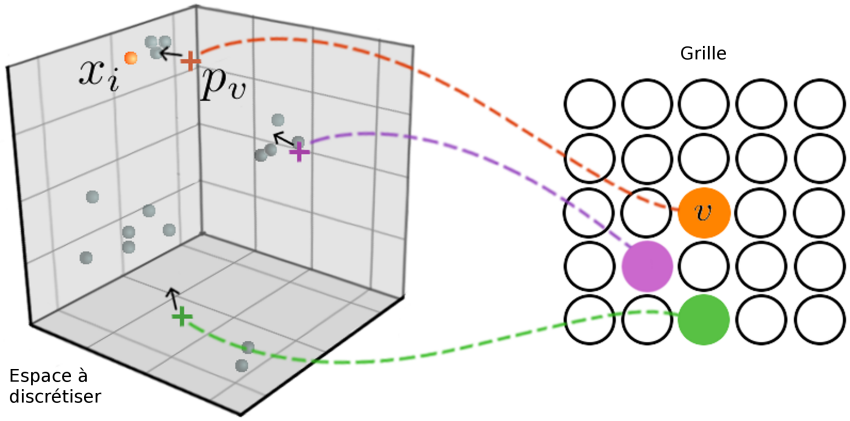

Figure 2.2 **Algorithme des cartes auto-organisatrices.** Chaque neurone de la grille est associé à un prototype qui le représente dans l'espace d'origine. Une observation x_i est choisie aléatoirement. L'observation x_i est affectée au neurone vainqueur v, *i.e.*, le neurone orange, dont le prototype p_v est le plus proche dans l'espace d'entrée. Le prototype vainqueur, p_v, ainsi que les prototypes voisins sont mis à jour dans la direction de x_i lors de l'étape de représentation.

Algorithm 1 SOM, version stochastique

Require: Données : $(x_i)_{i=1,\dots,n} \in \mathbb{R}^p$
1: Initialisation aléatoire de p_1^0,...,p_U^0 dans $\mathbb{R}^p$
2: **for** $t = 1 \to T$ **do**
3: Sélection aléatoire de $i \in \{1,\dots,n\}$
4: Affectation

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

5: Représentation pour $\forall, u = 1,\dots,U$

$$p_u^t = p_u^{t-1} + \mu_t h^t(f^t(x_i), u)\left(x_i - p_u^{t-1}\right)$$

6: **end for**
7: **return** $(p_u^T)_u$ et $(f^T(x_i))_i$

L'un des grands avantages des cartes auto-organisatrices est qu'elles permettent de visualiser la topologie des observations dans l'espace de départ. Ainsi, les prototypes de neurones voisins doivent être proches dans l'espace de départ et les observations proches dans l'espace de départ sont

habituellement classées dans des neurones voisins sur la grille. Cependant, l'algorithme, dans sa version initiale, n'est pas adapté aux données non vectorielles car les étapes d'affectation et de représentation sont explicitées dans $\mathbb{R}^p$.

Problématique abordée : Dans le contexte de ma thèse, j'ai abordé la problématique de l'instabilité des résultats du SOM stochastique : plusieurs exécutions de l'algorithme avec les mêmes paramètres peuvent conduire à des résultats assez différents. L'approche d'agrégation proposée, présentée dans le chapitre 4, prend en compte l'information de topologie, pour améliorer la qualité et réduire la variabilité des résultats du SOM, [105].

Cartes auto-organisatrices à noyau (K-SOM). Lorsque les données ne sont pas vectorielles, se posent les questions de la définition des prototypes dans l'espace initial ainsi que du calcul de la distance entre une observation et un prototype, nécessaire lors de l'étape d'affectation de l'algorithme. Pour cela, plusieurs extensions du SOM ont été proposées : le SOM médian [84, 87] et ses variantes [5, 50], le SOM relationnel [68] ou encore le SOM à noyau [65, 103]. Les limitations et les perspectives de ces méthodes sont discutées dans [148]. Ici, nous nous restreindrons au K-SOM, la version noyau du SOM, proposée dans sa version stochastique dans [103, 6]. Dans cette version, les prototypes sont représentés dans l'espace image $(\mathcal{H}, \langle ., . \rangle_{\mathcal{H}})$, car celui-ci est un espace vectoriel standard muni des opérateurs usuels, contrairement à l'espace initial $\mathcal{X}$ dans lequel évoluent les données. Ainsi, dans cette version de l'algorithme, les prototypes s'expriment comme des combinaisons convexes des images par ϕ des données initiales :

$$p_u = \sum_{i=1}^{n} \gamma_{ui} \phi(x_i) \qquad \text{avec} \qquad \gamma_{ui} \geq 0 \qquad \text{et} \qquad \sum_{i=1}^{n} \gamma_{ui} = 1. \tag{2.6}$$

L'étape d'affectation affecte l'observation x_i au neurone dont le prototype est le plus proche au sens de la distance dans l'espace image $\mathcal{H}$, et ceci, seulement en terme de valeurs du noyau, sans passer par le calcul de ϕ, grâce à l'astuce noyau :

$$\begin{aligned} f^t(x_i) &= \arg \min_{u=1,\dots,U} \|\phi(x_i) - p_u^{t-1}\|_{\mathcal{H}}^2 \\ &= \arg \min_{u=1,\dots,U} \|\phi(x_i) - \sum_{j=1}^{n} \gamma_{uj} \phi(x_j)\|_{\mathcal{H}}^2 \\ &= \arg \min_{u=1,\dots,U} \left(K_{ii} - 2 \sum_{j=1}^{n} \gamma_{uj} K_{ij} + \sum_{j,j'}^{n} \gamma_{uj} \gamma_{uj'} K_{jj'} \right). \end{aligned} \tag{2.7}$$

L'étape de représentation, quant à elle, s'écrit

$$p_u^t = p_u^{t-1} + \mu_t h^t(f^t(x_i), u) \left(\phi(x_i) - p_u^{t-1} \right),$$

ce qui est équivalent à

$$\gamma_u^t = \gamma_u^{t-1} + \mu_t h^t(f^t(x_i), u) \left(\mathbf{1}_i^n - \gamma_u^{t-1} \right),$$

avec $\mathbf{1}_i^n$ un vecteur de taille n avec pour seule valeur non nulle, l'entrée i égale à 1. La version complète de l'algorithme est présentée dans l'algorithme 2.

Les problèmes du K-SOM apparaissent lorsque le nombre d'observations, n, est grand. Le nombre total de coefficients γ_{ui} à apprendre est alors égal à $n \times U$, ce qui conduit à une complexité de $\mathcal{O}(n^2 U)$, pour une itération, au lieu de $\mathcal{O}(pU) + \mathcal{O}(npU)$ pour la version numérique standard du SOM dans $\mathbb{R}^p$. De plus, [126] montre que l'algorithme nécessite un nombre d'itérations de l'ordre de βn afin d'obtenir de bonnes propriétés de convergence. Ce qui conduit à une complexité globale du K-SOM de $\mathcal{O}(\beta n^3 U)$. Ceci restreint son utilisation aux jeux de données composés de quelques milliers d'observations. De plus, comme présenté dans [148], l'interprétation de la carte est plus complexe. En effet, dans le SOM standard, les prototypes peuvent être facilement interprétés car ils utilisent les mêmes variables que celles décrivant les observations du jeu de données d'entrée. Dans sa version à noyau, les prototypes

Algorithm 2 SOM à noyau, version stochastique

Require: Données : $(x_i)_{i=1,\dots,n} \in \mathbb{X}$

1: $\forall u = 1,\dots,U$ et $\forall i = 1,\dots,n$ initialiser γ_{ui}^0 aléatoirement dans $[0,1]$ tel que $\sum_{i=1}^n \gamma_{ui}^0 = 1$ **Result :** $p_u^0 = \sum_{i=1}^n \gamma_{ui}^0 \phi(x_i)$

2: **for** $t = 1 \to T$ **do**

3: Sélection aléatoire de $i \in \{1,\dots,n\}$

4: Affectation

$$f^t(x_i) = \arg\min_{u=1,\dots,U} \|\phi(x_i) - p_u^{t-1}\|_{\mathcal{H}}^2$$

5: Représentation pour $\forall, u = 1,\dots,U$

$$\gamma_u^t = \gamma_u^{t-1} + \mu_t h^t(f^t(x_i), u)\left(\mathbf{1}_i^n - \gamma_u^{t-1}\right),$$

avec $\mathbf{1}_i^n$ un vecteur de dimension n avec pour seule valeur non nulle, l'entrée i égale à 1.

6: **end for**

7: **return** $(p_u^T)_u$ et $(f^T(x_i))_i$

sont définis par leur proximité à toutes les observations, impliquant de comprendre ces relations, ce qui est probablement aussi difficile que d'analyser directement le jeu de données complet.

Problématique abordée : Trois contributions de ma thèse ont porté sur le K-SOM afin de réduire la complexité de l'algorithme tout en fournissant une interprétation facilitée des prototypes : le K-SOM accéléré [107], le *bagged* K-SOM [108] et le K-PCA SOM [107], respectivement présentés dans les chapitres 5, 6 et 7.

2.3 Méthodes pour le traitement de données massives

Les grands volumes de données biologiques produits nécessite une adaptation des approches statistiques standards qui ont très souvent été développées sans tenir compte de leur complexité algorithmique. Pour cela, les approches par sous-échantillonnage sont fréquemment utilisées. Elles comptent parmi elles les méthodes comme certaines extensions du *bagging* et l'approximation de Nyström qui sont présentées dans cette section.

Bagging Les approches de *bagging*, introduites par [22], consistent à tirer aléatoirement avec remise dans $(x_i)_{i=1,\dots,n}$, B échantillons de taille n, dit « bootstrap » sur lesquels un modèle différent est appris. Pour tout $b = 1,\dots,B$, on notera $\mathcal{T}_b$ l'ensemble des indices de $1,\dots,n$ des observations de l'échantillon *bootstrap* b. Souvent utilisé dans le contexte de la régression, le *bagging* agrège ensuite, par une simple moyenne, les B estimateurs obtenus à partir des B échantillons. Cette méthode est parallélisable et permet ainsi un gain en temps de calcul important, mais dépendant de l'infrastructure dont dispose l'utilisateur.

Cependant, la mise en œuvre de ces méthodes reste difficile lorsque n est grand, le nombre moyen d'observations différent dans un échantillon *bootstrap* étant de l'ordre de $0.632 \times n$. Pour répondre à ce problème, de nombreuses alternatives ont été proposées. [14] utilisent des échantillons *bootstrap* de taille m, avec $m \ll n$ et [83] proposent une approche, nommée « bag of little bootstrap », permettant de construire des échantillons bootstrap de tailles n mais ne contenant seulement $m \ll n$ observations différentes.

Approximation de Nyström La complexité des approches utilisant les noyaux est en général en $\mathcal{O}(n^2)$, voir même en $\mathcal{O}(n^3)$ dans le cas de la K-PCA. [181] utilise l'approximation de Nyström, pour

fournir une approximation de la décomposition en valeurs singulières d'un noyau K, dont la complexité est en $\mathcal{O}(n^3)$. Plus précisément, la décomposition spectrale de K, qui est supposé centré, est approchée en sélectionnant m observations sans remise, $\mathcal{T}_m$ parmi $(x_i)_{i=1,\dots,n}$, et en utilisant la décomposition spectrale de la matrice réduite $K^{(m)} = (K(x_i, x_j))_{i,j\in\mathcal{T}_m}$. Les observations sélectionnées, $\mathcal{T}_m$, peuvent être choisies de façon aléatoire ou par des stratégies plus efficaces comme celles décrites et proposées dans [89].

Si les valeurs propres et les vecteurs propres de $K^{(m)}$ sont respectivement notés $(\lambda_k^{(m)})_k$ et $(v_k^{(m)})_k$, pour tout $k \in 1, \dots, n$, alors les valeurs propres, $(\lambda_k)_k$, et les vecteurs propres, $(v_k)_k$, de K sont approchés par

$$\lambda_k \simeq \frac{n}{m}\lambda_k^{(m)} \qquad \text{and} \qquad v_{ki} \simeq \sqrt{\frac{m}{n}}\frac{1}{\lambda_k^{(m)}} K_{i.}^{(n,m)} v_k^{(m)},$$

avec $K_{i.}^{(n,m)}$ la i-ème ligne de la matrice $K^{(n,m)} = (K(x_j, x_{j'}))_{j=1,\dots,n,\, j'\in\mathcal{T}_m}$. Dans le cas où le rang de $K^{(m)}$ est identique à celui de la matrice d'origine K, l'approximation devient alors une égalité. Le coût de l'approximation de la décomposition spectrale de K par la décomposition spectrale de $K^{(m)}$ est de $\mathcal{O}(m^3) + \mathcal{O}(nm^2)$. Comme m est petit devant n, la complexité de la K-PCA utilisant l'approximation de Nyström est dominée par $\mathcal{O}(nm^2)$, au lieu de $\mathcal{O}(n^3)$ dans sa version standard.

Problématique abordée : Dans le cadre de mes travaux, j'ai utilisé les approches de *bagging* et l'approximation de Nyström afin de diminuer la complexité de K-SOM. Les méthodes proposées, le *bagged* K-SOM [108] et le K-PCA SOM [107], sont respectivement présentées dans les chapitres 6 et 7.

2.4 Intégration de données omiques

L'intérêt que représente les analyses intégratives, pour améliorer la compréhension des systèmes biologiques, n'est plus à démontrer [56]. Cependant, l'étude des relations et des interactions existant entre différents omiques représente encore aujourd'hui un défi. Pour y répondre, les formulations méthodologiques sont nombreuses et poursuivent des objectifs pouvant être très divers.

Dans [88], les approches d'intégration sont classées selon leurs objectifs : découvrir des mécanismes moléculaires, effectuer une classification non supervisée des observations ou encore prédire un phénotype ou une efficacité thérapeutique. [13] proposent une classification se basant sur les aspects mathématiques des méthodes, qui peuvent être plus ou moins génériques quant aux types de données qu'elles peuvent traiter. À titre d'exemple, *iCluster* [160] traite n'importe quel omique se présentant sous forme de valeurs quantitatives, et *Conexic* [4] est conçu pour analyser l'expression des gènes et leurs variabilité en nombre de copies. Une distinction peut aussi être faite selon que l'analyse est simultanée ou séquentielle, permettant au résultat d'un omique d'être amélioré à l'aide des résultats obtenus pour les autres. Ce type d'approches peut être pertinent dans le cas où certaines paires d'omiques ont un lien de cause à effet comme cela peut être le cas entre le génome et le transcriptome. Cependant, nous nous intéresserons dans la suite de cette section aux approches non séquentielles.

Pour cela, considérons M jeux de données $(x_i^m)_{i=1,\dots,n}$ (avec $m = 1, \dots, M$), prenant chacun valeur dans un espace arbitraire $(\mathcal{X}^m)_{m=1,\dots,M}$, tous obtenus sur les mêmes échantillons $i = 1, \dots, n$. Chaque jeu de données fournit une image spécifique des observations. La figure 2.3 présente une classification, proposée par [146], des différentes stratégies d'intégration en trois grandes familles : l'intégration par concaténation, par transformation et par agrégation de résultats de modèles. Ces trois familles sont présentées dans les trois sections suivantes.

Intégration par concaténation. Les méthodes utilisant cette première stratégie fusionnent les M jeux de données en une seule et même matrice. Les approches statistiques standards peuvent alors être

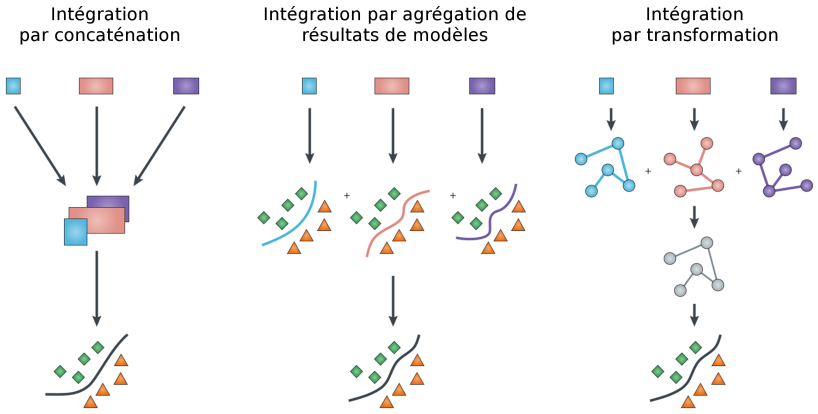

Figure 2.3 (a) L'intégration par concaténation implique la combinaison des jeux de données au niveau des matrices de données, qui peuvent être les données d'origine ou une représentation numérique de celles-ci. (b) L'intégration par agrégation de résultats de modèles nécessite d'analyser chaque jeu de données indépendamment avant d'agréger leurs résultats. (c) L'intégration par transformation projète ou transforme les données d'origine pour pouvoir les combiner. L'objet combiné, pouvant par exemple prendre la forme d'un graphe ou d'un noyau, peut alors être analysé par n'importe quelle méthode conçue pour traiter ce type d'objet. *(Figure extraite de [146])*.

utilisées pour modéliser les associations existantes entre jeux de données omiques et expliquer des variables observées. Pour cela, ces stratégies ont souvent recourt à la sélection de variables par LASSO avant d'apprendre un modèle, comme la régression de Cox [75]. Un enjeu important de l'intégration par concaténation consiste à déterminer une manière efficace de combiner les jeux de données. Dans [162], les auteurs étendent l'analyse canonique des corrélations parcimonieuse en introduisant une matrice d'association, déterminé par régression des moindres carrés, qui indique quels jeux de données doivent être connectés. Cependant, ces approches nécessitent que l'ensemble des données soient des matrices de variables continues.

Intégration par agrégation de résultats de modèles. Les méthodes de cette famille agrègent un ensemble de modèles, chacun généré sur un jeu de données différent. Pour cela, de nombreux algorithmes sont empruntés du domaine dit des ensembles d'experts. L'objectif de ces approches est très souvent de stabiliser les résultats d'un algorithme stochastique afin d'en améliorer les résultats, tout en diminuant leur variabilité. L'idée sous-jacente est qu'un ensemble d'experts, possédant chacun une expérience variée dans le même domaine, a une probabilité plus importante de fournir un résultat satisfaisant qu'un expert seul. À titre d'exemple, le *bagging*, présenté dans la section 2.3, fait partie de cette famille de méthodes. On notera $(\phi_b)_b = 1, \dots, B$ un ensemble de résultats de modèles obtenus sur les observations $(x_i)_{i=1,\dots,n}$. Dans un contexte de régression, l'agrégation peut se faire par la moyenne pondérée des prédictions. La fonction finale, $\phi^* : \mathcal{X} \to \mathbb{R}$, est alors définie par, $\forall\, x_i \in \mathcal{X}$, $\phi^*(x_i) = \sum_{b=1}^{B} \gamma_b \phi_b(x_i)$, avec $\gamma = (\gamma_1, \dots, \gamma_b) \in \mathbb{R}^B$ un vecteur de poids. Lorsque le modèle est un algorithme de classification supervisée, l'agrégation des B classifications en K classes peut se faire par vote majoritaire. La classification finale, $\phi^* : \mathcal{X} \to \{1, \dots, K\}$ est alors définie par, $\forall\, x_i \in \mathcal{X}$, $\phi^*(x_i) = \arg\max_{k=1,\dots,K} \sum_{b=1}^{B} \mathbb{I}_{\{\phi_b(x_i)=k\}}$.

Dans le cadre de la classification non supervisée, il n'y a pas de prédiction *a priori*, ce qui rend la comparaison de plusieurs résultats plus difficile, car il n'y a pas de transformation naturelle d'un étiquetage des classes à un autre. Ainsi, $\forall\, b = 1, \dots, B$, $\phi_b : \mathcal{X} \to \{1, \dots, K_b\}$ où le nombre de classes trouvées par l'algorithme ϕ_b, K_b, peut ou non dépendre de b. Dans ce cas, trois stratégies principales permettent de construire une classification finale ϕ^* : le ré-étiquetage suivie d'un vote majoritaire, le calcul d'une mesure de similarité entre les différentes classes et la recherche d'une partition médiane, [171].

Intégration par transformation. Cette dernière famille combine plusieurs jeux de données après les avoir transformées sous une forme intermédiaire, comme un graphe ou un noyau. Une fois les jeux de données convertis, un méta-graphe ou un méta-noyau peut être calculé et utilisé en entrée d'une méthode spécifique au traitement de ce type de représentation.

Dans le contexte des noyaux, l'apprentissage multi-noyaux regroupe un ensemble de méthodes permettant la combinaison convexe de M noyaux différents $(K^m)_{m=1,\dots,M}$ en un seul et même noyau K^*

$$K^* = \sum_{m=1}^{M} \beta_m K^m \quad \text{avec} \quad \left\{ \begin{array}{l} \beta_m \geq 0, \ \forall\, m = 1, \dots, M \\ \sum_{m=1}^{M} \beta_m = 1 \end{array} \right. . \tag{2.8}$$

Par définition, le noyau K^* résultant de la combinaison linéaire est aussi symétrique et positif et induit un espace image $\mathcal{H}$ et une fonction image associée ϕ^*. Afin d'éviter les effets d'échelle lors de l'intégration, les différents noyaux sont souvent préalablement normalisés à l'aide, par exemple, de la transformation cosinus [11] : $\tilde{K}_{ij} = \frac{K_{ij}}{\sqrt{K_{ii} K_{jj}}}$.

Les méthodes d'apprentissage multi-noyaux ont été développées dans le cadre supervisé, semi-supervisé et non supervisé. Cependant la grande majorité des recherches ont été conduites dans le cadre supervisé qui offre un objectif clair, *i.e.*, la minimisation de l'erreur de prédiction, pour la tâche de recherche de combinaison optimale.

Problématique abordée : Dans le travail présenté, trois propositions méthodologiques d'apprentissage multi-noyaux sont proposées dans un cadre non supervisé. Pour évaluer ces méthodes, huit jeux de données du projet *TARA oceans* ont été intégrés et analysés à l'aide d'une K-PCA. La description des méthodes et les résultats sont détaillés dans le chapitre 8, [106].

Chapitre 3
Contributions

Les contributions de ma thèse ont tout d'abord portés sur le SOM et sa version à noyau. Dans [105], je me suis intéressé à améliorer la stabilité du SOM numérique dans sa version stochastique, en utilisant des méthodes d'agrégation de résultats de modèles. J'ai ensuite proposé plusieurs variantes de la version stochastique du K-SOM afin de réduire sa complexité algorithmique [107] tout en améliorant l'interprétabilité du modèle résultant [108, 109]. Suite à ces travaux, je me suis intéressé à l'apprentissage multi-noyaux non supervisé afin de proposer des méthodes d'intégration de données omiques dans le contexte de l'écologie microbienne, [106].

La première section de ce chapitre présente succinctement les quatre contributions méthodologiques de ma thèse cherchant à améliorer le SOM et sa version à noyau. Le travail présenté dans la seconde section introduit trois nouvelles méthodes d'apprentissage multi-noyaux. L'efficacité de ces approches est illustrée en intégrant des données omiques de natures différentes dans un contexte d'écologie microbienne : le projet *TARA oceans*.

3.1 SOM, K-SOM et données massives

3.1.1 Améliorer la stabilité du SOM numérique

La méthode RoSyF, pour *Rotation and Symmetry Fusion*, cherche à améliorer la qualité et à réduire la variabilité des résultats du SOM numérique. Pour cela, elle agrège B résultats de SOM tout en préservant la topologie des cartes et en utilisant la variabilité de l'algorithme induite par différents états d'initialisation. Les B cartes entraînées sont tout d'abord ordonnées selon un critère de qualité, afin de sélectionner la carte référente : la meilleure carte selon ce critère, $\mathcal{M}^1$. Pour chacune des autres cartes, $(\mathcal{M}^b)_{b=2,\dots,B}$, la méthode recherche une transformation optimale, T_b^* dans l'ensemble des transformations, $\mathcal{T}$, composé de rotations et de symétries axiales :

$$T_b^* = \arg\min_{T\in\mathcal{T}} \frac{1}{U}\sum_{u=1}^{U} \|p_u^1 - T(p_u^b)\|^2.$$

Les prototypes finaux, $(p_u^*)_u$, sont calculés à l'aide de la moyenne des prototypes des B cartes obtenues par transformation optimale

$$\forall\, u = 1,\dots,U, \qquad p_u^* := \frac{1}{B}\sum_{b=1}^{B} T_b^*(p_u^b).$$

Différentes variantes de cette méthode sont proposées. Celles-ci définissent différents critères d'arrêt ainsi que différentes stratégies établissant l'ordre dans lequel les cartes sont fusionnées. RoSyF est comparée à plusieurs alternatives d'agrégation de résultats de SOM à l'aide de simulations dans le chapitre 4. Ce chapitre a fait l'objet d'une communication orale au $11^{ème}$ workshop WSOM 2016, [105].

3.1.2 Améliorer la complexité du K-SOM

Le K-SOM accéléré réduit la complexité de l'étape d'affectation du K-SOM de $\mathcal{O}(n^2U)$ à $\mathcal{O}(nU)$ pour une itération. Pour cela, cette phase d'affectation, définie par l'équation (2.7), peut être réécrite sous la forme

$$f^t(x_i) = \arg\min_{u\in 1,\dots,U} A_u^t - 2B_{ui}^t$$

où $A^t = \left(\sum_{j,j'=1}^{n} \gamma_{uj}^t \gamma_{uj'}^t K_{jj'}\right)_{u=1,\dots,U}$ est un vecteur de taille U et $B^t = \left(\sum_{j=1}^{n} \gamma_{uj}^t K_{i'j}\right)$ est une matrice de taille $(U \times n)$ pour tout $u = 1,\dots,U$ et $i' = 1,\dots,n$.

À l'initialisation de l'algorithme, A^0 et B^0 sont calculés et mis en mémoire pour un coût de stockage de $\mathcal{O}(U)$ et de $\mathcal{O}(nU)$. Cette étape, d'une complexité dominée par $\mathcal{O}(n^2)$ est réalisée une seule fois, et permet la mise à jour à faible coût computationnel de A^t et B^t à chaque itération lors de l'étape de représentation. Le gain de complexité ainsi obtenu permet d'accélérer de plus de 40 fois les temps de calcul pour traiter un jeu de données de 25000 observations, et ceci sans approximation.

La méthode correspondant à cette astuce, l'analyse de sa complexité ainsi que les simulations réalisées, sont détaillés dans le chapitre 5. Ce chapitre a fait l'objet d'une communication orale au XXV^{ème} symposium européen ESANN, [107].

3.1.3 Améliorer la complexité et l'interprétabilité du K-SOM

***Bagged* K-SOM.** Le *bagged* K-SOM exploite l'approche de *bagging*, présentée dans la section 2.3, qui permet d'obtenir une version du K-SOM qui est à la fois parallélisable et parcimonieuse. Pour cela, B

sous-échantillons de taille m sont obtenus, $(\mathcal{S}_b)_b$. Chacun d'entre eux est utilisé pour entraîner une carte de U neurones. Les prototypes, définis dans l'équation (2.6), peuvent alors s'écrire sous la forme $p_u^b = \sum_{x_i \in \mathcal{S}_b} \gamma_{ui}^b \phi(x_i)$ où ϕ est la fonction image associée au noyau K. Ces prototypes permettent alors de sélectionner les observations les plus pertinentes, qui sont utilisées pour entraîner une carte finale, dont le modèle permet la classification de l'ensemble des observations.

Cette approche est parallélisable, ce qui lui permet de réaliser des gains en temps de calcul important. Le *bagged* K-SOM permet aussi d'obtenir une représentation parcimonieuse des prototypes afin d'en faciliter l'interprétation. Pour cela, le nombre d'observations pertinentes sélectionnées doit rester faible. Dans le chapitre 6, la méthode est testée sur plusieurs jeux de données avant d'être comparée à des stratégies alternatives afin de pouvoir la valider. Ce chapitre a fait l'objet d'une communication orale au $10^{ème}$ workshop WSOM, [108].

K-PCA SOM Cette contribution cherche à définir les U prototypes du K-SOM dans un sous-espace engendré par la K-PCA, au lieu de l'espace image complet, $\mathcal{H}$. Plus précisément, les prototypes sont définis dans $A = \text{Span}\{a_1, \ldots, a_q\}$, où $(a_k)_{k=1,\ldots,q}$ correspond aux q premiers axes de la K-PCA réalisée sur les observations, $(x_i)_i$. Ces axes sont obtenus en résolvant le problème de décomposition en valeurs singulières présenté dans l'équation (2.5) : $\widetilde{K}\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}$. Les prototypes, définis dans l'équation (2.6), peuvent alors s'écrire sous la forme $p_u = \sum_{k=1}^{q} \gamma_{uk} a_k$. Cette approche correspond à la version numérique du SOM, avec pour entrée, la matrice $n \times q$, $\boldsymbol{\alpha}^\top \Lambda$, où $\Lambda = \text{Diag}\,(\lambda_1, \ldots, \lambda_q)$ et $\boldsymbol{\alpha}_{.i}$ est la i-ème colonne de $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_q]^\top$. La complexité du K-PCA SOM est ainsi réduite à la complexité de la version numérique du SOM dans $\mathbb{R}^q$, soit $\mathcal{O}(qU) + \mathcal{O}(nqU)$ pour une itération. Cependant, la complexité de la K-PCA est de $\mathcal{O}(n^3)$, complexité pouvant être ramenée à $\mathcal{O}(nm^2)$ grâce à l'approximation de Nyström, avec m le nombre d'observations sélectionnées pour réaliser l'approximation, comme présenté dans la section 2.3. L'interprétation des prototypes peut alors se faire de façon similaire à la PCA : les axes $a_1, \ldots, a_p$ des prototypes s'interprètent à l'aide des observations $(x_i)_i$ qui contribuent le plus à leur définition.

Dans le cadre de ce travail, le K-PCA SOM est comparé à une version directement parcimonieuse du K-SOM, présentée dans [124]. Cette méthode impose que chaque prototype, $u \in 1, \ldots, U$, soit défini à l'itération t par l'ensemble des observations les plus importantes déjà sélectionnées par l'algorithme, $I_u(t-1)$. Les prototypes s'écrivent alors sous la forme $p_u = \sum_{j \in I_u(t)} \gamma_{uj} \phi(x_i)$. Afin de maintenir la définition parcimonieuse des prototypes, ceux-ci sont régulièrement mis à jour et seulement les coefficients les plus importants sont conservés. L'algorithme complet enchaîne les étapes d'affectation, de représentation, toutes deux adaptées aux prototypes parcimonieux et une phase qui sélectionne les observations les plus importantes pour la définition des prototypes.

Les détails sur le K-PCA SOM ainsi que les simulations réalisées et la comparaison avec d'autres approches afin de valider la méthodes sont présentés dans le chapitre 7. Ce chapitre a fait l'objet d'une publication dans le journal *Neurocomputing*, [109].

3.2 Analyse exploratoire multi-omiques

Dans cette section, trois propositions méthodologiques multi-noyaux non supervisées sont présentées ainsi qu'une méthode permettant l'amélioration de l'interprétation de la K-PCA. Pour apprécier l'intérêt de ces méthodes, huit jeux de données du projet *TARA oceans* ont été intégrés et analysés à l'aide d'une K-PCA. Ces jeux de données sont de types différents, *i.e.*, tableaux de comptage et arbre phylogénétique, et de natures différentes, *i.e.*, micro-organismes planctoniques et variables physico-chimiques. Les résultats de ces travaux sont présentés dans le chapitre 8, qui fait l'objet d'une publication dans le journal *Bioinformatics*, [106].

3.2.1 Méthodes multi-noyaux non supervisées

Dans ce paragraphe, les méthodes proposées, nommées STATIS-UMKL, full-UMKL et sparse-UMKL, tirent profit des algorithmes d'apprentissage multi-noyaux présentés dans la section 2.4. Ces trois approches combinent, de façon générique, M noyaux différents, $(K^m)_{m=1,\dots,M}$, obtenus à partir de M jeux de données, en un seul et même noyau, K^*.

STATIS-UMKL. La première contribution propose un noyau consensus en utilisant une idée similaire à la méthode STATIS [134, 93]. Plus précisément, une mesure de similarité entre noyaux peut être obtenue en calculant leur cosinus selon le produit scalaire de Frobenius :

$$C_{mm'} = \frac{\langle K^m, K^{m'} \rangle_F}{\|K^m\|_F \|K^{m'}\|_F} = \frac{\text{Trace}(K^m K^{m'})}{\sqrt{\text{Trace}((K^m)^2)\text{Trace}((K^{m'})^2)}}.$$

La matrice $\mathbf{C} = (C_{mm'})_{m,m'=1,\dots,M}$ résultant peut être utilisée pour calculer le noyau K^* qui maximise la similarité moyenne entre tous les noyaux :

$$\begin{aligned} \text{maximise}_\beta \quad & \sum_{m=1}^{M} \left\langle K^*_{\mathbf{v}}, \frac{K^m}{\|K^m\|_F} \right\rangle_F = \mathbf{v}^\top \mathbf{C} \mathbf{v} \\ & \text{pour } K^*_{\mathbf{v}} = \sum_{m=1}^{M} v_m K^m \\ & \text{et } \mathbf{v} \in \mathbb{R}^M \text{ tel que } \|\mathbf{v}\|_2 = 1. \end{aligned} \tag{3.1}$$

Les coefficients $(\beta_m)_m$ peuvent alors être obtenus par $\beta = \frac{\mathbf{v}}{\sum_{m=1}^{M} v_m}$ afin de répondre à la contrainte de l'équation 8.1 du chapitre 2.

full-UMKL et sparse-UMKL. Les deux propositions méthodologiques de ce paragraphe préservent la topologie des données d'origine en calculant un noyau tel que la géométrie locale des données dans l'espace image, $\mathcal{H}$, définie pour l'observation x_i par

$$\Delta_i(\beta) = \left\langle \phi^*_\beta(x_i), \begin{pmatrix} \phi^*_\beta(x_1) \\ \vdots \\ \phi^*_\beta(x_N) \end{pmatrix} \right\rangle = \begin{pmatrix} K^*_\beta(x_i, x_1) \\ \vdots \\ K^*_\beta(x_i, x_N) \end{pmatrix},$$

soit similaire à celle des données d'origine. Cette dernière est approchée par le calcul de la matrice, $W = \sum_m^M A_k^m$, où A_k^m représente la matrice d'adjacence du graphe des k plus proche voisins du noyau K^m. Les coefficients $(\beta_m)_m$ peuvent alors être obtenus en résolvant le problème d'optimisation suivant :

$$\begin{aligned} \text{minimise}_\beta \quad & \sum_{i,j=1}^{N} W_{ij} \left\| \Delta_i(\beta) - \Delta_j(\beta) \right\|^2 \\ & \text{pour } K^*_\beta = \sum_{m=1}^{M} \beta_m K^m \\ & \text{et } \beta \in \mathbb{R}^M \text{ tel que } \beta_m \geq 0 \text{ et } \sum_{m=1}^{M} \beta_m = 1. \end{aligned} \tag{3.2}$$

Ici, l'approche sparse-UMKL applique une contrainte de norme L^1 sur les coefficients $(\beta_m)_m$, ce qui peut conduire à une solution parcimonieuse et ainsi permettre la sélection d'un sous-ensemble de noyaux. Pour une solution prenant en compte l'ensemble des noyaux disponibles, la version full-UMKL se fonde sur une contrainte sur la norme L^2.

3.2.2 Améliorer l'interprétabilité de la K-PCA

Dans ce travail, nous proposons une approche générique permettant d'étudier les variables influentes de l'ACP à noyau. Plus précisément, pour une mesure j, utilisé dans le calcul du noyau K^m, les valeurs observées sur cette mesure sont permutées aléatoirement entre tous les échantillons avant de recalculer le noyau perturbé, $\widetilde{K}^{m,j}$. Le méta-noyau perturbé est alors définie par $\widetilde{K}^* = \sum_{l \neq m} \beta_l K^l + \beta_m \widetilde{K}^{m,j}$. L'influence de la mesure j sur un sous-espace d'une composante principale, est calculée grâce à la distance de Crone-Crosby, [42], au niveau des axes : $\forall\, k = 1, \ldots, n$, $D_{cc}(\alpha_k, \widetilde{\alpha}_k) = \frac{1}{\sqrt{2}} \|\alpha_k - \widetilde{\alpha}_k\|$, avec α_k et $\widetilde{\alpha}_k$ représentant respectivement les vecteurs propres de la décomposition spectrale de K^* et de $\widetilde{K}^*$.

L'interprétation de la K-PCA se fait alors de la même façon que pour la PCA standard. Les axes $(a_k)_{k=1,\ldots,n}$, peuvent être interprétés par le biais des observations $(x_i)_{i=1,\ldots,n}$ qui contribuent le plus à leur définition, en considérant les variables importantes comme celles ayant les plus grandes distances de Crone-Crosby.

SOM, K-SOM et données massives

Chapitre 4

Aggregating Self-Organizing Maps with Topology Preservation

Résumé : Dans la version stochastique des cartes auto-organisatrices, différentes exécution de l'algorithme peuvent conduire à des résultats asses différents. Dans cet article, nous explorons une nouvelle approche qui agrège plusieurs résultats de l'algorithme afin d'améliorer la qualité et de réduire la variabilité des résultats. Notre approche utilise la variabilité de la méthode induite par différents états d'initialisation. A l'aide de simulations, nous démontrons que la méthode proposée parvient à améliorer les performances d'une exécution unique de l'algorithme tout en diminuant la variabilité de la solution finale. Une comparaison avec des méthodes existantes d'agrégation utilisant le bagging met aussi en évidence une bonne compétitivité des résultats.

Abstract : In the online version of Self-Organizing Maps, the results obtained from different instances of the algorithm can be rather different. In this paper, we explore a novel approach which aggregates several results of the SOM algorithm to increase their quality and reduce the variability of the results. This approach uses the variability of the algorithm that is due to different initialization states. We use simulations to show that our result is efficient to improve the performance of a single SOM algorithm and to decrease the variability of the final solution. Comparison with existing methods for bagging SOMs also show competitive results.

4.1 Introduction

Self-Organizing Maps (SOM), [86] have been shown to be powerful methods for analyzing high dimensional and complex data (see, for instance, [85] for applications of the method to many different areas). However, the method suffers from its lack of good convergence properties. In its original version, the theoretical convergence of the algorithm has only be proved in very limited cases [40] and even in the modified version in which the training of the SOM is expressed as an energy minimization problem [70], different runs of the algorithm give different results, that can be very dependent on the initialization. This problem is even more critical when the data set to be analyzed is complex or high dimensional.

This paper addresses the issue of aggregating several results of the SOM algorithm, all obtained on the same data set. Several attempts to combine SOMs while preserving their topological properties have been proposed in the literature [133, 153, 175, 10, 128]. In this paper, we present a novel method to combine several SOMs while preserving their topology. The proposed method combines several ideas taken from the different methods and allows to explore initialization states. It is both simple and efficient. We present a full comparison of the different options to aggregate the results of different SOMs and discuss the most relevant choices. Finally, we show that our approach is a competitive alternative to the existing methods on real data applications.

The remainder of the paper is organized as follows : in Section 4.2, an overview of aggregation methods for SOMs is presented. In Section 4.3, the proposed method is described. Finally, Section 4.4 presents experimental results and comparisons.

4.2 An overview of aggregation methods for SOMs

Suppose that B results of the SOM algorithm are given for the items $(x_i)_{i=1,\dots,n}$, $(\mathcal{M}^b)_{b=1,\dots,B}$. Each of these results, $\mathcal{M}^b$ is well defined by its set of prototypes $(p_u^b)_{u=1,\dots,U}$ and comes with an associated clustering function $\phi^b : x \in \mathbb{R}^d \to \arg\min_{u=1,\dots,U} \|x - p_u^b\|^2$. For the b-th SOM, the clusters will be denoted by $(\mathcal{C}_u^b)_{u=1,\dots,U}$, where $\mathcal{C}_u^b = \left\{x_i : \ \phi^b(x_i) = u\right\}$. The purpose is to build a *fused* or a *merged* map, $\mathcal{M}^*$, with prototypes $(p_u^*)_{u=1,\dots,U}$ and a clustering function ϕ^* which improves and summarizes the B maps into a unique consensual map. Note that all SOMs have been trained from the same data $(x_i)_{i=1,\dots,n}$ or from a subset (*e.g.*, a bootstrap sample) of this data set. They can also have been trained from different descriptors of the observations (*e.g.*, from different sets of variables observed on the same items) : in this case, the fused map thus corresponds to a map integrating the different descriptors. However, for the sake of simplicity, we will restrict our description and simulation to the first case (same observations, or eventually, bootstrap samples from the same observations and same descriptors).

As already explained in [133] in the context of a one-dimensional grid, there is no ground truth for cluster labelling in the unsupervised framework. A first strategy to overcome this issue is to perform a re-labelling of the clusters based on the clustering only : [153] merge together the clusters of different maps with a majority vote scheme. A "fused" prototype is defined as the centroid of the grouped cluster prototypes over $b = 1, \dots, B$ and a topology is deduced posterior to the definition of the clusters. Another approach that uses the different maps in an indirect way is described in [108] : in this paper, we proposed to use a subset of $(x_i)_i$, using the most representative observations of the set of B maps, to train a final SOM from a simpler and more robust data set. This method is well suited to handle very large data sets. However, both approaches do not necessary produce a map with a topology similar to the B merged SOMs and make use of only a small part of the information provided by the B learned SOMs.

Several attempts to explicitly take advantage of the prior (common) structure of the maps have been proposed in the literature. A first method consists in constraining the B SOMs to be as similar as possible by a common initialization. This initialization can be derived, for instance, from a PCA of $(x_i)_i$. Then, the different maps are fused by averaging the prototypes of the clusters situated at the same position the B SOMs [175] or by using a majority vote scheme to classify the observations [133]. Alternatively, [133, 10] also propose to make the B SOMs similar by initializing the b-th SOM with the final prototypes of the previous one. [10] improves this approach by weighting the averaging of the prototypes by a cluster quality index. Similarly, [67] uses a similar strategy to handle streaming or large data sets, splitting the data into several patches that are sequentially processed by a different SOM algorithm initialized with the result of the previous one. However, these methods do not allow to explore the possibilities of different initializations, which can be an issue in SOM. Moreover, a sequential initialization of the B SOMs prevents from training them in parallel, which can be an important issue if B is large : using a large B is advised for stabilizing the result of the algorithm.

Another approach to preserve the topology property of the map is to align the different maps on one of them, which serves as a reference for the topology : in [58], the map is chosen arbitrarily, and the other maps are fused sequentially to this first one, averaging the prototypes $(p_u^b)_u$ of the current map to the closest prototypes of the current fused map $(p_u^*)_u$. To leverage the problem of the choice of the map that is used to align the other maps, [128] proposes to choose a reference map that is the best one according to a given clustering quality criterion. However, this method makes the result strongly dependent on the choice of the first map because only its topology is used, whereas the topologies of the next maps are not utilized as such.

4.3 Description of the optimal transformation method

It is well known that the quality of the SOM strongly depends on its initialization. Given different maps obtained from different (random) initializations, we propose to find the "best" transformation that can be used to obtain two comparable results between two distinct maps. The optimal one-to-one transformation between prototypes in general might be difficult to define so we restrict ourselves to transformations that strictly preserve the topology of the map, *i.e.* the set of linear isometric transformations (rotation and/or symmetry). To do so, only square maps with m rows and columns are considered (*i.e.*, using the notations introduced in the previous section, $U = m^2$) : in these maps, the clusters are supposed to be positioned on a 2D grid at coordinates $\{(k_1, k_2)\}_{k_1,\, k_2=1,\dots,m}$.

Then, $\mathcal{T}$ denotes the set of all transformations, $T : \mathbb{R}^2 \to \mathbb{R}^2$, that let the map globally invariant : more precisely, $\mathcal{T}$ is composed of the set of rotations $\{r_\theta\}_{\theta\in\{0,\pi/2,\pi,3\pi/2\}}$ and of the transformations $\{r_\theta \circ s\}_\theta$, with s the symmetry with respect to the axis passing by the points $\left(\frac{m+1}{2}, 0\right)$ and $\left(\frac{m+1}{2}, m\right)$. For a given map $\mathcal{M}$ with prototypes $(p_u)_u$ and a given $T \in \mathcal{T}$, the transformed map $T(\mathcal{M})$ is the map in which the unit u, with coordinates (k_1^u, k_2^u) in $\mathbb{N}^2$, has a prototype denoted by p_u^T which is the prototype $p_{u'}$ of the original map, u' being the unit located at $T^{-1}(k_1^u, k_2^u)$.

When comparing two maps, the mean of the squared distances (in $\mathbb{R}^d$) between the prototypes of the two maps that are located at the same position is calculated. For two maps $\mathcal{M}$ and $\mathcal{M}'$, with respective prototypes $(p_u)_u$ and $(p'_u)_u$, we define a distance between two maps as the distance between their respective prototypes positionned at the same coordinates :

$$D\left(\mathcal{M}, \mathcal{M}'\right) = \frac{1}{m^2} \sum_{u=1}^{m^2} \|p_u - p'_u\|^2. \tag{4.1}$$

The best transformation between the current fused map and the next map to be fused is chosen according to this distance. The two maps are then fused using the optimal transformation before they

are merged, as described in Algorithm 3. The optimal transformation is found by computing the

Algorithm 3 Optimal transformation

1: *Initialization* $\mathcal{M}^{*,1} \leftarrow \mathcal{M}^1$
2: **for** $b : 2 \rightarrow B$ **do**
3: *Optimal transformation*

$$T_b^* := \arg\min_{T \in \mathcal{T}} D\left(\mathcal{M}^{*,b-1}, T(\mathcal{M}^b)\right)$$

4: *Fusion* between $\mathcal{M}^{*,b-1}$ and $T_b^*(\mathcal{M}^b)$. **Provides** : $\mathcal{M}^{*,b}$
5: **end for**
6: *Return* $\mathcal{M}^* := \mathcal{M}^{*,B}$

distance between the maps to be fused, $T_b^*(\mathcal{M}^b)$, and a reference map, which can be the first of the list, $\mathcal{M}^1$, for instance [1]. The fusion between the map is performed as suggested in [175] by averaging the prototypes located at the same position :

$$\forall\, u = 1, \ldots, m^2, \qquad p_u^* := \frac{1}{B} \sum_{b=1}^{B} p_u^{b,T}. \tag{4.2}$$

In the method described in the previous section, all maps are fused in an arbitrary order. However, as pointed out in [128], the maps may have very different qualities and may also be very different : merging a very peculiar map with a poor quality might lead to deterioration of the the results instead of improving them. In this section, two strategies are presented to leverage this problem.

The first one uses a measure of quality of the maps and first rank the maps from the one with the best quality to the one with the worse quality : $\mathcal{M}^{(1)}, \ldots, \mathcal{M}^{(B)}$. Standard quality measures for SOM can be used to perform this ranking [135] : i) the *quantization error* (QE), $\sum_{u=1}^{m^2} \sum_{i:\, x_i \in \mathcal{C}_u^*} \|x_i - p_u^*\|^2$, which is a clustering quality measure, disregarding the map topology ; ii) the *topographic error* (TE) which is the simplest of the topographic preservation measure : it counts the ratio of second best matching units that are in the direct neighborhood on the map of the best matching units for every $(x_i)_i$. However, for small maps and relatively simple problems, this measure has a small variability and can lead to many equally ranked maps.

Therefore, another approach is introduced to make a trade-off, while ranking the maps, between clustering and topographic qualities : the average rank of the maps is computed as :

$$r^b = \frac{r^b_{\text{quanti}} + r^b_{\text{topo}}}{2} \tag{4.3}$$

where r^b_{quanti} is the rank of the map $\mathcal{M}^b$ according to its quantization error (the best map is ranked first) and similarly for r^b_{topo} with the topographic error and the maps were finally ranked by increasing order of $(r^b)_b$.

Taking advantage of this ordering of the maps, the previous method can be modified using two different strategies :

1. the *similarity strategy* : following an idea similar to [128], the maps are merged by similarity : the merging process is initialized with the best map : $\mathcal{M}^{*,1} \leftarrow \mathcal{M}^{(1)}$. Then, this map is merged only with the maps that resemble this reference map. To do so, a simple ascending hierarchical clustering is performed between the maps $(T_b^*(\mathcal{M}^b))_{b=1,\ldots,B}$, with $(T_b^*)_b$ obtained by comparison with the reference map $\mathcal{M}^1$. This clustering is based on the distance introduced in (4.1) and the

1. The current fused map, $\mathcal{M}^{*,b-1}$ has also been used as a reference map, with no difference in the final result. Using $\mathcal{M}^1$ is thus a better strategy, because optimal transformation can be computed in parallel.

hierarchical tree is cut using the method described in [90]. Finally, the maps in the same cluster as $\mathcal{M}^{(1)}$ are fused to $\mathcal{M}^{*,1}$;

2. the *ordering strategy* : an alternative approach is performed sequentially by merging the maps by increasing rank $\mathcal{M}^{(1)}$, $\mathcal{M}^{(2)}$, ... The merging process is stopped at $\mathcal{M}^{(B')}$ with $B' \leq B$ (and usually $B' < B$) when the quality of the fused map $\mathcal{M}^{*,B'}$ would not increase anymore by merging it with $\mathcal{M}^{(B'+1)}$ (actually, two strategies are investigated : stopping when the quality measure is not increasing or stopping when the quality measure has not increased for the last 5%B fused maps).

4.4 Simulations

Methodology. In all the simulations, $B = 100$ maps are generated using the standard SOM. The optimal B has not been investigated in this paper and the number of fused maps was simply taken large enough so that the fusion makes sense. All maps were built with approximately $m = \sqrt{\frac{n}{10}}$ units and $5 \times n$ iterations of the stochastic algorithm and equipped with a Gaussian neighborhood controlled with the Euclidean distance between units on the grid. The size of the neighborhood was progressively decreased during the training. All simulations have been performed using the R package **SOMbrero**[2]. The 100 maps are then fused using one of the strategies described below and the performance of the different methods are finally assessed using various quality criteria for the resulting maps $\mathcal{M}^*$: i) two criteria already mentioned in Section 4.3 that are standard to measure the quality of the SOM : i) QE and TE ; ii) a criterion which uses the ground truth, when available (*i.e.*, an *a priori* group for the observations), the normalized mutual information (NMI) [43] between the unit of the map and the *a priori* group. This criterion quantifies the resemblance between the *a priori* group and the clustering provided by the SOM (it is comprised between 0 and 1, a value of 1 indicating a perfect matching between the two classifications). Note that this criterion must be interpreted with care because if the *a priori* groups are split between several units of the map, each of these units being composed of one group only (which is expected for SOM results), the criterion can be lower than when the groups are split between less units which are all composed of several groups (which would be a less expected result). Thus, this criterion has to be interpreted only together with the QE and the TE values.

The performance of the method is also assessed in term of *stability*. It is expected that several runs of one aggregating method give similar (thus stable) results. This stability is estimated in terms of : i) the distance between two final maps obtained from two different runs of the same method. If $\mathcal{M}^*$ and $\widetilde{\mathcal{M}}^*$ are two maps, the quantity $D(\mathcal{M}^*, T^*(\widetilde{\mathcal{M}}^*))$, where D is defined as in (4.1) and $T^* := \arg\min_{T \in \mathcal{T}} D(\mathcal{M}^*, T(\widetilde{\mathcal{M}}^*))$, is computed. This gives an estimation of the dissemblance between two maps from the prototype (hence the topological) perspective. If calculated over 250 different final maps, this quantity helps to quantify the stability of the final prototypes provided by a given aggregation method ; ii) the NMI between the final classes of two final maps obtained from two different runs of the same method. This gives an estimation of the dissemblance from the clustering perspective for a given aggregation method.

250 fusions for each method are performed using the methodology described above. This permits to compute average quality as stability criteria as well as to have an overview of the distribution of these criteria when the method is repeated.

Compared methods. The comparisons performed in this section aim at comparing our approach to existing ones (which are described in Section 4.2) as well as to investigate several options of the method (as discussed in Section 4.3).

2. `http://cran.r-project.org/web/packages/sombrero`, version 1.0.

First, our method, which merges several maps obtained from several initialization states, is compared to the standard bagging approach, in which several maps are trained from bootstrap samples from the similar initialization states. More precisely, bootstrap strategies are :

— the method denoted by **B-Rand**, which uses a common random initialization to learn $B = 100$ maps from 100 bootstrap samples coming from the original data set. Then, the prototypes that are positioned at the same coordinates, are averaged to obtain the final map $\mathcal{M}^*$ (as suggested in [175]) ;

— the method denoted by **B-PCA**, which uses a common PCA initialization to learn $B = 100$ maps from 100 bootstrap samples coming from the original data set (as suggested by [133]). The PCA initialization consists of initializing the prototypes by regularly positioning them along the coordinates of the projection of the data set on the first two axis of the PCA. Then, the prototypes that are positioned at the same coordinates, are averaged to obtain the final map $\mathcal{M}^*$;

— the method denoted by **B-Seq**, which uses a sequential initialization of the $B = 100$ maps : the first map is initialized randomly and trained with a bootstrap sample and the b-*th* map is initialized with the final prototypes of the $(b-1)$-*th* map and trained with another bootstrap sample. Finally, the final map $\mathcal{M}^*$, is obtained by averaging the prototypes of the $B = 100$ maps, that are positioned at the same coordinates, as suggested in [10].

These strategies are compared with our method and its bootstrap version, respectively denoted by **RoSyF** (for "Rotation and Symmetry Fusion") and **B-RoSyF**. **RoSyF** learns $B = 100$ maps, each from a different random initial state and using the whole data set $(x_i)_{i=1,\dots,n}$ and **B-RoSyF** learns $B = 100$ maps from 100 bootstrap samples coming from the original data set.

Finally, we also compare **RoSyF** with the approach consisting in selecting only one map from the B maps, the map supposed to be the best for instance. More precisely, using the $B = 100$ maps generated during the training of the **RoSyF** method, we selected one of the $B = 100$ maps i) randomly (this method is denoted by **Best-R**), ii) with the smallest QE (this method is denoted by **Best-QE** or iii) with the smallest TE (this method is denoted by **Best-TE**).

Datasets and results. This section compares the results obtained on two datasets coming from the UCI Machine Learning Repository [3] as available in the R package **mlbench** [4]. More precisely, the data "Glass" ($n = 214$, $d = 10$ and 7 *a priori* groups) [169] and the data "Vowel" ($n = 990$, $d = 10$ and 11 *a priori* groups) [121] are used. The SOM parameters are set to $m = 5$ and 1 000 iterations for "Glass" and $m = 10$ with 5 000 iterations for "Vowel". The different strategies, and especially the relevance of using different initial states instead of different bootstrap samples with the same initialization, is evaluated. The results are provided in Table 4.1.

First, note that for almost all quality criteria and datasets, **RoSyF** obtain better results than the methods based on different bootstrap samples (all differences are significant according to Wilcoxon test, risk 5%). **B-RoSyF** slightly deteriorates **RoSyF** performances. [39, 15] reported that the SOM algorithm is highly insensitive to initialization if run on the same data set as compared to what is obtained if bootstrap samples are used. However, it seems that the quality of the aggregated map is much better when different initial states are used on the same data set rather than different bootstrap samples with a common initial state, whatever this initial state is. Second, the TE obtained by **RoSyF** is always the lowest, just after the one obtained by **Best-TE** (which always selects the map with the lowest TE) but with a better QE and a better NMI. Again, all these differences are significant according to Wilcoxon tests (risk : 5%). On a clustering quality point of view, **RoSyF** is the method that obtains the second lowest quantization error, just after **Best-QE** which is designed to select the map with the lowest QE. Also, from a classification point of view, its performance is also very good : in average, **RoSyF** ranks first for the NMI criterion. Also note that all quality criteria have a low variability : the standard deviations is almost always the lowest : **RoSyF** is the method which has the best coefficient of variation (mean divided by the standard deviation) for all quality criteria.

3. `http://archive.ics.uci.edu/ml`
4. `http://cran.r-project.org/web/packages/mlbench`

Table 4.1 Method performance comparison (mean and standard deviation of different quality criteria ; QE has been multiplied by 100)

	B-Rand	B-PCA	B-Seq	B-RoSyF	RoSyF	Best-R	Best-QE	Best-TE
				"Glass"				
mean QE	855.10	855.93	854.97	609.84	597.81	595.09	**560.69**	593.68
sd QE	10.30	9.43	9.24	23.10	9.82	15.52	**5.45**	13.96
mean TE	11.95%	12.42%	11.77%	0.01%	0.01%	0.10%	0.04%	**0.00**%
sd TE	6.09%	6.53%	6.45%	0.04%	0.07%	0.24%	0.17%	**0.00**%
mean NMI	15.80%	15.77%	16.00%	**18.92**%	17.86%	15.64%	16.37%	15.87%
sd NMI	3.38%	3.15%	3.30%	2.09%	**1.38**%	2.20%	2.03%	2.21%
				"Vowel"				
mean QE	847.57	847.73	847.91	550.78	545.88	547.44	**531.30**	548.23
sd QE	11.82	10.88	11.63	5.18	**1.01**	7.10	2.39	6.72
mean TE	5.89%	6.06%	5.80%	0.07%	0.07%	0.19%	0.20%	**0.00**%
sd TE	3.62%	3.46%	3.37%	0.10%	0.08%	0.14%	0.14%	**0.00**%
mean NMI	7.11%	6.76%	7.03%	9.47%	9.57%	**9.64**%	9.53%	9.53%
sd NMI	1.44%	1.37%	1.49%	0.12%	**0.11**%	0.66%	0.54%	0.72%

Table 4.2 Method stability comparison (mean and standard deviation of different stability criteria ; D has been multiplied by 10 000)

	B-Rand	B-PCA	B-Seq	B-RoSyF	RoSyF	Best-R	Best-QE	Best-TE
				"Glass"				
mean D	70.85	67.22	**67.06**	149.65	67.07	2047.14	1302.27	1581.49
sd D	38.62	32.32	**31.24**	335.14	310.74	1557.08	1170.39	1186.28
mean NMI	64.77%	65.60%	65.88%	83.54%	**87.47**%	49.15%	54.41%	49.86%
sd NMI	6.37%	6.32%	6.23%	5.83%	**5.11**%	10.81%	9.57%	10.26%
				"Vowel"				
mean D	59.89	61.33	59.21	15.30	**11.07**	681.87	535.32	716.81
sd D	31.19	33.33	31.42	5.77	**3.87**	275.23	185.06	343.41
mean NMI	57.32%	56.83%	57.70%	90.83%	**92.39**%	72.53%	74.94%	72.11%
sd NMI	5.32%	5.21%	5.20%	1.59%	**1.33**%	3.29%	2.66%	3.37%

Table 4.2 (and Figure 4.1 for the dataset "Vowel") provides a comparison of the stability criteria. For this data set, **RoSyF** has the best stability, either in term of prototype stability (even though **B-PCA** and **B-Seq** also have a good prototype stability) and even more in term of class stability. These differences are significant according to Wilcoxon tests (risk : 5%). The results indicate that the method is indeed appropriate to improve the quality of the final map but also that it is very stable and gives very similar results if used several times, with different initializations of the prototypes and different training of the merged maps.

The relevance of stopping the merging process before all the maps have been fused has also been evaluated [5]. This comparison shows that there is only a small benefit in stopping the merging process before all maps have been used : most strategies lead to an highly deteriorated TE. Only stopping the training process when TE increases (**TE-Inc**) or based on the similarity strategy described in 6.2 are

5. For the sake of paper length, detailed results are not reported but only described.

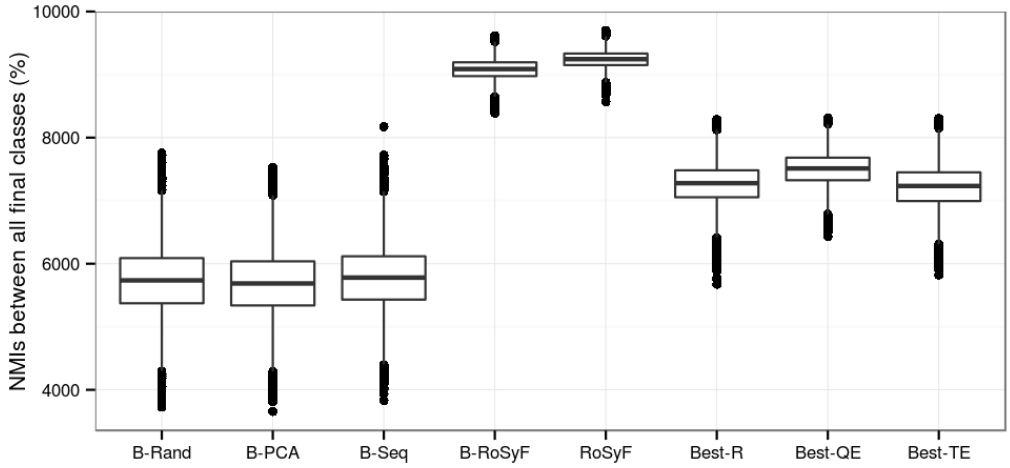

Figure 4.1 Normalized mutual information (NMI) between pairs of clusterings obtained from the 250 final maps generated by the different approaches.

valid approaches in terms of quality criteria. However, a stability analysis shows that all these strategies strongly deteriorate the stability of the final map : merging all maps is the approach that provides the best stability, either in term of prototype comparison than in term of class comparison, except for **TE-Inc** which provides a slightly more stable clustering but very different prototypes. All these strategies use only few maps (less than 10 maps in average), except again **TE-Inc** which uses 89.4 maps in average for the "Glass" dataset and is thus very close to the maximum number of available maps (100). Actually, additional simulations (not shown for the sake of paper length) merging more than 100 maps proved that the stability increases with the number of fused maps (up to a certain number which was for our dataset between 500 to 1000 maps). A trade-off has thus to be found between computational time required to generate a large number of maps and stability of the results. This question is still under study.

4.5 Conclusion

Although most work on SOM ensembles are based on bootstrapping techniques, this paper presents an approach allowing to explore different initial states for the map. The method improves the stability of the fused map, both in term of prototypes and in terms of clustering. We are currently investigating how to choose an optimal number B of maps to fuse as well as weighting schemes based on various quality criteria : this approach is already promising to improve the results, especially the stability of the final map.

Chapitre 5
Accelerating stochastic kernel SOM

Résumé : L'analyse de données non vectoriel est devenue commun dans de nombreuses applications modernes. Plusieurs méthodes se basant sur des prototypes ont été étendu afin de répondre à ce besoin en méthodes à noyau qui permettent d'embarquer les données dans un espace Euclidien implicitement définie. Un inconvénient majeur de ces approches réside dans leur complexité, qui est très souvent quadratique ou cubique en nombre d'observations. Dans cet article, nous proposons une méthode efficace afin de réduire la complexité de l'algorithme des cartes auto-organisatrices stochastique à noyau. Les résultats sont illustrés sur de grands jeux de données et sont comparés à la version standard de l'algorithme. L'approche est implémenté dans la dernière version du package R **SOMbrero**[a].

Abstract : Analyzing non vectorial data has become a common trend in a number of real-life applications. Various prototype-based methods have been extended to answer this need by means of kernalization that embed data into an (implicit) Euclidean space. One drawback of those approaches is their complexity, which is commonly of order the square or the cube of the number of observations. In this paper, we propose an efficient method to reduce complexity of the stochastic kernel SOM. The results are illustrated on large datasets and compared to the standard kernel SOM. The approach has been implemented in the last version of the R package SOMbrero.

a. `https://cran.r-project.org/web/packages/SOMbrero`, version 1.2

5.1 Introduction

In a number of real-life applications, data cannot be described by numerical variables or cannot be compared in a meaningful way using standard Euclidean metrics. This is the case, for instance, with graphs, trees, categorical data, ... A generic way to handle this type of dataset is to use a measure of similarity or dissimilarity between the data, which can be expertly designed.

Self-organizing maps (SOM) have first been extended to the framework of non numerical data through the median SOM approach [84, 34]. The method replaces the standard computation of the prototypes by an approximation in the original dataset. However, the representation of the prototypes is very restricted and generates representation issues and therefore poor performances. A different approach is taken in the relational and kernel versions of SOM [65, 103, 92, 67, 123]. In these versions, the original data are embedded into an (implicit) Euclidean or pseudo-Euclidean space in which the algorithm is performed using only the dissimilarity or the kernel. A side effect of this method is that it requires that the prototypes are expressed as a convex combination of the original data, leading to a quadratic complexity for the batch and the on-line version of the algorithm [148]. The induced computation cost is a serious bottleneck to analyze datasets with more than a few thousands observations.

In the present paper, we propose an efficient method to reduce the complexity of the stochastic kernel SOM (K-SOM) so that it is linear in the number of observations. The approach is based on an efficient on-line update of the prototypes, which only requires to store two matrices. The standard K-SOM algorithm is briefly described in Section 5.2 and our proposal for reducing its computational complexity is discussed in Sections 5.3 (kernel version) and 5.4 (adaptation to arbitrary dissimilarity datasets). Finally, the approach is illustrated in Section 5.5.

5.2 A brief description of the stochastic K-SOM

In the sequel, we consider a set of observations $(x_i)_{i=1,\dots,n}$ taking values in an arbitrary space $\mathcal{X}$. $\mathcal{X}$ is equipped with a kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that provides pairwise similarity between the observations, $K_{ij} := K(x_i, x_j)$. K is assumed symmetric and positive which ensures the existence of a unique Hilbert space, $(\mathcal{H}, \langle ., . \rangle_{\mathcal{H}})$, and a unique mapping ϕ from $\mathcal{X}$ into $\mathcal{H}$ such that $K(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$ [7].

In K-SOM [103, 92], the U prototypes $(p_u)_{u=1,\dots,U}$, representing the units of the low dimensional grid on which the observations are projected, are expressed as convex combinations of the images of $(x_i)_{i=1,\dots,n}$ by ϕ : $p_u = \sum_{i=1}^{n} \alpha_{ui} \phi(x_i)$, with $\alpha_{ui} \geq 0$ and $\sum_{i=1}^{n} \alpha_{ui} = 1$.

In the on-line version of the algorithm, the following two steps are performed for each iteration t, starting from randomly chosen values α_{ui}^0 :

— the assignment step in which an observation x_i is picked at random and affected to its closest prototype, $f^t(x_i) = \arg\min_{u=1,\dots,U} \|\phi(x_i) - p_u^t\|_{\mathcal{H}}^2$. This step is equivalent to $f^t(x_i) = \arg\min_{u=1,\dots,U} \sum_{j,j'=1}^{n} \alpha_{uj}^t \alpha_{uj'}^t K_{jj'} - 2 \sum_{j=1}^{n} \alpha_{uj}^t K_{ij}$, for $K_{ij} = K(x_i, x_j)$;

— the representation step in which the prototypes are updated according to a gradient descent-like approach : $\forall u = 1, \dots, U$, $p_u^{t+1} \leftarrow p_u^t + \mu(t) h^t(d(f^t(x_i), u))(\phi(x_i) - p_u^t)$, which is equivalent to $\alpha_u^{t+1} \leftarrow \alpha_u^t + \mu(t) h^t(d(f^t(x_i), u))(\mathbf{1}_i - \alpha_u^t)$ for d the distance between units on the grid, h^t a decreasing function such that $h^t(0) = 1$ and $\lim_{x \to +\infty} h^t(x) = 0$, $\mathbf{1}_i$ the n-dimensional vector in which only the entry indexed by i is non zero and equal to 1 and μ is a positive number. Usually $\mu(t)$ and h^t are chosen so as to vanish when t increases.

The complexity of the assignment and representation steps are, respectively, $\mathcal{O}(n^2 U)$ and $\mathcal{O}(nU)$, which leads to a total complexity of $\mathcal{O}(n^2 U)$ for one iteration. To obtain good convergence properties, the algorithm requires a number of iterations of the order of βn, as shown in [126], yielding a

complexity of $\mathcal{O}(\beta n^3 U)$. Hence, the K-SOM is not adapted to large datasets and cannot be used to analyze more than a few thousands observations. [109] have proposed two approximate versions to overcome this issue, using sparse representations of the prototypes or DR pre-processing techniques. In the present article, we propose a modification of the SOM update steps so as to produce a solution which is exactly equivalent to the original algorithm, but with a reduced computational time.

5.3 Reducing the complexity of stochastic K-SOM

Using a re-formulation of the assignment step, the computational cost of one iteration can be reduced to $\mathcal{O}(nU)$. At iteration t, an observation x_i is picked at random within $(x_{i'})_{i'=1,\dots,n}$ and the assignment step is written $f^t(x_i) = \arg\min_{u\in 1,\dots,U} A^t_u - 2B^t_{ui}$ in which $A^t = \left(\sum_{j,j'=1}^n \alpha^t_{uj}\alpha^t_{uj'}K_{jj'}\right)_{u=1,\dots,U}$ is a vector of size U and $B^t = \left(\sum_{j=1}^n \alpha^t_{uj}K_{i'j}\right)_{u=1,\dots,U,\, i'=1,\dots,n}$ is a $(U \times n)$-matrix.

The updates of A^t and B^t are performed during the representation step, which is thus equivalent to $\forall u = 1,\dots,U, \quad \alpha^{t+1}_u = (1-\lambda_u(t))\alpha^t_u + \lambda_u(t)\mathbf{1}_i$, in which $\lambda_u(t) = \mu(t)h(d(f^t(x_i),u))$. This leads to the following updates :

$$B^{t+1}_{ui'} = \sum_{j=1}^n \alpha^{t+1}_{uj}K_{i'j} = (1-\lambda_u(t))B^t_{ui'} + \lambda_u(t)K_{i'i},$$

and the vector A is modified using the equation given by

$$A^{t+1}_u = \sum_{j,j'=1}^n \alpha^{t+1}_{uj}\alpha^{t+1}_{uj'}K_{jj'} = (1-\lambda_u(t))^2 A^t_u + \lambda_u(t)^2 K_{ii} + 2\lambda_u(t)(1-\lambda_u(t))B^t_{ui}.$$

Thus, the computational cost of the algorithm can be decomposed into :

— computing $A^0_u = \sum_{j,j'=1}^n \alpha^0_{uj}\alpha^0_{uj'}K_{jj'}$ and $B^0_{ui'} = \sum_{j=1}^n \alpha^0_{uj}K_{i'j}$ for all $u = 1, \dots, U$ and all $i' = 1, \dots, n$. This step is performed only once and has a complexity of $\mathcal{O}(n^2)$ and $\mathcal{O}(n)$ for A^0 and B^0, respectively ;

— performing the assignment step which complexity does not depend on n since the distances are pre-computed and stored ;

— the update of A^t and B^t (representation step), which have respective complexities equal to $\mathcal{O}(U)$ and $\mathcal{O}(nU)$.

Since the assignment and representation steps are usually performed $\mathcal{O}(\beta n)$ times, the total complexity of the algorithm is dominated by $\mathcal{O}(\beta n^2 U)$. This computational cost is obtained using the additional storage of A^t and B^t which requires a memory of $\mathcal{O}(U)$ and $\mathcal{O}(nU)$, respectively.

5.4 The case of dissimilarity data

In some cases, the dataset is described by a measure of dissimilarity ($\delta(x_i, x_j) = \delta_{ij}$), rather than by a kernel. We will suppose that the dissimilarity is symmetric ($\delta_{ij} = \delta_{ji}$), with positive elements ($\delta_{ij} \geq 0$) and a null diagonal ($\delta_{ii} = 0$) but it might be not Euclidean. In [67, 123] are described relational versions of the SOM algorithm that are adapted to this case and are based on a pseudo-

Euclidean framework [62]. The principle is fairly similar to the one described in Section 5.2 except that the assignment step writes :

$$f^t(x_i) = \arg\min_{u=1,\dots,U} \left(\sum_{j=1}^{n} \alpha_{uj}^t \delta_{ij} - \frac{1}{2} \sum_{j,j'=1}^{n} \alpha_{uj}^t \alpha_{uj'}^t \delta_{jj'} \right)$$

(the representation step is unchanged compared to the kernel version). The complexity of the method is thus identical to the one of the kernel SOM. In the case where the dissimilarity is computed from a kernel K by $\delta_{ij} = K_{ii} + K_{jj} - 2K_{ij}$, the dissimilarity SOM and the relational SOM are identical.

The adaptation of kernel SOM described in Section 5.3 has a straightforward equivalent in the case of the relational SOM : $A^t = \left(\sum_{j,j'=1}^{n} \alpha_{uj}^t \alpha_{uj'}^t \delta_{jj'}\right)_{u=1,\dots,U}$ and $B^t = \left(\sum_{j=1}^{n} \alpha_{uj}^t \delta_{i'j}\right)_{u=1,\dots,U,i'=1,\dots,n}$. The assignment step is thus $f^t(x_i) = \arg\min_{u=1,\dots,U} \left(B_{ui}^t - \frac{1}{2} A_u^t\right)$ and the updates of A^t and B^t are performed during the representation step as described in Section 5.3.

5.5 Application

In this section, different simulations are performed to compare the standard K-SOM to the accelerated version described in Sections 5.3 and 5.4. To assess the computational cost of the methods, three large size datasets are used :

— a graph, denoted by "polblogs", in which the 1,222 nodes are blogs on US politics (recorded in 2005 by [2] [1]). The edges in this graph represent hyper-links between the blogs. For this dataset, the shortest path lengths between pairs of nodes have been computed and used in the relational version of this algorithm. This dissimilarity is not Euclidean.

— a DNA barcoding dataset, denoted by "cowrie", that contains 2,036 samples issued from the cowries family introduced in [116]. DNA barcoding data are composed of sequences of nucleotides, *i.e.*, sequences of "A", "C", "G", "T" letters in high dimension (hundreds or thousands of sites) allowing to assign biological specimens to a species. Only 1,414 samples were used (those corresponding to species with very few observations were removed). The Kimura-2P [82] dissimilarity is computed between pairs of sequences and used in the relational version of the algorithm. Again, this dissimilarity is not Euclidean.

— a (standard) numerical dataset, denoted by "wine", which is related to red variants of the Portuguese "Vinho Verde" wine [37] [2]. The dataset contains 4 898 observations of 12 numeric variables based on physicochemical tests, such as the pH, the sulphates or the residual sugar. The Gaussian kernel has been computed between any pair of wines : $K_{ij} = e^{-\sigma \|x_i - x_j\|^2}$ with σ equals to the median of $\left\{ \frac{1}{\|x_i - x_j\|^2} \right\}_{i<j}$.

Hence, the first two datasets are true relational dataset which requires to use an adapted version of the SOM algorithm to be processed, whereas the third one is a numeric dataset, which can be processed in a standard manner with the original SOM algorithm.

All simulations have been performed with the R package **SOMbrero** [3] that contains implementation of the standard numeric SOM and of the relational version of the algorithm. **SOMbrero** uses a stochastic learning, as is described in this article. Version 1.1 has been used to obtain the computational time provided by the original version of the algorithm and version 1.2 to obtain the computational time provided by the accelerated version. For a fixed random seed, the results of both versions (version

1. The graph is available at `http://www-personal.umich.edu/~mejn/netdata/polblogs.zip`.
2. The dataset is available at `https://archive.ics.uci.edu/ml/datasets/Wine+Quality`.
3. `https://cran.r-project.org/web/packages/SOMbrero`

Table 5.1 CPU time in seconds of the different versions of the SOM algorithm (average over 100 maps and standard deviation between parenthesis)

	numeric SOM	rSOM	acc. rSOM
"polblogs"	*NA*	1112.34 (165.86)	**36.99** (4.49)
"cowrie"	*NA*	1715.40 (249.60)	**42.52** (2.32)
"Wine"	**16.40** (2.13)	8527.16 (874.58)	206.32 (38.24)

1.1 with the standard implementation and version 1.2 with the accelerated implementation) were always exactly identical for the dissimilarity versions.

All maps were trained for a 10×10 grid with respectively 6000 ("polblogs"), 7000 ("cowrie") and 25000 ("wines") iterations. All grids were equipped with a piecewise linear neighborhood with the Euclidean distance between units on the grid calculated using the unit coordinates in $\mathbb{N}^{*2}$. Following the same methodology, 100 maps were trained using the standard numeric version of the SOM on "wine", the only numerical dataset, as a basis for comparison : the computational cost of the numeric version of the algorithm is $\mathcal{O}(\beta n \times Up)$ in which $p \ll n$ is the number of variables in the dataset (see [148]). It is thus expected that this direct approach is still faster than the relational version. Results (in terms of clustering on the map) are also different between the relational version and the numeric version but these differences are related to the choice of a good dissimilarity in a given dataset, which is out of the scope of this paper. We thus only report computational times here.

Table 5.1 provides the computational cost in seconds obtained over the 100 maps (mean and standard deviation) for the numeric SOM, the standard relational or kernel SOM and the accelerated version. *NA* (not applicable) values in the numeric SOM column of both "polblogs" and "cowrie" datasets come from the fact that these datasets are not in the framework of the numeric SOM.

Results show that the accelerated version allows to highly reduce the computational time of the relational or kernel SOM. When comparing the results obtained on the different datasets, the accelerated version is 30 times faster than the original approach on "polblogs" and more than 40 times faster on both "cowrie" and "wine". Compared to the numerical version, the computational cost is increased but still comparable.

5.6 Conclusion

We have proposed an efficient version of the stochastic K-SOM and relation SOM, with a complexity of $\mathcal{O}(\beta n^2 U)$. The experiments performed on several datasets demonstrate that the presented method strongly decreases the overall computational time. With the introduction of this method, a step is taken to allow the SOM algorithm to deal with massive non numerical datasets.

Chapitre 6
Bagged kernel SOM

Résumé : Dans de nombreuses applications modernes, l'utilisateur s'intéresse à l'analyse de données non vectorielles pour lesquelles les noyaux représentent un outils efficace, permettant d'embarquer les données dans un espace Euclidien implicitement définie. Cependant, lorsque ceux-ci sont utilisés par des méthodes se basant sur des prototypes, le temps de calcul devient fonction du nombre d'observations (car les prototypes s'expriment comme combinaisons convexes des données d'origines). Autre inconvénient de la méthode, les prototypes ne sont plus interprétable. Dans cet article, nous proposons une solution à ces deux problématiques par l'utilisation d'une approche de bagging. Les résultats sont illustrés sur des jeux de données simulés et sont comparés à des méthodes alternatives existantes dans la littérature.

Abstract : In a number of real-life applications, the user is interested in analyzing non vectorial data, for which kernels are useful tools that embed data into an (implicit) Euclidean space. However, when using such approaches with prototype-based methods, the computational time is related to the number of observations (because the prototypes are expressed as convex combinations of the original data). Also, a side effect of the method is that the interpretability of the prototypes is lost. In the present paper, we propose to overcome these two issues by using a bagging approach. The results are illustrated on simulated data sets and compared to alternatives found in the literature.

6.1 Introduction

In a number of real-life applications, the user is interested in analyzing data that are non described by numerical variables as is standard. For instance, in social network analysis, the data are nodes of a graph which are described by their relations to each others. Self-Organizing Maps (SOM) and other prototype based algorithms have already been extended to the framework of non numerical data, using various approaches. One of the most promising one is to rely on kernels to map the original data into an (implicit) Euclidean space in which the standard SOM can be used [103, 92, 18]. A closely related approach, called "relational SOM" [67, 123], extends this method to dissimilarity data which are pertaining to a pseudo-Euclidean framework, as demonstrated in [67]. Further, in [125], we addressed the issue of using several sources of (possibly non numeric) data by combining several kernels. The combination of kernels is made optimal with a stochastic gradient descent scheme that is included in the on-line version of the SOM algorithm.

However, while able to handle non Euclidean data, that can eventually come from different sources, these approaches suffer from two drawbacks : as pointed out in [110], when the data set is very large, the computational time of such approaches can be prohibitive. Indeed, prototypes are expressed as convex combinations of the original data and are thus expressed with a number of coefficients equal to the number of observations in the data set. Also, adding an extra gradient descent step to optimize the kernel combination requires to increase the number of iterations of the algorithm, which yields to an augmented computational time. The second drawback is emphasized in [72] : as the prototypes are expressed as a convex combination of the original data, they are no longer given as explicit representative points in the data space and the interpretability of the model is lost.

In the present paper, we propose to overcome these two issues by using a bagging approach in which only a small subset of the original data set is used. The results coming from several bags are combined to select the most representative observations that are then utilized to define the prototypes in a final map. This approach is both sparse (the resulting map is based on a small subset of observations only), fast and parallelizable, which makes it an interesting approach to analyze large samples. The rest of the paper is organized as follow : Section 6.2 describes the method and its relations to previous approaches in the literature. Section 6.3 provides the analysis of the results obtained on simulated data sets and on a real-world data set which is a graph.

6.2 Method

6.2.1 A brief description of the kernel SOM approach

Let us suppose that we are given input data, $(x_i)_{i=1,\dots,n}$ taking values in an arbitrary space $\mathcal{G}$. When $\mathcal{G}$ is not Euclidean, a solution to handle the data set $(x_i)_i$ with standard learning algorithms is to suppose that a *kernel* is known, i.e., a function $K : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ which is symmetric ($\forall\, z, z' \in \mathcal{G}$, $K(z, z') = K(z', z)$) and positive ($\forall\, N \in \mathbb{N}$, $\forall\, (z_j)_{j=1,\dots,N} \subset \mathcal{G}$ and $\forall\, (\alpha_j)_{j=1,\dots,N} \subset \mathbb{R}$, $\sum_{j,j'} \alpha_j \alpha_{j'} K(z_j, z_{j'}) \geq 0$). When such conditions are fulfilled, the so-called kernel defines a dot product in an underlying Hilbert space : more precisely, there exists a Hilbert space $(\mathcal{H}, \langle ., . \rangle_{\mathcal{H}})$, called the *feature space*, and a function $\phi : \mathcal{G} \to \mathcal{H}$, called the *feature map*, such that

$$\forall\, x, x' \in \mathcal{G}, \qquad \langle \phi(x), \phi(x') \rangle_{\mathcal{H}} = K(x, x')$$

(see [7]). Hence using the kernel as a mean to measure similarities between data yields to implicitly rely on the Euclidean structure of $\mathcal{H}$. Many algorithms have been *kernelized*, i.e., modified to handle

(possibly non vectorial) data described by a kernel. In particular, the general framework of kernel SOM is described in [103, 92, 18]. In this framework, as in the standard SOM, the data are clustered into a low dimensional grid made of U neurons, $\{1, \dots, U\}$ and these neurons are related to each other by a neighborhood relationship on the grid, h. Each neuron is also represented by a prototype p_u (for some $u \in \{1, \dots, U\}$) but unlike standard numerical SOM, this prototype does not take value in $\mathcal{G}$ but in the previously defined feature space $\mathcal{H}$. Actually, each prototype is expressed as a convex combination of the image of the input data by the feature map :

$$p_u = \sum_{i=1}^{n} \gamma_{ui} \phi(x_i), \qquad \text{with} \qquad \gamma_{ui} \geq 0 \text{ and } \sum_i \gamma_{ui} = 1.$$

In the on-line version of the algorithm, two steps are iteratively performed :

— **an affectation step** in which an observation x_i is picked at random and affected to its closest prototype using the distance induced by K :

$$f(x_i) = \arg\min_u \|p_u - \phi(x_i)\|^2_{\mathcal{H}},$$

where $\|p_u - \phi(x_i)\|^2_{\mathcal{H}} = K(x_i, x_i) - 2\sum_l \gamma_{uj} K(x_i, x_j) + \sum_{jj'} \gamma_{uj}\gamma_{uj'} K(x_j, x_{j'})$.

— **a representation step** in which the prototypes are updated according to their value at the previous step t and to the observation chosen in the previous step. A gradient descent-like step is used for this update :

$$\forall\, u = 1, \dots, U, \qquad \gamma_u^{t+1} = \gamma_u^t + \mu(t) h^t(f(x_i), u) \left(\delta_i^n - \gamma_u^t\right)$$

where δ_i^n is the n-dimensional vector in which only entries indexed by i is non zero and equal to 1, $\mu(t) \sim 1/t$ is a vanishing coefficient and, usually, the neighborhood relationship h^t also vanishes with until being restricted to the neuron itself.

Note that this algorithm has also been extended to the case where the observations are described by several kernels, each corresponding to one particular type of data, in the *multiple Kernel SOM* algorithm [125]. In this algorithm, an additional gradient descent step is added to the algorithm to tune the respective contribution of each kernel in an optimal way.

6.2.2 Ensemble of SOMs

Despite their generalization properties to complex data, kernel SOM and related methods are not well-suited for large data sets since the algorithms generally require the storage of the entire Gram matrix and since the prototypes are expressed as convex combinations of the input data and thus have a very high dimension. Another important drawback of the prototype representation is that, being expressed as a very large linear correlation of the mapped input data, they are not easy to interpret. Indeed, for non vectorial data, such as e.g., nodes in a graph whose similarities can be described by several types of kernels (see [163]), there is no way to describe the prototypes in terms of an object in the input space (here, the graph). As prototypes are commonly used to deciphter the clusters' meaning, one of the main interesting feature of the SOM algorithm is lost in the process, as pointed out in [72].

Several techniques have been proposed in the literature to overcome the dimensionality issues, which can be adapted to the kernel SOM framework : some are related to sparse representations and some to bootstraping and bagging. In [67], the large size of the data set is handled using "patch clustering", which is particularly suited for streaming data but can also be used to handle large dimensional data. The initial data set is randomly split into several patches, $\mathcal{P}_b$ and the algorithm processes each patch iteratively. At step b, the b-th patch is clustered until convergence. Each of the resulting prototypes is approximated by the closest P input data points. During the next step, the index

set of the P-approximations of all prototypes and the index set of the next patch $\mathcal{P}_{b+1}$ are put together into the extended patch $\mathcal{P}^\star_{b+1}$ and the clustering process is performed on all the observations indexed by $\mathcal{P}^\star_{b+1}$. This is iterated until all patches are clustered. This approach leads to good clustering results, however it is not parallelizable and the algorithm may be sensitive to the order in which patches are processed. Another technique for handling large data sets is to use bootstrap and bagging. In [133], bagging is applied to a batch version of the SOM algorithm for numerical data in a semi-supervised context. The prototypes of the map trained on the first bag are initialized to lie in the first principal component and each trained map is used to initialize the subsequent map for the subsequent bag. This procedure reduces the dimensionality and improves the classification error, but it is not parallelizable. Alternatively, [175, 10] propose by combining SOM based on separate bootstrap samples with a fusion of their prototypes. These approach, which can be used in parallel are however only valid if the prototypes are expressed on the same representation space, which is not directly generalizable when using kernel SOM in which prototypes are directly expressed with the bootstrap sample.

6.2.3 Bagged kernel SOM

Our proposal is to use a bagging approach that is both parallelizable and sparse. Bagging uses a large number of small sub-samples, all randomly chosen, to select the most relevant observations : B subsets, $(\mathcal{S}_b)_b$ each of size $n_B \ll n$, are built, at random, within the original data set $\{x_1, \ldots, x_n\}$. Using the on-line algorithm described in [173], a map with U neurons is trained, which results in the prototypes $p_u^b = \sum_{x_i \in \mathcal{S}_b} \gamma_{ui}^b \phi(x_i)$ where ϕ is the feature map associated with the kernel K. The most representative observations are chosen as the first P largest weights for each prototype : $\forall\, u = 1, \ldots, U$,

$$\mathcal{I}_u^b := \left\{ x_i : \ \gamma_{ui} \text{ is one of the first } P \text{ largest weights among } (\gamma_{uj}^b)_{x_j \in \mathcal{S}_b} \right\},$$

and $\mathcal{I}_b = \cup_u \mathcal{I}_u^b$. Alternative methods to select the most interesting prototypes are reported in [72] ; the one we chose is referred in this paper as the *K-convex hull* but it would be interesting to test other methods for selecting the most interesting observations.

Then, the number of times each observation $(x_i)_{i=1,\ldots,n}$ is selected in one sub-sample is computed : $\mathcal{N}(x_i) := \sharp\{b : \ x_i \in \mathcal{I}_b\}$ which is finally used as a quality criterion to select the most important variables which are the first $P \times U$ observations with the largest values for $\mathcal{N}(x_i)$:

$$\mathcal{S} := \{x_i : \mathcal{N}(x_i) \text{ is one of the first } PU \text{ largest numbers among } (\mathcal{N}(x_j))_{j \geq n}\}.$$

A final map is then trained with the selected observations in $\mathcal{S}$ which has prototypes expressed as $p_u = \sum_{x_i \in \mathcal{S}} \gamma_{ui} \phi(x_i)$. The final classification for all observations $(x_i)_{i=1,\ldots,n}$ is deduced from these prototypes by applying the standard affectation rule : $\mathcal{C}(x_i) := \arg\min_{u=1,\ldots,U} \|p_u - \phi(x_i)\|^2$ where $\|p_u - \phi(x_i)\|^2_{\mathcal{H}}$ is computed using K, as described in Section 6.2.1. The algorithm is described in Algorithm 4.

Note that, strictly speaking, only the sub-kernels $\mathbf{K}_{\bar{\mathcal{S}},\mathcal{S}} := (K(x_i, x_j))_{i \notin \mathcal{S}, j \in \mathcal{S}}$ and $\mathbf{K}_{\mathcal{S}} = (K(x_j, x'_j))_{j,j' \in \mathcal{S}}$ are required to perform the final affectation step because the closest prototype does not depend on the term $K(x_i, x_i)$ and thus the affectation step for $(x_i)_{i \notin \mathcal{S}}$ can be performed by computing :

$$-2\mathbf{K}_{\bar{\mathcal{S}},\mathcal{S}}\gamma + \mathbf{1}_{|\bar{\mathcal{S}}|} \left[\text{Diag}\left(\gamma^T \mathbf{K}_{\mathcal{S}} \gamma\right)\right]^T,$$

where $\gamma = (\gamma_{ui})_{u=1,\ldots,U, i \in \mathcal{S}}$ and $\mathbf{1}_{|\bar{\mathcal{S}}|}$ if the vector with all entries equal to 1 and having length the number of elements in $\bar{\mathcal{S}} = \{x_i : \ x_i \notin \mathcal{S}\}$.

The complexity of the approach is $\mathcal{O}(Un_B^2 B + U^2 P)$, compared to the direct approach which has a complexity equal to $\mathcal{O}(Un^2)$. Hence, the computational time is reduced as long as $Bn_B^2 + U^2 P < n^2$ and is even more reduced if the B sub-SOMs are performed in parallel. Usually, B is chosen to be large, n_B is small compared to n and P is only a few observations to obtain sparse representations

Algorithm 4 Multiple online kernel SOM

Initialize for all $i = 1, \dots, n$, $\mathcal{N}(x_i) \leftarrow 0$
for $b = 1 \to B$ **do**
  Sample randomly with replacement n_B observations in $(x_i)_{i=1,\dots,n}$ **return** $\mathcal{S}_b$
  Perform **kernel SOM** with $\mathcal{S}_b$ **return** prototypes $(p_u^b)_u \sim (\gamma_{ui}^b)_{ui}$
  for $u = 1 \to U$ **do**
    Select the P largest $(\gamma_{ui}^b)_{x_i \in \mathcal{S}_b}$ **return** $\mathcal{I}_u^b$ (set of the observations corresponding to the selected γ_{ui}^b)
  end for
  for $i = 1 \to n$ **do**
    if $x_i \in \cup_u \mathcal{I}_u^b$ **then**
      $\mathcal{N}(x_i) \leftarrow \mathcal{N}(x_i) + 1$
    end if
  end for
end for
Select the PU observations x_i corresponding to the largest $\mathcal{N}(x_i)$ **return** $\mathcal{S}$
Perform **kernel SOM** with $\mathcal{S}$ **return** prototypes $(p_u)_u \sim (\gamma_{ui})_{u=1,\dots,U, x_i \in \mathcal{S}}$ and classification $(f(x_i))_{x_i \in \mathcal{S}}$
Affect $(x_i)_{x_i \notin \mathcal{S}}$ with

$$f(x_i) := \arg\min_u \|\phi(x_i) - p_u\|_{\mathcal{H}}^2$$

return final classification $(f(x_i))_{i=1,\dots,n}$ and sparse prototypes $(p_u)_u \sim (\gamma_{ui})_{u=1,\dots,U, x_i \in \mathcal{S}}$

of the prototypes. However, the computational times are not directly comparable since the bagged approach can be performed in parallel, unlike the direct approach or the patch SOM.

6.3 Applications

Bagged kernel SOM on simulated data

First, a simple simulated dataset with 5000 observations is considered. The observations are randomly generated in $[0, 1]^{20}$ and are then mapped onto a 5×5 grid using the kernel SOM algorithm with a Gaussian kernel. Several algorithms are compared with varying parameters :

— the **patch SOM** with different numbers of patches (250, 375, 500, 1 000) and different values for P (2, 5, 10, 15, 20, 30 and 50). A last kernel SOM was trained with the selected observations to make the results (based on P selected observations) comparable with those of the patch SOM ;

— the **bagged SOM** with different values for n_B (5%, 7.5%, 10% and 20% of the original data set size) and for B (500 and 1000) and the same values for P as with the patch SOM ;

— a **full kernel SOM** used on the whole data set and aimed at being the reference method.

Figure 6.1 gives the quantization and topographic [135] errors of the resulting maps versus the value of P. In this figure, two classical quality criteria for SOM results are used : the quantization error (which assesses the quality of the clustering) and the topographic error (which assesses the quality of the organization ; see [135]). In some cases, the results can be even better than the full kernel SOM. Considering the bootstrap version, the performances are consistent with the full kernel SOM (for about $P \sim 5 - 10$, which corresponds to using only 250 observations at most, instead of 5000, to represent the prototypes).

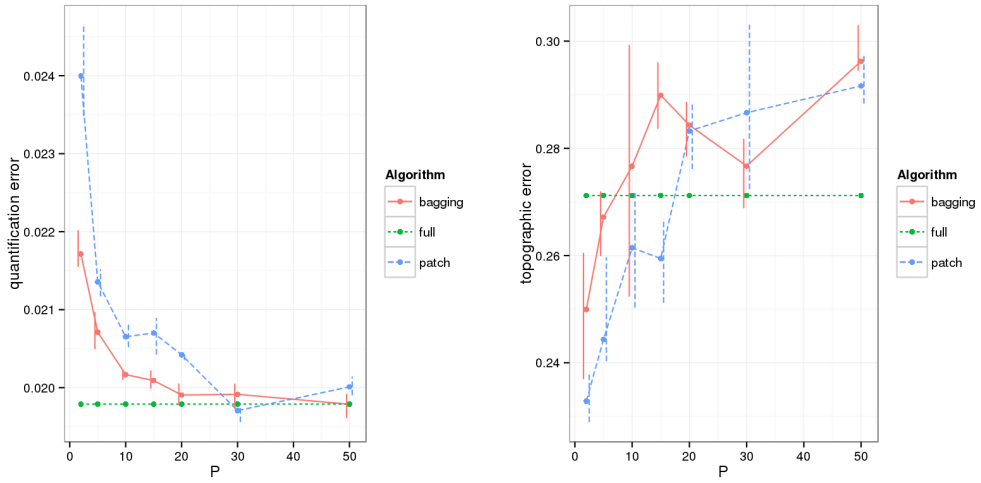

Figure 6.1 Evolution of the quantization (left) and topographic (right) errors versus P. Error bars indicates the first and last quantiles and dots the average values over all simulations.

The analysis of the other parameters of the algorithm (bag size n_B and number of bootstrap samples B) does not show any particular feature. This is explained because the final clustering is obtained from the PU most representative observations and thus P has a much greater impact on the performances than, e.g., n_B.

Application to ego-facebook© network : The bagging method is then applied to one of the ego-facebook© networks described in [112] [1]. The data used in this section are the ones extracted from the network number 107 : the largest connected component of the facebook© network was extracted, which contained 1 034 nodes. This section presents the comparaison of bagged SOM and standard SOM to map the nodes of the graph onto a two-dimensional grid (having sizes 10×10). As explained in [18, 149], using such mappings can provide a simplified representation of the graph, which is useful for the user to help him or her understand its macro-structure before focusing more deeply on some chosen clusters. The kernel used to compute similarities between nodes in the facebook© network was the *commute time kernel*, [55]. If the graph is denoted by $\mathcal{G} = (V, E, W)$, with $V = \{x_1, \ldots, x_n\}$ the set of vertices, E the set of edges which is a subset of $V \times V$ and W a weight matrix (a symmetric matrix with positive entries and null diagonal), the commute time kernel is the generalized inverse of the graph Laplacian, L, which is : $L_{ij} = \begin{cases} d_i & \text{if } i = j \\ -W_{ij} & \text{otherwise} \end{cases}$ where $d_i = \sum_j W_{ij}$ is the degree of node x_i. As explained in [102], the Laplacian is closely related to the graph structure and thus, it is not surprising that a number of kernel has been derived from this matrix [163]. As shown in [55], the commute kernel yields to a simple similarity interpretation because it computes the average time needed for a random walk on the graph to reach a node from another one.

Different approaches were compared : (i) the standard kernel SOM (on-line version), using all available data ; (ii) the bagged kernel SOM, as described in Section 6.2, with $B = 1000$ bootstrap sample, $n_B = 200$ in each sample and $P = 3$ observations selected per prototype and (iii) a standard kernel SOM trained with an equivalent number of randomly chosen observations. The relevance of the results was assessed using different quality measures. Some quality measures were related to the quality of the map (quantification error and topographic error) and some were related to a ground truth : some of the nodes have been indeed labeled by users to belong to one "list" (as named by facebook©). We confronted these groups to the clusters obtained on the map calculating (i) the average node purity (i.e., the mean over all clusters of the maximal proportion of one list in a given cluster ; only individuals belonging to one list were used to compute this quality measure) and (ii) the normalized mutual information [43] (also restricted to individuals belonging to one list only) and

1. available at `http://snap.stanford.edu/data/egonets-Facebook.html`

also to the graph structure using the modularity [120], which is a standard quality measure for node clustering.

The results are summarized in Table 6.1. Surprisingly, the maps trained with a reduced number

	Quantification Error ($\times 100$)	Topographic Error	Node Purity	Normalized Mutual Information	Modularity
bagged K-SOM	7.66	4.35	89.65	70.10	0.47
full K-SOM	9.06	5.22	86.53	53.79	0.34
random K-SOM	8.08	6.09	87.26	60.79	0.40

Table 6.1 Quality measures for different versions of kernel SOM (standard using all data, bagged, standard using randomly selected data) on facebook© data

of samples (bagged K-SOM and random K-SOM) obtain better quality measures than the map trained with all samples. Using a bootstraping approach to select the relevant observations also significantly improves all quality measures as compared to a random choice with the same number of observations. The results obtain with the bagged SOM are displayed in Figures 6.2 and 6.3 (from, respectively, the map and the network points of view). They show that the nodes are mainly dispatched into three big clusters, which correspond each to approximately only one "list", as defined by the user. The results

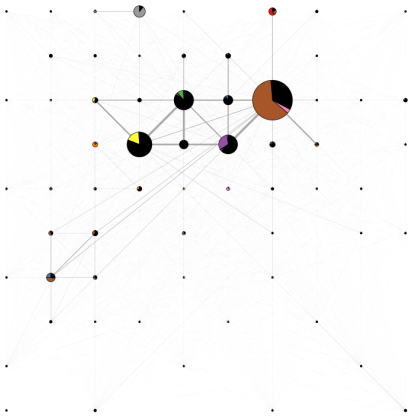

Figure 6.2 Simplified representation of the facebook© network projected on the map resulting from bagged K-SOM. The circle sizes are proportional to the number of nodes classified in the cluster and the edge width are proportional to the number of edges between the nodes in the two clusters. Colors correspond to the proportion of user-defined lists (black is used for "no list").

provided with the K-SOM using all the data tend to provide smaller communities and to scatter the biggest lists on the map. Using this approach, it is however hard to conclude if interpretability has been increased (i.e., if the selected observations used for training are representative of their cluster) as, in Figure 6.3, they do not seem to have a particularly high degree or centrality.

6.4 Conclusion

This paper presents a parallelizable bagged approach which results in a reduced computational time and a sparse representation of prototypes for kernel SOM. The simulations show good per-

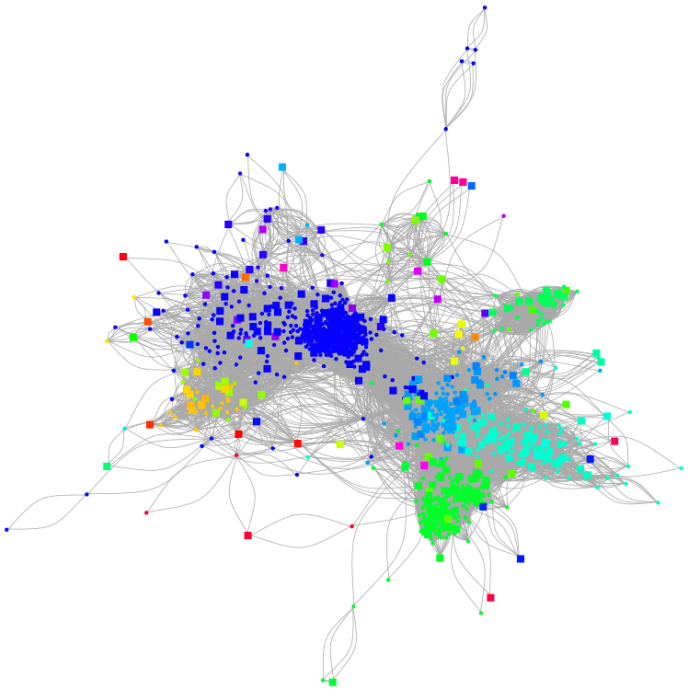

Figure 6.3 The facebook© network represented with a force-directed placement algorithm [57]. Colors represent the clusters on the map and selected nodes used to train the map are represented by squares (instead of circles)

formances and only a small loss of accuracy which is compensated by a faster computational time. Obtained prototypes are also easier to interpret, as based on a smaller number of observations.

Chapitre 7

Efficient interpretable variants of online SOM for large dissimilarity data

Résumé : L'algorithme des cartes auto-organisatrices représente un outils efficace dans l'exploration de données. Dans sa version originale, l'algorithme a été conçu pour des vecteurs numériques. Depuis, de nombreuses extensions ont été proposées afin de prendre en compte des jeux de données complexes décris par des (dis)similarités. La majorité de ces extensions représentent les prototypes sous forme d'une liste de (dis)similarités avec le jeu de données complet et souffre de nombreux inconvénients : leurs complexités est augmentées (elle devient quadratique au lieu de linéaire), la stabilité est diminuée et les prototypes ne sont plus interprétable.

Dans cet article, nous proposons et comparons deux extensions des cartes auto-organisatrices stochastiques pour des données de (dis)similarités : la première profite des propriétés de la version stochastique de l'algorithme afin to maintenir une représentation parcimonieuse des prototypes à chaque itération de l'algorithme, lorsque la seconde utilise une réduction de dimension dans l'espace image définie par la (dis)similarité. Nos contributions à l'analyse de données de (dis)similarité par des cartes topologiques sont de deux ordres : premièrement, nous présentons une nouvelle version des cartes auto-organisatrices qui impose une représentation parcimonieuse des prototypes lors de leurs mise à jour à chaque itération. Deuxièmement, cette approche est comparée sur de nombreux jeux de données de référence à une méthode de réduction de dimension standard (K-PCA), qui est elle-même adaptée aux jeux de données massifs grâce à l'approximation de Nyström.

Les résultats mettent en évidence que les deux approches permettent de réduire la dimensionnalité des prototypes tout en fournissant des résultats exacts en un temps raisonnable. La sélection d'une de ces deux stratégies dépend de la taille du jeu de données, du besoin d'une interprétation facile des résultats ainsi que de l'équipement informatique à disposition. La conclusion fournit quelques recommandations afin d'aider l'utilisateur à faire son choix.

Abstract : Self-organizing maps (SOM) are a useful tool for exploring data. In its original version, the SOM algorithm was designed for numerical vectors. Since then, several extensions have been proposed to handle complex datasets described by (dis)similarities. Most of these extensions represent prototypes by a list of (dis)similarities with the entire dataset and suffer from several drawbacks : their complexity is increased - it becomes quadratic instead of linear -, the stability is reduced and the interpretability of the prototypes is lost.

In the present article, we propose and compare two extensions of the stochastic SOM for (dis)similarity data : the first one takes advantage of the online setting in order to maintain a sparse representation of the prototypes at each step of the algorithm, while the second one uses a dimension reduction in a feature space defined by the (dis)similarity. Our contributions to the analysis of (dis)similarity data with topographic maps are thus twofolds : first, we present a new version of the SOM algorithm which ensures a sparse representation of the prototypes through online updates. Second, this approach is compared on several benchmarks to a standard dimension reduction technique (K-PCA), which is itself adapted to large datasets with the Nyström approximation.

Results demonstrate that both approaches lead to reduce the prototypes dimensionality while providing accurate results in a reasonable computational time. Selecting one of these two strategies depends on the dataset size, the need to easily interpret the results and the computational facilities available. The conclusion tries to provide some recommendations to help the user making this choice.

7.1 Introduction

7.1.1 State-of-the art on SOM for (dis)similarity data

Over the years, the self-organizing map (SOM) algorithm [86] was proved to be a powerful and convenient tool for clustering and visualizing data [132, 136, 155, 119, 187]. While the original algorithm had been designed for numerical vectors, the available data in the applications became more and more complex, being frequently too rich to be described by a fixed set of numerical attributes only. This is the case, for example, when data are described by relations between objects (individuals involved in a social network) or by measures of resemblance/dissemblance which are context specific (see [1, 51] for similarities between categorical sequences, [101] for similarities between microbial diversity distributions, [186] for similarities in gene expression data).

During the past twenty years, the SOM algorithm was extended to handle non numerical data. For example, SOM was adapted to categorical data analysis, by using a method similar to Multiple Correspondence Analysis in [38]. Another solution, called median SOM [84], addressed the issue of data described by pairwise relations (similarities or dissimilarities) : in this solution, the standard computation of the prototypes is replaced by an approximation within the original dataset. However, as prototypes are chosen among the data, their representation is very restrictive. In order to increase the flexibility of the prototypes, [35] proposed to represent a class by several prototypes, all chosen among the original dataset. But, this method seriously increases the computational time, while prototypes remain restricted to the original dataset and may generate possible sampling or sparsity issues.

A very different approach to handle relational data consists in relying on a (pseudo-)Euclidean framework, following the results of [7] (for data described by a kernel) or of [62] (for dissimilarity data). This approach was developed in the framework of kernel SOM (see [103] for the online version and [18] for the batch version), and in the framework of relational SOM (see [123] for the online version and [67] for the batch version). Kernel SOM and relational SOM are equivalent if the dissimilarity in relational SOM is the squared distance induced by the kernel. The key idea of this approach is to express prototypes as convex combinations of the images of the original data $(x_i)_{i=1,\dots,n}$ in a (pseudo-)Euclidean space in which the data are (implicitly) embedded by the kernel

(or the dissimilarity) : as stated in [148, 74], this solution yields several drawbacks due to the large dimensionality of the embedding space (which is equal to the number of observations, n). Firstly, the complexity (in n) is strongly increased and becomes at least quadratic. As stressed in [67], algorithms will be slow for datasets with 10,000 observations and impossible to run on a normal computer for 100,000 input data. Secondly, the results are highly unstable : especially in the online (also called stochastic) version of the algorithm, two different runs of the method can provide very different results. Thirdly, one of the most important features of the SOM algorithm is lost : in standard numerical SOM, clusters are represented by a single prototype valued in the data space. These prototypes help to interpret the obtained clusters and thus the overall map organization. In kernel/relational SOM, prototypes are given as n coefficients that correspond to a resemblance with each of the n observations : they do not correspond themselves to an observation in the original data space and as such, prototypes are not much more informative than the clustering itself.

In conclusion, kernel and relational extensions of the standard SOM algorithm are hardly practicable when the dataset is large. This is due partly to the number of observations, but also to the dimensionality of the (embedded) data which is directly related to this number. To address this issue, strategies usually used to handle large datasets or datasets with a high dimensionality are useful and they can even be combined.

7.1.2 Review of methods for large datasets and high dimensional datasets

Different strategies were developed and are available in the literature to handle large datasets (when the number of observations is large) and high-dimensional datasets (when the dimension of the dataset is large). For large datasets, standard approaches include i) *divide and conquer approaches* [31, 29, 145] in which data are split into several bits of data which are processed separately. The results are aggregated afterwards to obtain a final solution which is supposed to well approximate the solution that would have been obtained if the entire dataset had been processed at once ; ii) *subsampling methods* [24, 184, 83, 91, 115], which consist in using a restricted subset (usually carefully designed) of the original data, in order to approximate the solution that could have been obtained with the entire dataset and iii) *online* updates [154, 44], in which the results are updated with sequential steps, each having a low computational cost.

A particular case of the subsampling strategy is the Nyström approximation [181], which consists in sampling a small number of rows/columns in square matrices and in obtaining an approximation of its eigendecomposition at a very reduced computational cost. The eigendecomposition is even exact when the matrix is of low rank (when the size of the subsample is larger than the rank of the matrix). This method is frequently used for kernel and dissimilarity-based algorithms.

For high-dimensional data, the strategies are a bit different and include i) *sparse approaches* [71, 166], in which a subset of the variables is selected to build the final predictive model. This subset can be obtained from sequential exploration (stepwise strategies), from approximation heuristics or by using a sparse penalty term within the model (LASSO) ; ii) *dimension reduction (DR)* techniques, that can be linear (PCA for instance or random projections as in [185]) or nonlinear [95]. DR methods embed the data in a small dimensional space and are usually mainly used for visualization and exploratory analysis. However, if the embedding can be obtained at a low cost, it can be used as an approximation of the high-dimensional dataset on which more costly algorithms may be applied. SOM itself is a dimension reduction method but, as stressed before, the computational complexity of its kernel and relational versions is high. Finally, a particular case of DR techniques is *model-based clustering* methods, which use mixture distributions and embed the data in a low-dimensional subspace that is the best suited for clustering (see [19] for a review).

7.1.3 Kernel/relational extensions for large datasets

Several extensions for kernel and relational data of the standard SOM algorithm, or of related kernel/relational algorithms (such as, *e.g.*, k-means, LVQ, topographic maps...) have already been proposed in the literature. They use ideas coming from the strategies handling large and/or high-dimensional datasets cited above. Most of them seek a simplified/sparse representation of the prototypes and/or a reduced computational time.

In the relational k-means framework, [150] proposed a sparse extension of the batch algorithm : every prototype is represented by at most K (K fixed) observations by cluster, that are selected at each step of the algorithm. In the supervised framework, [74] used a similar strategy for batch LVQ, by selecting the most representative observations (with different methods to obtain them, including approximation heuristics and L^1 penalty) in every cluster and at each step of the algorithm. A similar method was used in [73], combined with the Nyström approximation of the LVQ algorithm, in order to obtain sparse prototypes at very low computational cost. The Nyström approximation was also used for obtaining faster versions of topographic mapping methods [60, 189] and for reducing the computational cost of the clustering. Another subsampling strategy was used in a nonlinear (kernel) DR framework to allow processing large datasets, in [59].

However, these approaches do not lead to a simplified (and thus interpretable) representation of the prototypes. Furthermore, all of them are restricted to the batch framework and most of them are performed after each iteration of a batch algorithm, *i.e.*, after all observations have been processed at least once. An alternative to these methods consists in splitting the data into several subsets on which independent algorithms are trained : in [67], the complexity is reduced using iterative "patch clustering" that mixes "divide and conquer" and "online updates" methods. First, the data are split into B patches of size n_B ($\ll n$, B fixed). A prototype-based clustering algorithm in batch version (neural gas or SOM) is then run on a patch $\mathcal{P}_t$. The resulting prototypes, which may be viewed as compressed representations of the data already seen, are then added as new data points to the next patch, $\mathcal{P}_{t+1}$. Moreover, the full vector of coefficients is replaced by the Q closest input data (Q fixed). With this method, linear time and constant space representation are obtained but the sequential training may influence the final result since all observations are not processed evenly.

In the same line of thoughts, [108] propose a bagging approach for kernel SOM. Data are split into B subsamples of size n_B ($\ll n$, B fixed), the online kernel SOM is trained on each subsample and, after training, the most representative Q observations are chosen for each prototype (Q fixed). Eventually, a final map is trained on the resulting most representative observations. In this method, parallel computing techniques can be used for reducing the computational time. However, the results of the B trained SOMs are not used as such but only to select the most representative observations in the dataset.

7.1.4 Contributions of the article

In the present article, we propose and compare two methods to obtain sparse prototypes and a reduced computational cost in the online SOM algorithm. The first one uses a reduction dimension in the embedding space, which can be efficiently performed with Nyström technique. This method combines ideas coming from the high dimension and the large data problems. However, it is not specific to the online setting. We thus compared it with another approach which takes advantage of the online framework to provide sparse prototypes at all iteration steps of the algorithm : the coefficients are interpreted similarly to an amount of mass and the most important observations are selected by using a fixed global probability mass, ν, such that only the largest coefficients summing to at most ν are kept. This second proposal also takes ideas from large data problems (online updates) and from high dimension (variable selection with sparsity).

For the sake of simplicity, most of the paper is written in the framework of kernel SOM but its extension to relational SOM is straightforward and briefly explained in Section 7.5. The rest of the

paper is organized as follows : Section 7.2 recalls the online kernel SOM (online K-SOM). Section 7.3 describes the dimension reduction approach for online K-SOM while Section 7.4 presents the direct sparse version of online K-SOM. Comparisons and numerical experiments are reported in Section 7.6.

7.2 Kernel SOM (K-SOM)

This section describes the theoretical background of the online version of the SOM algorithm and its extension to data described by a kernel. In the following, we consider a set of observations $(x_i)_{i=1,\dots,n}$ which take values in an arbitrary space $\mathcal{X}$. $\mathcal{X}$ is equipped with a kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that provides pairwise similarity between the observations, $K_{ij} := K(x_i, x_j)$. K is assumed symmetric ($K_{ij} = K_{ji}$) and positive ($\forall\, N \in \mathbb{N},\ \forall\, (\alpha_i)_{i=1,\dots,N} \subset \mathbb{R},\ \forall\, (x_i)_{i=1,\dots,N} \subset \mathcal{X},\ \sum_{i,i'} \alpha_i \alpha_{i'} K_{ii'} \geq 0$). According to [7], K is the dot product in a uniquely defined Hilbert space $(\mathcal{H}, \langle ., . \rangle)$ of the images of x_i and x_j by a uniquely defined feature map $\phi : \mathcal{X} \to \mathcal{H}$:

$$K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle.$$

The SOM algorithm aims at mapping the input data onto a low dimensional grid (usually a two-dimensional rectangle), composed of U units, each of them described by a prototype p_u, $u = 1, \dots, U$. The units are related together by a neighborhood relationship H, expressed as a function of the distance between the units on the grid, $d(u, u')$, $H : \mathbb{R} \to \mathbb{R}$. Classically, H is chosen such that $H(0) = 1$, $\lim_{x \to +\infty} H(x) = 0$ and is decreased during the training. Also, the distance on the grid, d, may be chosen as the length of the shortest path between the units or as the Euclidean distance between the coordinates of the units positioned at $(1, 1)$, $(1, 2)$, $\dots$, (m, m') (where $m \times m' = U$).

In standard (numerical) SOM, the data take values in $\mathcal{X} = \mathbb{R}^d$ and the kernel is the standard dot product in this space $K_{ij} = x_i^\top x_j$. In this case, the prototypes are also valued in $\mathbb{R}^d$. In kernel SOM, $\mathcal{X}$ is arbitrary and prototypes are not easily defined in this space, which may not be Euclidean or equipped with standard operations such as the sum. To allow for a more flexible representation of the prototypes (*i.e.*, a representation which is not restricted to the data already observed, $(x_i)_i$), the implicit feature space $(\mathcal{H}, \langle ., . \rangle)$ is used and the algorithm is re-defined in this space. The prototypes are expressed as convex combinations of the images by ϕ of the original data : $p_u = \sum_{i=1}^n \beta_{ui} \phi(x_i)$, $\beta_{ui} \geq 0$ and $\sum_i \beta_{ui} = 1$. As said before, the feature map is uniquely defined from the kernel K. However, it is usually not explicitly given and thus, all calculations are based on the values of the coefficients $\boldsymbol{\beta}_u = (\beta_{u1}, \dots, \beta_{un}) \in \mathbb{R}^n$ only.

The kernel version of online SOM [103, 150] is thus directly derived from the computations of the standard numerical SOM performed in the feature space $\mathcal{H}$: the norm and the dot product in $\mathcal{H}$ can be obtained using the kernel, without the need of explicitly knowing $\mathcal{H}$ or ϕ (this is the so-called "kernel trick"). Online K-SOM is provided in Algorithm 5. In this algorithm, $\mu(t)$ usually vanishes at the rate $\frac{1}{t}$ and the final clustering is defined as $(\mathcal{C}_u)_{u=1,\dots,U}$, where $\mathcal{C}_u = \{x_i :\ f(x_i) = u\}$ with $f := f^T$.

The issue with kernel SOM is that, when n is large, the total number of coefficients β_{ui} to learn is equal to $n \times U$, which yields a complexity of $\mathcal{O}(n^2 U)$ (for one iteration) instead of $\mathcal{O}(dU) + \mathcal{O}(ndU)$ for the standard numerical SOM in $\mathbb{R}^d$. Hence, this algorithm cannot be used to analyze datasets with more than a few thousands observations. Also, the representation of the prototypes is so flexible that the results are highly unstable with different final clusterings and different prototypes for each run of the algorithm. Finally, as stated in [148], one of the main challenges of this kind of algorithm is that the easiness in interpreting the resulting map is lost : in standard SOM, prototypes are easily interpreted because they are described by values of well known variables which are the variables also describing the observations in the input dataset. In kernel SOM, the prototypes are characterized by their proximity to all individuals in the dataset. Hence, interpreting the clustering requires to be able

Algorithm 5 Online kernel SOM

1: For all $u = 1, \dots, U$ and $i = 1, \dots, n$, initialize $(\beta_{ui}^0)_{u,i}$ such that $\beta_{ui}^0 \geq 0$ and $\sum_i^n \beta_{ui}^0 = 1$.
2: **for** $t = 1, \dots, T$ **do**
3: Randomly choose an input x_i
4: **Assignment step** : find the unit of the prototype closest to $\phi(x_i) \in \mathcal{H}$

$$f^t(x_i) \leftarrow \arg \min_{u=1,\dots,U} \|p_u^{t-1} - \phi(x_i)\|^2$$

which is equivalent to minimize, over u, $\sum_{j,j'} \beta_{uj}^{t-1} \beta_{uj'}^{t-1} K_{jj'} - 2 \sum_j \beta_{uj}^{t-1} K_{ij}$;
5: **Representation step** : $\forall\, u = 1, \dots, U$,

$$\boldsymbol{\beta}_u^t \leftarrow \boldsymbol{\beta}_u^{t-1} + \mu(t) H^t(d(f^t(x_i), u)) \left(\mathbf{1}_i - \boldsymbol{\beta}_u^{t-1}\right)$$

where $\mathbf{1}_i$ is a vector with a single non null coefficient at the ith position, equal to one.
6: **end for**

to understand the relationships of the prototypes with all individuals, which is probably as difficult as analyzing the entire dataset directly.

7.3 A dimension reduction technique for K-SOM

This section describes a first approach to reduce the dimensionality of the prototypes, thus improving the computational complexity as well as the interpretability of the prototypes. Our proposal is to rely on a preprocessing of the data based on PCA in the feature space (Kernel PCA, denoted by K-PCA and described in Section 7.3.1) and then to express the SOM algorithm in the subspace of the feature space $\mathcal{H}$ which is spanned by the first eigenvectors of the K-PCA (Section 7.3.2). Finally, the computational complexity of the approach can be further reduced by performing the K-PCA thanks to the Nyström approximation (Section 7.3.3).

7.3.1 Kernel PCA (K-PCA)

K-PCA, introduced in [158], is a PCA analysis performed in the feature space $(\mathcal{H}, \langle ., . \rangle)$ induced by the kernel. A centered data matrix is first computed using :

$$\begin{aligned} \widetilde{K}_{ij} &:= \left\langle \phi(x_i) - \frac{1}{n}\sum_{l=1}^n \phi(x_l), \phi(x_j) - \frac{1}{n}\sum_{l=1}^n \phi(x_l) \right\rangle \\ &= K_{ij} - \frac{1}{n}\sum_{l=1}^n (K_{il} + K_{jl}) + \frac{1}{n^2}\sum_{l,l'=1}^n K_{ll'}, \end{aligned} \tag{7.1}$$

which yields to the modified centered kernel $\widetilde{K} = K - \frac{1}{n}\mathbf{I}_n^T K - \frac{1}{n} K \mathbf{I}_n + \frac{1}{n^2}\mathbf{I}_n^T K \mathbf{I}_n$, in which $\mathbf{I}_n$ is the vector with n entries equal to 1. The eigenvectors $(\boldsymbol{\alpha}_k)_{k=1,\dots,n} \in \mathbb{R}^n$ and the corresponding eigenvalues $(\lambda_k)_{k=1,\dots,n}$ are obtained from this centered matrix by solving the following eigenvalue problem :

$$\widetilde{K}\boldsymbol{\alpha} = \lambda \boldsymbol{\alpha}. \tag{7.2}$$

This problem is equivalent to finding eigenvectors, in the feature space $\mathcal{H}$, of the covariance matrix of the (centered) images of the original data by ϕ (the feature map associated to the centered kernel $\widetilde{K}$). These vectors, $(a_k)_{k=1,\dots,n} \in \mathcal{H}$ lie in the span of $\{\phi(x_i)\}_{i=1,\dots,n}$ and can be expressed as :

$$a_k = \sum_{i=1}^{n} \alpha_{ki}\phi(x_i)$$

where $\boldsymbol{\alpha}_k = (\alpha_{ki})_{i=1,\dots,n}$. $\boldsymbol{a}_k = (a_{ki})_{i=1,\dots,n}$ are orthonormal in $\mathcal{H}$:

$$\forall\, k, k',\ \langle a_k, a_{k'}\rangle = \boldsymbol{\alpha}_k^\top \widetilde{K}\boldsymbol{\alpha}_{k'} = \delta_{kk'} \qquad \text{with} \qquad \delta_{kk'} = \begin{cases} 0 & \text{if } k \neq k' \\ 1 & \text{otherwise} \end{cases}.$$

The principal components are the coordinates of the projections of the images of the original data, $(\phi(x_i))_i$ onto the eigenvectors $(a_k)_{k\geq 1}$ which can be expressed as :

$$\langle a_k, \phi(x_i)\rangle = \sum_{j=1}^{n} \alpha_{kj}\widetilde{K}_{ji} = \widetilde{K}_{i.}\boldsymbol{\alpha}_k,$$

where $\widetilde{K}_{i.}$ is the i-th row of the kernel $\widetilde{K}$. According to Equation (7.2), we thus obtain that $\langle a_k, \phi(x_i)\rangle = \lambda_k\alpha_{ki}$. The original data are projected on these axes with :

$$\mathcal{P}_{a_k}(\phi(x_i)) = \langle a_k, \phi(x_i)\rangle a_k = (\lambda_k\alpha_{ki})a_k.$$

The result of K-PCA can be used to approximate the data in a reduced space by selecting p axes $(a_k)_{k=1,\dots,p}$ in the feature space $\mathcal{H}$ (with $p \ll n$), on which the data can be projected.

7.3.2 K-PCA SOM

We suppose in this section that the kernel K is centered. Otherwise, without loss of generality, the centering procedure described in Equation (7.1) can be applied.

The main idea developed in this section is to define the U prototypes of the SOM in $A = \text{Span}\{a_1, \dots, a_p\}$ instead of the entire feature space. A prototype can thus be written as :

$$p_u = \sum_{k=1}^{p} \beta_{uk}a_k$$

with $\beta_{uk} \geq 0$ and $\sum_k \beta_{uk} = 1$ and the two steps of the SOM algorithm can thus be rewritten as :

— *assignment step* : when the observation x_i is picked at random, it is affected to :

$$f(x_i) := \arg\min_u \|p_u - \phi(x_i)\|^2$$

in which :

$$\begin{aligned} \|p_u - \phi(x_i)\|^2 &= \sum_{k,k'=1}^{p} \beta_{uk}\beta_{uk'}\langle a_k, a_{k'}\rangle - 2\sum_{k=1}^{p} \beta_{uk}\langle a_k, \phi(x_i)\rangle + \|\phi(x_i)\|^2 \\ &= \sum_{k,k'=1}^{p} \beta_{uk}\beta_{uk'}\langle a_k, a_{k'}\rangle - 2\sum_{k=1}^{p} \beta_{uk}\lambda_k\alpha_{ki} + \|\phi(x_i)\|^2 \\ &= \boldsymbol{\beta}_u^\top\boldsymbol{\beta}_u - 2\boldsymbol{\beta}_u^\top\Lambda\boldsymbol{\alpha}_{.i} + \|\phi(x_i)\|^2, \end{aligned} \tag{7.3}$$

where $\boldsymbol{\beta}_u = (\beta_{uk})_{k=1,\dots,p}$, $\Lambda = \text{Diag}\,(\lambda_1, \dots, \lambda_p)$ and $\boldsymbol{\alpha}_{.i}$ is the i-th column of $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_p]^\top$. This step is thus equivalent to minimize $\boldsymbol{\beta}_u^\top\boldsymbol{\beta}_u - 2\boldsymbol{\beta}_u^\top\Lambda\boldsymbol{\alpha}_{.i}$ over $u \in \{1, \dots, U\}$ and is also equivalent to minimize $\|p_u - \mathcal{P}_A(\phi(x_i))\|^2$ over u. This result was expected since, by

the definition of $\mathcal{P}_A(\phi(x_i))$, we have that $\|p_u - \phi(x_i)\|^2 = \|p_u - \mathcal{P}_A(\phi(x_i))\|^2 + \|\mathcal{P}_A(\phi(x_i)) - \phi(x_i)\|^2$, in which the second term does not depend on u ;

— *representation step* : the gradient descent like step is given by

$$\begin{aligned} p_u &= p_u + \mu H(d(f(x_i), u)) \left(\mathcal{P}_A(\phi(x_i)) - p_u\right) \\ &= \sum_k \beta_{uk} a_k + \mu H(d(f(x_i), u)) \left(\sum_k (\lambda_k \alpha_{ki}) a_k - \sum_k \beta_{uk} a_k \right), \end{aligned}$$

which is equivalent to update the coefficients as :

$$\boldsymbol{\beta}_u = \boldsymbol{\beta}_u + \mu H(d(f(x_i), u)) \left(\Lambda \boldsymbol{\alpha}_{.i} - \boldsymbol{\beta}_u\right).$$

This approach is simply the standard (numerical) SOM with entries the $n \times p$ matrix $\boldsymbol{\alpha}^\top \Lambda$. The computational complexity of this approach is thus reduced from $\mathcal{O}(n^2 UT)$ (online K-SOM, T iterations) to $\mathcal{O}(pUT) + \mathcal{O}(npU)$ (online numeric SOM in $\mathbb{R}^p$, cost of the T iterations and cost of the final clustering computation), once the K-PCA is given. Hence, since T is generally of the order of n, this gives a linear complexity for the algorithm. However, K-PCA itself can have a large complexity. This issue is addressed in the next section.

The interpretation of the prototypes is done similarly as for a standard PCA : the axes a_1, ..., a_p of the prototypes are interpreted with respect to the observations $(x_i)_i$ which contribute most to their definition. Then, prototypes are interpreted by means of their coefficients on these axis.

7.3.3 K-PCA from Nyström approximation

Kernel-based methods, and especially K-PCA, do not scale well when the number of observations, n, is large. For instance, the computational complexity of the eigendecomposition of K is $\mathcal{O}(n^3)$. An effective approach for improving the scalability of kernel methods is the Nyström approximation [181] and a similar technique is used in [189] for improving the computational cost of topographic maps for dissimilarity data. In the latter reference, the authors use the Nyström approximation to reduce the computational cost of $\|p_u - \phi(x_i)\|^2$ from $\mathcal{O}(n^2)$ to $\mathcal{O}(m^2 n)$, in which m is a number of observations taken at random within the original dataset. m is supposed to be small compared to n and close to the kernel rank.

In the approach introduced in Section 7.3.2 and unlike article [189], it is the pre-processing step which is addressed by the Nyström approximation, while the computational cost of calculating the clustering of the map is handled by the use of the numerical version of the algorithm based on the K-PCA pre-processing. More precisely, the eigendecomposition of a kernel matrix K (which is supposed to be centered) is approximated by selecting m observations, $\mathcal{T}_m$ among $(x_i)_{i=1,\dots,n}$, and by using the eigendecomposition of the reduced matrix $K^{(m)} = (K(x_i, x_j))_{i,j \in \mathcal{T}_m}$. In practice, the selected observations $\mathcal{T}_m$ are chosen at random, although more efficient sampling techniques such as the ones described and evaluated in [89] could also be used.

If the eigenvalues and the (orthonormal) eigenvectors of $K^{(m)}$ are denoted by $(\lambda_k^{(m)})_k$ and $(v_k^{(m)})_k$ respectively, then the eigenvalues and (orthonormal) eigenvectors of K are given by

$$\lambda_k \simeq \frac{n}{m} \lambda_k^{(m)} \qquad \text{and} \qquad v_{ki} \simeq \sqrt{\frac{m}{n}} \frac{1}{\lambda_k^{(m)}} K_{i.}^{(n,m)} v_k^{(m)},$$

with $K_{i.}^{(n,m)}$ the i-th row of the matrix $K^{(n,m)} = (K(x_j, x_{j'}))_{j=1,\dots,n,\, j' \in \mathcal{T}_m}$. If the rank of $K^{(m)}$ is equal to the rank of the original matrix K, the approximation even becomes an equality. Then, assuming that the kernel K (which is supposed to be centered) is known or at least that any of the pairs $K(x_i, x_j)$ (i, $j = 1, \dots, n$) can be computed at low cost, the K-PCA requires to obtain the entries

$(\lambda_k \alpha_{ki})_{k=1,\dots,p} \in \mathbb{R}^p$ for all $i = 1, \dots, n$, where $(\boldsymbol{\alpha}_k)_k$ are the eigenvectors of K which are orthogonal with respect to the norm induced by the kernel. We can easily show that

$$\boldsymbol{\alpha}_k = \frac{v_k}{\sqrt{\lambda_k}} \tag{7.4}$$

because, by definition, as v_k are eigenvectors of K, so are $\boldsymbol{\alpha}_k$ and, in addition, using the equality of Equation (7.4), we have that :

$$\boldsymbol{\alpha}_k^\top K \boldsymbol{\alpha}_{k'} = \frac{1}{\sqrt{\lambda_k}\sqrt{\lambda_{k'}}} v_k^T K v_{k'} = \frac{1}{\sqrt{\lambda_k}\sqrt{\lambda_{k'}}} v_k^T \lambda_{k'} v_{k'} = \delta_{kk'}.$$

Therefore, K-PCA can be computed with entries the rows of the $n \times p$ matrix $\boldsymbol{\alpha}^\top \Lambda$ with $\forall\, i = 1, \dots, n$ and $\forall\, k = 1, \dots, p$,

$$\lambda_k \alpha_{ki} = \sqrt{\lambda_k} v_{ki} = \frac{1}{\sqrt{\lambda_k^{(m)}}} K_{i.}^{(n,m)} v_k^{(m)} = K_{i.}^{(n,m)} \boldsymbol{\alpha}_k^{(m)} \tag{7.5}$$

with $\boldsymbol{\alpha}_k^{(m)} = \frac{v_k^{(m)}}{\sqrt{\lambda_k^{(m)}}}$. This simplified K-PCA is a good approximation of the K-PCA with kernel K. The complexity of the preprocessing is reduced from $\mathcal{O}(n^3)$ (standard K-PCA) to $\mathcal{O}(m^3) + \mathcal{O}(nm^2)$ (respectively, K-PCA based on the reduced kernel and approximation). The complete algorithm is provided in Algorithm 6.

Algorithm 6 Online K-PCA SOM

Nyström approximation of K-PCA
Select at random m observations $\mathcal{T}_m = \{x_{i(1)}, \dots, x_{i(m)}\}$ from the original dataset
Compute the first p eigenvalues and orthonormal eigenvectors of $K^{(m)}$, $(\lambda_k^{(m)})_{k=1,\dots,p}$, $(v_k^{(m)})_{k=1,\dots,p}$ and obtain, for $k = 1, \dots, p$, $\boldsymbol{\alpha}_k^{(m)} = \frac{v_k}{\sqrt{\lambda_k}}$
Compute $\left(K^{(n,m)} \boldsymbol{\alpha}_k^{(m)}\right)_{k=1,\dots,p}$, which is an $n \times p$ matrix $B = (b_{ik})_{i=1,\dots,n,\, k=1,\dots,p}$
K-PCA SOM
Initialize prototypes randomly : $\forall\, u = 1, \dots, U$, $\boldsymbol{\beta}_u^0 \in [0,1]^p$ and $\sum_{k=1}^p \beta_{uk}^0 = 1$
for $t = 1, \dots, T$ **do**
    Randomly choose an input $i \in \{1, \dots, n\}$
    Assignment step : find the unit of the prototype closest to i-th row of B, $\mathbf{b}_i$:

$$f^t(x_i) \leftarrow \arg \min_{u=1,\dots,U} \|\boldsymbol{\beta}_u^{t-1} - \mathbf{b}_i\|_{\mathbb{R}^p}^2$$

    Representation step : $\forall\, u = 1, \dots, U$,

$$\boldsymbol{\beta}_u^t \leftarrow \boldsymbol{\beta}_u^{t-1} + \mu(t) H^t(d(f^t(x_i), u)) \left(\mathbf{b}_i - \boldsymbol{\beta}_u^{t-1}\right)$$

end for

7.4 Direct sparse K-SOM

In this section, we present an alternative approach to obtain sparse prototypes while taking advantage of the online updates of the stochastic version of the K-SOM algorithm. Here, unlike in K-PCA SOM, the prototypes are directly written as convex combinations of the observations, but, in this

case, they are restricted to the input data already fed to the algorithm and, more particularly, to the most important of them. This algorithm has already been described in the framework of dissimilarity data and tested on restricted datasets in [124].

7.4.1 Description of the approach

To ensure sparsity through the algorithm, the prototypes are initialized at random among the input data. Thus, the method first selects at random U observations that are used as initial values for the U prototypes. Then, at a given step t of the algorithm, a prototype is written as a convex combination of the most important past observations : if $I_u(t-1)$ denotes the set of the most important observations selected for prototype p_u before step t, p_u writes $p_u = \sum_{j \in I_u(t)} \beta_{uj} \phi(x_i)$. Finally, the distance in the feature space $\mathcal{H}$ between a new input, x_i selected at random, and p_u is given by :

$$\|\phi(x_i) - p_u\|^2 = \sum_{j,j' \in I_u(t-1)} \beta_{uj}\beta_{uj'} K(x_j, x_{j'}) - 2 \sum_{j \in I_u(t-1)} \beta_{uj} K(x_j, x_i) + K(x_i, x_i). \tag{7.6}$$

In order to maintain prototypes as sparse combinations of the input data, they are periodically updated and the most important coefficients only are kept. The update instants may be performed throughout the iterations using various strategies : for instance, they can be uniformly distributed during the learning process or distributed according to some geometric distribution. The parameter of the geometric distribution may be fixed, during the whole training, or varying (in ascending or descending fashion) with the iterations. We ran several simulations on various data sets, using these four scenarios (results not shown). The results were globally similar, with a slight advantage for the "ascending random" strategy. In this case, the parameter of the geometric distribution was $\gamma_t = \frac{1-\mu(t)}{\kappa}$, where κ is a constant to be set.

As suggested by [74] for a batch sparse LVQ method, sparsity could be achieved by selecting the first Q most important coefficients, where Q is a fixed integer. However, in order to allow for more flexibility in the expression of the prototypes, the most important coefficients are selected according to their value, by fixing a threshold : let $0 < \nu \leq 1$ be the selected threshold. At time step t at which an update occurs and for every $u = 1, ..., U$, the coefficients of the prototype p_u are first ordered in descending order, $\beta_{u,(1)} \geq ... \geq \beta_{u,(\sharp I_u(t))}$. Then, the integer N_u such that

$$N_u = \arg \min_{k=1,...,\sharp I_u(t)} \left\{ \sum_{i=1}^{k} \beta_{u,(i)} \geq \nu \right\}$$

is introduced. The most important coefficients are finally updated as follows

$$\beta_{u,(i)} = \begin{cases} \frac{\beta_{u,(i)}}{\sum_{j=1}^{N_u} \beta_{u,(j)}} & \text{if } (i) \leq N_u \\ 0 & \text{if } (i) > N_u \end{cases},$$

and $I_u(t)$ is updated accordingly afterwards by keeping the observations that correspond to non zero coefficients only.

The sparse relational SOM algorithm is entirely described in Algorithm 7.

Contrary to K-PCA SOM, in this version, the prototypes might directly be interpreted through the observations that are used in their representation. Also, the sparsity is updated during the training and the induced dimension reduction is thus not constrained to the efficiency of a given dimension reduction technique such as K-PCA. However, due to the sparse representation step, the algorithm can be computationally more expensive than K-PCA SOM and the amount of information preserved in the sparse representation is not as well controlled as in K-PCA projection. Finally, the complexity of the method, for T iterations, is not easily obtained : if a total mass equal to ν results in no more than $Q(\nu)$ observations for every prototype at each step, then the global complexity of the method has an upper bound of order $\mathcal{O}\left((Q(\nu))^2 UT\right) + \mathcal{O}\left(n(Q(\nu)^2\right)$ (respectively for the iterations and the final clustering

Algorithm 7 Sparse online K-SOM

For all $u = 1, \dots, U$, initialize p_u^0 among U randomly selected observations in $(x_i)_i$. Initialize $I_u(0) = \{i(u)\}$, with $i(u) \in \{1, \dots, n\}$ for all u and $\boldsymbol{\beta}_u^0 = 1$.

for $t = 1, \dots, T$ **do**

Randomly choose an input x_i, $i \in \{1, ..., n\}$.

Assignment step : find the unit with the prototype closest to $\phi(x_i)$:

$$f^t(x_i) \leftarrow \arg \min_{u=1,\dots,U} \left[(\boldsymbol{\beta}_u^{t-1})^\top \mathbf{K}_{I_u(t-1)} \boldsymbol{\beta}_u^{t-1} - 2 \sum_{j \in I_u(t-1)} \beta_{uj}^{t-1} K(x_j, x_i) \right]$$

where $\mathbf{K}_{I_u(t-1)} = (K(x_j, x_{j'}))_{j,j' \in I_u(t-1)}$.

Representation step : $\forall\, u = 1, \dots, U$

if $i \in I_u(t-1)$, **then**

$\boldsymbol{\beta}_u^t \leftarrow \boldsymbol{\beta}_u^{t-1} + \mu(t) H^t(d(f^t(x_i), u)) \left(\mathbf{1}_i - \boldsymbol{\beta}_u^{t-1} \right)$

$I_u(t) = I_u(t-1)$

else if $i \notin I_u(t-1)$, **then**

$$\boldsymbol{\beta}_u^t \leftarrow \left[1 - \mu(t) H^t(d(f^t(x_i), u)) \right] \left(\boldsymbol{\beta}_u^{t-1}, 0 \right) + \mu(t) H^t(d(f^t(x_i), u)) (\underbrace{0, ..., 0}_{\sharp I_u(t-1)}, 1)$$

$I_u(t) = I_u(t-1) \cup \{i\}$.

end if

Sparse representation

if t is an update instant **then**

Sparsely update the prototypes : $\forall\, u = 1, \dots, U$ and with $\beta_{u,(1)}^t \geq ... \geq \beta_{u,(\sharp I_u(t))}^t$, set

$$N_{t,u} = \arg \min_{k=1,\dots,\sharp I_u(t)} \left\{ \sum_{i=1}^{k} \beta_{u,(i)}^t \geq \nu \right\}$$

and $\forall\, (i)$, st $i \in I_u(t)$,

$$\beta_{u,(i)}^t \leftarrow \begin{cases} \frac{\beta_{u,(i)}^t}{\sum_{j=1}^{N_{t,u}} \beta_{u,(j)}^t} & \text{if } (i) \leq N_{t,u} \\ 0 & \text{if } (i) > N_{t,u} \end{cases}$$

and $I_u(t) \leftarrow \{i : (i) \leq N_{t,u}\}$.

end if

end for

computation). However, the relation between ν and $Q(\nu)$ is hard to know in advance and can depend on the dataset distribution and on the training itself.

7.4.2 Variants of the sparse updates

In the line of [74], variants of the update step (step 15 in Algorithm 7) can be introduced. More precisely, some of the approaches introduced in [74] can be used almost directly in the online framework. The main difference is that the number of iterations in the online version is much larger than in the batch version. In practice, this implies that the sparse approximation must have a low computational complexity because it is performed several times during the training. We thus restrict ourselves to the following two proposals which are the less computationally costly and can be used instead of step 15 in Algorithm 7 :

— *simple numerical heuristic approximation* (N-**num**) : instead of selecting the first coefficients which sum to a given amount of total mass, one can simply select the first N coefficients for a given N. This changes step 15 into

$$\beta^t_{u,(i)} = \begin{cases} \frac{\beta^t_{u,(i)}}{\sum_{j=1}^{N} \beta^t_{u,(j)}} & \text{if } (i) \leq N \\ 0 & \text{if } (i) > N \end{cases} ,$$

— *simple geometric heuristic approximation* (N-**geom**) : as an alternative, one can search through the observations in I_u the first N observations closest to prototype p_u. Additionally to what is described in [74], we set the coefficients $\beta_{u,i}$ afterwards, in accordance to the new observations used to represent the prototype p_u. Again, we used the previous values of the coefficients as *a priori* for the new ones. Step 15 re-writes :

1. $\tilde{I}_u(t)$ is the set of the first N observations, x_i, in $I_u(t)$ which minimize $\|p_u - \phi(x_i)\|^2$
2. $\boldsymbol{\beta}^t_{u,i} \leftarrow \begin{cases} \frac{\beta^t_{u,i}}{\sum_{j \in \tilde{I}_u(t)} \beta^t_{u,j}} & \text{if } (i) \in \tilde{I}_u(t) \\ 0 & \text{if } i \notin \tilde{I}_u(t) \end{cases} ,$
3. $I_u(t) \leftarrow \tilde{I}_u(t)$

Both approaches described above lead to a fixed number of observations for representing all the prototypes, contrary to what is proposed in Section 7.4.1. The advantage of our proposal is that prototypes may be represented by a number of observations which varies with the density of the data on the map : when a given cluster is isolated from its neighbors, this should yield to a more accurate way to represent the cluster. Also, for all three methods, the sparse approximation is maintained during all the training, as long as the sparse updates are performed frequently enough. This yields to a reduced computational cost of the assignment step (Step 4 in Algorithm 7) and, to a lesser extent, of the representation step (Step 10 in Algorithm 7). As already stated in the previous section, the computational gain is not easily obtained and is a trade-off between the sparsity constraint imposed on the data (which itself depends on the total mass ν or on the number of observations N and on the number of update instants) and the complexity of the sparse representation step itself (Step 15 in Algorithm 7). Further discussions about computational time and efficiency of the different alternatives are provided in the experiments (Section 7.6).

7.5 The case of dissimilarity data

For the sake of clarity, the entire paper has been written in the framework of kernel data. However, its extension to dissimilarity data is almost straightforward, even in the case where the dissimilarity

is not Euclidean. The present section briefly describes the modifications that must be made in this setting.

In the sequel, Δ will denote a dissimilarity matrix with entries $\delta(x_i, x_j)$, the dissimilarity between the observations x_i and x_j. Δ is supposed to have some very basic properties, such as the positiveness of all entries ($\delta(x_i, x_j) \geq 0$), symmetry ($\delta(x_i, x_j) = \delta(x_j, x_i)$) and a null diagonal $\delta(x_i, x_i) = 0$. However, it may not necessarily be Euclidean (see, for instance [123] for a discussion on the additional requirements which make this dissimilarity Euclidean). The extension of SOM to this case is called "relational SOM" and was described in [67, 123]. It is very similar to the kernel case, except that the assignment step of Algorithm 5 is replaced by

$$f^t(x_i) \leftarrow \arg\min_{u=1,\dots,U} \left[\left(\beta_u^{t-1}\right)^\top \mathbf{\Delta}_i - \frac{1}{2} \left(\beta_u^{t-1}\right)^\top \Delta \beta_u^{t-1} \right], \tag{7.7}$$

where $\mathbf{\Delta}_i$ is the i-th row in matrix Δ. This equation is justified by the pseudo-Euclidean framework described in [62] : Δ can be expressed as the difference between dot products of images of the original data by two mapping functions, ψ_1 and ψ_2 into two Euclidean spaces, $\mathcal{E}_1$ and $\mathcal{E}_2$:

$$\delta(x_i, x_j) = \|\psi_1(x_i) - \psi_1(x_j)\|_{\mathcal{E}_1} - \|\psi_2(x_i) - \psi_2(x_j)\|_{\mathcal{E}_2} .$$

In this framework, Equation (7.7) is the exact distance calculation in the space $\mathcal{E}_1 \otimes \mathcal{E}_2$ equipped with the pseudo dot product $\langle ., . \rangle_{\mathcal{E}_1} - \langle ., . \rangle_{\mathcal{E}_2}$. Moreover, if Δ is Euclidean, then the similarity defined by

$$K(x_i, x_j) = -\frac{1}{2} \left(\delta(x_i, x_j) - \frac{1}{n} \sum_{k=1}^{n} (\delta(x_i, x_k) + \delta(x_k, x_j)) + \frac{1}{n^2} \sum_{k,\, k'=1}^{n} \delta(x_k, x_{k'}) \right), \tag{7.8}$$

as suggested in [95], is a kernel and the distances between observations in the feature space induced by this kernel are given by Δ. Thus, Equation (7.7) is exactly equivalent to the assignment step in standard SOM for the kernel K defined by Equation (7.8).

Hence, sparse K-SOM can straightforwardly be extended to dissimilarity data using the assignment step restricted to selected observations in the sparse representation as given in Equation (7.7). For SOM based on dimension reduction, the extension is also easy to obtain. For a non-Euclidean dissimilarity, Δ, the eigendecomposition of the similarity K defined by Equation (7.8) leads to positive and negative eigenvalues (because, in this case, K is not a kernel anymore). If $n_+ \leq n$ denotes the number of positive eigenvalues, the projection of the original data by K-PCA should be restricted to the first p components associated with the p largest eigenvalues such that $p \leq n_+$. This restriction is equivalent to performing K-PCA on a kernel pre-processed with the standard "clip" approach as suggested in [30].

7.6 Experimental results

7.6.1 Methodology

The present section is devoted to the evaluation of the proposed methods on several datasets and from several points of view. First, in Section 7.6.2, the methods presented in this article are assessed on both small and large datasets. This allowed us to compare the performances and computational efficiency of these methods with those of the standard K-SOM algorithm, in an extensive way. In the same section, the sparse approach is also compared, on the two small datasets, to the simple numerical and geometric heuristic approximation methods described in Section 7.4.2. Finally, in Section 7.6.3, the two new methods are used to produce a map on a very large dataset. In Section 7.6.3, computational times are reported and compared, the influence of the size of the prototypes is assessed and the

Nyström approximation is tested. This example also serves as a use case to show how the results can be interpreted.

The datasets were chosen in the framework of this article, *i.e.*, most of them are not standard numeric datasets. They include graphs, genomic sequences and categorical time series to illustrate the approach on a wide range of application and type of (dis)similarities. For every method and every set of parameters that we tested, the experiments were performed with 100 maps, so as to evaluate the variability of our conclusions.

Finally, performances are reported in terms of :

— standard quality measures used to evaluate SOM (see [135] for a description of quality measures in SOM) : i) the *quantization error* (QE), $\sum_{u=1}^{U} \sum_{i:\, x_i \in \mathcal{C}_u} \|\phi(x_i) - p_u\|^2$ with $\|\phi(x_i) - p_u\|^2$ given by the kernel as in Equation (7.3) (K-PCA SOM) or as in Equation (7.6) (sparse K-SOM). This measure is a clustering quality measure, disregarding the map topology ; ii) the *topographic error* (TE) which is the simplest of the topographic preservation measures : it computes the ratio over $(x_i)_i$ of the second best matching units that are in the direct neighborhood of the best matching units on the map.
Since for the K-PCA SOM version, the distances $\|\phi(x_i) - p_u\|^2$ depend on the number of selected axes (the smaller the number of axes, the smaller the distances between observations in the corresponding projected subspace), we have normalized all QE by the average of the squared distances between all pairs of observations in the feature space, $\|\phi(x_i) - \phi(x_j)\|^2$ (for $i \neq j$) to leverage the impact of the feature space metric itself. However, the performances remain difficult to compare directly even with such a normalization, as shown in the results of the simulations. Thus, the *average intra-cluster inertia* (ICI) is also provided. This measure is equal to the average over the units u of $\sum_{i:\, x_i \in \mathcal{C}_u} \|\phi(x_i) - G_{\mathcal{C}_u}\|^2$ in which $G_{\mathcal{C}_u} = \frac{1}{\sharp \mathcal{C}_u} \sum_{i:\, x_i \in \mathcal{C}_u} \phi(x_i)$ is the center of gravity of $\mathcal{C}_u$. We can easily show that it is equal to $\frac{1}{\sharp \mathcal{C}_u} \sum_{i:\, x_i \in \mathcal{C}_u} K(x_i, x_i) - \frac{1}{(\sharp \mathcal{C}_u)^2} \sum_{i,i':\, x_i, x_{i'} \in \mathcal{C}_u} K(x_i, x_{i'})$, for kernel data and to $\frac{1}{2(\sharp \mathcal{C}_u)^2} \sum_{i,i':\, x_i, x_{i'} \in \mathcal{C}_u} \delta(x_i, x_j)$ for dissimilarity data (see Appendix 7.8 for a proof) ;

— quality measures which take advantage of a prior external information : i) if an *a priori* classification of the observations is given, we use the *normalized mutual information* (NMI, [43]) between this *a priori* classification and the clustering induced by the SOM map. A perfect match between the two would correspond to an NMI equal to 1 whereas independent classifications would provide an NMI equal to 0 ; ii) when the studied dataset is a graph and no *a priori* classification is given, the quality of the clustering of the SOM map can nevertheless be evaluated by computing its *modularity* [120]. A high modularity indicates a good clustering with respect to the graph structure ;

— quality measures which aim at providing an indication of how much the results are stable between two runs of the algorithm. To do so, we provide the average NMI between any pair of clustering results obtained from the 100 runs, for a given method and with a given set of parameters. In the sequel, this measure is called *stability*.

7.6.2 Evaluation of the different sparse approaches on various datasets

Description of the datasets : This section aims at evaluating and comparing the two approaches described in Sections 7.3 and 7.4 on various datasets : two small datasets, for an intensive analysis, and three larger datasets, for a better evaluation of the gain in computational time.

More precisely, the two small datasets used for the experiments presented in this section are :

— a graph that gives the frequencies of co-appearance in a same chapter between characters in the novel “Les Misérables” from the French author Victor Hugo. The 77 vertices of the graph are the characters and the 254 edges are weighted by the number of co-appearances in a same chapter.

For this graph, a kernel obtained from the shortest-path lengths is computed [1]. The similarity described in Section 7.5 is used to perform K-PCA SOM : in this case, the similarity K defined in Equation (7.8) is not positive (only the first 67 components are positive) ;

— a dataset that contains 465 input data issued from ten unbalanced sampled species of Amazonian butterflies. This dataset was previously used by [69] to demonstrate the synergy between DNA barcoding and morphological-diversity studies. The notion of DNA barcoding comprises a wide family of molecular and bioinformatics methods aimed at identifying biological specimens and assigning them to a species. DNA barcoding data are composed of sequences of nucleotides, *i.e.*, sequences of "A", "C", "G", "T" letters in high dimension (hundreds or thousands of sites). Specific distances and dissimilarities such as the Kimura-2P [82] are usually computed and used in the experiments. A similarity was obtained from the Kimura-2P dissimilarity as described in Equation (7.8) to perform K-PCA. Again, this similarity is not positive. Only 246 eigenvalues are positive in the eigendecomposition of K.

These two datasets will be denoted, respectively, by "lesmis" and "astraptes" in the sequel.

Additionally, three larger datasets are also used for comparison :

— a graph whose edges model hyper-links between blogs on US politics, recorded in 2005 by [2] [2]. The 1,222 vertices of the graph represent the political blogs and are labeled by the political leaning (liberal or conservative). The graph contains 19,089 edges. The similarity used for this graph is again obtained from Equation (7.8), using the shortest-path lengths as the original dissimilarity measure. The undirected version of the shortest-path length was used to provide a symmetric dissimilarity, even though the original graph is directed. Only the first 779 eigenvalues of this similarity are positive ;

— a DNA dataset similar to the "astraptes" data, except with a larger size. This dataset contains 614 sites of the CoI locus for 2,036 samples issued from the cowries family. The data were introduced in the DNA barcoding context in [116]. In order to assess the ability for clustering of the proposed algorithms, the species with very few observations were removed. The final dataset used for simulations contained 1,414 samples and 47 species, the number of observations per species varying from 11 to 140 observations. The species were used as *a priori* classes for computing the NMI. Here also, the Kimura-2P dissimilarity [82] was used for deriving the similarity matrix, for which only the first 476 eigenvalues were positive.

— a (standard) numerical dataset related to red variants of the Portuguese "Vinho Verde" wine [37] [3]. The dataset contains 4 898 observations of 12 numeric variables based on physicochemical tests, such as the pH, the sulphates or the residual sugar. Additionally, the quality (a score between 0 and 10) of the wines is provided, which is used as an *a priori* class to compute the quality measure (NMI) described in Section 7.6.1 (and thus it is not used to compute the similarity). To stuck to the framework of the article, the Gaussian kernel was used to obtain a measure of similarity between wines : $K_{ij} = e^{-\sigma \|x_i - x_j\|^2}$ with σ equal to the median of $\left\{ \frac{1}{\|x_i - x_j\|^2} \right\}_{i<j}$.

In the sequel, these datasets will be refered as "polblogs", "cowrie" and "wines", respectively.

Evaluation of K-PCA SOM and direct sparse K-SOM : More precisely, the following methods are compared on these five datasets :

— a standard kernel SOM, K-SOM (or relational SOM if the dissimilarity is not Euclidean) ;

— K-PCA SOM, as described in Section 7.3. The datasets used in this section are not large enough to allow a relevant use of the Nyström approximation of the K-PCA ;

— direct sparse K-SOM (or relational SOM) as described in Section 7.4, and denoted by sparse K-SOM in the sequel.

1. The graph as well as the shortest path-lengths are included in the R package **SOMbrero** [16].
2. The graph is available at `http://www-personal.umich.edu/~mejn/netdata/polblogs.zip`.
3. The dataset is available at `https://archive.ics.uci.edu/ml/datasets/Wine+Quality`.

Table 7.1 Performance results of K-PCA SOM and sparse K-SOM (average over 100 maps and standard deviation between parenthesis) for the "lesmis" dataset. Parameters for the methods are given between parenthesis after the method name (% of entropy preserved in the projection for K-PCA SOM and maximum mass, ν, and update parameter, κ, for random ascending updates in sparse K-SOM).

Methods	QE ($\times 100$)	ICI ($\times 100$)	TE (%)	Modularity ($\times 100$)	Stability (%)	Dimension
K-SOM	23.40 (0.44)	47.57 (2.53)	3.01 (2.52)	31.76 (2.80)	85.04 (2.13)	77
K-PCA (80%)	18.30 (0.46)	47.31 (2.51)	3.30 (3.01)	32.02 (3.27)	86.20 (2.21)	27
K-PCA (60%)	10.32 (0.55)	48.09 (2.41)	3.36 (3.65)	**32.17** (2.63)	89.56 (2.17)	11
K-PCA (40%)	2.55 (0.25)	52.62 (2.42)	**2.64** (3.30)	25.96 (1.77)	91.85 (2.08)	4
K-PCA (20%)	**0.92** (0.18)	52.35 (2.21)	2.83 (3.32)	24.53 (1.92)	**92.68** (2.02)	2
sparse (90%, 1)	29.34 (1.20)	48.80 (2.88)	9.64 (7.00)	31.44 (4.09)	77.72 (2.94)	4
sparse (90%, 50)	23.25 (5.61)	**46.63** (2.70)	4.45 (3.62)	30.96 (3.03)	83.64 (2.34)	8
sparse (99%, 1)	23.36 (4.06)	**46.63** (2.71)	3.25 (2.85)	31.40 (3.03)	84.64 (2.18)	13
sparse (99%, 50)	23.40 (4.49)	47.07 (2.57)	3.31 (2.72)	31.74 (3.41)	84.99 (2.35)	15

All methods were trained for a 5×5 grid ("lesmis"), a 8×8 grid ("astraptes") and a 10×10 grid ("polblogs", "cowrie" and "wines") with respectively 500 ("lesmis"), 2500 ("astraptes"), 6000 ("polblogs"), 7000 ("cowrie") and 8000 ("wines") iterations. These choices approximately correspond to default parameters in **SOMbrero**. All grids were equipped with a piecewise linear neighborhood with the distance between units calculated as the Euclidean distance between the unit coordinates in $\mathbb{N}^{*2}$.

Furthermore, different sets of parameters, corresponding to different levels of sparsity, were tested :

— for K-PCA SOM, the only parameter to set was the dimension of the projection, p. This parameter was chosen with respect to a minimal ratio of entropy preserved in the projection and this ratio itself was varied in $\{20\%, 40\%, 60\%, 80\%\}$;

— for the direct sparse K-SOM, two parameters had to be set : the first one is the mass parameter ν which was varied in $\{90\%, 99\%\}$ for the two smallest datasets ("lesmis" and "astraptes") and in $\{95\%, 99\%\}$ for the three largest datasets ("polblogs", "cowrie" and "wines") to ensure sparse results. The second parameter calibrated the update instants : random ascending updates were performed with $\kappa \in \{1, 50\}$ for both datasets.

Tables 7.1-7.5 provide the results obtained over 100 maps (mean and standard deviation) for different quality criteria. For the smallest datasets, we focused on providing many different measures of the quality of the resulting map, as compared to the one obtained by the standard K-SOM : normalized QE, ICI, TE and modularity ("lesmis") or NMI ("astraptes"), as described in Section 7.6.1, are given and the last column of the results provides the final dimension of the prototypes (either the exact dimension for K-PCA SOM or the average over all prototypes for the sparse K-SOM). The computational time is not reported because, for these small datasets, it is not relevant.

For the three largest datasets, the computational time is reported but the results are less exhaustive on quality criteria (only the most important ones are reported : ICI, TE, modularity/NMI and stability). Also, for K-PCA SOM, only the clustering time is given. The computational time for the K-PCA itself is ~ 2.59 seconds for "polblogs", ~ 3.73 seconds for "cowrie" and ~ 5.65 seconds for "wines", which is less than the computational time needed to train the map in all cases.

Discussion on the comparison : Results demonstrate a high efficiency of both the DR method and the sparse approach to decrease data dimensionality while producing accurate results in a reasonable computational time. For almost all quality criteria, both approaches introduced in the manuscript are in the range of or outperform the results obtained with the direct K-SOM at a very reduced computational time and a limited dimensionality of the resulting prototypes.

Table 7.2 Performance results of K-PCA SOM and sparse K-SOM (average over 100 maps and standard deviation between parenthesis) for the "astraptes" dataset. Parameters for the methods are given between parenthesis after the method name (% of entropy preserved in the projection for K-PCA SOM and maximum mass, ν, and update parameter, κ, for random ascending updates in sparse K-SOM).

Methods	QE ($\times 100$)	ICI ($\times 10^4$)	TE (%)	NMI (%)	Stability (%)	Dimension
K-SOM	1.02 (0.06)	3.37 (0.43)	**1.39** (1.99)	82.19 (0.59)	94.57 (0.88)	459
K-PCA (80%)	0.21 (0.02)	4.34 (0.70)	1.79 (2.20)	**90.73** (2.86)	**95.20** (1.05)	5
K-PCA (60%)	0.10 (0.01)	6.40 (1.42)	2.71 (2.64)	80.99 (0.67)	94.73 (0.96)	3
K-PCA (40%)	0.06 (0.01)	9.33 (1.52)	2.64 (2.27)	79.14 (0.59)	94.49 (0.95)	2
K-PCA (20%)	**0.01** (0.00)	23.15 (2.06)	24.55 (8.84)	73.02 (0.61)	94.70 (0.85)	1
sparse (90%, 1)	2.08 (1.46)	3.59 (8.97)	2.94 (3.09)	86.34 (1.24)	91.87 (1.40)	5
sparse (90%, 50)	0.99 (0.10)	**3.08** (5.70)	3.72 (4.16)	82.60 (0.66)	93.62 (0.86)	8
sparse (99%, 1)	0.98 (0.05)	3.30 (4.90)	1.51 (2.54)	82.22 (0.58)	94.66 (0.84)	25
sparse (99%, 50)	20.99 (0.06)	3.38 (4.55)	**1.39** (1.63)	82.16 (0.59)	94.73 (0.77)	25

Table 7.3 Performance results of K-PCA SOM and sparse K-SOM (average over 100 maps and standard deviation between parenthesis) for the "polblogs" dataset. Parameters for the methods are given between parenthesis after the method name (% of entropy preserved in the projection for K-PCA SOM and maximum mass, ν, and update parameter, κ, for random ascending updates in sparse K-SOM).

Methods	ICI ($\times 100$)	TE (%)	NMI (%)	Stability (%)	CPU time	Dimension
K-SOM	84.01 (0.85)	21.93 (1.51)	20.56 (0.30)	64.81 (0.73)	18269 (2378)	1599
K-PCA (80%)	86.39 (0.84)	14.01 (1.42)	20.93 (0.27)	69.68 (0.79)	27.60 (4.45)	257
K-PCA (60%)	86.61 (0.77)	14.87 (1.35)	20.96 (0.26)	70.62 (0.82)	18.88 (2.12)	121
K-PCA (40%)	89.35 (2.28)	13.56 (2.12)	21.14 (0.32)	60.27 (1.73)	16.55 (2.45)	50
K-PCA (20%)	91.46 (0.87)	**12.06** (1.51)	**21.30** (0.25)	**77.19** (0.77)	**13.35** (2.35)	13
sparse (95%, 1)	85.20 (1.40)	34.03 (2.80)	20.64 (0.41)	55.14 (0.41)	60.13 (2.50)	8
sparse (95%, 50)	**84.00** (0.89)	32.61 (2.31)	20.66 (0.32)	60.39 (0.32)	67.89 (7.86)	12
sparse (99%, 1)	84.18 (0.75)	23.95 (1.66)	20.46 (0.30)	63.98 (0.30)	88.19 (5.30)	34
sparse (99%, 50)	84.06 (0.76)	24.15 (1.62)	20.51 (0.30)	64.30 (0.30)	96.73 (6.75)	34

Table 7.4 Performance results of K-PCA SOM and sparse K-SOM (average over 100 maps and standard deviation between parenthesis) for the "cowrie" dataset. Parameters for the methods are given between parenthesis after the method name (% of entropy preserved in the projection for K-PCA SOM and maximum mass, ν, and update parameter, κ, for random ascending updates in sparse K-SOM).

Methods	ICI ($\times 100$)	TE (%)	NMI (%)	Stability (%)	CPU time	Dimension
K-SOM	2.07 (0.24)	1.57 (1.29)	**90.45** (0.86)	**94.79** (1.04)	3771 (1534)	1414
K-PCA (80%)	2.05 (0.19)	1.92 (1.39)	90.26 (0.73)	94.55 (0.89)	16.68 (1.45)	23
K-PCA (60%)	1.94 (0.21)	2.33 (1.33)	89.11 (0.81)	94.04 (0.80)	15.19 (1.31)	10
K-PCA (40%)	3.15 (0.30)	3.85 (1.96)	85.72 (0.80)	90.49 (0.93)	15.35 (2.41)	4
K-PCA (20%)	6.26 (0.28)	5.11 (2.18)	74.47 (0.48)	87.99 (1.17)	**15.04** (2.27)	2
sparse (95%, 1)	2.09 (0.26)	2.04 (1.31)	89.64 (0.97)	91.78 (1.18)	70.03 (1.23)	8
sparse (95%, 50)	**1.84** (0.20)	2.38 (1.82)	90.37 (0.70)	93.78 (0.92)	62.66 (3.27)	13
sparse (99%, 1)	20.36 (0.22)	1.55 (1.36)	90.34 (0.79)	94.45 (0.91)	90.73 (4.16)	37
sparse (99%, 50)	20.00 (0.22)	**1.54** (1.24)	90.37 (0.79)	94.50 (0.93)	81.14 (2.89)	37

More specifically, QE is always better for K-PCA SOM with the smallest dimensionality. However, this is merely an effect of the dimensionality of the input data and is not a reliable criterion to compare results. ICI is a better way to measure the cluster homogeneity and shows that in almost all cases, K-PCA

Table 7.5 Performance results of K-PCA SOM and sparse K-SOM (average over 100 maps and standard deviation between parenthesis) for the "wines" dataset. Parameters for the methods are given between parenthesis after the method name (% of entropy preserved in the projection for K-PCA SOM and maximum mass, ν, and update parameter, κ, for random ascending updates in sparse K-SOM).

Methods	ICI ($\times 100$)	TE (%)	NMI (%)	Stability (%)	CPU time	Dimension
K-SOM	22.10 (0.50)	10.37 (1.34)	**11.86** (0.31)	74.23 (0.81)	13480 (10575)	1222
K-PCA (80%)	**21.94** (0.51)	10.15 (1.31)	11.78 (0.30)	74.96 (0.77)	20.00 (3.29)	28
K-PCA (60%)	22.43 (0.61)	8.80 (1.30)	11.72 (0.34)	75.37 (0.88)	17.95 (1.74)	9
K-PCA (40%)	25.44 (0.74)	7.08 (1.33)	**11.86** (0.26)	77.65 (0.88)	17.32 (2.33)	4
K-PCA (20%)	34.99 (0.36)	**0.13** (0.72)	11.01 (0.23)	**85.75** (1.68)	**15.99** (1.88)	2
sparse (95%, 1)	47.62 (1.46)	14.29 (1.35)	11.75 (0.36)	66.21 (0.79)	81.48 (5.02)	8
sparse (95%, 50)	45.90 (1.08)	11.95 (1.39)	11.76 (0.38)	70.72 (0.77)	97.60 (14.28)	13
sparse (99%, 1)	44.14 (0.95)	11.26 (1.36)	11.80 (0.30)	74.00 (0.73)	141.17 (12.28)	40
sparse (99%, 50)	44.17 (0.93)	11.58 (1.24)	11.83 (0.32)	74.13 (0.77)	146.93 (14.98)	39

SOM and sparse K-SOM have comparable or even better ICI than the original approach (standard K-SOM) with a slight advantage for sparse K-SOM. Only, sparse K-SOM with the "wines" dataset exhibits a poor performance for this criterion.

The quality of the organization of the map is measured with TE, which is often very comparable in both approaches to the direct K-SOM. However, it is again poor for the sparse K-SOM applied to the "wines" dataset and frequently tends to increase when ICI decreases. A good compromise between ICI and TE is reached for K-PCA SOM with an preserved entropy rate of 60% and for sparse K-SOM with ν equal to 90%/95% and κ equal to 50 for all datasets.

When comparing the different results with the *a priori* external information (through modularity or NMI), again the proposed approaches are comparable to or even better than the direct sparse K-SOM. However, in some cases ("lesmis", "astraptes", "cowrie"), it tends to highly deteriorate when the dimensionality is decreasing. This shows that a good tradeoff has to be found between interpretability (small dimensionality) and preservation of most of the information from the original dataset.

Finally, the stability of the results is often increased by K-PCA SOM (except for the "cowrie" dataset). This is an expected behavior : since the dimensionality of the input dataset is reduced, the prototypes lies in a reduced dimensionality space with less degrees of freedom. On the contrary, sparse K-SOM is not as appealing for this criterion because, even if the prototype representation is constrained, it is expressed in the original data space, which has a high dimension.

These good performance of our approaches come along with a high computational efficiency : K-PCA SOM and sparse K-SOM allows to highly reduce the computational times. Even if the K-PCA itself is a problem with a high complexity, it does not affect much the efficiency of K-PCA SOM on large datasets with a few thousands observations. The reason is that relational SOM is itself a time consuming algorithm, as explained in the introduction of this article. For a more challenging dataset (with more than ten thousands observations), a comparison and discussion is provided in Section 7.6.3.

In conclusion, both approaches introduced in this article are valid alternative to the direct K-SOM for large relational datasets. They both provide simplified prototype representation at a very reduced computational time. The dimensionality of the final prototypes is in all our examples 10 times or even 100 times smaller than in the direct sparse SOM version with no loss in accuracy of the resulting map. However, it should be noted that the easiness to interpret the prototypes is not exactly equivalent in K-PCA SOM and in sparse K-SOM : in the first approach, prototypes are expressed on the axes of the K-PCA which have to be interpreted in a previous step of the analysis, whereas, in sparse K-SOM, they are directly related to (a few number of) original observations. A detailed study is then presented in Sections 7.6.3 and 7.6.3.

Comparison with other approaches : In this section, the datasets "lesmis" and "astraptes" presented in Section 7.6.2 are used to compare the sparse approach to the simple numerical heuristic

Table 7.6 Comparison between the different variants for the sparse updates (average over 100 maps and standard deviation between parenthesis) for the "lesmis" dataset. Parameters for the methods are given between parenthesis after the method name (maximum mass, ν, and update parameter, κ, for random ascending updates).

Methods	QE ($\times 100$)	ICI ($\times 100$)	TE (%)	Modularity ($\times 100$)	Stability (%)
N-num (4, 1)	31.36 (1.66)	50.12 (3.16)	20.72 (9.37)	31.13 (4.23)	75.31 (3.49)
N-num (8, 50)	23.35 (0.53)	46.89 (2.60)	4.71 (3.36)	31.39 (3.14)	83.62 (2.20)
N-num (13, 1)	23.45 (0.45)	46.99 (2.55)	3.38 (2.99)	31.85 (3.07)	**84.85** (2.32)
N-num (15, 50)	23.40 (0.44)	47.04 (2.28)	**3.21** (2.44)	**31.92** (3.20)	84.41 (2.41)
N-geom (4, 1)	35.09 (3.31)	51.56 (4.95)	22.31 (12.91)	25.90 (3.45)	72.14 (4.93)
N-geom (8, 50)	23.76 (0.81)	**45.00** (2.66)	7.49 (4.90)	24.40 (3.06)	81.06 (2.47)
N-geom (13, 1)	25.32 (1.37)	46.26 (2.39)	5.48 (4.08)	27.27 (2.54)	82.10 (2.40)
N-geom (15, 50)	**23.28** (0.53)	46.45 (2.56)	4.77 (3.60)	29.12 (2.89)	83.43 (2.14)

Table 7.7 Comparison between the different variants for the sparse updates (average over 100 maps and standard deviation between parenthesis) for the "astraptes" dataset. Parameters for the methods are given between parenthesis after the method name (maximum mass, ν, and update parameter, κ, for random ascending updates).

Methods	QE ($\times 100$)	ICI ($\times 10^4$)	TE (%)	NMI (%)	Stability (%)
N-num (5, 1)	3.27 (0.82)	3.80 (0.90)	6.56 (5.47)	**88.08** (1.56)	90.88 (1.98)
N-num (8, 50)	1.25 (0.13)	**3.28** (0.62)	2.88 (3.33)	83.51 (0.79)	93.14 (0.98)
N-num (25, 1)	1.01 (0.72)	3.36 (0.49)	1.99 (2.80)	82.20 (0.65)	94.42 (0.90)
N-num (25, 50)	**1.00** (0.07)	3.34 (0.48)	**1.71** (2.37)	82.10 (0.59)	**94.44** (0.82)
N-geom (5, 1)	28.44 (12.16)	27.08 (11.37)	35.95 (14.65)	77.44 (6.99)	75.30 (6.72)
N-geom (8, 50)	2.23 (1.36)	5.82 (1.13)	11.45 (6.99)	83.77 (1.22)	92.71 (1.21)
N-geom (25, 1)	29.86 (7.45)	20.12 (7.96)	22.00 (9.73)	81.47 (2.51)	88.51 (2.90)
N-geom (25, 50)	1.74 (0.96)	5.01 (0.79)	8.72 (5.93)	83.19 (0.79)	93.84 (1.08)

approximation (denoted N-num) and the simple geometric heuristic approximation (denoted N-geom) described in Section 7.4.2. Both strategies lead to a fixed number of observations for representing all the prototypes. Thus, for a fair comparison, this number was chosen considering the average final dimensionality of the prototypes obtained with the sparse method (last column in Tables 7.1 and 7.2). The random ascending updates are performed using the same setting than in the original sparse version (*i.e.*, random ascending updates $\kappa \in \{1, 50\}$).

Results obtained with N-num and N-geom over 100 maps are provided in Table 7.6 for "lesmis" and in Table 7.7 for "astraptes". In average, for both datasets, N-num gives better results than N-geom. As observed with the sparse method, both N-num and N-geom obtain best results with a final dimension equal to 8 for "lesmis". For all quality criteria except for the ICI, our variant of the sparse updates slightly improves the heuristic approximation approaches. This conclusion is supported by the results on "astraptes", except that N-num gives a better classification than our variant of the sparse updates.

For the same level of sparsity, the results obtained with both N-num and N-geom slightly deteriorate the map quality compared to what can be observed with our version of the sparse updates.

7.6.3 Using K-PCA SOM and sparse K-SOM for mining job trajectories

This section presents the experiments performed on a more realistic dataset, with a larger sample size, obtained from the survey "Generation 98" [38, 33]. The dataset contains information on career paths of 16,040 young people monitored during 94 months after having graduated in 1998. Nine categories are used to describe labor market statuses : permanent-labor contract, fixed-term contract, apprenticeship contract, public temporary-labor contract, on-call contract, unemployed, inactive, military service, education. Dissimilarities between career trajectories were computed using the optimal matching [118, 1] on the 12,560 unique career paths. This resulted in a non positive dissimilarity (6,651 eigenvalues out of 12 500 were found positive).

To assess the accuracy and the computational cost of both K-PCA SOM and sparse SOM, 100 maps were trained, using an R implementation of the methods, on the same 40-nodes computer without concurrent access. All maps were trained for a 10×10 grid, equipped with a piecewise linear neighborhood, with 60,000 iterations. The entropy ratio preserved by K-PCA SOM was varied in $\{20\%, 40\%, 60\%, 80\%\}$. For sparse K-SOM, the mass parameter ν was varied in $\{95\%, 99\%\}$ and the update parameter κ was varied in $\{1, 50\}$. Only 10 maps were trained using the standard K-SOM due to its very high computational cost (more than ten days for each map).

Table 7.8 presents the results obtained in terms of QE, ICI, TE and CPU time (only the clustering time is reported, the K-PCA computational time is $\sim 8,000$ seconds. A detailed study to address this issue is made in Section 7.6.3). The last column provides the final dimension or number of coefficients of the prototypes.

As observed in Section 7.6.2, results demonstrate a high efficiency, in term of computational cost, of both K-PCA SOM and sparse K-SOM, while producing accurate results. Direct K-SOM takes more than ten days to train one map, whereas the slowest alternative strategy only requires sixteen minutes (*i.e.*, the sparse K-SOM with $\nu = 99\%$ and $\kappa = 1$). The results also confirm what was found with the toy datasets : K-PCA SOM provides a good trade-off between a good map quality and low dimensional prototypes and outperforms sparse K-SOM. The best results for K-PCA SOM are obtained with 20% entropy-rate preserved. However, this strategy selects only two dimensions, which increases the redundancy in the data and tends to produce clusters with few observations. Thus, the K-PCA SOM preserving 40% entropy should be preferred. The best results for the sparse K-SOM are obtained with a mass equal to 95%.

Both K-PCA SOM and sparse K-SOM provide accurate results in a reasonable computational time. For sparse K-SOM, prototypes can be interpreted by inspecting the properties of the few observations used to represent them. For K-PCA SOM, the projection of the data on a subspace requires to interpret the axes of the K-PCA as an extra step in order to understand the meaning of the prototypes. A detailed study showing how the results of both approaches can be interpreted is performed in Sections 7.6.3 and 7.6.3.

Analysis of the influence of the prototype size : The job trajectory dataset was further used to assess the influence of the size of the prototypes on different characteristics. More precisely, we conducted an experiment to analyze the relation between the dimension of the projection (in K-PCA SOM) or the average number of coefficients per prototype (in sparse K-SOM) and some numerical characteristics of the algorithm : ICI, TE, CPU time and stability. A larger set of parameters, corresponding to varying dimensions of the prototypes, were tested by training 20 maps with each value of the parameters :

— for K-PCA SOM, the number of dimensions used for the projection was varied in $\{5, 10, 15, 20, 25, 50, 75, 100, 250, 500, 1000, 5000\}$. Figure 7.1 displays the evolution of different numerical characteristics of the maps versus the projection dimension ;

— for sparse K-SOM, following the results given in Table 7.8, we set the random ascending update parameter κ to 1 and varied the maximum mass, ν in $\{0.9, 0.95, 0.975, 0.99, 0.995\}$. Figure 7.2

Table 7.8 Performance results of K-PCA SOM and sparse K-SOM (average over 100 maps and standard deviation between parenthesis) for the "trajectories" dataset. Parameters for the methods are given between parenthesis after the method name (% of entropy preserved in the projection for K-PCA SOM and maximum mass, ν, and update parameter, κ, for random ascending updates in sparse K-SOM).

Methods	QE ($\times 100$)	ICI	TE (%)	CPU time	Stability (%)	Dimension
K-SOM	20.99 (0.12)	**23.94** (0.24)	7.91 (0.66)	949582 (1 373)	77.65 (3.66)	12 500
K-PCA (80%)	23.49 (0.10)	24.07 (0.31)	8.27 (0.92)	251 (78)	75.81 (1.82)	392
K-PCA (60%)	15.36 (0.12)	24.32 (0.32)	8.57 (0.77)	136 (44)	75.72 (1.95)	44
K-PCA (40%)	5.61 (0.09)	26.26 (0.37)	6.98 (0.75)	114 (40)	77.13 (3.24)	8
K-PCA (20%)	**0.37** (0.00)	31.92 (0.95)	**0.82** (0.86)	**112** (33)	**86.31** (5.69)	2
sparse (95%, 1)	32.40 (0.74)	28.66 (1.36)	30.86 (6.88)	378 (3)	55.79 (1.37)	14
sparse (95%, 50)	25.36 (0.35)	26.57 (0.59)	11.15 (1.86)	655 (32)	63.64 (0.88)	14
sparse (99%, 1)	25.11 (0.17)	27.09 (0.46)	5.26 (0.72)	1025 (194)	68.08 (1.25)	50
sparse (99%, 50)	26.76 (0.36)	27.36 (0.71)	31.59 (4.53)	381 (28)	59.81 (0.90)	8

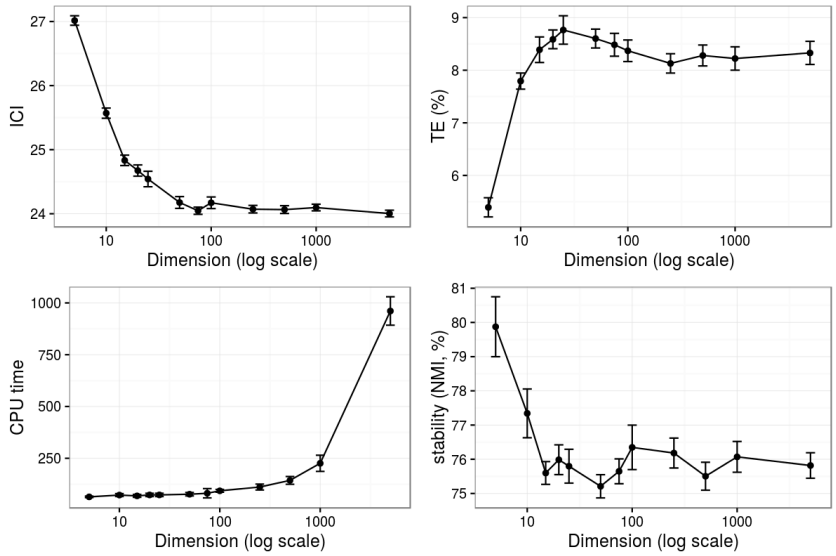

Figure 7.1 **K-PCA SOM**. Average performances over 20 maps (ICI, TE, computational time in seconds and stability as measured by NMI) versus the dimensionality. Error bars correspond to the standard error $\frac{\text{standard deviation}}{\sqrt{20}}$.

provides the evolution of different numerical characteristics of the maps versus the average number of coefficients per prototype.

For K-PCA SOM, up to approximately 50 (over a maximum of $12,500$), the dimension seems to have only a very limited effect on the quality of the map. All quality criteria stabilize after this value, with a slight tendency to improve and then to deteriorate again for very high dimensions. A strong computational time benefit can be observed when decreasing the dimensionality below $1,000$: this is a direct consequence of the quadratic complexity of K-SOM.

For sparse K-SOM, all characteristics, except for CPU time, tend to improve when the number of coefficients increases. However, TE seems to stabilize for approximately 50 coefficients per prototypes ($\nu = 0.99$). However, since the computational time is not reduced from the same amount than in K-PCA SOM, testing the value of ν is not a good strategy. Moreover, $\nu = 0.995$ gives prototypes with approximately 80 coefficients, which must not be increased to preserve interpretability.

Interpretability of K-PCA SOM : In this section, the map with the lowest ICI, among the 100 generated by the K-PCA SOM with 40% preserved entropy rate, is used to show how the results of

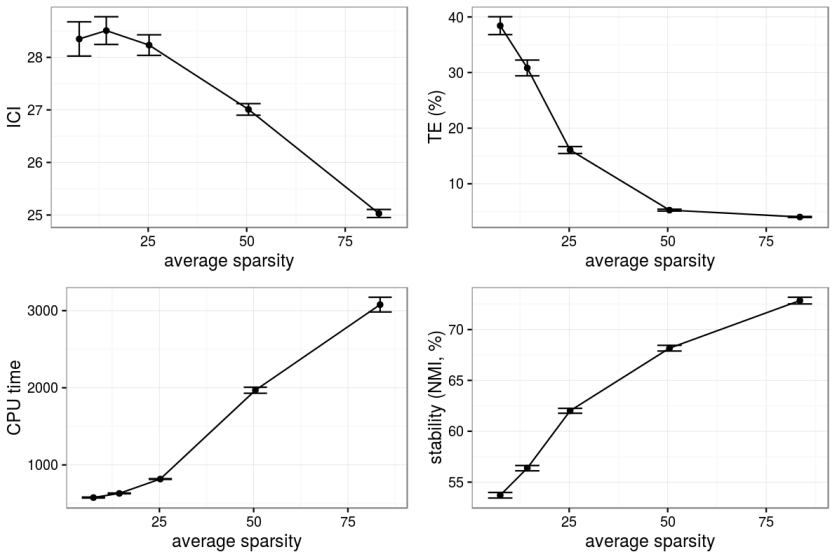

Figure 7.2 **Sparse K-SOM**. Average performances over 20 maps (ICI, TE, computational time in seconds and stability as measured by NMI) versus the average sparsity (average number of coefficients per prototypes) for different values of ν. Error bars correspond to the standard error $\frac{\text{standard deviation}}{\sqrt{20}}$.

K-PCA SOM can be interpreted, despite the K-PCA pre-processing. Its performances are equal to 5.74 (QE), 25.42 (ICI) and 7.66 (TE). The projection of the data on a subspace requires to interpret the K-PCA axes first. Figure 7.3 (left) presents the entropy supported by the first 15 axes and shows that the first two axes are enough to provide relevant information on the data. Figure 7.3 (right) displays the projections of the observations on the first two principal axes. The first axis represents 16.90% of the total entropy and opposes permanent-labor and fixed term contracts. This is supported by Figure 7.4 which shows the distribution of the 25 career paths with the smallest and the highest coordinates on the first two axes of the K-PCA. Stable job trajectories have the smallest coordinates on the first axis when fixed-term contract or unemployed have the highest. Figure 7.3 (right) also demonstrates that the second axis separates two kinds of precarious situations. Fixed-term contracts are opposed to highly precarious contracts such as unemployment, inactivity, on-call contract and public temporary-labor contract. The same observations can be made regarding the first 25 career paths with the smallest and highest coordinates on the second K-PCA axis, as shown in Figure 7.4.

The distribution of the job trajectories within each neuron of the map is represented in Figure 7.5. First note that the presented map is comparable in term of topology to the one described in [123]. Different typologies can be highlighted : a fast access to permanent contracts (clear blue) on the bottom-left corner of the map, a transition through fixed-term contracts before obtaining stable ones (dark and then clear blue) on the map top-left corner, temporary jobs (dark blue) on the top-middle neurons, a long period of inactivity (yellow) or unemployment (red) on the map bottom-right corner.

The map organization is in accordance with the axis interpretation. Figure 7.6 (top) displays the average coordinates on the first and second axes in every cluster of the map. Results show a gradient of the observation coordinates on the first K-PCA axis between the bottom-left and the right side of the map. This gradient can also be seen on the map shown in Figure 7.5 and on the heatmaps presented on Figure 7.6 (bottom) which represent the cluster average of the career path modes [4]. This confirms that the first principal component (and corresponding diagonal on the map) separates permanent contracts from instable career paths. In Figure 7.6 (top), a gradient can also be observed for the second K-PCA axis between the top-left, where trajectories correspond to a fast access to permanent-labor contracts and the bottom-left corner of the map, where trajectories pertaining to precarious jobs are gathered.

4. To compute mode averages, job market contracts have been converted to numerical labels from 1, for the permanent-labor contract, to 9 for education.

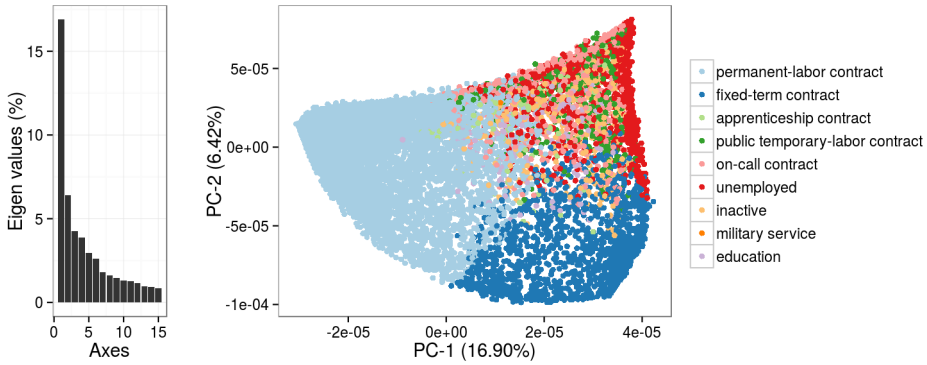

Figure 7.3 Entropy preserved by the 15 first axes on the left and projection of the observations on the first two principal components on the right. Colors represent the contract that appears the most often (mode) in the trajectory.

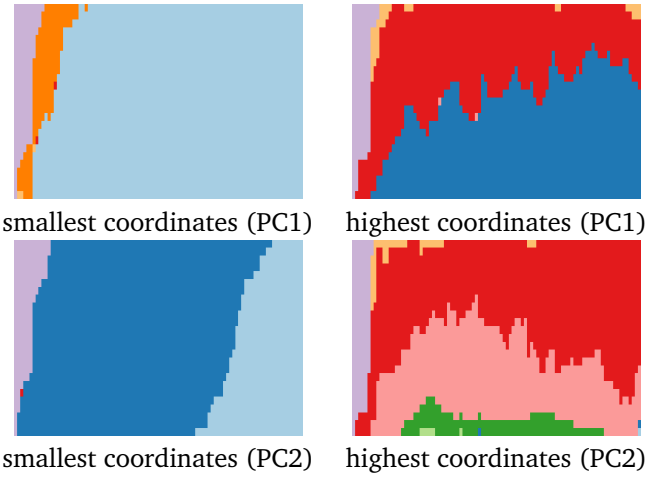

Figure 7.4 Distribution of the 25 career paths with smallest and highest coordinates on the first two axes of the K-PCA.

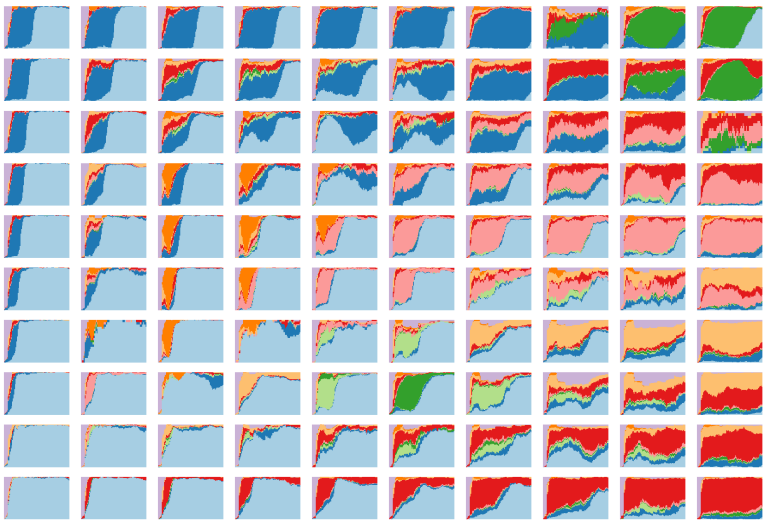

Figure 7.5 **K-PCA SOM**. For each neuron of the map, job trajectories distribution is represented using the observations classified in the corresponding unit. Colors represent the type of contract.

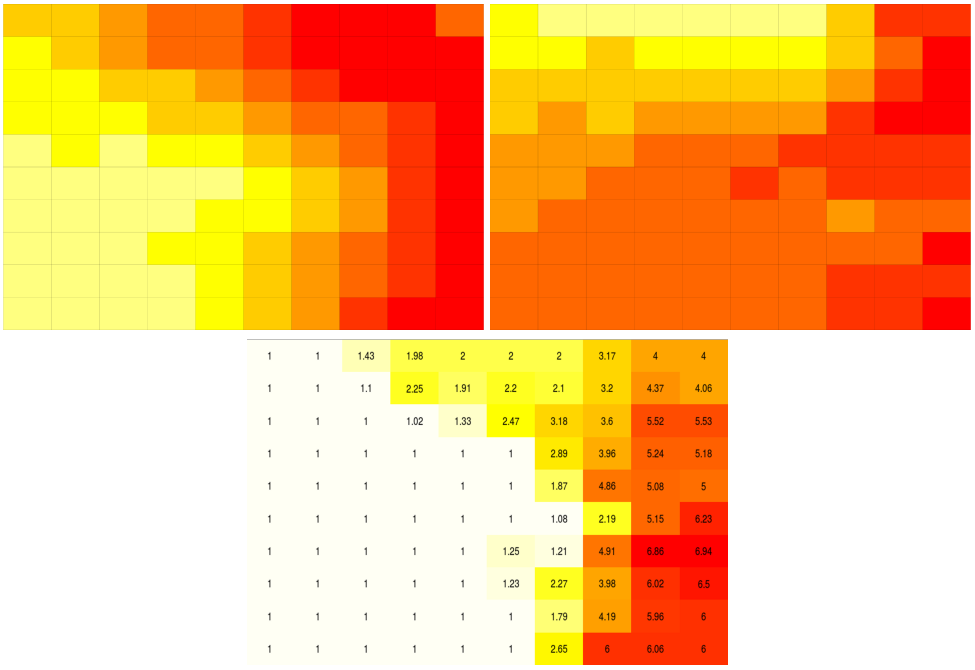

Figure 7.6 **K-PCA SOM**. Representation of the SOM map with neurons filled using colors according to the average coordinate of the observations for the first (on the top-left) and the second (on the top-right) principal component. On the bottom, the map neurons are filled using colors according to the average of the career path modes.

Interpretability of sparse K-SOM : Similarly to the previous section, the present section provides a short discussion about one of the final results obtained from sparse K-SOM. The selected map is again the one with the smallest ICI among all maps obtained with $\nu = 95\%$ and $\kappa = 50$. It gives better performances in term of ICI (24.87) than K-PCA SOM but QE (25.27) and TE (10.8) are increased.

The resulting distribution of the job trajectories within the clusters of the map is provided in Figure 7.7. This distribution is fairly similar to the one obtained in Section 7.6.3 : the left hand side of the map corresponds to a fast access to permanent contracts whereas the right hand side corresponds to different types of precarious situations. Two main differences can be highlighted : first, the class are more homogeneous in sparse K-SOM, especially at the border of the map. This is a direct effect of the dimension reduction in K-PCA SOM : since the dimension reduction increases redundancy in the dataset, some clusters (mostly located at the borders of the map) contain more observations and are thus less homogeneous. Second, the precarious situations (on the right hand side of the map) are organized a bit differently (with on-call contracts in the middle or the bottom of the map). However, both representations are realistic, with most of the clusters in the map being homogeneous.

A similar representation is provided in Figure 7.8 (left) but restricted to the observations which are involved in the prototype definition. The right part of this figure displays the number of such observations. Two conclusions can be derived from these graphics : the first one is that the observations involved in the prototype definition have a distribution very similar to the distribution of the entire set of observations included in the corresponding cluster. They are thus a selected subset of observations representative of their cluster. Moreover, as their number is very restricted compared to the total number of observations included in a cluster (approximately 14/15 observations as shown in the left part of Figure 7.8), they are a convenient way for the user to make sense of the prototypes and thus, of the corresponding cluster, since an exhaustive inspection of these observations becomes possible.

Nyström approximation : To evaluate the relevance of using a Nyström approximation of the K-PCA, the K-PCA SOM with 40% preserved entropy rate is used as a reference. Table 7.9 presents K-PCA SOM results using a Nyström approximation with different rates of observations sampled to perform the approximation. This rate was varied in $\{100\%, 25\%, 10\%, 5\%, 1\%\}$, in which the 100%-results are

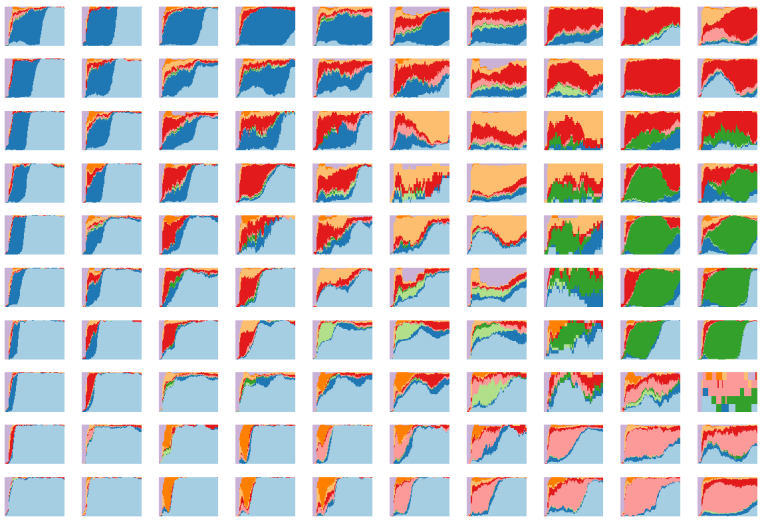

Figure 7.7 **Sparse K-SOM**. For each neuron of the map, job trajectories distribution is represented using the observations classified in the corresponding unit. Colors represent the type of contract.

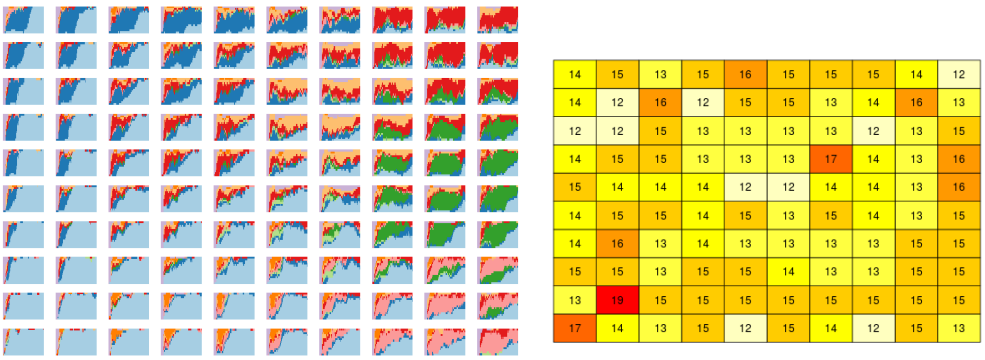

Figure 7.8 **Sparse K-SOM**. For each neuron of the map, job trajectories distribution for the observations with a positive coefficients for the corresponding neuron (left) and number of such observations (right).

reported from Table 7.8. The coefficients given to the K-PCA SOM are restricted to the first eight K-PCA axes everywhere, to avoid any bias related to data dimensionality. The computational time is reported in Table 7.9 and gives the time needed to perform the K-PCA only, excluding the training and clustering times.

Results demonstrate a high efficiency in terms of computational time of the Nyström approximation while producing accurate results. In fact, none of the tested values lead to deteriorate the map quality in term of QE, ICI and TE, while the K-PCA is ~ 1000 times faster when using 10% of observations. The best ICI is even obtained using only 1% of the observations. The clustering stability decreases with the number of observations used by the Nyström approximation, even if the stability is still high when using at least 10% of the observations.

The maps with the smallest ICI among the 100 maps generated from a Nyström approximation using 1% and 5% of the observations are displayed in Figure 7.9. Results show the ability of the Nyström approximation to preserve a realistic representation of the dataset while reducing the computational time.

The map obtained with 5% of the observations shows an organization similar to the one presented in Section 7.6.3, except for one fact : precarious situations, located on the right side of the map, are organized differently. Unemployment and public temporary-labor contracts are inverted between

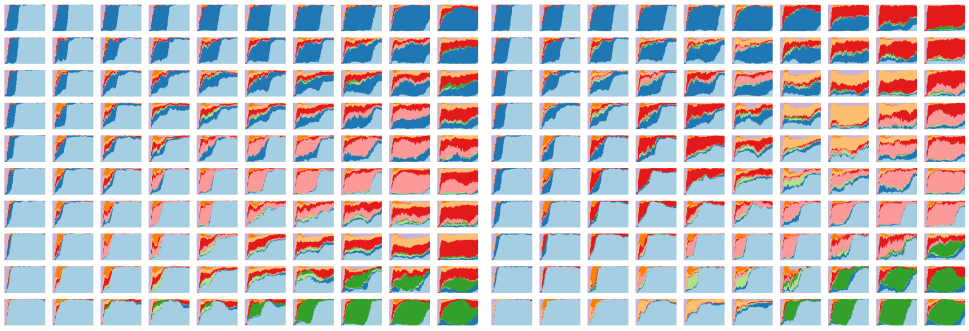

Figure 7.9 K-PCA SOM performed through a Nyström approximation using 1% (left) and 5% (right) of the observations : For each neuron of the map, job trajectories distribution is represented using the observations classified in the corresponding unit. Colors represent the type of contract.

Table 7.9 Performance results of the K-PCA SOM with K-PCA performed through a Nyström approximation (average over 100 maps and standard deviations between parenthesis) for the "trajectories" dataset. After the method name and between parenthesis, the percentage of observations used to perform the approximation is given.

Methods	QE ($\times 100$)	ICI	TE (%)	CPU time	Stability (%)
K-PCA (100%)	**5.61** (0.09)	26.26 (0.37)	**6.98** (0.75)	8 153 (205)	**77.13** (3.24)
K-PCA (25%)	5.62 (0.09)	26.11 (0.40)	7.12 (0.77)	101.38 (18.96)	75.84 (2.38)
K-PCA (10%)	5.62 (0.13)	26.13 (0.40)	7.00 (0.76)	7.39 (1.23)	74.68 (2.25)
K-PCA (5%)	5.64 (0.15)	26.05 (0.45)	7.11 (0.92)	0.86 (0.38)	73.10 (1.72)
K-PCA (1%)	5.65 (0.18)	**25.99** (0.47)	7.02 (1.02)	**0.02** (0.01)	69.32 (1.26)

the top and the bottom of the map. With a Nyström approximation using 1% of the observations, trajectories mostly containing fixed-term contracts are located on the right hand side of the map, what is not observed on the map obtained with 5% of the observations and on maps presented in Section 7.6.3 and in Section 7.6.3. As expected, clusters obtained using 1% of the observations are less homogeneous than those obtained with 5%. The differences between these two maps might have different causes. Firstly, the instability of the SOM algorithm can explain the differences in terms of map organization : different runs of the algorithm give different results. This is particularly critical when the dataset to be analyzed is high dimensional as can be the "Generation 98" survey (even with a subsampling rate of 1%, the dimensionality of the problem is still larger than 100). This issue could be addressed by aggregating strategies, as described in [105]. Secondly, the differences between the two maps in Figure 7.9 might be explained by the high redundancy of trajectories with fixed-term contracts in the dataset : a very small subsampling might enforce the over-representation of these trajectories and affect the result. Such a problem could be addressed by using more efficient sampling techniques such as the ones described in [89].

The choice of the ratio m/n of observations to select in order to obtain accurate results highly depends on the quality of the kernel approximation provided by the Nyström technique. This quality is strongly influenced by the rank of the kernel, which can not easily be obtained when n is very large. Adaptative sampling technique for the Nyström algorithm, such as the one described in [48, 61] are based on an unequal probability sampling, which is performed iteratively and depends on the reconstruction error. [89] proposes an improved version in which the full kernel is not even needed to estimate the reconstruction error. Such methods could be relevant to assess the evolution of the quality reconstruction in a growing sample and to stop the Nyström sampling when this quality is considered good enough.

7.7 Conclusion

The contributions of the present manuscript to the analysis of (dis)similarity data with topographic maps are twofolds : firstly, we have proposed a new version of the kernel and relational SOM algorithms, called sparse K-SOM, which ensures a sparse representation of the prototypes. Secondly, this approach has been compared to a preprocessing of the data by a dimension-reduction technique (K-PCA). We have also investigating the use of a Nyström approximation technique to ensure a better scalability of the method.

The experiments performed on several real datasets showed that both presented methods allow to strongly decrease the overall computational time on large datasets. The interpretability of the results is also improved since the prototypes have a much lower dimensionality. Moreover, the accuracy of the final map, in terms of cluster homogeneity, quality of the organization or adequacy to external *a priori* information is not deteriorated.

In average, K-PCA SOM gives better results in term of map and clustering quality than the sparse approach. However, prototypes returned by sparse K-SOM can be directly interpreted by inspecting the properties of the few observations used to represent them. In K-PCA SOM, the axes of the K-PCA have to be interpreted as an extra step to understand the prototypes meaning : this step is fairly standard in K-PCA. In conclusion, when selecting one or the other method, the user should also consider his/her need to easily interpret the result.

With the introduction of these methods, a further step is taken in allowing relational SOM to deal with massive data sets. Future works should investigate the multiple relational SOM, presented in [123], with a view to integrate several sources of data of different types, while preserving the interpretability of the results.

7.8 Appendix : formula for the average intra-cluster intertia

The intra cluster inertia is the average over $u \in \{1, \ldots, U\}$ of the quantities

$$\mathcal{I}(u) = \frac{1}{\sharp \mathcal{C}_u} \sum_{i:\, x_i \in \mathcal{C}_u} \|\phi(x_i) - G_{\mathcal{C}_u}\|^2$$

in which $G_{\mathcal{C}_u} = \frac{1}{\sharp \mathcal{C}_u} \sum_{i:\, x_i \in \mathcal{C}_u} \phi(x_i)$ is the center of gravity of $\mathcal{C}_u$.

Expansion for kernel data

For kernel data, this quantity equals

$$\begin{aligned}
\mathcal{I}(u) &= \frac{1}{\sharp\mathcal{C}_u}\sum_{i:\, x_i\in\mathcal{C}_u}\left\|\phi(x_i)-\frac{1}{\sharp\mathcal{C}_u}\sum_{i':\, x_{i'}\in\mathcal{C}_u}\phi(x_{i'})\right\|^2 \\
&= \frac{1}{\sharp\mathcal{C}_u}\sum_{i:\, x_i\in\mathcal{C}_u}\left[\|\phi(x_i)\|^2-2\frac{1}{\sharp\mathcal{C}_u}\sum_{i':\, x_{i'}\in\mathcal{C}_u}\langle\phi(x_i),\phi(x_{i'})\rangle+\left\|\frac{1}{\sharp\mathcal{C}_u}\sum_{i':\, x_{i'}\in\mathcal{C}_u}\phi(x_{i'})\right\|^2\right] \\
&= \frac{1}{\sharp\mathcal{C}_u}\sum_{i:\, x_i\in\mathcal{C}_u}K(x_i,x_i)-2\frac{1}{(\sharp\mathcal{C}_u)^2}\sum_{i,i':\, x_i,x_{i'}\in\mathcal{C}_u}K(x_i,x_{i'})+\frac{1}{(\sharp\mathcal{C}_u)^2}\sum_{i,i':\, x_i,x_{i'}\in\mathcal{C}_u}K(x_i,x_{i'}) \\
&= \frac{1}{\sharp\mathcal{C}_u}\sum_{i:\, x_i\in\mathcal{C}_u}K(x_i,x_i)-\frac{1}{(\sharp\mathcal{C}_u)^2}\sum_{i,i':\, x_i,x_{i'}\in\mathcal{C}_u}K(x_i,x_{i'}).
\end{aligned}$$

Expansion for dissimilarity data

Using the result of Equation (2) in [123], we get that

$$\mathcal{I}(u)=\frac{1}{\sharp\mathcal{C}_u}\sum_{i:\, x_i\in\mathcal{C}_u}\left[\boldsymbol{\Delta}_i\nu_u-\frac{1}{2}\nu_u^T\Delta\nu_u\right],$$

in which $\nu_u=\frac{1}{\sharp\mathcal{C}_u}\mathbf{1}_{\mathcal{C}_u}$ where the entries of $\mathbf{1}_{\mathcal{C}_u}$ are equal to 1 for indexes i' such that $x_{i'}\in\mathcal{C}_u$ and to 0 otherwise. Thus

$$\begin{aligned}
\mathcal{I}(u) &= \frac{1}{\sharp\mathcal{C}_u}\sum_{i,i':\, x_i,x_{i'}\in\mathcal{C}_u}\frac{1}{\sharp\mathcal{C}_u}\delta(x_i,x_{i'})-\frac{1}{2(\sharp\mathcal{C}_u)^2}\sum_{j,j':\, x_j,x_{j'}\in\mathcal{C}_u}\delta(x_j,x_{j'}) \\
&= \frac{1}{2\sharp(\mathcal{C}_u)^2}\sum_{i,i':\, x_i,x_{i'}\in\mathcal{C}_u}\delta(x_i,x_{i'}).
\end{aligned}$$

Analyse exploratoire multi-omiques

Chapitre 8

Integrating TARA Oceans datasets using unsupervised multiple kernel learning

Résumé : Dans les analyses de métagénomique, l'intégration de sources d'informations différentes est une tâche difficile dû fait de l'hétérogénéité des jeux de données. Ces jeux de données peuvent être composés de tableaux de comptages, qui nécessitent d'être analysés avec des distances, mais aussi des tableaux d'abondances, des réseaux d'interactions ou encore des informations phylogénétiques qui se sont révélées être pertinent pour améliorer la comparaison entre communautés. Les méthodes d'intégration standard peuvent tirer profit d'information externe mais ne permettent pas d'analyser plusieurs jeux de données omiques hétérogène de façon générique.

Nous proposons trois méthodes multi-noyaux qui permettent d'intégrer plusieurs jeux de données de types différents en une seule analyse exploratoire. Plusieurs solutions sont mises à disposition pour apprendre un méta-noyau consensus ou un méta-noyau qui préserve la topologie des données d'origine. Ce méta-noyau est ensuite analysé à l'aide d'une K-PCA afin de fournir une visualisation rapide et exacte des similarités entre échantillons, dans un espace non linéaire et d'un point de vue de plusieurs sources de données. Une méthode générique est aussi proposée pour améliorer l'interprétabilité de la K-PCA au travers des données d'origine. Nous avons appliqué nos méthodes aux différents jeux de données collectés durant l'expédition *TARA Oceans*. Les simulations mettent en évidence que nos méthodes sont capables de retrouver les résultats précédemment observés à l'aide d'une seule analyse et de fournir une nouvelle image de la structure des échantillons lorsque un plus grand nombre de jeux de données sont inclus dans l'analyse.

L'ensemble des méthodes proposées sont disponibles sur le CRAN dans le package R **mixKernel**. Il est entièrement compatible avec le package **mixOmics** et un tutoriel décrivant l'approche peut être trouvé sur le site internet de **mixOmics** : `http://mixomics.org/mixkernel/`.

Abstract : In metagenomic analysis, the integration of various sources of information is a difficult task since produced datasets are often of heterogeneous types. These datasets can be composed of species counts, which need to be analysed with distances, but also species abundances, interaction networks or phylogenetic information which have been shown relevant to provide a better comparison between communities. Standard integration methods can take advantage of external information but do not allow to analyse heterogenous multi-omics datasets in a generic way.

We propose a multiple kernel framework that allows to integrate multiple datasets of various types into a single exploratory analysis. Several solutions are provided to learn either a consensus meta-kernel or a meta-kernel that preserves the original topology of the datasets. This kernel is subsequently used in kernel PCA to provide a fast and accurate visualisation of similarities between samples, in a non linear space and from the multiple source point of view. A generic procedure is also proposed to improve the interpretability of the kernel PCA in regards with the original data. We applied our framework to the multiple metagenomic datasets collected during the *TARA* Oceans expedition. We demonstrate that our method is able to retrieve previous findings in a single analysis as well as to provide a new image of the sample structures when a larger number of datasets are included in the analysis.

Proposed methods are available in the R package **mixKernel**, released on CRAN. It is fully compatible with the **mixOmics** package and a tutorial describing the approach can be found on **mixOmics** web site `http://mixomics.org/mixkernel/`.

8.1 Introduction

The development of high-throughput sequencing technologies has substantially improved our ability to estimate complex microbial communities composition, even for organisms that cannot be cultured. The sequence reads, produced by amplicon sequencing such as 16S rRNA sequencing, can be taxonomically classified into taxa or clustered into operational taxonomic units (OTUs). Important insights have been gained from the analysis of such data by profiling microbial communities and differences between communities in a wide range of applications from the human enterotypes [8] to the plankton [17]. In microbiome studies, differences among various samples are often extracted to understand associations between organisms and external factors [183, 41]), or to characterize microbial diversity patterns [77, 54].

However, the analysis of metagenomic datasets is complex due to their sparse and compositional structure : OTU counts are often converted to relative, rather than absolute, abundances because the sequencing depth strongly varies between samples. The resulting measures are constrained to a simplex space and the standard Euclidean distance is thus irrelevant to compare samples [3]. As a consequence, directly using standard statistical methods on these data may lead to spurious results [104]. The most widely used approaches to address this issue include transforming the compositional datasets using log-ratio in order to release the simplex constrain [94] or using β-diversity measures to assess the dissimilarity between communities. These dissimilarity measures compute absolute [79] or relative [21] overlaps between two communities. In microbiome studies, they are often used as inputs for an ordination analysis, such as the Principal Coordinates Analysis (PCoA, or Multidimensional Scaling), to identify features that explain differences between studied communities.

However, [130] shows that integrating information about differences among species in the analysis (*i.e.*, by means of phylogenetic dissimilarity) is relevant to reveal phylogenetic patterns in comparing communities. Integrating the philogenetic information is usually performed by using specific dissimilarities, such as the Unifrac and weighted Unifrac measures [100, 101] in ordination methods. Alternatively, [131] propose the DPCoA to analyze the relations between the abundance data and external information corresponding to differences among species (phylogenetic, morphological,

biological...). [47] extend this approach to also integrate external variables measured on communities, using a prior clustering of the communities based on these variables. [141] and [27] show that these methods can be generalized by using a kernel framework and extend them to incorporate context-dependent non-Euclidean structures with abundance data into a regression framework.

In the present work, we use a similar kernel framework to propose a generic approach that can incorporate various types of external information to metagenomic data or that can integrate multiple metagenomic datasets. More precisely, β-diversity measures or phylogenetic-based dissimilarities or any other dissimilarity measuring a specific kind or dissemblance between two samples are viewed as kernels and integrated using an unsupervised multiple kernel approach. Such a kernel can be subsequently used in combination with KPCA [158] for exploratory analysis. To improve the interpretability of our approach, indexes of the importance of the various features of the samples are proposed. The method is evaluated on the *TARA* Oceans expedition datasets [80, 17]. Results show that not only our approach allows to retrieve the main conclusions stated in the different *TARA* Oceans papers in a single and fast analysis, but that, integrating a larger number of information, it can also provide a different overview of the datasets.

8.2 Methods

8.2.1 Unsupervised multiple kernel learning

Kernels and notations : For a given set of observations $(x_i)_{i=1,\dots,N}$, taking values in an arbitrary space $\mathcal{X}$, we call "kernel" a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that provides pairwise similarities between the observations : $K_{ij} := K(x_i, x_j)$. Moreover, this function is assumed to be symmetric ($K_{ij} = K_{ji}$) and positive ($\forall\, n \in \mathbb{N},\ \forall\, (\alpha_i)_{i=1,\dots,n} \subset \mathbb{R},\ \forall\, (x_i)_{i=1,\dots,n} \subset \mathcal{X},\ \sum_{i,i'=1}^{n} \alpha_i \alpha_{i'} K_{ii'} \geq 0$). According to [7], this ensures that K is the dot product in a uniquely defined Hilbert space $(\mathcal{H}, \langle .,. \rangle)$ of the images of $(x_i)_i$ by a uniquely defined feature map $\phi : \mathcal{X} \to \mathcal{H} : K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle$. In the sequel, the notation K will be used to denote either the kernel itself or the evaluation matrix $(K_{ij})_{i,j=1,\dots,N}$ depending on the context.

This setting allows us to deal with multiple source datasets in a uniform way, provided that a relevant kernel can be calculated from each dataset (examples are given in Section 8.3.3 for standard numeric datasets, philogenetic tree, ...). Suppose now that M datasets $(x_i^m)_{i=1,\dots,N}$ (for $m = 1, \dots, M$) are given instead of just one, all obtained on the same samples $i = 1, \dots, N$. M different kernels $(K^m)_{m=1,\dots,M}$ provide different views of the datasets, each related to a specific aspect.

Multiple kernel learning (MKL) refers to the process of linearly combining the M given kernels into a single kernel K^* :

$$K^* = \sum_{m=1}^{M} \beta_m K^m \quad \text{subject to} \quad \begin{cases} \beta_m \geq 0,\ \forall\, m = 1, \dots, M \\ \sum_{m=1}^{M} \beta_m = 1 \end{cases} . \tag{8.1}$$

By definition, the kernel K^* is also symmetric and positive and thus induces a feature space and a feature map (denoted by ϕ^* in the sequel). This kernel can thus be used in subsequent analyses (SVM, KPCA, ...) as it is supposed to provide an integrated summary of the samples.

A simple choice for the coefficients β_m is to set them all equal to $1/M$. However, this choice treats all the kernels similarly and does not take into account the fact that some of the kernels can be redundant or, on the contrary, atypical. Sounder choices aim at solving an optimization problem so as to better integrate all informations. In a supervised framework, this mainly consists in choosing weights that minimize the prediction error [63]. For clustering, a similar strategy is used in [188],

optimizing the margin between the different clusters. However, for other unsupervised analyses (such as exploratory analysis, KPCA for instance), such criteria do not exist and other strategies have to be used to choose relevant weights.

As explained in [190], propositions for unsupervised multiple kernel learning (UMKL) are less numerous than the ones available for the supervised framework. Most solutions (see, *e.g.*, [190, 99]) seek at providing a kernel that minimizes the distortion between all training data and/or that minimizes the approximation of the original data in the kernel embedding. However, this requires that the datasets $(x_i^m)_i$ $(m = 1, \ldots, M)$ are standard numerical datasets : the distortion between data and the approximation of the original data are then directly computed in the input space (which is $\mathbb{R}^d$) using the standard Euclidean distance as a reference. Such a method is not applicable when the input dataset is not numerical (*i.e.*, is a phylogenetic tree for instance) or when the different datasets $(x_i^m)_i$ $(m = 1, \ldots, M)$ do not take value in a common space.

In the sequel, we propose two solutions that overcome this problem : the first one seeks at proposing a consensual kernel, which is the best consensus of all kernels. The second one uses a different point of view and, similarly to what is suggested in [190], computes a kernel that minimizes the distortion between all training data. However, this distortion is obtained directly from the M kernels, and not from an Euclidean input space. Moreover, it is used to provide a kernel representation that preserve the original data topology. Two variants are described : a sparse variant, which also selects the most relevant kernels, and a non sparse variant, when the user does not want to make a selection among the M kernels.

A consensus multiple kernel : Our first proposal, denoted by STATIS-UMKL, relies on ideas similar to STATIS [134, 93]. STATIS is an exploratory method designed to integrate multi-block datasets when the blocks are measured on the same samples. STATIS finds a consensus matrix, which is obtained as the matrix that has the highest average similarity with the relative positions of the observations as provided by the different blocks. We propose to use a similar idea to learn a consensus kernel.

More precisely, a measure of similarity between kernels can be obtained by computing their cosines [1] according to the Frobenius dot product : $\forall\, m,\, m' = 1,\, \ldots,\, M$,

$$C_{mm'} = \frac{\langle K^m, K^{m'} \rangle_F}{\|K^m\|_F \|K^{m'}\|_F} = \frac{\text{Trace}(K^m K^{m'})}{\sqrt{\text{Trace}((K^m)^2)\text{Trace}((K^{m'})^2)}}. \tag{8.2}$$

$C_{mm'}$ can be viewed as an extension of the RV-coefficient [147] to the kernel framework, where the RV-coefficient is computed between $(\phi^m(x_i^m))_i$ and $(\phi^{m'}(x_i^{m'}))_i$ (where ϕ^m is the feature map associated to K^m).

The similarity matrix $\mathbf{C} = (C_{mm'})_{m,m'=1,\ldots,M}$ provides information about the resemblance between the different kernels and can be used as such to understand how they complement each other or if some of them provide an atypical information. It also gives a way to obtain a summary of the different kernels by choosing a kernel K^* which maximizes the average similarity with all the other kernels :

$$\begin{aligned} \text{maximize}_\beta \quad & \sum_{m=1}^{M} \left\langle K_{\mathbf{v}}^*, \frac{K^m}{\|K^m\|_F} \right\rangle_F = \mathbf{v}^\top \mathbf{C} \mathbf{v} \\ & \text{for } K_{\mathbf{v}}^* = \sum_{m=1}^{M} v_m K^m \\ & \text{and } \mathbf{v} \in \mathbb{R}^M \text{ such that } \|\mathbf{v}\|_2 = 1. \end{aligned} \tag{8.3}$$

The solution of the optimization problem of Equation (8.3) is given by the eigen-decomposition of $\mathbf{C}$. More precisely, if $\mathbf{v} = (v_m)_{m=1,\ldots,M}$ is the first eigenvector (with norm 1) of this decomposition, then its entries are all positive (because the matrices K^m are positive) and are the solution of the

1. Cosines are usually preferred over the Frobenius dot product itself because they allow to re-scale the different matrices at a comparable scale. It is equivalent to using the kernel $\widetilde{K}^m = \frac{K^m}{\|K^m\|_F}$ instead of K^m.

maximization of $\mathbf{v}^\top \mathbf{C}\mathbf{v}$. Setting $\beta = \frac{\mathbf{v}}{\sum_{m=1}^{M} v_m}$ thus provides a solution satisfying the constrains of Equation (8.1) and corresponding to a consensual summary of the M kernels.

Note that this method is equivalent to performing multiple CCA between the multiple feature spaces, as suggested in [177] in a supervised framework, or in [142] for multiple kernel PCA. However, only the first axis of the CCA is kept and a L^2-norm constrain is used to allow the solution to be obtained by a simple eigen-decomposition. This solution is better adapted to the case where the number of kernels is small.

A sparse kernel preserving the original topology of the data : Because it focuses on consensual information, the previous proposal tends to give more weights to kernels that are redundant in the ensemble of kernels and to discard the information given by kernels that provide complementary informations. However, it can also be desirable to obtain a solution which weights the different images of the dataset provided by the different kernels more evenly. A second solution is thus proposed, which seeks at preserving the original topology of the data. This method is denoted by sparse-UMKL in the sequel.

More precisely, weights are optimized such that the local geometry of the data in the feature space is the most similar to that of the original data. Since the input datasets are not Euclidean and do not take values in a common input space, the local geometry of the original data cannot be measured directly as in [190]. It is thus approximated using only the information given by the M kernels. To do so, a graph, the k-nearest neighbor graph (for a given $k \in \mathbb{N}^*$), $\mathcal{G}^m$, associated with each kernel K^m is built. Then, a $(N \times N)$-matrix $\mathbf{W}$, representing the original topology of the dataset is defined such that W_{ij} is the number of times the pair (i,j) is in the edge list of $\mathcal{G}^m$ over $m = 1, \ldots, m$ (*i.e.*, the number of times, over $m = 1, \ldots, M$, that x_i^m is one of the k nearest neighbors of x_j^m or x_j^m is one of the k nearest neighbors of x_i^m).

The solution is thus obtained for weights that ensure that $\phi^*(x_i)$ and $\phi^*(x_j)$ are "similar" (in the feature space) when W_{ij} is large. To do so, similarly as [99], we propose to focus on some particular features of $\phi^*(x_i)$ which are relevant to our problem and correspond to their similarity (in the feature space) with all the other $\phi^*(x_j)$. More precisely for a given $\beta \in \mathbb{R}^M$, we introduce the N-dimensional vector $\Delta_i(\beta) = \left\langle \phi^*_\beta(x_i), \begin{pmatrix} \phi^*_\beta(x_1) \\ \vdots \\ \phi^*_\beta(x_N) \end{pmatrix} \right\rangle = \begin{pmatrix} K^*_\beta(x_i, x_1) \\ \vdots \\ K^*_\beta(x_i, x_N) \end{pmatrix}$. But, contrary to [99], we do not rely on a distance in the original space to measure topology preservation but we directly use the information provided by the different kernels through $\mathbf{W}$. The following optimization problem is thus solved :

$$\begin{aligned} \text{minimize}_\beta \quad & \sum_{i,j=1}^{N} W_{ij} \left\| \Delta_i(\beta) - \Delta_j(\beta) \right\|^2 \\ & \text{for } K^*_\beta = \sum_{m=1}^{M} \beta_m K^m \\ & \text{and } \beta \in \mathbb{R}^M \text{ such that } \beta_m \geq 0 \text{ and } \sum_{m=1}^{M} \beta_m = 1. \end{aligned} \tag{8.4}$$

The optimization problem of Equation (8.4) expands as

$$\begin{aligned} \text{minimize}_\beta \quad & \sum_{m,m'=1}^{M} \beta_m \beta_{m'} S_{mm'} \\ & \text{for } \beta \in \mathbb{R}^M \text{ such that } \beta_m \geq 0 \text{ and } \sum_{m=1}^{M} \beta_m = 1, \end{aligned} \tag{8.5}$$

for $S_{mm'} = \sum_{i,j=1}^{N} W_{ij} \langle \Delta_i^m - \Delta_j^m, \Delta_i^{m'} - \Delta_j^{m'} \rangle$ and $\Delta_i^m = \begin{pmatrix} K^m(x_i, x_1) \\ \vdots \\ K^m(x_i, x_N) \end{pmatrix}$. The matrix $\mathbf{S} = (S_{mm'})_{m,m'=1,\dots,M}$ is positive and the problem is thus a standard Quadratic Programming (QP) problem with linear constrains, which can be solved by using the R package **quadprog**. Since the constrain $\sum_{m=1}^{M} \beta_m = 1$ is an L^1 constrain in a QP problem, the produced solution will be sparse : a kernel selection is performed because only some of the obtained $(\beta_m)_m$ are non zero. While desirable when the number of kernels is large, this property can be a drawback when the number of kernels is small and that using all kernels in the integrated exploratory analysis is expected. To address this issue, a modification of Equation (8.5) is proposed in the next section.

A full kernel preserving the original topology of the data : To get rid of the sparse property of the solution of Equation (8.5), an L^2 constrain can be used to replace the L^1 constrain, similarly to Equation (8.3) :

$$\begin{aligned} \text{minimize}_{\mathbf{v}} \quad & \sum_{m,m'=1}^{M} v_m v_{m'} S_{mm'} \\ & \mathbf{v} \in \mathbb{R}^M \text{ such that } v_m \geq 0 \text{ and } \|\mathbf{v}\|_2 = 1, \end{aligned} \tag{8.6}$$

and to finally set $\beta = \frac{\mathbf{v}}{\sum_m v_m}$. This problem is a Quadratically Constrained Quadratic Program (QCQP), which is known to be hard to solve. For a similar problem, [99] propose to relax the problem into a semidefinite programming optimization problem. However, a simpler solution is provided by using ADMM (Alterning Direction Method of Multipliers ; [20]). More precisely, the optimization problem of Equation (8.6) is re-written as

$$\begin{aligned} \text{minimize}_{\mathbf{x} \text{ and } \mathbf{z}} \quad & \mathbf{x}^T \mathbf{S} \mathbf{x} + \mathbb{I}_{\{\mathbf{x} \geq 0\}}(\mathbf{x}) + \mathbb{I}_{\{\mathbf{z} \geq 1\}} \\ & \text{such that } \mathbf{x} - \mathbf{z} = 0 \end{aligned}$$

and is solved with the method of multipliers. Final weights are then obtained by re-scaling the solution $\beta := \frac{\mathbf{z}}{\sum_m z_m}$. The method is denoted by full-UMKL in the sequel.

8.2.2 Kernel PCA (KPCA) and enhanced interpretability

Short description of KPCA : KPCA, introduced in [158], is a PCA analysis performed in the feature space induced by the kernel K^*. It is equivalent to standard MDS (*i.e.*, metric MDS or PCoA ; [167]) for Euclidean dissimilarities. Without loss of generality, the kernel K^* is supposed centered [2]. KPCA simply consists in an eigen-decomposition of K^* : if $(\boldsymbol{\alpha}_k)_{k=1,\dots,N} \in \mathbb{R}^N$ and $(\lambda_k)_{k=1,\dots,N}$ respectively denote the eigenvectors and corresponding eigenvalues (ranked in decreasing order) then the PC axes are, for $k = 1, \dots, N$, $a_k = \sum_{i=1}^{N} \alpha_{ki} \phi^*(x_i)$, where $\boldsymbol{\alpha}_k = (\alpha_{ki})_{i=1,\dots,N}$. $\boldsymbol{a}_k = (a_{ki})_{i=1,\dots,N}$ are orthonormal in the feature space induced by the kernel : $\forall\, k, k'$, $\langle a_k, a_{k'} \rangle = \boldsymbol{\alpha}_k^\top K^* \boldsymbol{\alpha}_{k'} = \delta_{kk'}$ with $\delta_{kk'} = \begin{cases} 0 & \text{if } k \neq k' \\ 1 & \text{otherwise} \end{cases}$. Finally, the coordinates of the projections of the images of the original data, $(\phi^*(x_i))_i$, onto the PC axes are given by : $\langle a_k, \phi^*(x_i) \rangle = \sum_{j=1}^{N} \alpha_{kj} K^*_{ji} = K^*_{i.} \boldsymbol{\alpha}_k = \lambda_k \alpha_{ki}$, where $K^*_{i.}$ is the i-th row of the kernel K^*.

These coordinates are useful to represent the samples in a small dimensional space and to better understand their relations. However, contrary to standard PCA, KPCA does not come with a variable representation, since the samples are described by their relations (via the kernel) and not by standard numeric descriptors. PC axes are defined by their similarity to all samples and are thus hard to interpret.

2. if K^* is not centered, it can be made so by computing $K^* - \frac{1}{N} K^* \mathbf{I}_N + \frac{1}{N^2} \mathbf{I}_N^\top K^* \mathbf{I}_N$, with $\mathbf{I}_N$ a vector with N entries equal to 1.

Interpretation : There is few attempts, in the literature, to help understand the relations of KPCA with the original measures. When the input datasets take values in $\mathbb{R}^d$, [144] propose to add a representation of the variables to the plot, visualizing their influence over the results from derivative computations. However, this approach would make little sense for datasets like ours, *i.e.*, described by discrete counts.

We propose a generic approach that assesses the influence of variables and is based on random permutations. More precisely, for a given measure j, that is used to compute the kernel K^m, the values observed on this measure are randomly permuted between all samples and the kernel is re-computed : $\widetilde{K}^{m,j}$. For species abundance datasets, the permutation can be performed at different phylogeny levels, depending on the user interest. Then, using the weights found with the original (non permuted) kernels, a new meta-kernel is obtained $\widetilde{K}^* = \sum_{l \neq m} \beta_l K^l + \beta_m \widetilde{K}^{m,j}$. The influence of the measure j on a given PC subspace is then assessed by computing the Crone-Crosby distance [42] at the axis level : $\forall\, k = 1, \ldots, N$, $D_{cc}(\alpha_k, \tilde{\alpha}_k) = \frac{1}{\sqrt{2}} \|\alpha_k - \tilde{\alpha}_k\|$, where α_k and $\tilde{\alpha}_k$ respectively denote the eigenvectors of the eigen-decomposition of K^* and $\tilde{K}^*$.[3]

Finally, the KPCA interpretation is done similarly as for a standard PCA : the interpretation of the axes $(a_k)_{k=1,\ldots,N}$ is done with respect to the observations $(x_i)_{i=1,\ldots,N}$ which contribute the most to their definition, when important variables are the ones leading to the largest Crone-Crosby distances.

8.2.3 Unsupervised multiple kernel and KPCA in **mixOmics**

Methods presented in the paper are available on CRAN in the R package **mixKernel** and a full tutorial on the **mixOmics** R package WEB site at http ://mixomics.org/mixkernel/. Kernels can be computed using the function **compute.kernel** that allows to choose between linear, phylogenic and abundance kernels. Unifrac and weighted Unifrac distances are processed using functions taken from the **phyloseq** package [114]. Bray-Curtis dissimilarities are computed with the **vegan** package. The function **combine.kernels** implements methods described in Section 8.3.2 and returns a meta-kernel which can be used as an input for the function **kernel.pca**. The KPCA result can then be displayed using the **mixOmics** plot function **plotInd**.

To assess variable influence in the different datasets, the function **kernel.pca.permute** computes Crone-Crosby distances resulting from permutations. In this function, the user can specify the level at which the permutations must be performed. The most important variables can then be plotted using the **plotVar mixOmics** function. A subset of *TARA* Oceans datasets and a tutorial are also provided in the package to help users processing their own data. In addition, the tutorial is also available on the **mixOmics** web site `http://mixomics.org/mixkernel/` and the method is scheduled to be part of the next version of **mixOmics**.

8.3 Implementation on *TARA* Oceans datasets

8.3.1 Overview on *TARA* Oceans

The *TARA* Oceans expedition [80, 17] facilitated the study of plankton communities by providing oceans metagenomic data combined with environmental measures to the scientific community. During the expedition, 579 samples were collected for morphological, genetic and environmental analyses, from 75 stations in epipelagic and mesopelagic waters across eight oceanic provinces. The *TARA*

3. Note that a similar distance can be computed at the entire projection space level but, since axes are naturally ordered in PCA, we chose to restrict to axis-specific importance measures.

Oceans consortium partners analyzed prokaryotic [165], viral [23] and eukaryotic-enriched [170] size fractions and provided an open access to the raw datasets and processed materials.

Some integrated analyses have already been performed with these datasets : by integrating prokaryotic, eukaryotic and viral datasets, [98] created the global plankton interactome, *i.e.*, a taxon-taxon co-occurrence network. This integrated network, associated to a sparse partial least square analysis, allowed [66] to detect associations between genomic datasets and carbon export. A similar co-occurrence strategy is used in [174] to perform an integrated analysis across domains of life to study the environmental characteristics of the Agulhas rings.

So far, all articles related to *TARA* Oceans that aim at integrating prokaryotic, eukaryotic and viral communities, took advantage of the datasets only by using co-occurrence associations. The integration analysis of the whole material aims at providing a more complete overview of the relations between all collected informations.

8.3.2 Selected samples

Ocean samples used in [165, 170, 23, 151] were collected at various locations, representing all main oceanic regions at different depth layers. Collected samples were located in height different oceans or seas : indian ocean (IO), mediterranean sea (MS), north atlantic ocean (NAO), north pacific ocean (NPO), red sea (RS), south atlantic ocean (SAO), south pacific ocean (SPO) and south ocean (SO).

[165] focused on 139 prokaryotic-enriched samples collected from 68 stations and spread across three depth layers : the surface (SRF), the deep chlorophyll maximum (DCM) layer and the mesopelagic (MES) zones. In [170], 334 size-fractionated samples were analyzed from 47 stations at two water-column depths of the photic-zone : SRF and DCM. The different size-fractions filters used during the sampling allowed to split samples into four major eukaryotic organism sizes : piconanoplankton, nanoplankton, microplankton and mesoplankton. Finally, [23] and [151] analyzed respectively 43 and 89 viral-fractioned samples, collected from 45 stations from the SRF, the DCM and the MES layers.

To evaluate the performances of the proposed methods from different points of view, two analyses were performed. First, the 139 prokaryotic samples were used as inputs of the proof-of-concept analysis presented in Section 8.4.3. Then, a more complete analysis is presented in Section 8.4.4. This analysis was performed on the whole available material but only samples for which all the prokaryotic, eukaryotic and viral information was available. As shown in Figure 8.1, this resulted in 48 common sampling locations which included two depth layers (SRF and MES) and 31 stations.

8.3.3 Dissimilarities and kernels for *TARA* Oceans datasets

Using selected samples, described in Section 8.3.2, 8 (dis)similarities were computed :

— The **phychem** kernel is a similarity measure obtained from environmental variables. To compute this kernel, 22 numerical features were used, including, *e.g.*, temperature, salinity, ... This dataset was extracted from Table W8, available on the companion website of [165] [4]. Missing values were previously imputed using a k-nearest neighbor approach, as implemented in the R package **DMwR** (for $k = 5$). Finally, the linear kernel, $K(x_i, x_j) = x_i^T x_j$, was computed between pairs of ocean samples from this dataset ;

— The **pro.phylo** dissimilarity describes the phylogenetic dissimilarities between ocean samples. The companion website of [165][2] gives access to the abundance table of 35,650 OTUs summarized at different taxonomic levels as well as to the OTUs of 16S ribosomal RNA gene sequences. A phylogenomic tree was built from these data using fasttree [138]. The weighted Unifrac distance was then computed using the R package **phyloseq** [114] : $d_{wUF}(x_i, x_j) = \frac{\sum_e l_e |p_e - q_e|}{\sum_e (p_e + q_e)}$, in

4. `http://ocean-microbiome.embl.de/companion.html`

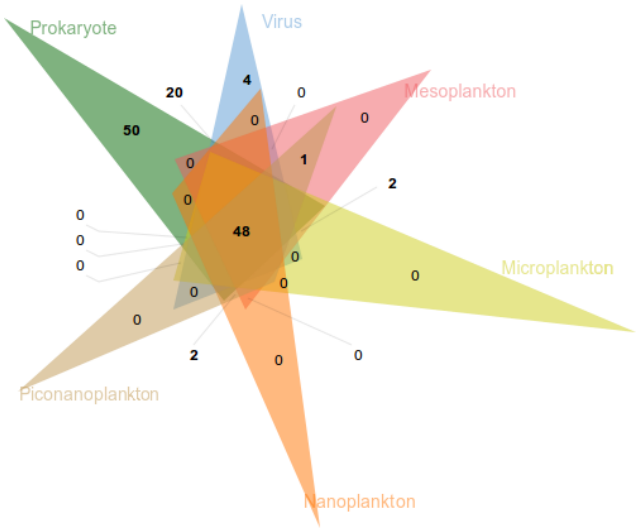

Figure 8.1 Common sampling locations among prokaryotic, eukaryotic and viral samples. Figure was obtained using jvenn [9].

which, for each branch e, l_e is the branch length and p_e (respectively q_e) is the fraction of the community of ocean sample x_j (respectively of ocean sample x_j) below branch e ;

— The **pro.NOGs** dissimilarity provides a measure of prokaryotic functional processes dissimilarities between ocean samples. It was obtained using the Bray-Curtis dissimilarity

$$d_{BC}(x_i, x_j) = \frac{\sum_s |n_{is} - n_{js}|}{\sum_s (n_{is} + n_{js})}, \tag{8.7}$$

computed on the gene abundances of 39,246 bacterial genes. In Equation (8.7), n_{is} is the number of counts of bacterial gene number s in ocean sample x_i. Genes were annotated using the ocean microbial reference gene catalog[2] and summarized at eggNOG gene families (genes annotated by eggNOG version 3 database : [137]). The gene abundance table is freely available from the companion website of [165][2] ;

— The ocean eukaryotic aspect is assessed by four dissimilarities, one for each eukaryotic organism size collected : **euk.pina** for piconanoplankton, **euk.nano** for nanoplankton, **euk.micro** for microplankton and **euk.meso** mesoplankton. The Bray-Curtis dissimilarity, defined in Equation (8.7), is computed on the abundance table of $\sim 150,000$ eukaryotic plankton OTUs. The dataset can be downloaded from the companion website of [170] [5] ;

— The **vir.VCs** dissimilarity measures ocean viral communities and was computed using the Bray-Curtis dissimilarity, defined in Equation (8.7), on the abundance table of 867 Viral Clusters (VCs) available from the supplementary materials of [151].

All dissimilarities, d, described above (**pro.phylo**, **pro.NOGs**, **euk.pina**, **euk.nano**, **euk.micro**, **euk.meso** and **vir.VCs**) were transformed into similarities as suggested in [95] : $K_{ij} = -\frac{1}{2}\left(d(x_i, x_j) - \frac{1}{N}\sum_{k=1}^{N}(d(x_i, x_k) + d(x_k, x_j)) + \frac{1}{N^2}\sum_{k,\,k'=1}^{N} d(x_k, x_{k'})\right)$, where d is the weighted Unifrac distance or the Bray-Curtis dissimilarity. The eight similarities obtained are all positive and are thereby kernels, which are all centred by definition. To avoid scaling effects in kernel integration, all kernels were scaled using the standard cosine transformation [11] : $\tilde{K}_{ij} = \frac{K_{ij}}{\sqrt{K_{ii}K_{jj}}}$.

5. `http://taraoceans.sb-roscoff.fr/EukDiv/`

8.4 Results and discussion

This section is divided into two parts : Section 8.4.3 performs the exploratory analysis only with the datasets studied in [165]. The results described in this paper are used as a ground truth to validate the relevance of our strategy. A further step is taken in Section 8.4.4 in which a larger set of datasets are analyzed to illustrate the use of the method and its efficiency to perform an integrated exploratory analysis. In both sections, analyses are performed with the full-UMKL approach presented in Section 8.2.1. An analysis of the correlation between kernels is provided in Section 8.4.1 and a comparison with the other multiple kernel strategies that explains the choice of full-UMKL is discussed in Section 8.4.2.

8.4.1 Similarities between kernels

To have a general overview on the 8 datasets to integrate, the similarity measure between kernels defined in Equation (2) is computed. The pairwise values are displayed in Figure 8.2.

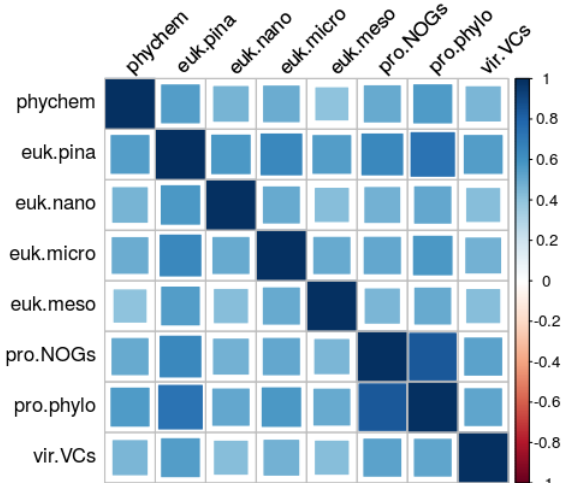

Figure 8.2 Similarities between kernels computed using the STATIS-UMKL approach.

The figure shows that **pro.phylo** and **pro.NOGs** are the most correlated pair of kernels. This result is expected as both kernels provide a summary of prokaryotic communities. Second, the kernel that is the less correlated (in average) with the other ones is **euk.meso** and the kernel that is the most correlated (in average) with the other ones is **euk.pina**. These facts are supported by the conclusions stated in [170] : mesoplanktonic communities are strongly geographically structured, according to their basin of origin, whereas piconanoplankton communities are more homogeneous across the world oceans.

When focusing on similarities to environmental and physical variables, as measured by **phychem**, the figure shows that the kernels that are the most correlated to this kernel are **pro.phylo** and **euk.pina** kernels and that, again, **euk.meso** provides a different image of the oceans. These results are supported by a conclusion made in [165] and [170] : the vertical stratification of the ocean microbiome is mainly driven by temperature rather than geography, but geography plays a strong role to structure communities with respect to the large organism size fractions.

Finally, **vir.VCs** is also more similar to small size organisms kernels than kernels representing larger ones. This is explained by the fact that the biographical structure of viruses is due to host community structure and to a passive transport by oceanic currents [23].

These results confirm the discussion reported in Section 8.4.2 : STATIS-UMKL allows to have an overview on the different datasets and should be used when the integrated analysis focuses on correlated informations.

8.4.2 Comparison of the different integration options

In the following section, the different methods proposed and especially the relevance of using a specific approach to perform the integration is evaluated. To perform this analysis, environmental, prokaryotic, eukaryotic and viral datasets are integrated together using the three proposed approaches : full-UMKL, sparse-UMKL and STATIS-UMKL. The weights obtained for each methods are presented in Figure 8.3.

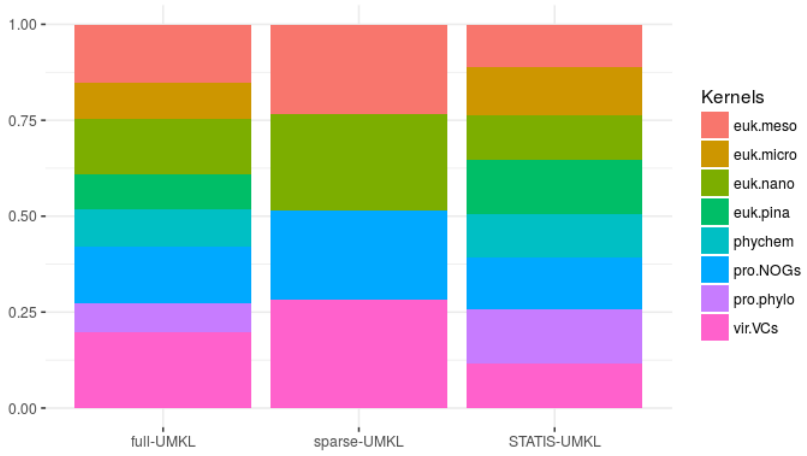

Figure 8.3 Kernels weights obtained for the three proposed approaches : full-UMKL, sparse-UMKL and STATIS-UMKL. Colors represent the different kernels.

First, note that, Figure 8.3 shows that STATIS-UMKL gives more weights to **euk.micro**, **euk.pina**, **pro.NOGs** and **pro.phylo**, meaning that these kernels are strongly correlated. In the contrary, full-UMKL gives more importance to atypical kernels, *i.e.*, **euk.meso**, **euk.micro**, **pro.NOGs** and **vir.VCs**, which are the only kernels selected by the sparse-UMKL approach, the other ones being discarded from the final meta-kernel.

Results show that the three proposed methods are complementary and can be used depending on the research question and the analysis step. The STATIS-UMKL approach allows to have an overview on the correlation between the different datasets to analyze and to integrate them in a consensual way. sparse-UMKL can be used to focus on a more even contribution of the various images provided by the different kernels and to remove redundant informations. Finally, a similar goal is achieved with the full-UMKL method, that should be preferred when the analysis requires to be performed on the whole material.

8.4.3 Proof of concept with a restricted number of datasets

In the present section, only the datasets analyzed in [165] are analyzed. These kernels are the environmental kernel, **phychem**, and the two prokaryotic kernels, **pro.phylo** and **pro.NOGs**, all computed on the 139 prokaryotic samples described in Section 8.3. Figure 8.4 (left) provides the sample projection of the first two axes of the KPCA (full-UMKL kernel). The 10 most important variables for each dataset are displayed in Figure 8.4 (first axis) and in Figure 8.5 (second axis). Both figures were obtained by randomly permuting the 22 environmental variables, the eggNOG gene families

at 23 functional levels of the gene ontology and the *proteobacteria* abundances at 102 order levels. Additionally, the explained variance supported by the first 15 axes is provided in Figure 8.6.

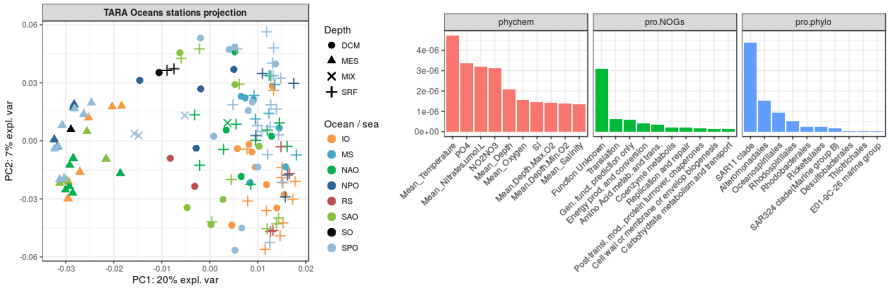

Figure 8.4 **Only datasets of [165].** Left : Projection of the observations on the first two KPCA axes. Colors represent the oceanic regions and shapes the depth layers. Right : The 10 most important variables for the first KPCA axis, ranked by decreasing Crone-Crosby distance.

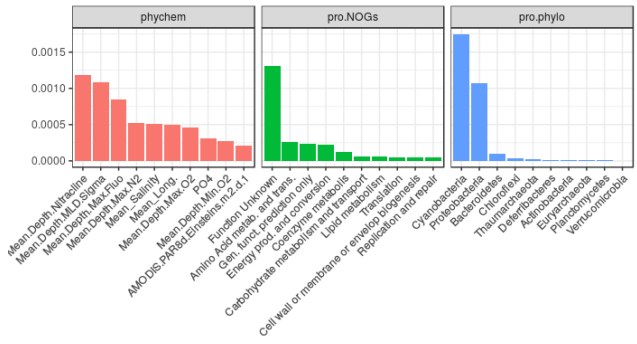

Figure 8.5 **Only datasets of [165].** The 10 most important variables for the second KPCA axis, ranked by decreasing Crone-Crosby distance. Variables of the **pro.phylo** kernel were permuted at the phylum level.

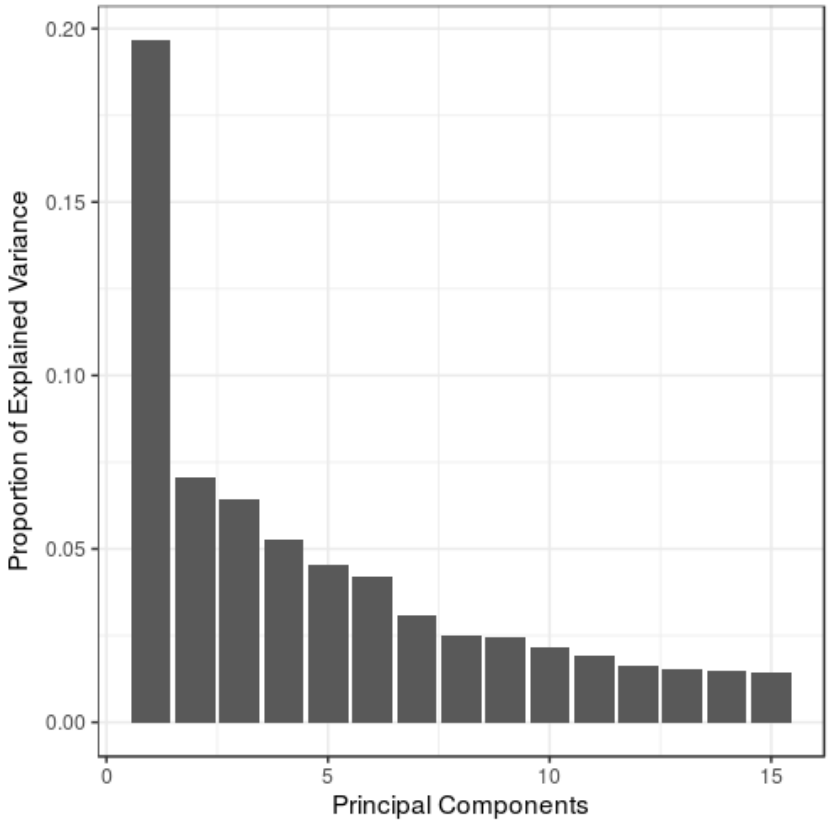

Figure 8.6 **Only datasets of [165]**. Entropy preserved by the 15 first axes of the KPCA performed on the meta-kernel obtained using the full-UMKL approach.

First, note that Figure 8.4 shows very similar results to the ones returned by the PCA performed on community composition dissimilarities (Bray-Curtis) presented in [165] : samples are separated by their depth layer of origin, *i.e.*, SRF, DCM or MES, with stronger differences for MES samples.

Figure 8.4 exhibits that both the abundance of *clade SAR11* and the temperature lead to the largest Crone-Crosby distances, meaning that they contribute the most to the first KPCA axis definition. This result is validated by displaying the values of this variable on the KPCA projection (see Figure 8.7 and Figure 8.8). On both figures, a gradient can be observed on the first KPCA axis between the left (lowest abundances of *clade SAR11* and lowest temperatures), and the right (highest values of these variables). Those results are similar to the ones presented in [165] : the vertical stratification of prokaryotic communities is mostly driven by temperature and *proteobacteria* (more specifically *clade SAR11* and *clade SAR86*) dominate the sampled areas.

Similarly, Figure 8.5 shows that *cyanobacteria* abundance and the nitracline mean depth (*i.e.* water layer in which the nitrate concentration changes rapidly with depth) contribute the most to the second KPCA axis definition. The display of the nitracline mean depth on KPCA projection (Figure 8.9) shows a gradient on the second KPCA axis. Figure 8.10, displaying *cyanobacteria* abundance, shows a gradient between the top-left and the bottom-right of the KPCA projection, because *cyanobacteria* abundance also ranks as the third important variable on the first axis (see Figure 8.11). Those results are consistent with findings of [165] : *cyanobacteria* were found abundant and the nitracline strongly correlated to the taxonomic composition (p-value i 0.001). On both first two axes of the KPCA, unknown functions lead to the largest Crone-Crosby distances between variables used to compute the **pro.NOGs** kernel. Again, this result is in agreement with a conclusion made in [165] : a large fraction of the ocean gene families encode for unknown functions.

These results demonstrate that the proposed method gives a fast and accurate insight to the main variability explaining the differences between the different samples, viewed through different omics

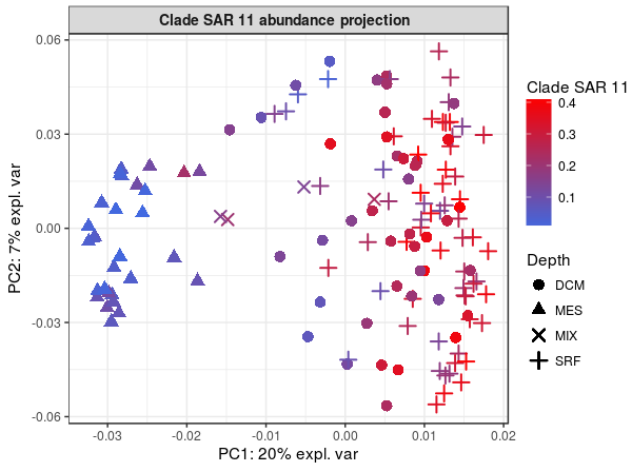

Figure 8.7 **Only datasets of [165]**. Projection of the observations on the first two KPCA axes. Colors represent the relative abundance of *clade SAR11* : blue for low values and red for high values.

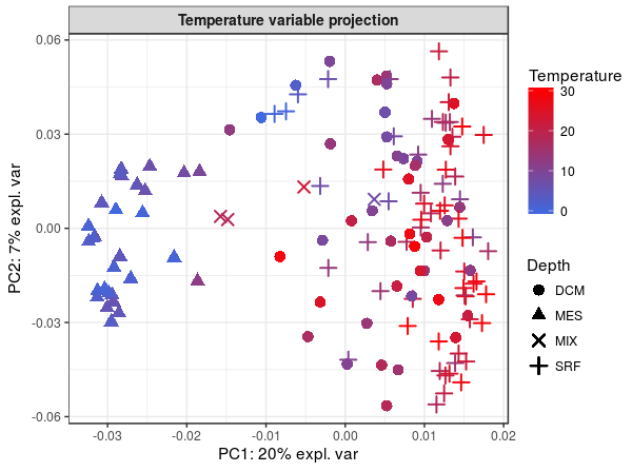

Figure 8.8 **Only datasets of [165]**. Projection of the observations on the first two KPCA axes. Colors represent the temperature : blue for cold waters and red for warm waters.

datasets. In particular, for both **pro.phylo** and **phychem** kernels, the most important variables are those used in [165] to state the main conclusions.

8.4.4 Integrating environmental, prokaryotic, eukaryotic and viral datasets

In this section, environmental, prokaryotic, eukaryotic and viral datasets are integrated together into a meta-kernel obtained using the full-UMKL method. Figure 8.18 (left) displays the projection of the samples on the first two axes of the KPCA. Figure 8.18 (right) and Figure 8.12 provide the 5 most important variables for each datasets, respectively for the first and the second axes of the KPCA. To obtain these figures, abundance values were permuted at 56 prokaryotic phylum levels for the **pro.phylo** kernel, at 13 eukaryotic phylum levels for **euk.pina**, **euk.nano**, **euk.micro** and **euk.meso** and at 36 virus family levels for the **vir.VCs** kernel. Variables used for **phychem** and **pro.NOGs** were the same than in Section 8.4.3. Additionally, the explained variance supported by the first 15 axes is provided in Figure 8.13.

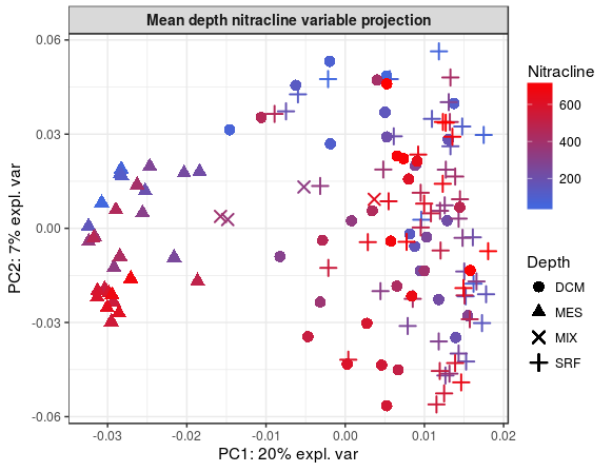

Figure 8.9 **Only datasets of [165]**. Projection of the observations on the first two KPCA axes. Colors represent the nitracline mean depth : blue for low values and red for high values.

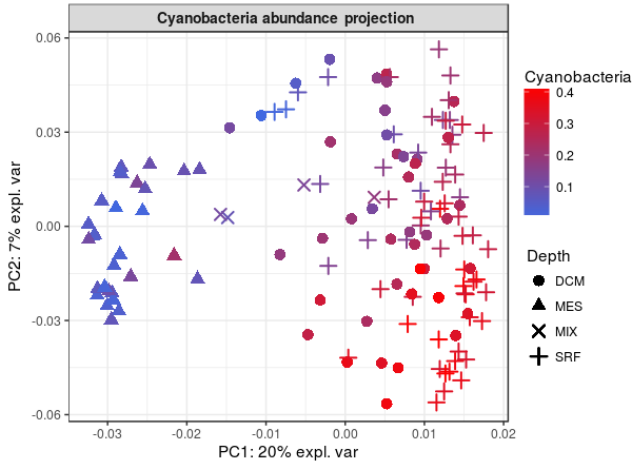

Figure 8.10 **Only datasets of [165]**. Projection of the observations on the first two KPCA axes. Colors represent the relative abundance of *cyanobacteria* : blue for low values and red for high values.

First, note that Figure 8.18 does not highlight anymore any particular pattern in terms of depth layers but it does in terms of geography. SO samples are gathered in the bottom-center of the KPCA projection and SPO samples are gathered on the top-left side. Second, Figure 8.18 shows that the most important variables come from the **phychem** kernel (especially the longitude) and from kernels representing the eukaryotic plankton. More specifically, large size organisms are the most important : *rhizaria* phylum for **euk.meso** and *alveolata* phylum for **euk.nano**. The abundance of *rhizaria* organisms also ranks first between important variables of the second KPCA axis, followed by the *opisthokonta* phylum for **euk.nano**. The display of these variables on the KPCA projection reveals a gradient on the first axis for both the *alveolata* phylum abundance (Figure 8.14) and the longitude (Figure 8.15) and on the second axis for *rhizaria* (Figure 8.16) and *opisthokonta* (Figure 8.17) abundances. This indicates that SO and SPO epipelagic waters mainly differ in terms of *Rhizarians* abundances and both of them differ from the other studied waters in terms of *alveolata* abundances.

The integration of *TARA* Oceans datasets shows that the variability between epipelagic samples is mostly driven by geography rather than environmental factors and that this result is mainly explained by the strong geographical structure of large eukaryotic communities. Studied samples were all

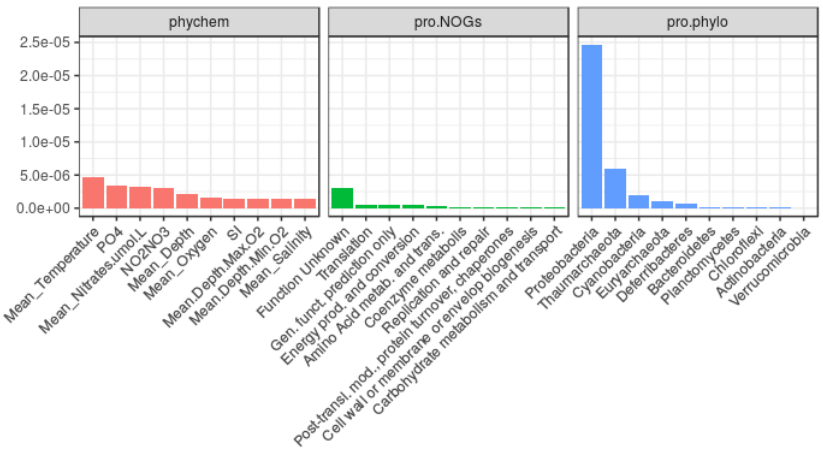

Figure 8.11 **Only datasets of [165].** The 10 most important variables for the second axis of KPCA, ranked by decreasing Crone-Crosby distance. Variables of the **pro.phylo** kernel were permuted at the phylum level.

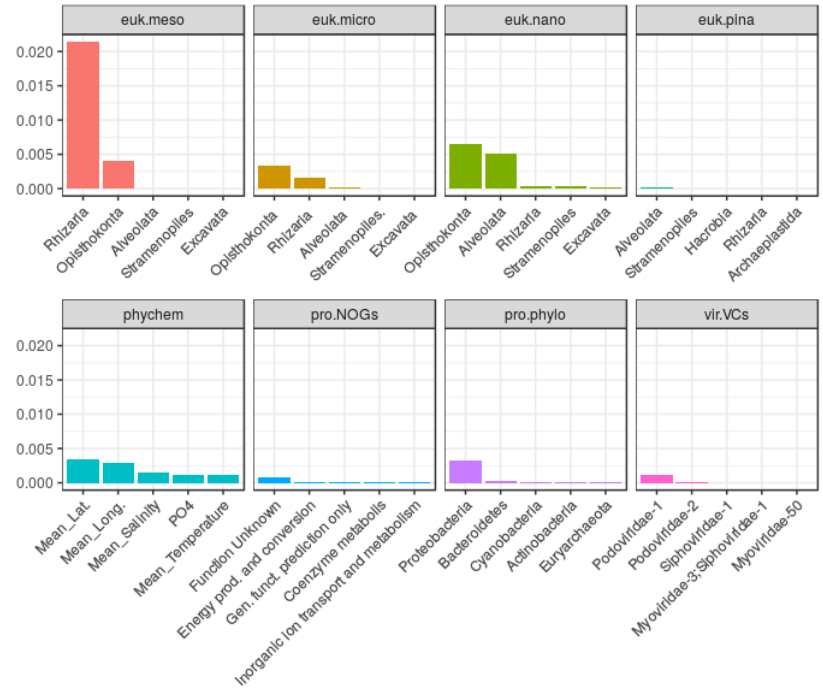

Figure 8.12 The 5 most important variables for the second axis of the KPCA and for each of the 8 datasets, ranked by decreasing Crone-Crosby distance.

collected from epipelagic layers, where water temperature does not vary much, which explains the poor influence of the prokaryotic dataset in this analysis.

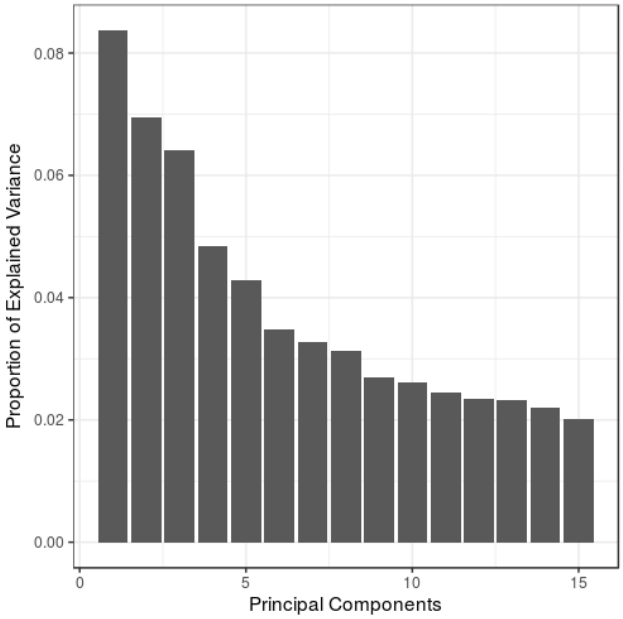

Figure 8.13 Entropy preserved by the 15 first axes of the KPCA performed on the meta-kernel obtained using the full-UMKL approach and environmental, prokaryotic, eukaryotic and viral datasets.

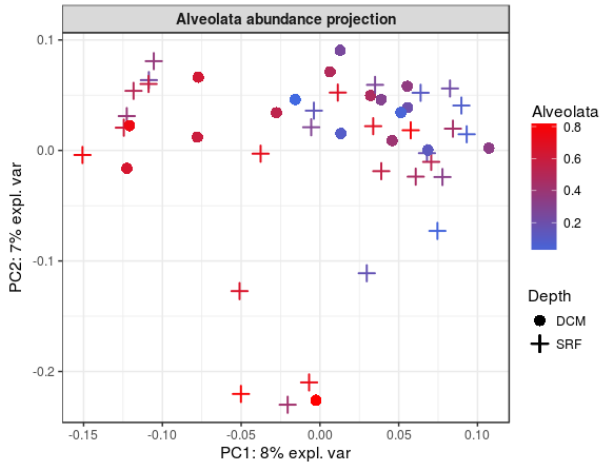

Figure 8.14 Projection of the observations on the first two KPCA axes. Colors represent the relative abundance of *alveolata* organisms in the nanoplanktonic community : blue for low values and red for high values.

8.5 Conclusion

The contributions of the present manuscript to the analysis of multi-omics datasets are twofolds : firstly, we have proposed three unsupervised kernel learning approaches to integrate multiple datasets from different types, which either allow to learn a consensus meta-kernel or a meta-kernel preserving the original topology of the data. Secondly, we have improved the interpretability of the KPCA by assessing the influence of input variables in a generic way.

The experiments performed on *TARA* Oceans datasets showed that presented methods allow to give a fast and accurate insight over the different datasets within a single analysis. However, the approach is not restricted to KPCA analyses : the meta-kernel presented in this article could have been

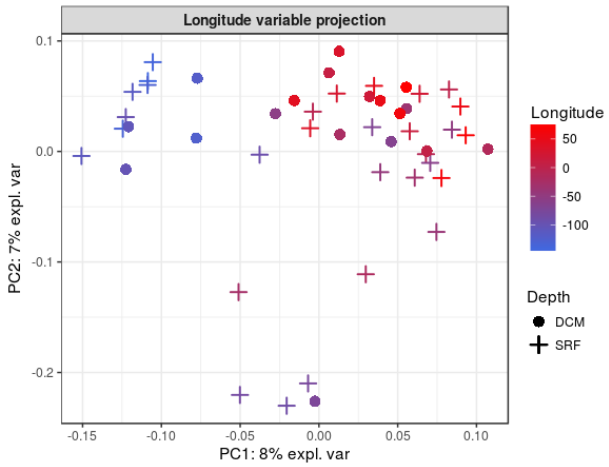

Figure 8.15 Projection of the observations on the first two KPCA axes. Colors represent the longitude : blue for low values and red for high values.

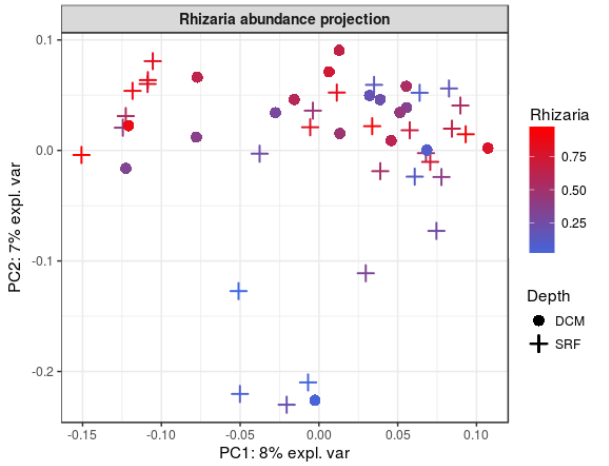

Figure 8.16 Projection of the observations on the first two KPCA axes. Colors represent the relative abundance of *rhizaria* organisms in the mesoplanktonic community : blue for low values and red for high values.

used in combination with kernel clustering methods or with kernel supervised models, to integrate multi-omics datasets.

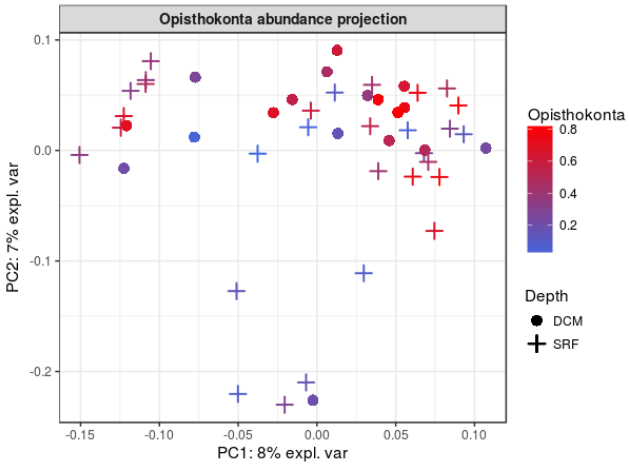

Figure 8.17 Projection of the observations on the first two KPCA axes. Colors represent the relative abundance of *opisthokonta* organisms in the nanoplanktonic community : blue for low values and red for high values.

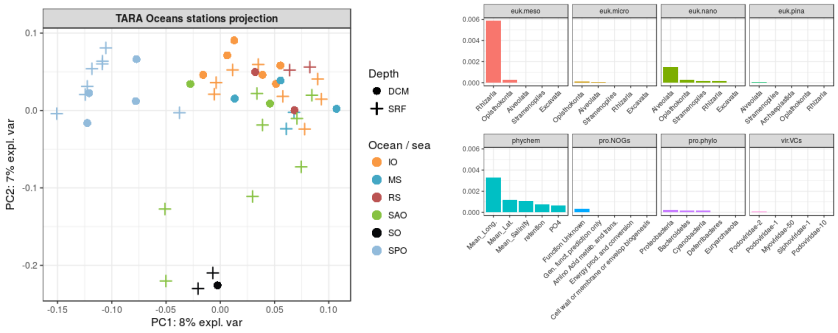

Figure 8.18 Left : Projection of the observations on the first two KPCA axes. Colors represent the oceanic regions and shapes the depth layers. Right : The 5 most important variables for each of the eight datasets, ranked by decreasing Crone-Crosby distance.

Chapitre 9
Conclusion et perspectives

Dans le cadre de cette thèse, plusieurs contributions méthodologiques sont proposées pour permettre l'exploration simultanée de plusieurs jeux de données omiques de natures hétérogènes. Pour aborder cette question, je me suis intéressé aux noyaux sous deux aspects différents : l'apprentissage multi-noyaux pour combiner plusieurs jeux de données et l'adaptation des méthodes à noyaux afin de leur permettre de traiter de grands volumes de données.

Dans un premier temps, j'ai proposé une méthode permettant de stabiliser la version stochastique du SOM numérique, en agrégeant un ensemble de résultats tout en préservant la topologie des cartes. Cette approche favorise l'utilisation de l'algorithme pour le traitement de données biologiques, qui nécessite très souvent des résultats reproductibles. Pour permettre le passage à l'échelle du K-SOM au domaine des omiques, j'ai proposé trois variantes de cet algorithme, qui présentent des intérêts différents suivant l'objectif recherché par l'utilisateur. Une première méthode accélère le K-SOM, sans aucune approximation, à l'aide d'un jeu de réécriture de l'équation de l'étape d'affectation. Bien que cette approche n'aborde pas la problématique d'interprétabilité du modèle, elle s'avère très utile lorsque la classification et la projection des données sont les seuls objectifs de l'utilisateur. Dans le cas où des besoins d'interprétation sont nécessaires, deux autres variantes sont proposées. La première exploite l'approche de *bagging* pour permettre sa parallélisation et impose une représentation parcimonieuse des prototypes. Cette représentation permet une interprétation direct du modèle en inspectant les propriétés des observations utiles à la définition des prototypes. La seconde contribution réduit la dimension des données d'entrée en les pré-traitant par K-PCA. Dans cette version, comprendre les prototypes demande d'interpréter au préalable les axes de la K-PCA.

Bien que ces deux méthodes facilitent l'interprétation des prototypes, elles restent toutefois difficiles à interpréter en raison de la phase de représentation dans l'espace image. Dans le cas où les données d'origine sont vectorielles, redonner l'accès aux variables reste le meilleur moyen pour permettre à l'utilisateur de comprendre le modèle généré. Pour cela, j'ai proposé une approche générique permettant d'étudier les variables influentes de la K-PCA, dans un contexte multi-omiques.

Les troix méthodes d'apprentissage multi-noyaux proposées permettent d'intégrer de façon générique plusieurs jeux de données de nature hétérogènes. La première propose un noyau consensus qui maximise la similarité moyenne entre tous les noyaux, les deux autres préservent la topologie des données d'origine et fournissent une solution parcimonieuse ou non. En pratique, les utilisateurs sont souvent confrontés au problème de données manquantes, ce qui peut conduire à l'exclusion de nombreux échantillons pour lesquels une partie de l'information n'est pas disponible. À titre d'exemple, afin de pouvoir intégrer la totalité des jeux de données rendus disponibles par les partenaires du consortium *TARA oceans*, j'ai exclu un grand nombre de stations pour lesquelles les données de comptage d'eucaryotes ou de virus étaient manquantes. L'analyse réalisée se focalise sur l'étude des échantillons collectés en surface des océans, laissant inexploitée des données disponibles pour les autres couches océaniques. Ce problème requiert l'adaptation de la méthode actuelle pour tenir compte des échantillons manquants, soit par imputation de ceux-ci, soit par des méthodes permettant la prise en compte directe de l'information « manquant/présent ».

Choisir un noyau adapté à ses données et le paramétrer est évidemment le point critique et qui requiert à la fois une connaissance experte et, dans certains cas, une phase de calibration. Dans ce contexte, le développement de noyaux adaptés aux divers omiques, mais aussi d'outils permettant de sélectionner le noyau le plus adapté aux données d'entrée est une question qui pourrait prolonger mon travail récent.

Plus généralement, bien que j'aie concentré mon attention sur l'exploration de structures microbiennes, les méthodes proposées peuvent être utilisées pour intégrer d'autres types de données, et trouve des applications dans d'autres types de problèmes en biologie. Par exemple, l'annotation fonctionnelle de gènes par intégration de plusieurs types de données transcriptomiques et de données d'ontologie ou encore l'étude de méthodes d'exploration de la typologie des ARNs non codants. Là encore, c'est un sujet que je compte aborder dans un avenir proche.

Bibliographie

[1]A. Abbott et J. Forrest. “Optimal matching methods for historical sequences”. In : *Journal of Interdisciplinary History* 16 (1986), p. 471–494 (cf. p. 70, 88).

[2]L.A. Adamic et N. Glance. “The political blogosphere and the 2004 US election : divided they blog”. In : *Proceedings of the 3rd LINKDD Workshop*. New York, NY, USA : ACM Press, 2005, p. 36–43 (cf. p. 58, 83).

[3]J. Aitchison. “The statistical analysis of compositional data”. In : *Journal of the Royal Statistical Society, Series B* 44.2 (1982), p. 139–177 (cf. p. 21, 100).

[4]Uri David Akavia, Oren Litvin, Jessica Kim et al. “A new look at the statistical model identification”. In : *Cell* 143.6 (2010), p. 1005–17 (cf. p. 36).

[5]C. Ambroise et G. Govaert. “Analyzing dissimilarity matrices via Kohonen maps”. In : *Proceedings of 5th Conference of the International Federation of Classification Societies (IFCS 1996)*. T. 2. Kobe (Japan), 1996, p. 96–99 (cf. p. 34).

[6]P. Andras. “Kernel-Kohonen networks”. In : *International Journal of Neural Systems* 12 (2002), p. 117–135 (cf. p. 34).

[7]N. Aronszajn. “Theory of reproducing kernels”. In : *Transactions of the American Mathematical Society* 68.3 (1950), p. 337–404 (cf. p. 28, 30, 56, 62, 70, 73, 101).

[8]Manimozhiyan Arumugam, Jeroen Raes, Eric Pelletier et al. “Enterotypes of the human gut microbiome”. In : *Nature* 473 (2011), p. 174–180 (cf. p. 100).

[9]Philippe Bardou, Jérôme Mariette, Frédéric Escudié, Christophe Djemiel et Christophe Klopp. “jvenn : an interactive Venn diagram viewer”. In : *BMC bioinformatics* 15.1 (2014), p. 293 (cf. p. 107).

[10]B. Baruque et E. Corchado. *Fusion methods for unsupervised learning ensembles*. T. 322. Studies in Computational Intelligence. Springer, 2011 (cf. p. 48, 49, 52, 64).

[11]A. Ben-Hur et J. Weston. *Data Mining Techniques for the Life Sciences*. T. 609. Methods in Molecular Biology. Springer-Verlag, 2010, 223–239 (cf. p. 38, 107).

[12]Wolfgang H. Berger et Frances L. Parker. “Diversity of Planktonic Foraminifera in Deep-Sea Sediments”. In : *Science* 3937.168 (1970), 1345–1347 (cf. p. 22).

[13]Matteo Bersanelli, Ettore Mosca, Daniel Remondini et al. “Methods for the integration of multi-omics data : mathematical aspects”. In : *BMC Bioinformatics* 15.17 (2016) (cf. p. 36).

[14]P. Bickel, F. Götze et W. van Zwet. “Resampling fewer than n observations : gains, losses and remedies for losses.” In : *Statistica Sinica* 1.7 (1997), p. 1–31 (cf. p. 35).

[15]E. de Bodt, M. Cottrell et M. Verleisen. “Statistical tools to assess the reliability of self-organizing maps”. In : *Neural Networks* 15.8-9 (2002), p. 967–978 (cf. p. 52).

[16]J. Boelaert, L. Bendhaïba, M. Olteanu et N. Villa-Vialaneix. “SOMbrero : an R package for numeric and non-numeric self-organizing maps”. In : *Advances in Self-Organizing Maps and Learning Vector Quantization (Proceedings of WSOM 2014)*. (2–4 juil. 2014). Sous la dir. de T. Villmann, F.M. Schleif, M. Kaden et M. Lange. T. 295. Advances in Intelligent Systems and Computing. Mittweida, Germany : Springer Verlag, Berlin, Heidelberg, 2014, p. 219–228 (cf. p. 83).

[17]P. Bork, C. Bowler, C. de Vargas et al. “Tara Oceans studies plankton at planetary scale”. In : *Science* 348.6237 (2015), p. 873–873. eprint : `http://science.sciencemag.org/content/348/6237/873.full.pdf` (cf. p. 19, 100, 101, 105).

[18]R. BOULET, B. JOUVE, F. ROSSI et N. VILLA. “Batch kernel SOM and related Laplacian methods for social network analysis”. In : *Neurocomputing* 71.7-9 (2008), p. 1257–1273 (cf. p. 62, 63, 66, 70).

[19]C. BOUVEYRON et C. BRUNET-SAUMARD. “Model-based clustering of high-dimensional data : a review”. In : *Computational Statistics & Data Analysis* 71 (2014), p. 52–78 (cf. p. 71).

[20]S. BOYD, N. PARIKH, E. CHU, B. PELEATO et J. ECKSTEIN. “Distributed optimization and statistical learning via the alterning direction method of multipliers”. In : *Foundations and Trends in Machine Learning* 3.1 (2011), p. 1–122 (cf. p. 104).

[21]R.J. BRAY et J.T. CURTIS. “An Ordination of the Upland Forest Communities of Southern Wisconsin”. In : *Ecological Monographs* 27.4 (1957), p. 325–349 (cf. p. 23, 100).

[22]L. BREIMAN. “Bagging predictors”. In : *Machine Learning* 24 (1996), p. 123–140 (cf. p. 35).

[23]J.R. BRUM, J.C. IGNACIO-ESPINOZA, S. ROUX et al. “Patterns and ecological drivers of ocean viral communities”. In : *Science* 348.6237 (2015). eprint : `http://science.sciencemag.org/content/348/6237/1261498.full.pdf` (cf. p. 106, 108).

[24]M. BĂDOIU, S. HAR-PELED et P. INDYK. “Approximate clustering via core-sets”. In : *Proceedings of the 34th annual ACM Symposium on Theory of Computing*. Sous la dir. de J. REIF. 250-257. Montreal, QC, Canada : ACM New York, NY, USA, 2002 (cf. p. 71).

[25]J.G. CAPORASO, J. KUCZYNSKI, J. STOMBAUGH et al. “QIIME allows analysis of high-throughput community sequencing data.” In : *Nature Methods* 7 (2010), 335–336 (cf. p. 20).

[26]A. CHAO et T.J. SHEN. “Nonparametric estimation of Shannon’s index of diversity when there are unseen species in sample”. In : *Environmental and Ecological Statistics* 4.10 (2003), 429–443 (cf. p. 22).

[27]J. CHEN et H. LI. “Kernel methods for regression analysis of microbiome compositional data”. In : *Topics in Applied Statistics*. Sous la dir. de L. HU, Y. LIU et J. LIN. T. 55. Springer Proceedings in Mathematics & Statistics (PROMS). New York, NY, USA : Springer, 2013, p. 191–201 (cf. p. 101).

[28]J. CHEN, K. BITTINGER, E. CHARLSON et al. “Associating microbiome composition with environmental covariates using generalized UniFrac distances”. In : *Bioinformatics* 28.16 (2012), 2106–2113 (cf. p. 23).

[29]X. CHEN et M.G. XIE. “A split-and-conquer approach for analysis of extraordinarily large data”. In : *Statistica Sinica* 24 (2014), p. 1655–1684 (cf. p. 71).

[30]Y. CHEN, E.K. GARCIA, M.R. GUPTA, A. RAHIMI et L. CAZZANTI. “Similarity-based classification : concepts and algorithm”. In : *Journal of Machine Learning Research* 10 (2009), p. 747–776 (cf. p. 29, 81).

[31]C.T. CHU, S.K. KIM, Y.A. LIN et al. “Map-Reduce for machine learning on multicore”. In : *Advances in Neural Information Processing Systems (NIPS 2010)*. (6–11 déc. 2010). Sous la dir. de J.D. LAFFERTY, C.K.I. WILLIAMS, J. SHAWE-TAYLOR, R.S. ZEMEL et A. CULOTTA. T. 23. Hyatt Regency, Vancouver, Canada, 2010, p. 281–288 (cf. p. 71).

[32]J.R. COLE, Q. WANG, J.A. FISH et al. “Ribosomal Database Project : data and tools for high throughput rRNA analysis.” In : *Nucleic Acids Research* 42 (2014), p. 633–642 (cf. p. 20).

[33]E. CÔME, M. COTTRELL et P. GAUBERT. “Analysis of professional trajectories using disconnected self-organizing maps”. In : *Neurocomputing* 147 (2015). Advances in Self-Organizing Maps Subtitle of the special issue : Selected Papers from the Workshop on Self-Organizing Maps 2012 (WSOM 2012), p. 185–196 (cf. p. 88).

[34]B. CONAN-GUEZ, F. ROSSI et A. EL GOLLI. “Fast algorithm and implementation of dissimilarity self-organizing maps”. In : *Neural Networks* 19.6-7 (2006), p. 855–863 (cf. p. 56).

[35]B. CONAN-GUEZ, F. ROSSI et A. EL GOLLI. “Fast algorithm and implementation of dissimilarity self-organizing maps”. In : *Neural Networks* 19.6-7 (2006), p. 855–863 (cf. p. 70).

[36]The Human Microbiome Project CONSORTIUM. “A framework for human microbiome research”. In : *Nature* 286 (2012), 215–221 (cf. p. 19).

[37]P. CORTEZ, A. CERDEIRA, F. ALMEIDA, T. MATOS et J. REIS. “Modeling wine preferences by data mining from physicochemical properties”. In : *Decision Support Systems* 47.4 (2009), p. 547–553 (cf. p. 58, 83).

[38]M. COTTRELL et P. LETRÉMY. “How to use the Kohonen algorithm to simultaneously analyse individuals in a survey”. In : *Neurocomputing* 63 (2005), p. 193–207 (cf. p. 70, 88).

[39]M. COTTRELL, E. de BODT et M. VERLEISEN. “A statistical tool to assess the reliability of self-organizing maps”. In : *Advances in Self-Organizing Maps (Proceedings of WSOM 2001)*. (13–15 juin 2001). Sous la dir. de N. ALLINSON, H. YIN, J. ALLINSON et J. SLACK. Lincoln, UK : Springer Verlag, 2001, p. 7–14 (cf. p. 52).

[40]M. COTTRELL, J.C. FORT et G. PAGÈS. “Theoretical Aspects of the SOM Algorithm”. In : *Neurocomputing* 21 (1998), p. 119–138 (cf. p. 48).

[41]Diana L. COX-FOSTER, Sean CONLAN, Edward C. HOLMES et al. “A Metagenomic Survey of Microbes in Honey Bee Colony Collapse Disorder”. In : *Science* 318.5848 (2007), p. 283–287 (cf. p. 100).

[42]L. J. CRONE et D. S. CROSBY. “Statistical Applications of a Metric on Subspaces to Satellite Meteorology”. In : *Technometrics* 37.3 (1995), p. 324–328. eprint : `http://amstat.tandfonline.com/doi/pdf/10.1080/00401706.1995.10484338` (cf. p. 43, 105).

[43]L. DANON, A. DIAZ-GUILERA, J. DUCH et A. ARENAS. “Comparing community structure identification”. In : *Journal of Statistical Mechanics* (2005), P09008 (cf. p. 51, 66, 82).

[44]M. DENIL, D. MATHESON et N. de FREITAS. “Consistency of online random forests”. In : *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*. 2013, p. 1256–1264 (cf. p. 71).

[45]T.Z. DESANTIS, P. HUGENHOLTZ, N. LARSEN et al. “Greengenes, a Chimera-Checked 16S rRNA Gene Database and Workbench Compatible with ARB”. In : *Applied Environmental Microbiology* 72 (2006), p. 5069–5072 (cf. p. 20).

[46]M.A. DILLIES, A. RAU, J. AUBERT et al. “A comprehensive evaluation of normalization methods for Illumina high-throughput RNA sequencing data analysis”. In : *Briefings in Bioinformatics* (2012) (cf. p. 22).

[47]S. DRAY, S. PAVOINE et D. Aguirre de CÁRCER. “Considering external information to improve the phylogenetic comparison of microbial communities : a new approach based on constrained double principal coordinates analysis (cDPCoA)”. In : *Molecular Ecology Resources* 15.2 (2014), p. 242–249 (cf. p. 19, 101).

[48]P. DRINEAS, M. MAHONEY et S. MUTHUKRISHNAN. “Relative-error CUR matrix decompositions”. In : *SIAM Journal on Matrix Analysis and Applications* 30.2 (2008), p. 844–881 (cf. p. 94).

[49]E.R. DUSKO et MetaHIT CONSORTIUM. “etagenomics of the intestinal microbiota : potential applications”. In : *Gastroentérologie Clinique et Biologique* 34 (2010), p. 23–28 (cf. p. 19).

[50]A. EL GOLLI, B. CONAN-GUEZ et F. ROSSI. “A self organizing map for dissimilarity data”. In : 2004, p. 61–68 (cf. p. 34).

[51]C.H. ELZINGA. “Sequence similarity : a nonaligning technique”. In : *Sociological Methods and Research* 32.3-29 (2003) (cf. p. 70).

[52]F. ESCUDIE, L. AUER, M. BERNARD et al. “FROGS : Find Rapidly OTU with Galaxy Solution”. In : *In : The environmental genomic Conference*. 2015 (cf. p. 24).

[53]Karoline FAUST, J. Fah SATHIRAPONGSASUTI, Jacques IZARD et al. “Microbial Co-occurrence Relationships in the Human Microbiome”. In : *PLOS Computational Biology* 8.7 (2012), e1002606 (cf. p. 24).

[54]Noah FIERER, Jonathan W. LEFF, Byron J. ADAMS et al. “Cross-biome metagenomic analyses of soil microbial communities and their functional attributes”. In : *Proceedings of the National Academy of Sciences* 109.52 (2012), p. 21390–21395 (cf. p. 100).

[55]F. FOUSS, A. PIROTTE, J.M. RENDERS et M. SAERENS. “Random-walk computation of similarities between nodes of a graph, with application to collaborative recommendation”. In : *IEEE Transactions on Knowledge and Data Engineering* 19.3 (2007), p. 355–369 (cf. p. 66).

[56]Eric A. FRANZOSA, Tiffany HSU, Alexandra SIROTA-MADI et al. “Sequencing and beyond : integrating molecular ’omics’ for microbial community profiling”. In : *Nature Reviews Microbiology* 13 (2015), 360–372 (cf. p. 36).

[57]T. FRUCHTERMAN et B. REINGOLD. “Graph drawing by force-directed placement”. In : *Software, Practice and Experience* 21 (1991), p. 1129–1164 (cf. p. 68).

[58]A. GEORGAKIS, H. LI et M. GORDAN. “An ensemble of SOM networks for document organization and retrieval AKRR (2005)”. In : *Proceedings of International Conference on Adaptive Knowledge Representation and Reasoning (AKRR 2005)*. 2005 (cf. p. 49).

[59]A. GISBRECHT, A. SHULTZ et B. HAMMER. “Parametric nonlinear dimensionnality reduction using kernel t-SNE”. In : *Neurocomputing* 147 (2015), p. 71–82 (cf. p. 72).

[60]A. GISBRECHT, B. MOKBEL et B. HAMMER. “The Nyström approximation for relational generative topographic mappings”. English. In : *NIPS workshop on challenges of Data Visualization*. (11 déc. 2010). Whistler BC, Canada, 2010 (cf. p. 72).

[61]A. GITTENS et M.W. MAHONEY. “Revisiting the Nystrom method for improved large-scale machine learning”. In : *Journal of Machine Learning Research* 28.3 (2013), p. 567–575 (cf. p. 94).

[62]L. GOLDFARB. “A unified approach to pattern recognition”. In : *Pattern Recognition* 17.5 (1984), p. 575–582 (cf. p. 30, 58, 70, 81).

[63]M. GÖNEN et E. ALPAYDIN. “Multiple kernel learning algorithms”. In : *Journal of Machine Learning Research* 12 (2011), p. 2211–2268 (cf. p. 101).

[64]S. GOODWIN, John D. MCPHERSON et W. Richard MCCOMBIE. “Coming of age : ten years of next-generation sequencing technologies”. In : *Nature* 17 (2016), 333–351 (cf. p. 18).

[65]T. GRAEPEL, M. BURGER et K. OBERMAYER. “Self-organizing maps : generalizations and new optimization techniques”. In : *Neurocomputing* 21 (1998), p. 173–190 (cf. p. 34, 56).

[66]L. GUIDI, S. CHAFFRON, L. BITTNER et al. “Plankton networks driving carbon export in the oligotrophic ocean”. In : *Nature* 532 (2016), 465–470 (cf. p. 24, 106).

[67]B. HAMMER et A. HASENFUSS. “Topographic mapping of large dissimilarity data sets”. In : *Neural Computation* 22.9 (2010), p. 2229–2284 (cf. p. 49, 56, 57, 62, 63, 70–72, 81).

[68]B. HAMMER, A. HASENFUSS, F. ROSSI et M. STRICKERT. “Topographic processing of relational data”. In : *Proceedings of the 6th Workshop on Self-Organizing Maps (WSOM 07)*. Sous la dir. de Bielefeld University NEUROINFORMATICS GROUP. Bielefeld, Germany, 2007 (cf. p. 34).

[69]P.D.N. HEBERT, E.H. PENTON, J.M. BURNS, D.H. JANZEN et W. HALLWACHS. “Ten species in one : DNA barcoding reveals cryptic species in the neotropical skipper butterfly astraptes fulgerator”. In : *Genetic Analysis* 101.41 (2004), p. 14812–14817 (cf. p. 83).

[70]T. HESKES. “Energy functions for self-organizing maps”. In : *Kohonen Maps*. Sous la dir. d’E. OJA et S. KASKI. Amsterdam : Elsevier, 1999, p. 303–315 (cf. p. 48).

[71]R.R. HOCHKING. “The analysis and selection of variables in linear regression”. In : *Biometrics* (1976) (cf. p. 71).

[72]D. HOFMANN et B. HAMMER. “Sparse approximations for kernel learning vector quantization.” In : (2013). Sous la dir. de M. VERLEYSEN, 549–554 (cf. p. 62–64).

[73]D. HOFMANN, A. GISBRECHT et B. HAMMER. “Efficient approximations of robust soft learning vector quantization for non-vectorial data”. In : *Neurocomputing* 147 (2015), p. 96–106 (cf. p. 72).

[74]D. HOFMANN, F.M. SCHLEIF, B. PAASS EN et B. HAMMER. “Learning interpretable kernelized prototype-based models”. In : *Neurocomputing* 141 (2014), p. 84–96 (cf. p. 71, 72, 78, 80).

[75]Emily R. HOLZINGER, Scott M. DUDEK, Alex T. FRASE, Sarah A. PENDERGRASS et Marylyn D. RITCHIE. “Transitivity in structural models of small groups”. In : *Bioinformatics* 30.5 (2014), 698–705 (cf. p. 37).

[76]L.A. HUG, Brett J. BAKER, Karthik ANANTHARAMAN et al. “A new view of the tree of life”. In : *Nature Microbiology* 1.16048 () (cf. p. 18).

[77]C. HUTTENHOWER, D. GEVERS, R. KNIGHT et al. “Structure, function and diversity of the healthy human microbiome”. In : *Nature* 486.7402 (2012), p. 207–214 (cf. p. 100).

[78]T. JAAKKOLA, M. DIEKHANS et D. HAUSSLER. “A discriminative framework for detecting remote protein homologies”. In : *Journal of Computational Biology* 1-2.7 (1912), p. 95–114 (cf. p. 28).

[79]P. JACCARD. “The distribution of the flora in the alpine zone.” In : *New Phytologist* 11.2 (1912), p. 37–50 (cf. p. 22, 100).

[80]E. KARSENTI, S.G. ACINAS, P. BORK et al. “A holistic approach to marine eco-systems biology”. In : *PLoS Biology* 9.10 (2011), e1001177 (cf. p. 19, 101, 105).

[81]L. KAUFMAN et P.J. ROUSSEEUW. “Clustering by means of medoids”. In : *Statistical Data Analysis Based on the L1-Norm and Related Methods*. Sous la dir. d’Y. DODGE. North-Holland, 1987, p. 405–416 (cf. p. 30).

[82]M. KIMURA. “A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences”. In : *Journal of Molecular Evolution* 16 (1980), p. 111–120 (cf. p. 58, 83).

[83]A. KLEINER, A. TALWALKAR, P. SARKAR et M.I. JORDAN. “A scalable bootstrap for massive data”. In : *Journal of the Royal Statistical Society : Series B (Statistical Methodology)* 76.4 (2014), p. 795–816 (cf. p. 35, 71).

[84]T. KOHOHEN et P.J. SOMERVUO. “Self-organizing maps of symbol strings”. In : *Neurocomputing* 21 (1998), p. 19–30 (cf. p. 34, 56, 70).

[85]T. KOHONEN. *MATLAB Implementations and Applications of the Self-Organizing Map*. Helsinki, Finland : Unigrafia Oy, 2014 (cf. p. 48).

[86]T. KOHONEN. *Self-Organizing Maps, 3rd Edition*. T. 30. Berlin, Heidelberg, New York : Springer, 2001 (cf. p. 32, 48, 70).

[87]T. KOHONEN et P.J. SOMERVUO. “How to make large self-organizing maps for nonvectorial data”. In : *Neural Networks* 15.8 (2002), p. 945–952 (cf. p. 34).

[88]Vessela N. KRISTENSEN, Ole Christian LINGJAERDE, Hege G. RUSSNES et al. “Principles and methods of integrative genomic analyses in cancer”. In : *Nature Reviews Cancer* 14 (2014), 299–313 (cf. p. 36).

[89]S. KUMAR, M. MOHRI et A. TALWALKAR. “Sampling techniques for the Nyström method”. In : *Journal of Machine Learning Research* 13 (2012), p. 981–1006 (cf. p. 36, 76, 94).

[90]P. LANGFELDER, B. ZHANG et S. HORVATH. “Defining clusters from a hierarchical cluster tree : the dynamic tree cut package for R”. In : *Bioinformatics* 24.5 (2008), p. 719–720 (cf. p. 51).

[91]N. LAPTEV, K. ZENG et C. ZANIOLO. “Early accurate results for advanced analytics on mapreduce”. In : *Proceedings of the 28th International Conference on Very Large Data Bases*. (27–31 août 2012). T. 5. Proceedings of the VLDB Endowment 10. Istanbul, Turkey, 2012 (cf. p. 71).

[92]K.W. LAU, H. YIN et S. HUBBARD. “Kernel self-organising maps for classification”. In : *Neurocomputing* 69 (2006), p. 2033–2040 (cf. p. 56, 62, 63).

[93]Christine LAVIT, Yves ESCOUFIER, Robert SABATIER et Pierre TRAISSAC. “The ACT (STATIS method)”. In : *Computational Statistics & Data Analysis* 18.1 (1994), p. 97 –119 (cf. p. 42, 102).

[94]K.A. LÊ CAO, M.E. COSTELLO, V.A. LAKIS et al. “mixMC : a multivariate statistical framework to gain insight into microbial communities”. In : *PloS One* 11.8 (2016), e0160169 (cf. p. 100).

[95]J.A. LEE et M. VERLEYSEN. *Nonlinear Dimensionality Reduction*. Information Science and Statistics. New York ; London : Springer, 2007 (cf. p. 29, 31, 71, 81, 107).

[96]C. LESLIE, E. ESKIN, A. COHEN, J. WESTON et W. STAFFORD NOBLE. “Mismatch string kernels for discriminative protein classification”. In : *Bioinformatics* 20.4 (2004), 467–476 (cf. p. 28).

[97]C. LESLIE, E. ESKIN et W.S. NOBLE. “The spectrum kernel : a string kernel for SVM protein classification”. In : *Procs of the Pacific Symposium on Biocomputing*. 2002, p. 2–7 (cf. p. 28).

[98]Gipsi LIMA-MENDEZ, Karoline FAUST, Nicolas HENRY et al. “Determinants of community structure in the global plankton interactome”. In : *Science* 348.6237 (2015). eprint : `http://science.sciencemag.org/content/348/6237/1262073.full.pdf` (cf. p. 24, 106).

[99]Y. LIN, TL. LIU et Fuh CS. “Multiple kernel learning for dimensionality reduction”. In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33 (2010), p. 1147–1160 (cf. p. 102–104).

[100]C. LOZUPONE et R. KNIGHT. “UniFrac : a new phylogenetic method for comparing microbial communities”. In : *Applied and Environmental Microbiology* 71.12 (2005), p. 8228–8235 (cf. p. 23, 100).

[101]C.A. LOZUPONE, M. HAMADY, S.T. KELLEY et R. KNIGHT. “Quantitative and qualitative β diversity measures lead to different insights into factors that structure microbial communities”. In : *Applied and Environmental Microbiology* (2007), p. 1576–1585 (cf. p. 23, 70, 100).

[102]U. von LUXBURG. “A tutorial on spectral clustering”. In : *Statistics and Computing* 17.4 (2007), p. 395–416 (cf. p. 66).

[103]D. MAC DONALD et C. FYFE. “The kernel self organising map.” In : *Proceedings of 4th International Conference on knowledge-based intelligence engineering systems and applied technologies*. 2000, p. 317–320 (cf. p. 34, 56, 62, 63, 70, 73).

[104]S. MANDAL, W. van TREUREN, R. WHITE et al. “Analysis of composition of microbiomes : a novel method for studying microbial composition”. In : *Microbial Ecology in Health and Disease* 26 (2015), p. 27663 (cf. p. 100).

[105]J. MARIETTE et N. VILLA-VIALANEIX. “Aggregating Self-organizing maps with topology preservation”. In : *Advances in Self-Organizing Maps and Learning Vector Quantization (Proceedings of WSOM 2016)*. (6–8 jan. 2016). Sous la dir. d’E. MERÉNYI, M.J. MENDENHALL et O’Driscoll P. T. 428. Advances in Intelligent Systems and Computing. Houston, TX, USA : Springer International Publishing Switzerland, 2016, p. 27–37 (cf. p. 9, 34, 39, 40, 94).

[106]J. MARIETTE et N. VILLA-VIALANEIX. “Unsupervised multiple kernel learning for heterogeneous data integration”. In : *Bioinformatics* (2017), btx682 (cf. p. 9, 38, 39, 41).

[107]J. MARIETTE, F. ROSSI, M. OLTEANU et N. VILLA-VIALANEIX. “Accelerating stochastic kernel SOM”. In : *XXVth European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2017)*. (26–28 avr. 2017). Sous la dir. de M. VERLEYSEN. Bruges, Belgium : d-side publications, 2017, p. 269–274 (cf. p. 9, 35, 36, 39, 40).

[108]J. MARIETTE, M. OLTEANU, J. BOELAERT et N. VILLA-VIALANEIX. “Bagged Kernel SOM”. In : *Proceedings of WSOM*. Forthcoming. Mittweida, Germany, 2014 (cf. p. 9, 35, 36, 39, 41, 48, 72).

[109]J. MARIETTE, M. OLTEANU et N. VILLA-VIALANEIX. “Efficient interpretable variants of online SOM for large dissimilarity data”. In : *Neurocomputing* 225 (2017), p. 31–48 (cf. p. 9, 32, 39, 41, 57).

[110]S. Massoni, M. Olteanu et N. Villa-Vialaneix. “Which distance use when extracting typologies in sequence analysis ? An application to school to work transitions”. In : *International Work Conference on Artificial Neural Networks (IWANN 2013)*. (12–14 juin 2013). Puerto de la Cruz, Tenerife, 2013 (cf. p. 62).

[111]Frederick A. Matsen IV et Steven N. Evans. “Edge Principal Components and Squash Clustering : Using the Special Structure of Phylogenetic Placement Data for Sample Comparison”. In : *PLOS ONE* 8.3 (mar. 2013), p. 1–15 (cf. p. 19).

[112]J. McAuley et J. Leskovec. “Learning to discover social circles in ego networks”. In : *NIPS Workshop on Social Network and Social Media Analysis*. 2012 (cf. p. 66).

[113]Paul J. McMurdie et Susan Holmes. “Waste Not, Want Not : Why Rarefying Microbiome Data Is Inadmissible”. In : *PLOS Computational Biology* 10.4 (2014), p. 1–12 (cf. p. 22).

[114]P.J. McMurdie et S. Holmes. “phyloseq : an R package for reproducible interactive analysis and graphics of microbiome census data”. In : *PloS One* 8.4 (2013), e61217 (cf. p. 105, 106).

[115]X. Meng. “Scalable simple random sampling and stratified sampling”. In : *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*. T. 28. JMLR : W&CP. Georgia, USA, 2013 (cf. p. 71).

[116]C.P. Meyer et G. Paulay. “DNA barcoding : error rates based on comprehensive sampling”. In : *PLoS Biology* 3.12 (nov. 2005) (cf. p. 58, 83).

[117]F. Meyer, D. Paarmann, M. D’Souza et al. “The metagenomics RAST server – a public resource for the automatic phylogenetic and functional analysis of metagenomes”. In : *BMC Bioinformatics* 9.386 (2008) (cf. p. 24).

[118]S. Needleman et C. Wunsch. “A general method applicable to the search for similarities in the amino acid sequence of two proteins”. In : *Journal of Molecular Biology* 48.3 (1970), p. 443–453 (cf. p. 88).

[119]A. Neme, J.R.G. Pulido, Muñoz A., S. Hernández et T. Dey. “Stylistics analysis and authorship attribution algorithms based on self-organizing maps”. In : *Neurocomputing* 147 (2015). Advances in Self-Organizing Maps Subtitle of the special issue : Selected Papers from the Workshop on Self-Organizing Maps 2012 (WSOM 2012), p. 147–159 (cf. p. 70).

[120]M.E.J. Newman et M. Girvan. “Finding and evaluating community structure in networks”. In : *Physical Review, E* 69 (2004), p. 026113 (cf. p. 67, 82).

[121]M. Niranjan et F. Fallside. “Neural networks and radial basis functions in classifying static speech patterns”. In : *Computer Speech & Language* 4.3 (1990), p. 275–289 (cf. p. 52).

[122]T.M.W. Nye. “Principal components analysis in the space of phylogenetic trees”. In : *Annals of Statistics* 39.5 (2011), p. 2716–2739 (cf. p. 19).

[123]M. Olteanu et N. Villa-Vialaneix. “On-line relational and multiple relational SOM”. In : *Neurocomputing* 147 (2015), p. 15–30 (cf. p. 56, 57, 62, 70, 81, 90, 95, 96).

[124]M. Olteanu et N. Villa-Vialaneix. “Sparse online self-organizing maps for large relational data”. In : *Advances in Self-Organizing Maps and Learning Vector Quantization (Proceedings of WSOM 2016)*. (6–8 jan. 2016). Sous la dir. d’E. Merényi, M.J. Mendenhall et O’Driscoll P. T. 428. Advances in Intelligent Systems and Computing. Houston, TX, USA : Springer International Publishing Switzerland, 2016, p. 27–37 (cf. p. 41, 78).

[125]M. Olteanu, N. Villa-Vialaneix et C. Cierco-Ayrolles. “Multiple kernel self-organizing maps”. In : *XXIst European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*. (24 avr. 2023–26 avr. 2008). Sous la dir. de M. Verleysen. Bruges, Belgium : d-side publications, 2013, p. 83–88 (cf. p. 62, 63).

[126]M. Olteanu, N. Villa-Vialaneix et M. Cottrell. “On-line relational SOM for dissimilarity data”. In : *Advances in Self-Organizing Maps (Proceedings of WSOM 2012)*. (12–14 déc. 2012). Sous la dir. de P.A. Estevez, J. Principe, P. Zegers et G. Barreto. T. 198. AISC (Advances in Intelligent Systems and Computing). Santiago, Chile : Springer Verlag, Berlin, Heidelberg, 2012, p. 13–22 (cf. p. 34, 56).

[127]R. Ounit, S. Wanamaker, T.J. Close et S. Lonardi. “CLARK : fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers”. In : *BMC Genomics* 16.1 (2015), p. 236 (cf. p. 20).

[128]L.A. Pasa, J.A.F. Costa et M. Guerra de Medeiros. “Fusion of Kohonen maps ranked by cluster validity indexes”. In : *Proceedings of the 9th International Conference on Hybrid Artificial Intelligence Systems (HAIS 2014)*. Sous la dir. de M. Polycarpou, A.C.P.L.F. de Carvalho, J.S. Pan et al. T. 8480. Salamanca, Spain : Springer International Publishing Switzerland, 2014, p. 654–665 (cf. p. 48–50).

[129]Joseph N. Paulson, Colin Stine, Hector Corrada Bravo et Mihai Pop. “Robust methods for differential abundance analysis in marker gene surveys”. In : *Nature Methods* 10.12 (2013), p. 1200–1202 (cf. p. 22).

[130]S. Pavoine. “A guide through a family of phylogenetic dissimilarity measures among sites”. In : *Oikos* 125 (2016), p. 1719–1732 (cf. p. 100).

[131]S. Pavoine, A.B. Dufour et D. Chessel. “From dissimilarities among species to dissimilarities among communities : a double principal coordinate analysis”. In : *Journal of Theoretical Biology* 228.4 (2004), p. 523–537 (cf. p. 19, 24, 32, 100).

[132]B.S. Penn. “Using self-organizing maps to visualize high-dimensional data”. In : *Computers & Geosciences* 31.5 (2005), p. 531–544 (cf. p. 70).

[133]L. Petrakieva et C. Fyfe. “Bagging and Bumping Self Organising Maps”. In : *Computing and Information Systems Journal* 9 (2003), p. 69–77 (cf. p. 48, 49, 52, 64).

[134]H. L’Hermier des Plantes. “Structuration des tableaux à trois indices de la statistique”. Thèse de troisième cycle. Thèse de doct. Université de Montpellier, 1976 (cf. p. 42, 102).

[135]G. Polzlbauer. “Survey and comparison of quality measures for self-organizing maps”. In : *Proceedings of the Fifth Workshop on Data Analysis (WDA’04)*. Sous la dir. de J. Paralic, G. Polzlbauer et A. Rauber. Sliezsky dom, Vysoke Tatry, Slovakia : Elfa Academic Press, 2004, p. 67–82 (cf. p. 50, 65, 82).

[136]M. Pölzlbauer, M. Dittenbach et A. Rauber. “Advanced visualization of self-organizing maps with vector fields”. In : *Neural Networks* 19.6-7 (2006). Advances in Self Organising Maps - WSOM’05, p. 911–922 (cf. p. 70).

[137]Sean Powell, Damian Szklarczyk, Kalliopi Trachana et al. “eggNOG v3.0 : orthologous groups covering 1133 organisms at 41 different taxonomic ranges”. In : *Nucleic Acids Research* 40.D1 (2012), p. D284 (cf. p. 107).

[138]M.N. Price, P.S. Dehal et A.P. Arkin. “FastTree 2 - approximately maximum-likelihood trees for large alignments”. In : *PloS One* 5.3 (2010), e9490 (cf. p. 106).

[139]V. Pylro, F. Roesch, D.K. Morais et al. “Data analysis for 16S microbial profiling from different benchtop sequencing platforms”. In : *Journal of Microbiological Methods* 107 (2014), p. 30–37 (cf. p. 20).

[140]Christian Quast, Elmar Pruesse, Pelin Yilmaz et al. “The SILVA ribosomal RNA gene database project : improved data processing and web-based tools”. In : *Nucleic Acids Research* 41 (2013), p. 590–596 (cf. p. 20).

[141]T.W. Randolph, S. Zhao, W. Copeland, M. Hullar et A. Shojaie. “Kernel-penalized regression for analysis of microbiome data”. Preprint arXiv :1511.00297. 2017 (cf. p. 101).

[142]S. Ren, P. Ling, M. Yang, Y. Ni et Z. Zong. “Multi-kernel PCA with discriminant manifold for hoist monitoring”. In : *Journal of Applied Sciences* 13.20 (2013), p. 4195–4200 (cf. p. 103).

[143]Liam J. Revell. “Size-correction and principal components for interspecific comparative studies”. In : *International Journal of Organic Evolution* 63.12 (2009), 3258–3268 (cf. p. 19).

[144]F. Reverter, E. Vegas et J. Oller. “Kernel-PCA data integration with enhanced interpretability”. In : *BMC Systems Biology* 8 (2014) (cf. p. 105).

[145]S. del Rio, V. López, J.M. Beniítez et F. Herrera. “On the use of MapReduce for imbalanced big data using random forest”. In : *Information Sciences* 285 (2014), p. 112–137 (cf. p. 71).

[146]M.D. Ritchie, E.R. Holzinger, R. Li, Pendergrass S.A. et D. Kim. “Methods of integrating data to uncover genotype-phenotype interactions”. In : *Nature Reviews Genetics* (2015) (cf. p. 36, 37).

[147]P. Robert et Y. Escoufier. “A unifying tool for linear multivariate statistical methods : the RV-coefficient”. In : *Applied Statistics* 25.3 (1976), p. 257–265 (cf. p. 102).

[148]F. Rossi. “How many dissimilarity/kernel self organizing map variants do we need ?” In : *Advances in Self-Organizing Maps and Learning Vector Quantization (Proceedings of WSOM 2014)*. (2–4 juil. 2014). Sous la dir. de T. Villmann, F.M. Schleif, M. Kaden et M. Lange. T. 295. Advances in Intelligent Systems and Computing. Mittweida, Germany : Springer Verlag, Berlin, Heidelberg, 2014, p. 3–23 (cf. p. 34, 56, 59, 71, 73).

[149]F. Rossi et N. Villa-Vialaneix. “Optimizing an organized modularity measure for topographic graph clustering : a deterministic annealing approach”. In : *Neurocomputing* 73.7-9 (2010), p. 1142–1163 (cf. p. 66).

[150]F. Rossi, A. Hasenfuss et B. Hammer. “Accelerating relational clustering algorithms with sparse prototype representation”. In : *Proceedings of the 6th Workshop on Self-Organizing Maps (WSOM 07)*. Bielefeld, Germany : Neuroinformatics Group, Bielefeld University, 2007 (cf. p. 30, 72, 73).

[151]Simon Roux, Jennifer R Brum, Bas E. Dutilh et al. “Ecogenomics and biogeochemical impacts of uncultivated globally abundant ocean viruses”. In : *Nature* 537 (2016), p. 689–693 (cf. p. 106, 107).

[152]Simon Roux, Jeremy Tournayre, Antoine Mahul, Debroas Didier et Enault François. “Metavir 2 : new tools for viral metagenome comparison and assembled virome analysis”. In : *BMC bioinformatics* 76 (2014), p. 15 (cf. p. 24).

[153]C. Saavedra, R. Salas, S. Moreno et H. Allende. “Fusion of self organizing maps”. In : *Proceedings of the 9th International Work-Conference on Artificial Neural Networks (IWANN 2007)*. 2007 (cf. p. 48).

[154]A. Saffari, C. Leistner, J. Santner, M. Godec et H. Bischof. “On-line random forests”. In : *Proceedings of IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE. 2009, p. 1393–1400 (cf. p. 71).

[155]P. Sarlin et S. Rönnqvist. “Cluster coloring of the self-organizing map : an information visualization perspective”. In : *18th International Conference on Information Visualisation*. (16–18 juil. 2013). London, UK : IEEE, 2013, p. 532–538 (cf. p. 70).

[156]P.D. Schloss, S.L. Westcott, T. Ryabin et al. “Introducing mothur : open-source, platform-independent, community-supported software for describing and comparing microbial communities.” In : *Applied and Environmental Microbiology* 75.23 (2009), p. 7537–7541 (cf. p. 20).

[157]B. Schölkopf, K. Tsuda et J.P. Vert. *Kernel methods in computational biology*. London : MIT Press, 2004 (cf. p. 29, 30).

[158]B. Schölkopf, A. Smola et K.R. Müller. “Nonlinear component analysis as a kernel eigenvalue problem”. In : *Neural Computation* 10 (1998), p. 1299–1319 (cf. p. 31, 74, 101, 104).

[159]C.E. SHANNON. “A mathematical theory of communication”. In : *The Bell System Technical Journal* 27 (1948), 379–423 and 623–656 (cf. p. 22).

[160]R. SHEN, A.B. OLSHEN et M. LADANYI. “Integrative clustering of multiple genomic data types using a joint latent variable model with application to breast and lung cancer subtype analysis”. In : *Bioinformatics* 22.25 (2009), p. 2906–12 (cf. p. 36).

[161]E.H. SIMPSON. “Measurement of diversity”. In : *Nature* 163 (1949), p. 688 (cf. p. 22).

[162]A. SINGH, B. GAUTIER, C.P. SHANNON et al. “DIABLO-an integrative, multi-omics, multivariate method for multi-group classification”. In : *BioRxiv* (2016) (cf. p. 37).

[163]A.J. SMOLA et R. KONDOR. “Kernels and regularization on graphs”. In : *Proceedings of the Conference on Learning Theory (COLT) and Kernel Workshop*. Sous la dir. de M. WARMUTH et B. SCHÖLKOPF. Lecture Notes in Computer Science. 2003, p. 144–158 (cf. p. 63, 66).

[164]T. SØRENSEN. “A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons”. In : *Kongelige Danske Videnskabernes Selskab* 4.5 (1948), 1–34 (cf. p. 22).

[165]S. SUNAGAWA, L.P. COELHO, S. CHAFFRON et al. “Structure and function of the global ocean microbiome”. In : *Science* 348.6237 (2015). Sous la dir. d’Emmanuel BOSS, Chris BOWLER, Michael FOLLOWS et al. eprint : `http://science.sciencemag.org/content/348/6237/1261359.full.pdf` (cf. p. 106–114).

[166]R. TIBSHIRANI. “Regression shrinkage and selection via the lasso”. In : *Journal of the Royal Statistical Society, series B* 58.1 (1996), p. 267–288 (cf. p. 71).

[167]W.S. TOGERSON. *Theory & Methods of Scaling*. New York, NY, USA : Wiley, 1958 (cf. p. 104).

[168]Warren S. TORGERSON. “Metabolic sensing by p53 : keeping the balance between life and death”. In : *Psychometrika* 17.4 (1952), 401–419 (cf. p. 24).

[169]G. TOWELL et J.W. SHAVLIK. “Interpretation of artificial neural networks : mapping knowledge-based neural networks into rules”. In : *Proceedings of Advances in Neural Information Processing Systems* 4 (1992) (cf. p. 52).

[170]C. de VARGAS, S. AUDIC, N. HENRY et al. “Eukaryotic plankton diversity in the sunlit ocean”. In : *Science* 348.6237 (2015). Sous la dir. d’Emmanuel BOSS, Michael FOLLOWS, Lee KARP-BOSS et al. eprint : `http://science.sciencemag.org/content/348/6237/1261605.full.pdf` (cf. p. 24, 106–108).

[171]S. VEGA-PONS et J. RUIZ-SCHULCLOPER. “A survey of clustering ensemble algorithms”. In : *International Journal of Pattern Recognition and Artificial Intelligence* (2011) (cf. p. 37).

[172]J.P. VERT. “A tree kernel to analyse phylogenetic profiles”. In : *Bioinformatics* 18 (2002), S276–S284 (cf. p. 28).

[173]N. VILLA et F. ROSSI. “A comparison between dissimilarity SOM and kernel SOM for clustering the vertices of a graph”. In : *6th International Workshop on Self-Organizing Maps (WSOM)*. (3–6 sept. 2007). Bielefeld, Germany : Neuroinformatics Group, Bielefeld University, 2007 (cf. p. 64).

[174]Emilie VILLAR, Gregory K. FARRANT, Michael FOLLOWS et al. “Environmental characteristics of Agulhas rings affect interocean plankton transport”. In : *Science* 348.6237 (2015). eprint : `http://science.sciencemag.org/content/348/6237/1261447.full.pdf` (cf. p. 24, 106).

[175]B.L. VRUSIAS, L. VOMVORIDIS et L. GILLAM. “Distributing SOM ensemble training using grid middleware”. In : *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN 2007)*. 2007, p. 2712–2717 (cf. p. 48–50, 52, 64).

[176]J. WAGNER, F. CHELAR, J. KANCHERL et al. “Metaviz : interactive statistical and visual analysis of metagenomic data”. In : *BioRxiv* (2017) (cf. p. 24).

[177]Z. WANG, S. CHEN et T. SUN. “MultiK-MHKS : a novel multiple kernel learning algorithm”. In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.2 (2008), p. 348–353 (cf. p. 103).

[178]J. WARD. “Hierarchical Grouping to Optimize an Objective Function”. In : *Journal of the American Statistical Association* 301.58 (1963), p. 236–244 (cf. p. 24, 30).

[179]H. WHITE. “Statistical Methods for Detecting Differentially Abundant Features in Clinical Metagenomic Samples”. In : *PLoS Computational Biology* 5.4 (2009), p. 1–11 (cf. p. 22).

[180]M. Van de WIEL, M. NEERINCX, T. BUFFART, D. SIE et H. VERHEUL. “Shrinkbayes : a versatile r-package for analysis of count-based sequencing data in complex study designs”. In : *BMC Bioinformatics* 15 (2014) (cf. p. 22).

[181]C.K.I. WILLIAMS et M. SEEGER. “Using the Nyström method to speed up kernel machines”. In : *Advances in Neural Information Processing Systems (Proceedings of NIPS 2000)*. Sous la dir. de T.K. LEEN, T.G. DIETTERICH et V. TRESP. T. 13. Denver, CO, USA : Neural Information Processing Systems Foundation, 2000 (cf. p. 35, 71, 76).

[182]D.E. WOOD et S.L. SALZBERG. “Kraken : ultrafast metagenomic sequence classification using exact alignments”. In : *Genome Biology* 15.3 (2014), R46 (cf. p. 20).

[183]Gary D. WU, Jun CHEN, Christian HOFFMANN et al. “Linking Long-Term Dietary Patterns with Gut Microbial Enterotypes”. In : *Science* 334.6052 (2011), p. 105–108 (cf. p. 100).

[184]D. YAN, L. HUANG et M.I. JORDAN. “Fast approximate spectral clustering”. In : *Proceedings of the 15th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining*. Sous la dir. de J. ELDER, F. SOULIÉ-FOGELMAN, P. FLACH et M. ZAKI. ACM New York, NY, USA, 2009, p. 907–916 (cf. p. 71).

[185]Z. YU, P. LUO, J. YOU et al. “Incremental semi-supervised clustering ensemble for high dimensional data clustering”. In : *IEEE Transactions on Knowledge and Data Engineering* 28.3 (2016), p. 701–714 (cf. p. 71).

[186]Z. YU, J. YOU, L. LI, H.S. WONG et G. HAN. “Representative distance : a new similarity measure for class discovery from gene expression data”. In : *IEEE Transactions on NanoBioscience* 11.4 (2012), p. 341–351 (cf. p. 70).

[187]Z. YU, H.S. WONG, J. YOU et G. HAN. “Visual query processing for efficient image retrieval using a SOM-based filter-refinement scheme”. In : *Information Science* 203 (2012), p. 83–101 (cf. p. 70).

[188]B. ZHAO, J.T. KWOK et C. ZHANG. “Multiple kernel clustering”. In : *Proceedings of the 2009 SIAM International Conference on Data Mining (SDM)*. Sous la dir. de C. APTE, H. PARK, K. WANG et M.J. ZAKI. Philadelphia, PA : SIAM, 2009, p. 638–649 (cf. p. 101).

[189]X. ZHU, A. GISBRECHT, F.M. SCHLEIF et B. HAMMER. “Approximation techniques for clustering dissimilarity data”. In : *Neurocomputing* 90 (2012), p. 72–84 (cf. p. 72, 76).

[190]J. ZHUANG, J. WANG, S.C.H. HOI et X. LAN. “Unsupervised multiple kernel clustering”. In : *Journal of Machine Learning Research : Workshop and Conference Proceedings* 20 (2011), p. 129–144 (cf. p. 102, 103).