Projet Clustering

Florian Guillaume, Paul Lancelin et Lucas Robin

26 décembre 2021

Table des matières

1 Exercice 1

1.1 Question 1

Pour cet exercice, nous avons décidé de prendre le jeu de données 'Lsun'. C'est un jeu de données à deux dimensions donc simple à représenter. Il possède 400 individus/points répartis en 3 classes et ayant chacune une forme particulière. De plus, les groupes sont distincts et séparés entre eux, ce qui nous donne visuellement ceci :

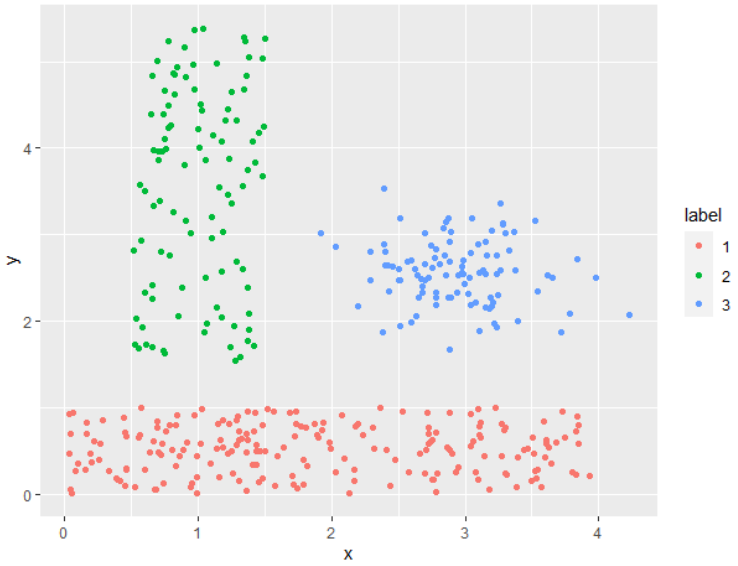

FIGURE 1 – Séries temporelles pour notre étude

Implémenter deux méthodes de partitionnement de votre choix (i.e. méthode de classification non supervisée vue en cours) :

Nous avons décidé d'appliquer la méthode du clustering K-Means à noyaux et le clustering spectral, ayant des propriétés semblables, les méthodes seront expliquées par la suite. Ainsi, pour ces deux méthodes, nous avons essayé différentes initialisations de l'algorithme, mais également réalisé un clustering K-Means classique pour comparer les résultats et les méthodes.

Les méthodes mises en place sont les suivantes :

- K-Means classique (point de comparaison)
- K-Means à noyau polynomial
- K-Means à noyau radial
- Clustering Spectral à noyau polynomial
- Clustering Spectral à noyau radial

1.2 Question 2

Les résultats des clustering mis en place sont présentés sur le graphique ci-dessous :

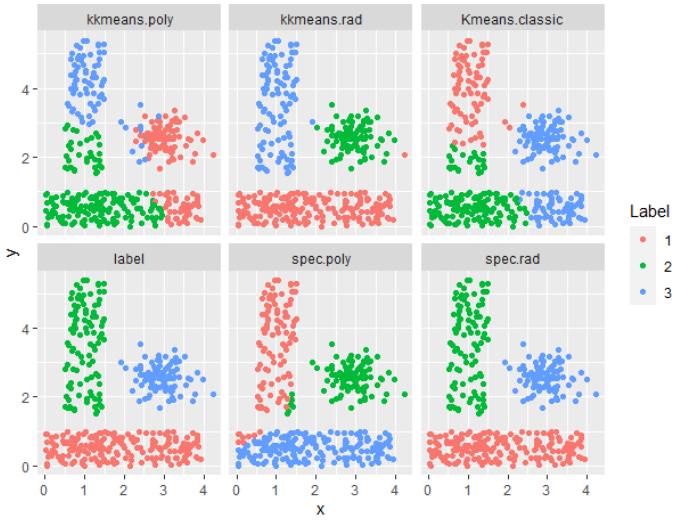

FIGURE 2 – Clustering mis en place

Pour mesurer notre taux de réussite du clustering nous utilisons le critère ARI qui calcule une mesure de similarité entre deux regroupements. Pour ce faire, il considère toutes les paires d'échantillons et compte les paires qui sont attribuées dans les mêmes clusters ou dans des clusters différents pour les labels prédits et les vrais labels. En effet, on ne pas utiliser un taux de classification classique comme l'accuracy. Ce qui nous donne :

Modèles \ K	2	3	4	5
K-Means (classique)	0.6660385	0.4358084	0.5868596	0.6477917
K-Means (radial)	0.1137455	0.7723172	0.2987194	0.5963249
K-Means (poly)	0.5769406	0.7124627	0.4993525	0.4998053
Spectral (radial)	0.5283688	1	0.9429530	0.9356646
Spectral (poly)	0.4102563	0.9070894	0.6033119	0.5263149

1.3 Question 3

1.3.1 Les méthodes à noyau

L'intérêt des méthodes à noyau est de projeter nos données dans un espace de bien plus grande dimension, voir de dimension quasi-infinie. Projeter nos données dans un espace bien plus grand a pour effet d'améliorer la performance des modèles de clustering sur des données non-linéairement séparables.

De plus, la mécanique des méthodes à noyau, aussi appelée "astuce du noyau", consiste a ne jamais calculé les coordonées des points dans le nouvel espace mais de calculer les produits internes entre les images de toutes les paires de données dans l'espace de plus grande dimension. Cette étape de calcul est beaucoup moins couteuse que le calcul explicite des coordonées.

Exemple : le noyau radial

$$\forall x, y \in \mathbb{R}^d, \kappa(x, y) = exp(-\frac{||x - y||^2}{\sigma^2})$$

Initialisation des centroïdes :
Afin d'initialiser nos centroïdes, nous avons utilisé une initialisation aléatoire. Le principe est de tirer aléatoirement k centroïdes dans nos données. D'autre part, comme les centroïdes sont choisis) partir des points de nos données, chaque centroïde est associé à des points de nos données à la fin. Il existe d'autres méthodes d'initialisation On peut penser à la méthode K-Means++ qui sélectionne également des points de nos données, mais avec la contrainte qu'ils soient le plus éloignés les uns des autres.

1.3.2 Méthode à noyau K-Means :

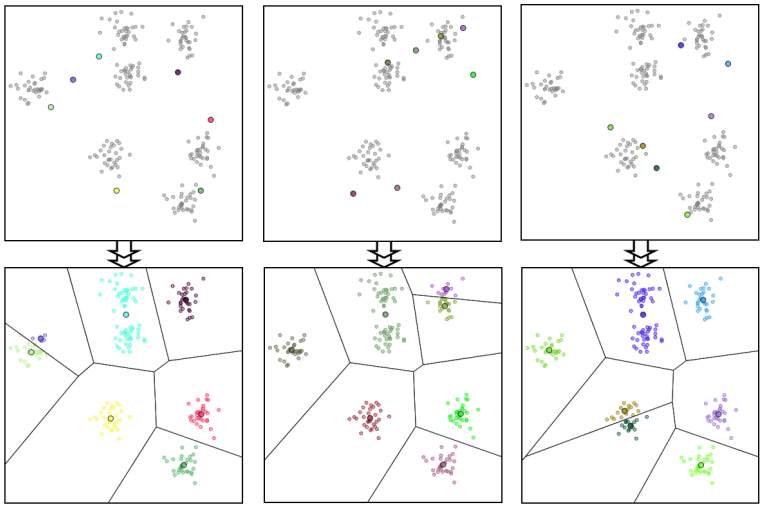

FIGURE 3 – Principe du K-Means

La méthode des K-Means comme son nom l'indique va séparer les données en K groupes. Chaque groupe sera séparé de façon à ce que les données qui y sont présentes soient proches les unes des autres et qu'elles partagent la même "moyenne", cette dernière représente le groupe. Cet Algorithme est aussi appelé algorithme de Lloyd. Il fonctionne en deux étapes répétées. L'algorithme prend en entrée les centroïdes initialisés comme décrit ci-dessus. Dans une première étape de la boucle, il associes les points des données au centroïde le plus proche en utilisant la distance Euclidienne. Vient ensuite la seconde étape de la boucle, où les centroïdes sont mis à jour pour chaque groupe. Par ailleurs, le nombre de données ajoutées à nos nouveaux centroïdes est également compté afin de savoir si la boucle doit être répétée à nouveau. L'algorithme continue d'itérer tant qu'une différence significative est constatée entre les anciens et les nouveaux groupes.

La méthode des K-Means à noyau va tout d'abord calculer la matrice de Gram et donc projeter nos données dans un espace de plus grande dimension pour un noyau donné. Cette étape

a pour intérêt de possiblement mieux rendre compte de la proximité entre les données que dans l'espace initial. Puis, pour des centroïdes initialisés, elle va mettre en place l'algorithme de Lloyd dans le nouvel espace de points. L'intérêt de la projection est toujours d'améliorer la performance de l'algorithme en comparaison à son application sur l'espace des données de départ.

1.3.3 Clustering Spectral

Le clustering spectral est un algorithme de classification non supervisé qui a pour but de déterminer des clusters de noeuds sur un graphe pour des données individus/variables. Cet algorithme est basé sur la décomposition spectrale du Laplacien (normalisé) d'une matrice de similarité et il peut être résumé comme suit (*tiré du cours de Laurent Rouvière*) :

Entrées :

- tableau de données $n \times p$
- K un noyau
- k le nombre de clusters.

1. Calculer la matrice de **similarités** W sur les données en utilisant le **noyau** K
2. Calculer le **Laplacien normalisé** L_{norm} à partir de W.
3. Calculer les k **premiers vecteurs propres** $u_1, \ldots, u_k$ de L_{norm}. On note U la matrice $n \times k$ qui les contient.
4. Calculer la matrice T en **normalisant les lignes** de U : $t_{ij} = u_{ij}/(\sum_\ell u_{i\ell}^2)^{1/2}$.
5. Faire un k-**means** avec les points $y_i, i = 1, \ldots, n$ (i-ème ligne de T) $\Longrightarrow A_1, \ldots, A_k$.

Sortie : clusters $C_1, \ldots, C_k$ avec

$$C_j = \{i | y_i \in A_j\}.$$

FIGURE 4 – Graphe connexe

La méthode du clustering spectral commence par transformer notre tableau de données en un graphe pondéré et non-orienté afin de pouvoir y appliquer la méthode de partition de graphe spectral.

Etape 1 : Le clustering spectral utilise le noyau pour calculer la matrice de similarité. On obtient alors un graphe.

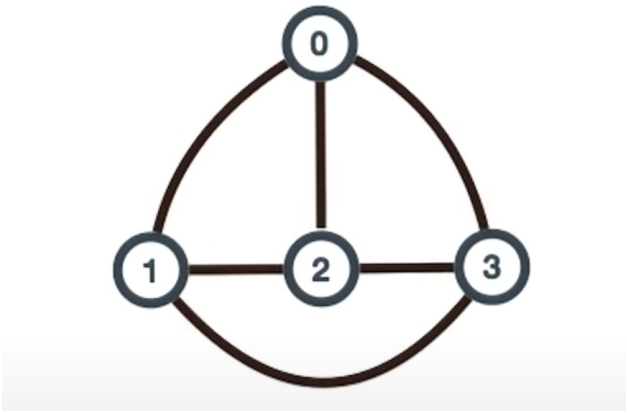

FIGURE 5 – Graphe connexe

Etape 2 : Construire la matrice D (diagonale) qui contient les dégrés de chaque sommet.

$$\mathbf{D} = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

Etape 3: Construire la matrice W (d'adjacence) qui marque toutes les arretes.

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & -1 & -1 \\ -1 & 0 & -1 & -1 \\ -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}$$

Remarque : Ces deux matrices on une forme très régulière car notre graphe est entièrement connecté.

Etape 4: Obtention le Laplacien du graphe

$$L(G) = D - W = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}$$

Mais quelle est le lien avec notre problème de partitionnement ?

Pour faire le lien avec le partitionnement on utilise les éléments suivants :

1) Laplacien L(G) est symétrique
2) L(G) a des valeurs propres réelles et non-négatives
   L(G) a des vecteurs propres orthogonaux et à valeurs réelles

3) G possède k composantes connexes si et seulement si $\lambda_0 = ... = \lambda_{k-1} = 0$ (Fiedler, 1973)

Il existe donc un lien entre le spectre de L(G) et la connexivité de G !

On peut donc utiliser ce point pour nous aguiller sur le nombre de groupe lors de notre clustering. En effet, dans notre cas, il n'y aura qu'une seule valeur propre nulle (étant en présence d'un graphe complètement connecté) mais le trou spectral peut nous donner une indication du nombre de groupes.

4) On calcule la matrice U des k vecteurs propres (en colonne) associés à 0 ou à défault au nombre k de composantes connexes supposé. De plus, la méthode du laplacien normalisé suppose la normalisation des lignes de cette matrice et l'obtention de la matrice T.
5) La dernière étape consiste à effectuer un K-Means sur les vecteurs $y_i, i = 1, ..., n$ ($i^{ème}$ ligne de T).
On obtient en sortie $A_1, ..., A_k$.

Finalement, on obtient nos clusters $C_1, ..., C_k$ avec :

$$C_j = \{i | y_i \in A_j\}$$

2 Exercice 2

Pour cet exercice, nous avons pris le jeu de donnée StarLightCurve possédant des courbes de lumière stellaire de la luminosité d'un objet céleste en fonction du temps. L'étude des courbes de lumière en astronomie est associée à l'étude de la variabilité des sources. Il s'agit d'un ensemble de données avec 1 000 courbes de lumière des étoiles (soi-disant) alignées en phase d'une longueur de 1 024, dont la classe a été déterminée par un expert (pour plus de précisions : https://arxiv.org/abs/0905.3428).

Afin de nous alléger en terme de calcul nous avons pris seulement 70 courbes et nous avons agrégé avec un pas de 4. C'est-à-dire nous avons pris les séries temporelles $(x_1, x_2, \ldots, x_{1024})$ et remplacer par leurs valeurs moyennes : $\left(\frac{x_1+x_2+x_3+x_4}{4}, \frac{x_5+x_6+x_7+x_8}{4}, \ldots\right)$. Ce qui nous donne graphiquement, avec leurs vrais labels, les courbes suivantes :

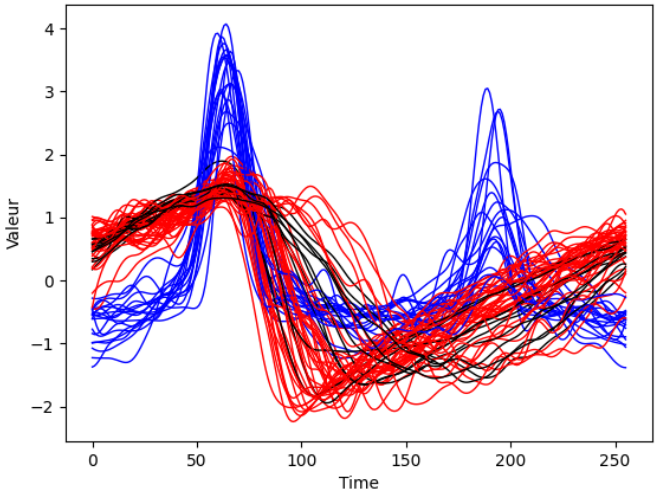

FIGURE 6 – Series temporelle de notre étude

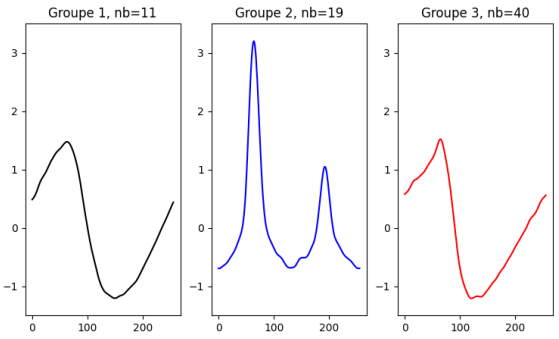

FIGURE 7 – série moyenne arimétiques par groupe

2.1 Clustering par distance

2.1.1 Similarité de séries chronologiques

En clustering, beaucoup d'algorithmes se reposent sur la notion de distance/similarité entre les individus. Depuis ces dernières années la communauté scientifique a étendue cette notion pour des séries chronologiques. Ainsi, les mesures les plus connues pour calculer la distance entre deux séries sont Dynamic Time Warping (DTW), l'inconvénient est que cette mesure n'est pas différentiable partout. Les chercheurs ont alors créé Soft-DTW qui résout ce problème, mais ce n'est pas une divergence définie et elle peut être négative puisque non minimisée lorsque les séries temporelles sont égales. Dans l'article 'Differentiable Divergences between Time Series' de Mathieu Blondel, Arthur Mensch et Jean-Philippe Vert, les auteurs proposent une nouvelle mesure de similarité entre deux séries temporelles à taille variable appelée divergence soft-DTW, qui repose sur soft-DTW et corrige les problèmes cités précédemment. Cette mesure se définit comme telle :

Définition divergence soft-DTW : Considérons deux séries temporelles $\mathbf{x} = (x_0, \ldots, x_n)$ et $\mathbf{y} = (y_0, \ldots, y_n)$ de même longueur.

$$D^\gamma(\mathbf{x}, \mathbf{y}) = \text{soft-}DTW^\gamma(\mathbf{x}, \mathbf{y}) - \frac{1}{2}\left(\text{soft-}DTW^\gamma(\mathbf{x}, \mathbf{x}) + \text{soft-}DTW^\gamma(\mathbf{y}, \mathbf{y})\right) \tag{1}$$

Pour plus de détails sur cette mesure, voir l'article 'Differentiable Divergences between Time Series' de Mathieu Blondel, Arthur Mensch, Jean-Philippe Vert : https://arxiv.org/abs/2010.08354

2.1.2 Clustering

Nous avons donc une distance de similarité entre nos séries, il devient facile de faire un clustering. En effet, nous avons mis en place l'algorithme de Lloyd Kmeans avec cette divergence soft-DTW. Afin d'évaluer nos résultats et puisque que nous connaissons les vrais labels, nous utilisons l'indice de Rand qui calcule une mesure de similarité entre deux regroupements. Mais dans un contexte de clustering nous ne savons pas à l'avance les étiquettes de chacun des groupes, ainsi nous utilisons le coefficient de silhouette, où nous utilisons la distance divergence Soft-DTW énoncée précédemment. Ainsi, nous utilisons les Kmeans avec cette nouvelle distance pour les différents groupes possibles et évalués notre résultat par ARI et silhouette, ce qui nous donne comme résultat :

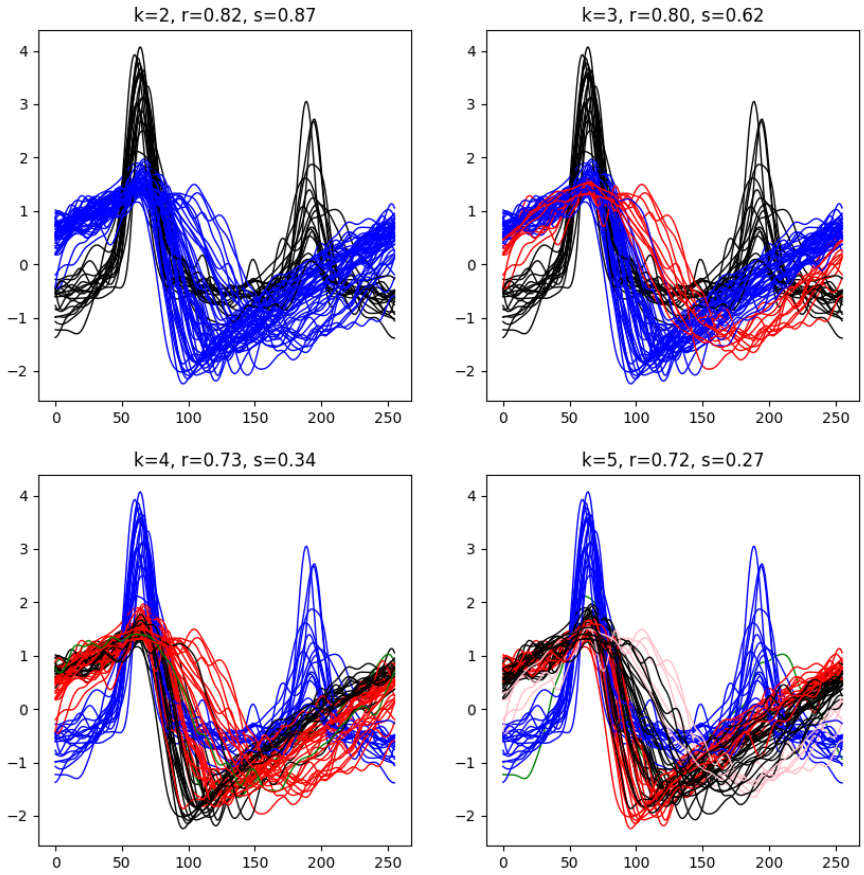

FIGURE 8 – Résultat de l'algorithme Kmeans

r: indice de rand s: score silhouette

Par cette méthode, nous arrivons assez bien a retrouver les vrais labels selon le critère ARI (mais biaisé par un groupe majoritaire). Le critère silhouette est néanmoins le plus fort avec K = 2, ce qui est normal au vu de la ressemblance entre les groupes 1 et 3.

De plus, nous pouvons créer une matrice de distance entre nos séries et utiliser le clustering Hiérarchique ou spectral(implementer dans le code et resultats dans les images).

2.2 ToMATo

Nous présentons maintenant un autre clustering, basé sur la notion de persistance. Il se nomme ToMATo. Pour résumer, cet algorithme utilise un estimateur de densité et un graphe de voisinage. Il commence par une phase de recherche de mode afin de construire des clusters initiaux, et finit par fusionner des clusters en fonction de leur importance. La phase de fusion dépend d'un paramètre qui est la proéminence minimale dont un cluster a besoin pour éviter d'être fusionné dans un autre cluster plus grand. Ainsi, si un cluster n'a pas assez d'importance, pas assez de proéminence, il sera comme absorbé par un cluster plus important. Diminuer le seuil de proéminence définit une hiérarchie de clusters : si 2 points sont dans des clusters séparés quand on a k clusters, ils sont toujours dans des clusters différents pour k+1 clusters. Nous avons utilisé le package `gudhi` possédant cet algorithme et permettant l'affichage du diagramme de persistance. Il s'agit d'un outil graphique pratique pour aider à décider du nombre de clusters final. Après avoir initialisé les paramètres de l'algorithme ToMATo, nous obtenons le diagramme de persistance suivant :

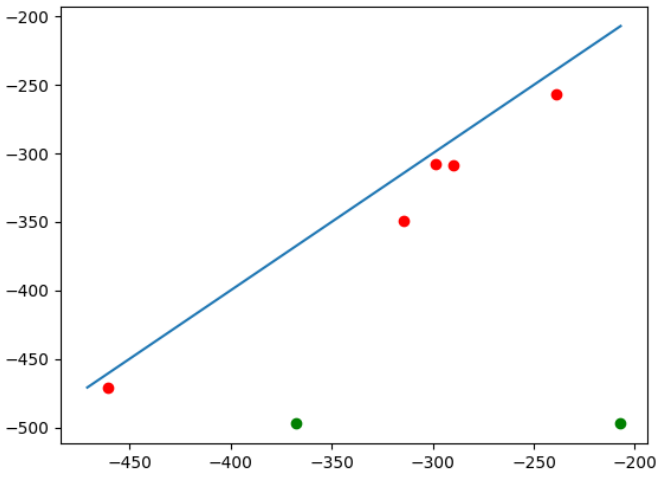

FIGURE 9 – Diagramme de persistance

Ce diagramme montre l'importance des clusters comme leur distance à la diagonale. Il y a toujours un point infiniment loin, ce qui fait qu'il y a au moins un cluster. Parmi les autres, un point semble nettement plus éloigné de la diagonale que les autres, ce qui indique que diviser les points en 2 groupes peut être une idée judicieuse. Mais nous allons évaluer notre clustering selon différents groupes et selon les mêmes critères ARI et silhouette ce qui nous donne :

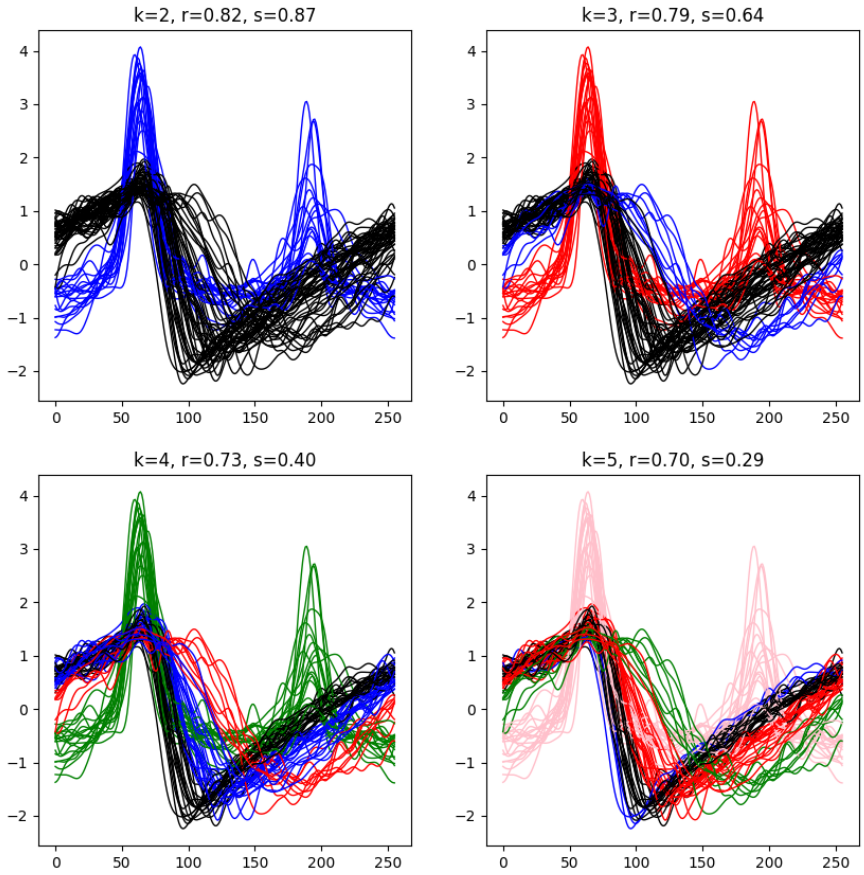

FIGURE 10 – Résultat de l'algorithme ToMATo

Ainsi, nous pouvons voir que nous avons les mêmes scores que dans la partie précédente avec un score de silhouette plus performant pour k = 4 et k = 5. Une possibilité envisageable selon la documentation du package `gudhi` est de pouvoir utiliser un graphe fourni par l'utilisateur dans l'algorithme de ToMATo, graphe que nous pouvons créer par la matrice de distance/divergence Soft-DTW.

2.3 Conclusion et critiques

Ainsi, nous avons mis en place deux types d'algorithmes : un basé sur la notion de distance de série temporelle (kmeans Lloyd avec la divergence Soft-DTW) et un autre sur la persistance (ToMATo). Ces deux algorithmes ont présenté une bonne classification au vu des critères que nous avons mis en place. Cependant, nous avons réduit notre problème en diminuant le nombre d'individus (70 à la place de 1000) et la dimension de notre série temporelle, afin d'alléger nos calculs. En effet, le calcul de la divergence Soft_DTW peut être long pour des séries de grande longueur, mais également lorsqu'il y a beaucoup d'individus. Ainsi, nous avons rencontré un problème computationnel dû à la grande dimension de nos séries. Ce problème est présent principalement dans les modèles de clustering avec cette notion de distance.

Une solution serait une optimisation de code ou alors l'utilisation d'autres méthodes de clustering adaptées pour ces données. L'article 'Functional Data Clustering: A Survey' de Julien Jacques et Cristian Preda (Lien : https://www.researchgate.net/publication/271677889_Functional_Data_Clustering_A_Survey) passe en revue un ensemble de méthodes de clustering sur des données fonctionnelles, dont une intéressante qui se base d'abord par l'approximation des séries dans une base de fonctions connues comme la base de Fourier, la polynomiale ou encore B-spline (très efficace). Ces méthodes permettent une certaine réduction de la dimension. Une fois nos séries exprimées dans cette base de fonctions, on effectue un algorithme de clustering classique sur les coefficients ou encore sur les composantes fonctionnelles (ACP pour les variables fonctionnelles). Le fichier se nommant `fda_clust` applique cette technique sur l'entièrté du jeu de données (1 000 courbes) et s'exécute en très peu de temps.