Algorithme du k-means

Cf. Annexe pour l'algorithme en pseudo-code

Cf. méthode lancerAlgorithme() de la classe kmeans pour l'implémentation en Java

Jeux de données simples

Nous avons deux jeux de données à deux dimensions : exemple1.txt et exemple2.txt

Exemple 1

Pour k=2 clusters, deux itérations complètes de l'algorithme avec une méthode de calcul des distances euclidiennes :

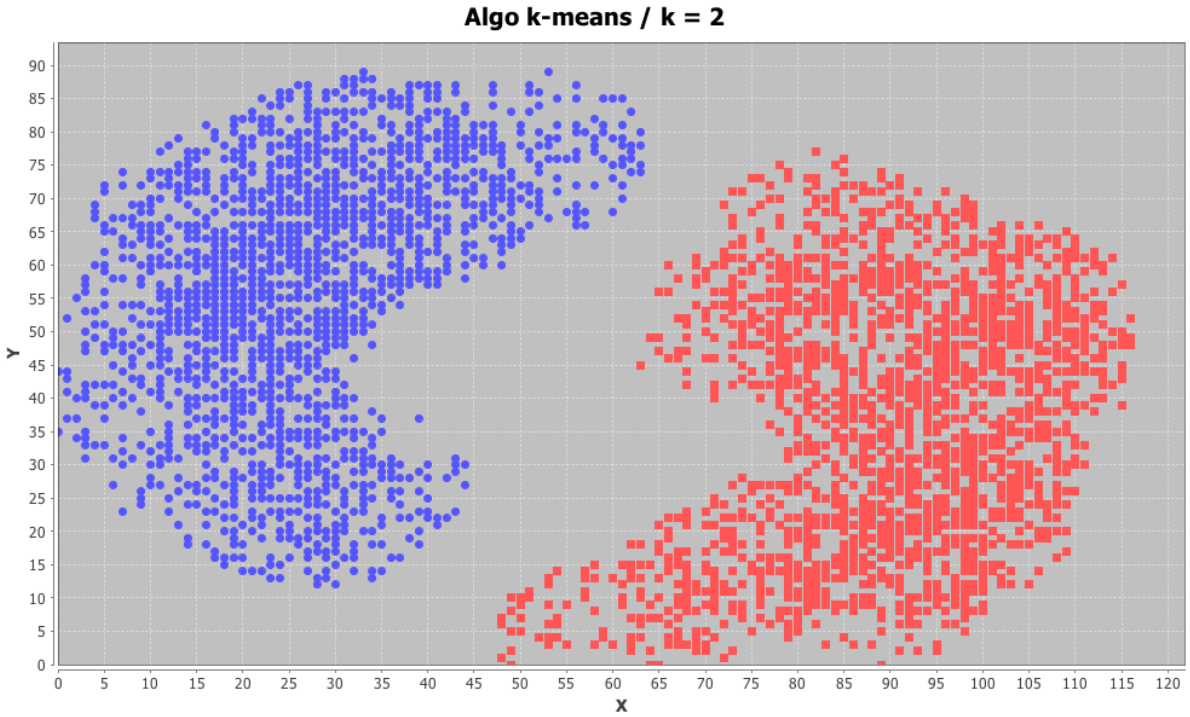


Figure 1 : exemple 1, k=2

Nous voyons avec les figures 1 et 2 que le résultat est le même. Relancé de nombreuses autres fois, l'algorithme converge toujours vers le même découpage. Le jeu de données présentant déjà une répartition en deux parties distinctes couplé à un choix de deux clusters explique très certainement ce résultat.

En effet, lorsqu'on augmente le nombre de cluster (k=3 par exemple), on observe (figures 3 et 4) que le découpage final n'est pas nécessairement le même. Sans calcul de similarité/dissimilarité, on ne peut déterminer quel résultat est le plus "optimal".

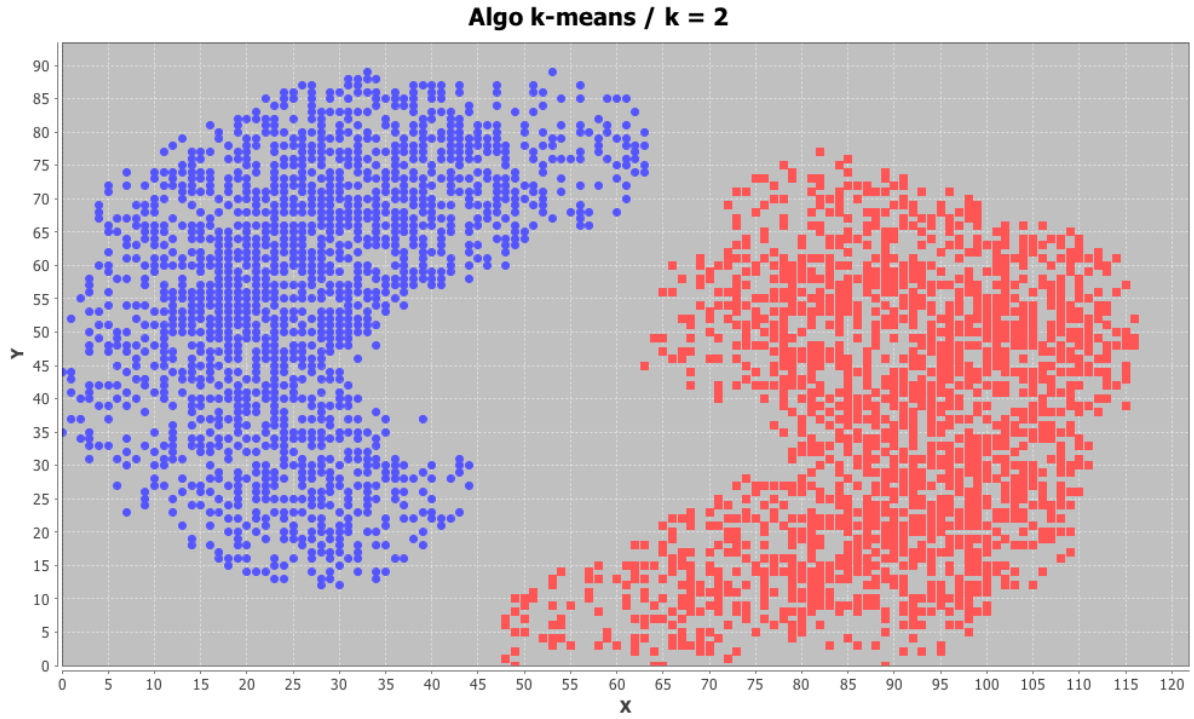


Figure 2 : exemple 1, k = 2

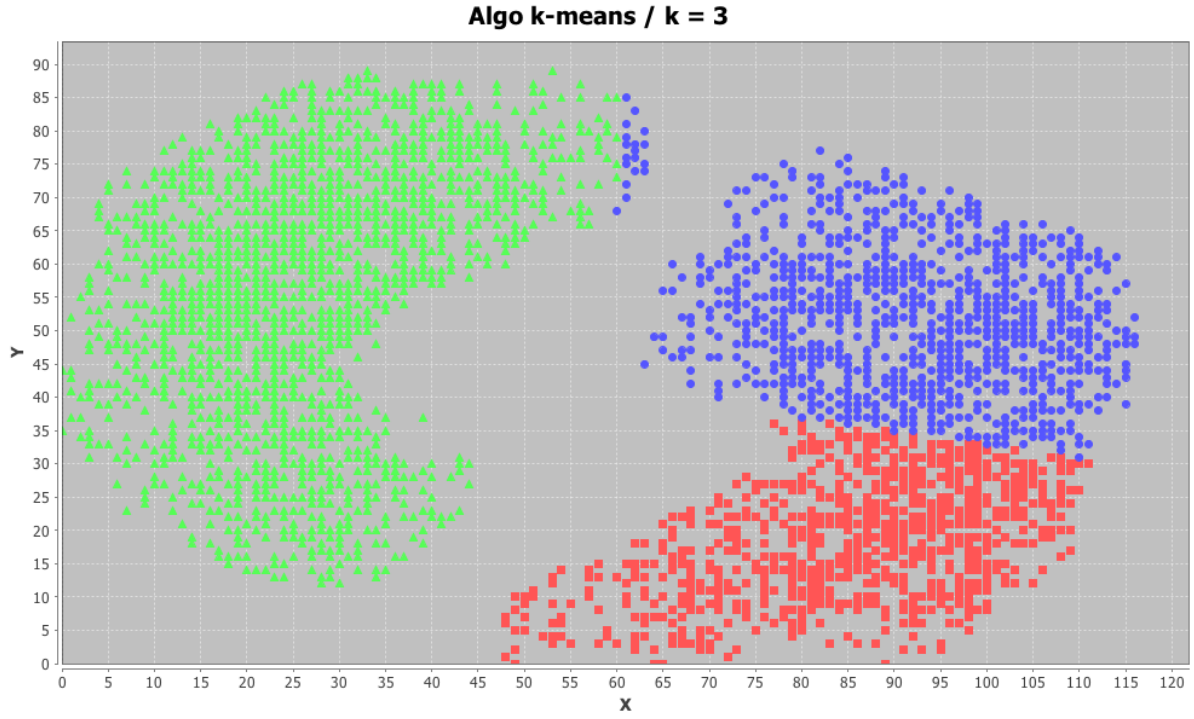


Figure 3 : exemple 1, k = 3

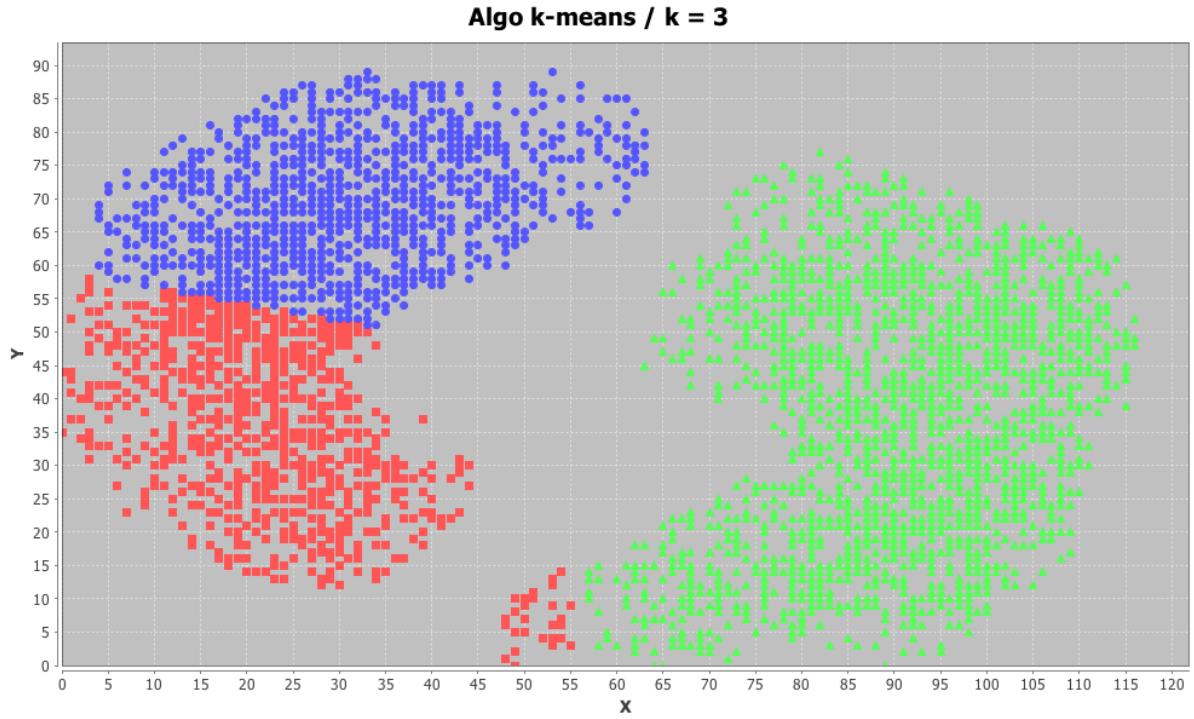


Figure 4 : exemple 1, k = 3

Sur la figure 3 par exemple,, on observe que quelques individus qu'on aurait *a priori* attribué au cluster vert appartiennent en pratique au cluster bleu avec cette application de l'algorithme. On pourrait dire la même chose des quelques individus attribués au cluster rouge sur la figure 4.

Ces exemples montrent l'intérêt de mesures de similarité pour obtenir une classification pertinente.

Exemple 2

La classification pour deux clusters du jeu de données de l'exemple 2 révèle encore plus cet état de fait : intuitivement, on serait tenté de répartir les données en deux, le centre d'une part et la couronne d'autre part. Pourtant, plusieurs classifications pour deux clusters de notre algorithme k-means (sans mesure de similarité) coupe littéralement en deux les données (figures 5 et 6).

On peut noter qu'avec k=5 le rond central est toujours découpé, que de k=6 à k=9 on arrive à obtenir des classifications plutôt cohérentes, et qu'à partir de k=10 on retombe sur un nombre de clusters trop élevé pour avoir un rond central cohérent (figure 7).

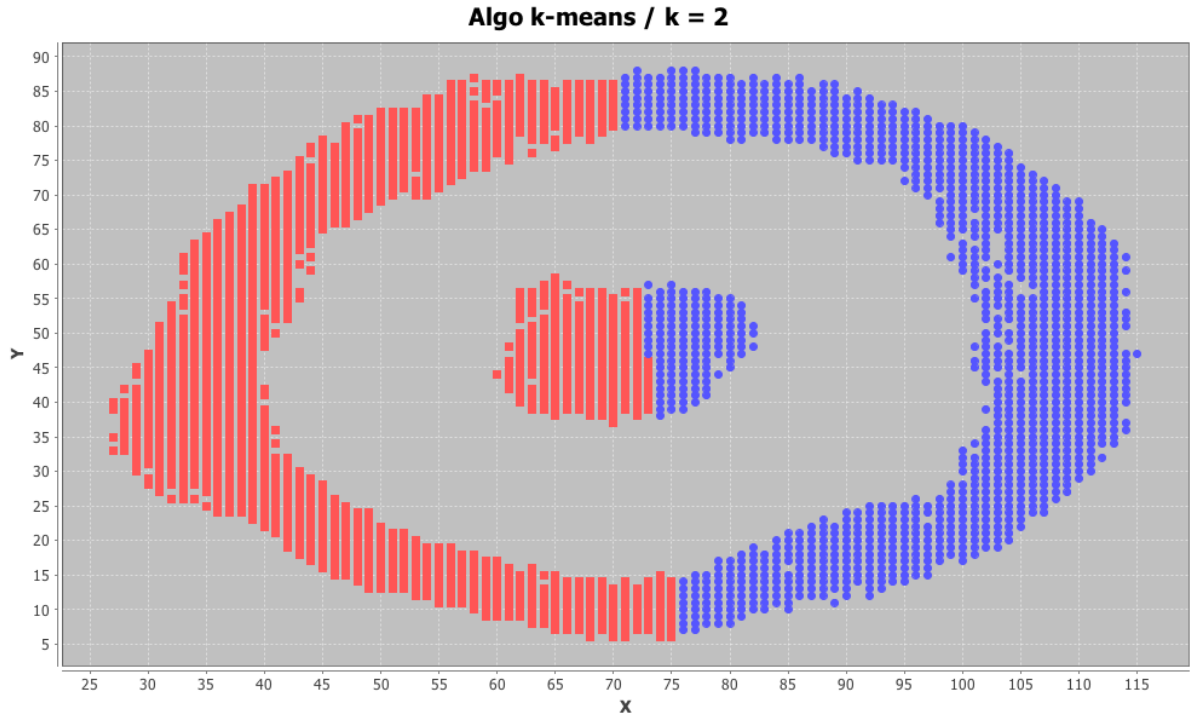


Figure 5 : exemple 2, k = 2

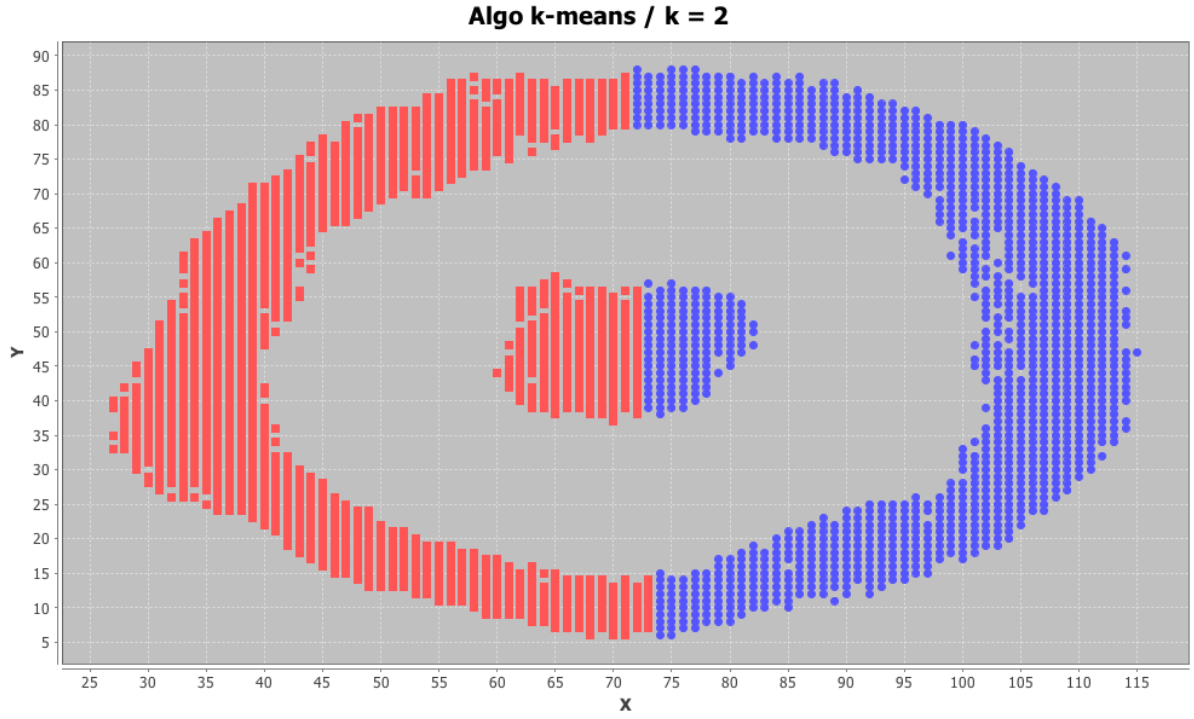
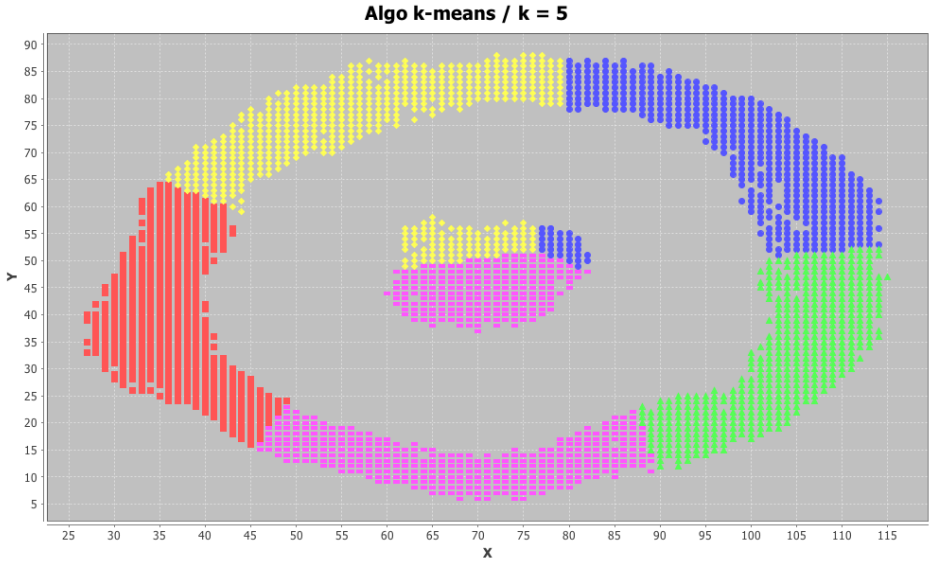
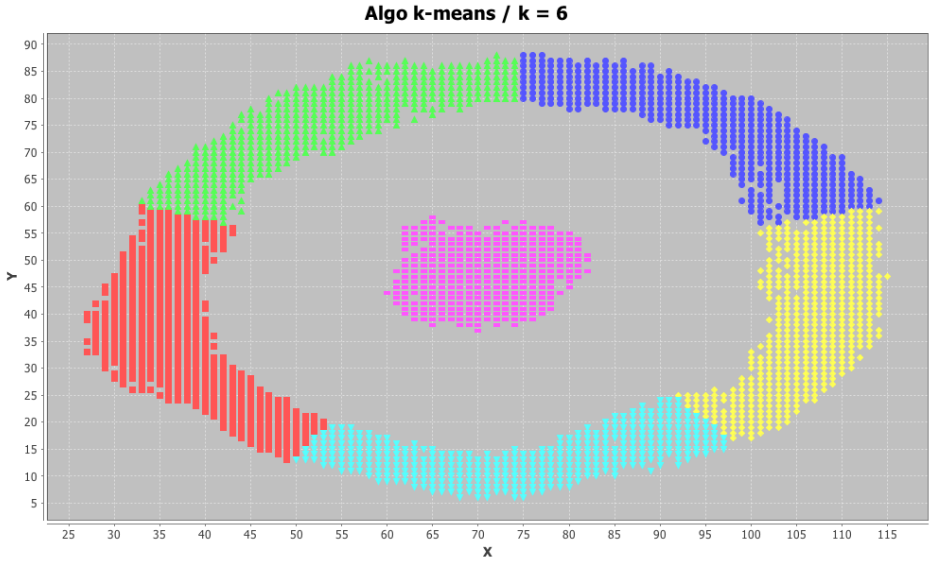


Figure 6 : exemple 2, k = 2





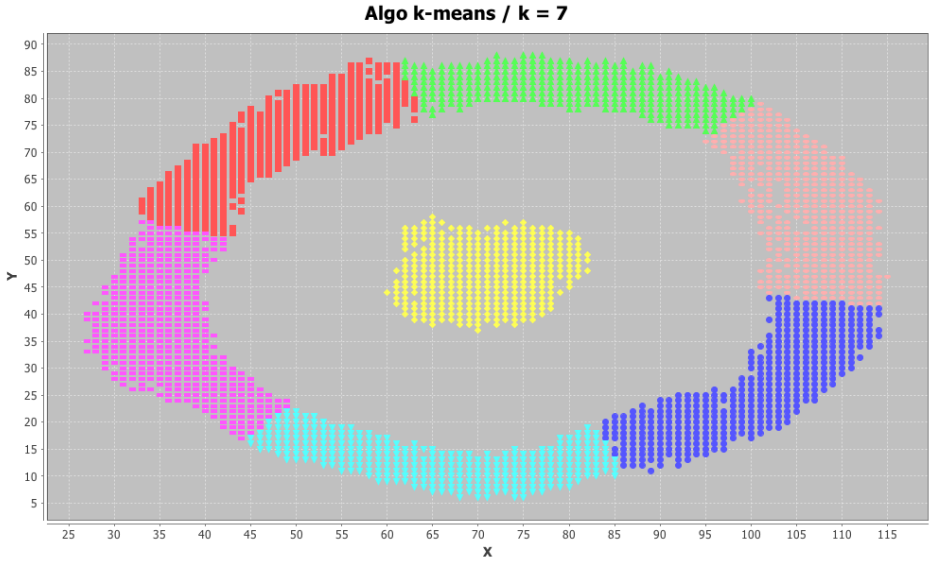
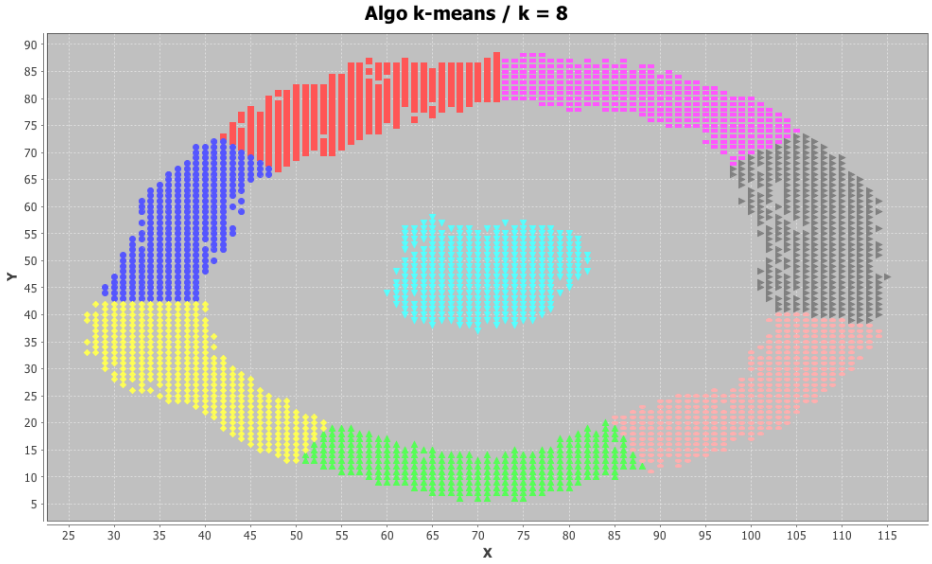
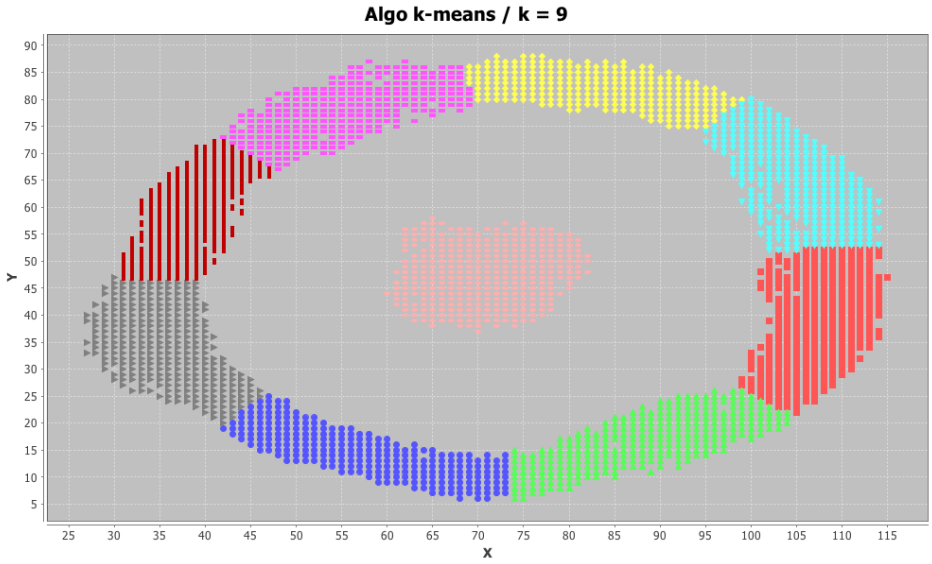


Figure 7.1 : exemple 2





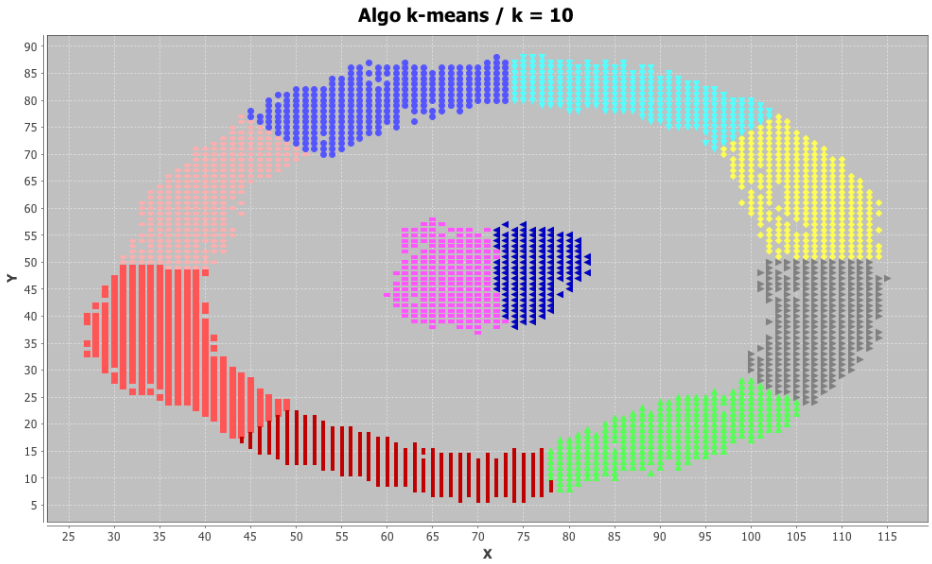


Figure 8.2 : exemple 2

Calcul de similarité

La méthode similarity calcule et affiche l'inertie inter et intra groupe. Idéalement, il faut ensuite utiliser ces valeurs pour, à k fixé, déterminer lequel des n lancements complets de l'algorithme permet d'avoir l'inertie inter maximale et intra minimale.

Jeux de données complexes

Le traitement est correctement effectué (il faut "zapper" la première ligne du fichier contenant le jeu de données puisqu'il s'agit de texte). Pour pouvoir visualiser graphiquement ces données, il s'agit de faire une Analyse en Composantes Principales. En effet, le nombre de notes est trop élevé pour permettre tel quel une représentation dans un graphique en deux dimensions (même 3).

Il faut tout d'abord calculer la matrice des coefficients de corrélation pour les matières, ainsi que la matrice des variances-covariances permettant d'établir les valeurs propres. A partir de ces dernières, on peut établir une matrice des corrélations variables-facteurs (les "facteurs" apparaissant lorsque les valeurs propres sont établies). On peut ainsi obtenir deux facteurs qui corrèlent positivement ou négativement chaque matière — permettant ainsi une représentation graphique.

Puis, on peut établir la matrice correspondante pour les individus : en fonction des notes obtenues dans chaque matière, et en fonction des facteurs obtenus précédemment, on se retrouve avec pour chaque individu deux variables représentant leurs résultats (un facteur se rapprochant de leur moyenne générale, un autre de leur résultats homogènes ou non selon les disciplines). Chaque facteur étant positionné sur un axe (X,Y) on peut obtenir une représentation graphique.

L'appartenance à un cluster est enfin, comme jusqu'à alors, représentée par une couleur.

Malheureusement, si nous avons pu vérifier cette belle théorie pour un jeu de données test sous Matlab, nous n'avons eu le temps de l'implémenter en Java avec notre algorithme.