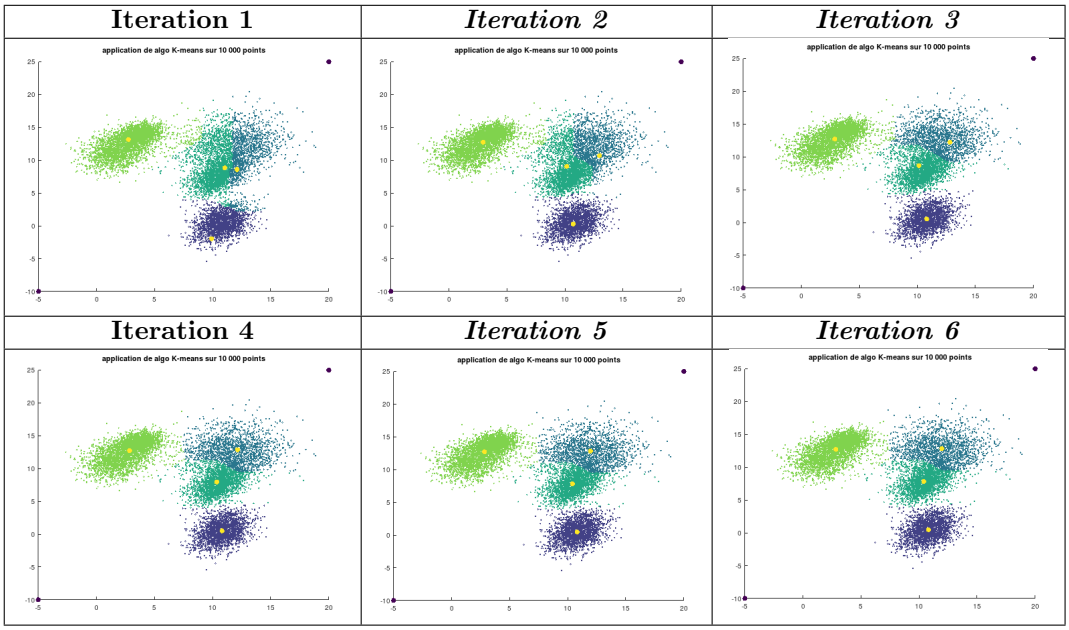
# TP2 K-means

Florent Diet

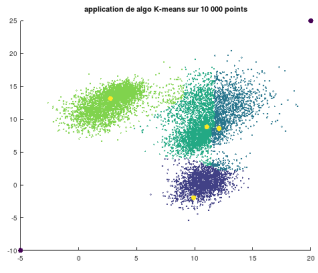
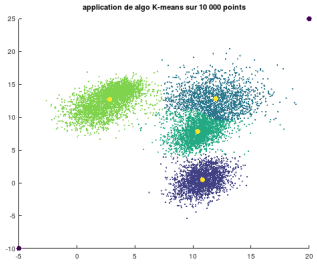
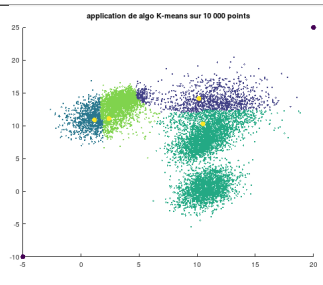
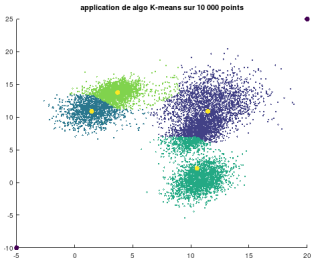
Novembre 2022

## 1 Observation sur le rendu

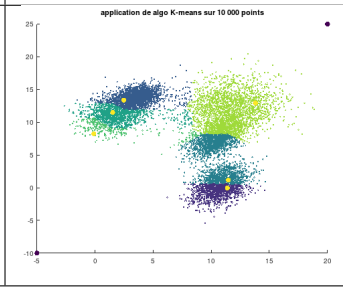
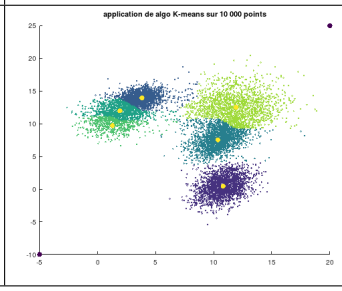



Ici les points jaunes sont les centres des régions depuis lesquels on fait les calculs de distance et les points noirs sont des points de construction il ne faut pas y faire attention. On peut voir que l'algorithme s'arrête bien quand il estime que la distance entre le centre de la région est equidistant par rapport à tous les points appartenant à sa propre région.

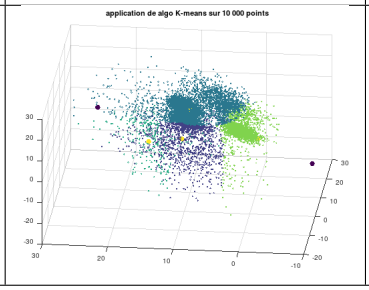
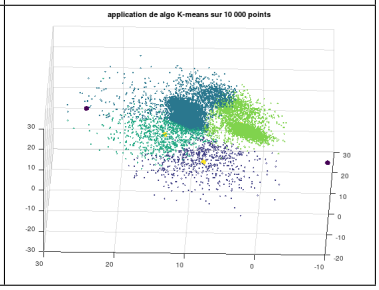
Les résultats peuvent différer en fonction des centres des régions pris au hasard parmi tout les points du jeu de donnée pour la toute premiere iteration.

| | *Iteration 1* | *Iteration 6* |
|---|---|---|
| Exemple 1 |   |   |
| | *Iteration 1* | *Iteration 6* |
| Exemple 2 |   |   |

Le nombre de region peut aussi varier.

| | *Iteration 1* | *Iteration 6* |
|---|---|---|
| Kmeans avec 6 regions |   |   |

Mon application fonctionne aussi dans différentes dimensions même si le résultat n'est pas très net je vous recommande d'utiliser mon code sous octave si vous voulez l'observer de plus près.

| | *Iteration 1* | *Iteration 12* |
|---|---|---|
| Kmeans en 3D |   |   |

# 2 Observation de l'algorithme

## 2.1 Analyse de l'algorithme

Chaque itération possède 3 grandes boucles qui consiste à :

- j'attribue à chaque point un identifiant qui représente son appartenance à une région.
- je remplis un tableau de la somme des coordonnées de mes points selon leurs régions d'appartenance.
- je calcule les nouveaux centres des régions en faisant la moyenne de leurs coordonnées.

## 2.2 Convergence et arrêt

Mon code s'arrête quand les centres de mes régions n'ont pas été grandement modifié par rapport a l'iteration d'avant. Pour faire cela je mets les 2 matrices en entier et je les compare et si jamais toutes les valeurs sont similaires je peux sortir de ma boucle. Avec cette méthode le nombre d'itérations va de 3 jusqu'à 9 itérations avant de s'arrêter avec 4 régions.

J'ai aussi essayé ce même processus avec des valeurs réelles et le nombre d'iteration monte aux alentours de 16 quand on a 4 régions et bien sûr le résultat est plus précis qu'avec des entiers même si on ne le distingue pas vraiment a l'oeil nue.

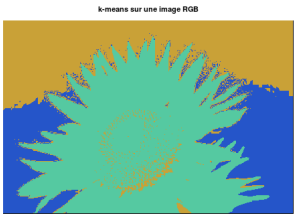
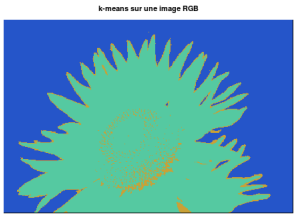
## 2.3 Qualité du résultat

Le résultat me semble cohérent et en ayant mesuré le temps d'exécution de l'application avec 4 régions et 5 itérations cela me donnent 10 secondes. Donc environ 2 secondes par itérations.

# 3 K-means avec des images RGB

Figure 1: Image de tournesol

J'ai aussi réussi a implémenter l'algorithe k-means sur des images RGB. Ici j'ai un tournesol ou je peux distinguer 3 groupe de couleur (si on reste simple). Cela a pris 3 min 30 environ pour faire les 18 itération sur une image rgb en 266x400.

| **Iteration 1** | ***Iteration 7*** | ***Iteration 18*** |
|---|---|---|
|  |  |  |