Density clustering

Théo Lopès-Quintas

1 Présentation et arithmétique

Le but du jeu de la séance est de se présenter, présenter le déroulé des différentes séances à venir puis de commencer à présenter les notions de bases de la programmation à travers la problématique de la décomposition en facteur premier d'un nombre.

Exercice 1.1 (Nombre premier)**.** *Nous souhaitons savoir si un nombre est premier ou non. On rappelle qu'un nombre est premier s'il est divisible uniquement par* 1 *et par lui-même. Ainsi,* 2 *est un nombre premier mais pas* 4*. Par convention* 1 *n'est pas un nombre premier. Construire un script qui renvoie si un nombre est premier ou non.*

Exercice 1.2 (Liste des nombres premiers)**.** *Construire une fonction qui renvoie la liste des nombres premier jusqu'au nombre renseigné par l'utilisateur.*

Exercice 1.3 (Liste des diviseurs)**.** *Construire une fonction qui renvoie la liste des diviseurs d'un nombre. Une fois que c'est fait, réécrire la fonction qui donne la primalité d'un nombre.*

Exercice 1.4 (Liste des diviseur premier)**.** *Construire une fonction qui renvoie la liste des diviseurs premier d'un nombre renseigné par l'utilisateur. Si un nombre premier est plusieurs fois diviseur d'un nombre, alors il apparaîtra plusieurs fois. Par exemple, pour le nombre* 16*, on attend la réponse :* $[2, 2, 2, 2]$.

Exercice 1.5 (Liste de valeur et d'occurence)**.** *On souhaite connaître pour une liste donnée les différentes valeurs et occurrences de ses valeurs. Par exemple, pour la liste* $[1, 2, 4, 3, 3, 1]$*, on souhaite obtenir la liste :* $[[1, 2], [2, 1], [4, 1], [3, 2]]$

Exercice 1.6 (Décomposition en facteur premier)**.** *A partir d'un nombre renseigné par l'utilisateur, construire une fonction qui permet de renvoyer sa décomposition en nombre premier.*

Pour aller plus loin, on peut explorer les idées suivantes :

1. Trouver une manière plus rapide de trouver si un nombre est premier.
2. En prenant le résultat de la décomposition, trouver une manière d'afficher à l'utilisateur la décomposition comme si elle était écrite à la main.

2 Mouvement Brownien et ?

Le but de la séance est d'apprendre à coder proprement un mouvement brownien, plusieurs, corrélés entre eux puis d'exploiter cette compétence pour pricer des options avec la méthode de Monte-Carlo. Sous cette justification, nous allons produire des automatismes et des fonctions qui seront utile pour le TP sur la stratégie d'option.

2.1 Mouvement Brownien

Il faudra probablement faire quelques rappels mathématiques sur le mouvement brownien en début de séance. Ensuite il faudra présenter la partie random de numpy et comment elle s'exploite pour accélérer lors du TP.

Exercice 2.1 (Réalisation d'un mouvement brownien)**.** *1. Écrire une fonction qui renvoie une réalisation d'un mouvement brownien standard en prenant en entrée le nombre de point renseigné par l'utilisateur.*

2. *Reprendre la fonction et ajouter la possibilité de choisir la volatilité et le point de départ du mouvement brownien.*

3. *Améliorer la fonction pour qu'elle permette de renvoyer plusieurs mouvement brownien sous forme d'une matrice, où chaque colonne représente une réalisation et chaque ligne un temps.*

Exercice 2.2 (Visualisation d'un mouvement brownien)**.** *On souhaite visualiser les trajectoires d'un mouvement brownien.*

1. *Écrire une fonction qui prend en entrée un ou plusieurs mouvement brownien sous forme d'une matrice et qui affiche le ou les trajectoires.*

2. *Bonus : Adapter l'opacité des lignes en fonction du nombre de mouvement brownien à afficher.*

3 Conjecture de Syracuse ou suite logistique avec Pierre Lancien

Le but de la séance est d'apprendre à perfectionner sa qualité de code avec les conseils de Pierre Lancien à travers une étude simple de la conjecture de Syracuse ou la suite logistique.

4 Stratégie d'option avec Léa Mina

Le but de la séance est d'obtenir une visualisation d'une stratégie d'option avec des données réelles. On apprend dans la séance à exploiter un peu plus la programmation fonctionnelle et on s'applique à coder très proprement un code qui peut vide devenir massif et difficile à maintenir.
Il faut également apprendre à composer avec une librairie externe. A voir si les fonctions pour récupérer les prix sont données ou non.

Voici les grandes étapes :

- Écrire une fonction call qui va renvoyer (modulo des arguments) un *call*. Faire de même pour un *put*.
- Définir la structure de données et les informations nécessaire pour décrire une option.
- Définir les grandes lignes du programme à développer, soit :
  - Récupérer les prix des options nécessaires
  - Définir un intervalle de valeur pour tracer la stratégie
  - A l'aide des fonctions call et put construire un vecteur de valeur de la stratégie
  - Tracer la stratégie
- Développer chacun des points en utilisant la méthode de programmation fonctionnelle.

Pour aller plus loin dans cette séance qui est assez courte, on peut aussi essayer de se donner comme challenge :

- Calculer et faire apparaître les points de break-even (expliqué par Léa)
- Faire apparaître toutes options et la stratégie sur le graphique, avec une légende dynamique, en faisant attention à avoir un graphique lisible
- Faire apparaître clairement les zones de gain et de perte de la stratégie
- Faire apparaître une courbe gaussienne de probabilité pour avoir une idée de la faisabilité de gain de l'option

5 ? et présentation des projets

Voici la liste des projets synthétiquement :

- Coder et décoder le code RSA
- Jeu de la vie de Conway
- Créer un visualiseur amélioré d'une stratégie d'option en prenant en compte les grecs