Université de Montpellier **HLMA310**
2019/2020 **Pascal Azerad, Florent Bascou, Joseph Salmon**

TP n° 6 : (TP Noté) Chaînes de Markov et gestion de l'aléatoire

Pour ce travail vous devez déposer un **unique** fichier au format `.ipynb`, dont le nom est `tp_note_hmla310_group_?_prenom_nom.ipynb`, le tout en minuscule. Vous remplirez votre nom, prénom et le groupe qui vous concerne (?=A,B ou C), en suivant l'affectation des groupes donnée sur Moodle.

Vous devez téléverser votre fichier avant le **vendredi 25/10/2019, 23h59**, sur Moodle, dans la rubrique "TP noté - Python". Vous pouvez effectuer un premier téléversement pour vérifier que tout marche bien et écraser ensuite à chaque fois jusqu'à la version définitive (pratique pour ne pas avoir un fichier vierge à la date limite).

La note totale est sur **20** points répartis comme suit :

- qualité des réponses aux questions : **14** pts,
- qualité de rédaction et d'orthographe : **1** pts,
- qualité des graphiques (légendes, couleurs) : 1 pt
- style PEP8 valide : **2** pts,
- qualité d'écriture du code (noms de variable clairs, commentaires, code synthétique, etc.) : **1** pt
- Notebook reproductible (i.e., "Restart & Run all" marche correctement sur la machine du correcteur) et absence de bug : **1** pt

Les personnes qui n'auront pas soumis leur devoir sur Moodle avant la limite obtiendront **zéro**.

Rappel : aucun travail par mail ne sera accepté !

Exercice 1. (algèbre linéaire / chaîne de Markov)

1) Créer la matrice

$$M = \begin{pmatrix} 0.448 & 0.054 & 0.011 \\ 0.484 & 0.699 & 0.503 \\ 0.068 & 0.247 & 0.486 \end{pmatrix} .$$

2) Donner la valeur de la somme de chaque colonne.

On définit la suite de vecteurs $X_{n+1} = M \cdot X_n$ où $X_n = \begin{pmatrix} u_n \\ v_n \\ w_n \end{pmatrix}$. La suite est initialisée par un vecteur $X_0 = \begin{pmatrix} u_0 \\ v_0 \\ w_0 \end{pmatrix}$.

3) Créer une fonction qui prend en argument :
   - un entier : n
   - un vecteur initial : X_0

   et qui retourne
   - X, la matrice des itérés, c'est-à-dire la matrice dont les lignes sont les vecteurs $X_0^\top$, $X_1^\top, \dots, X_n^\top$. Noter que cette matrice a $(n+1)$ lignes et 3 colonnes, et donc X peut s'écrire :

$$X = \begin{pmatrix} X_0^\top \\ X_1^\top \\ \vdots \\ X_n^\top \end{pmatrix} .$$

4) On donne $X_0 = \begin{pmatrix} 20 \\ 70 \\ 10 \end{pmatrix}$ et $n = 20$. Afficher pour ce X_0 et ce n la matrice des itérés de la question précédente.

5) Calculer la somme de chaque ligne de la matrice X (avec les même X_0 et n que ci-dessus).

Interprétation en sociologie

On suppose pour simplifier que la société est divisée en 3 classes : 1) favorisée, 2) moyenne, 3) défavorisée. Les coefficients de la matrice $M = (m_{i,j})$ représentent les pourcentages de passage d'une classe à l'autre en une unité de temps fixe, correspondant en gros à une génération. Ainsi,

- $m_{1,1} = 0.448$ signifie que 44.8% d'une classe favorisée reste favorisée au bout d'une unité de temps,
- $m_{2,1} = 0.484$ signifie que 48.4% d'une classe favorisée passera dans la classe moyenne au bout d'une unité de temps,
- $m_{3,1} = 0.068$ signifie que 6.8% d'une classe favorisée passera dans la classe défavorisée au bout d'une unité de temps.
- $m_{1,3} = 0.011$ signifie que 1.1% d'une classe défavorisée passera dans la classe favorisée au bout d'une unité de temps.
- $m_{1,2} = 0.054$ signifie que 5.4% d'une classe moyenne passera dans la classe favorisée au bout d'une unité de temps...

Le vecteur X_0 représente la répartition initiale de la population. $X_0 = \begin{pmatrix} 20 \\ 70 \\ 10 \end{pmatrix}$, signifie qu'au début la population compte 20 personnes favorisées, 70 personnes de classe moyenne et 10 personnes défavorisées.

$$X_{n+1} = M \cdot X_n$$

modélise l'évolution de la population au fur et à mesure des générations. cela se comprend bien en regardant par exemple la première composante

$$u_{n+1} = 0.448 \cdot u_n + 0.054 \cdot v_n + 0.011 \cdot w_n$$

NB : les chiffres sont arbitraires, bien qu'issus d'un livre de sociologie (Glass and Hall, *Social mobility in UK*)

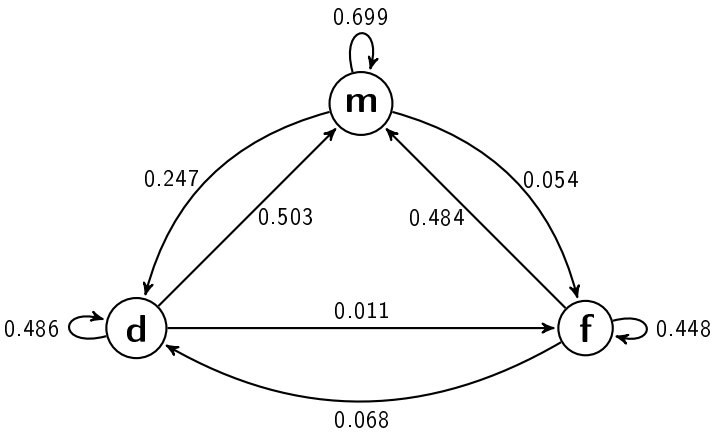

FIGURE 1 – Illustration du graphe illustrant l'évolution sociologique.

6) Tracer sur une même figure l'évolution des trois classes jusqu'à la n^e génération, c'est-à-dire afficher les trois suites $u_0, \dots, u_{10}$; $v_0, \dots, v_{10}$ et $w_0, \dots, w_{10}$. Construisez un graphique qui reproduit la Figure 2, en utilisant la fonction `fill_between` (comme vu au TP3).
7) Calculer M^{25} et obtenir directement X_{25}, la population à la 25^e génération.
8) Obtenir la population à la 25^e génération en prenant comme condition initiale $X_0 = \begin{pmatrix} 0 \\ 0 \\ 100 \end{pmatrix}$ puis $X_0 = \begin{pmatrix} 0 \\ 100 \\ 0 \end{pmatrix}$. Que constatez vous ?

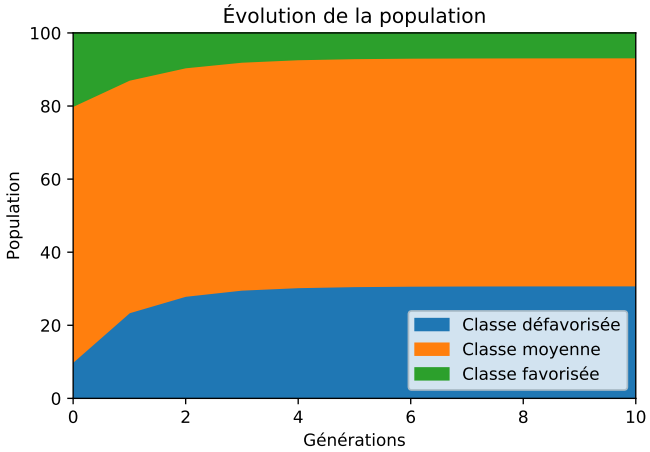

FIGURE 2 – Figure à reproduire

9) Diagonaliser la matrice M avec `numpy`. Expliquer pourquoi M^n converge quand $n \to +\infty$.

10) Facultatif : Comment expliquer le fait que la répartition limite ne dépende pas de la condition initiale ?

EXERCICE 2. (Génération de nombre aléatoires)

On s'intéresse ici aux générateurs congruentiels linéaires, une des méthodes permettant de générer des nombres pseudo-aléatoires par un ordinateur.

Un générateur congruentiel linéaire forme une suite de nombres, définie par la récurrence suivante :

$$X_{n+1} = (aX_n + c) \mod m \ , \tag{1}$$

où :

- $a \in \mathbb{N}$ est le multiplicateur,
- $c \in \mathbb{N}$ est l'incrément,
- $m \in \mathbb{N}$ le module.

L'initialisation X_0 est souvent appelé la **graine**, et permet de générer la suite.

On peut à l'aide d'un tel générateur de nombre aléatoire, générer une suite de nombres dans $[0, 1[$ en considérant simplement :

$$\overline{X}_n = \frac{X_n}{m} \ . \tag{2}$$

1) Écrire une fonction `module`, permettant de coder la relation de récurrence précédente, qui prend en entrée x, et les trois arguments optionnels $a = 14, c = 11$, et $m = 95$, et qui renvoie en sortie : $(ax+c) \mod m$. Vérifier votre fonction retourne la valeur 87 pour $a = 14, c = 11, m = 95$ et $x = 19$.

2) Écrire une fonction `suite` qui prend en entrée un nombre x_0, un entier n, et des paramètres optionnels $a = 14, c = 11, m = 95$, et qui renvoie la liste $[x_0, \ldots, x_n]$ obtenue pour la suite de nombres aléatoires associés. Pour $n = 38$, quelle particularité à la liste produite par cette fonction ?

Observez ce que produite la commande suivante :

```
x0 = 0
n = 57
print(suite(x0, n, a=14, c=11, m=2000))
```

Sur cette liste, on observe que la suite se répète à partir d'un certain rang. En fait, dès qu'un nombre réapparaît, la séquence se répète à l'identique. Le nombre d'itération avant que la suite ne se répète se nomme la **période** et un "bon" générateur de nombre aléatoire (c'est-à-dire un bon choix de a, c et m) doit produire une période élevée.

3) Écrire une fonction qui prend en entrée une liste, et renvoie deux nombres : 1) l'indice du premier nombre qui se répète 2) la période associée (pour le cas sans répétition on sortira 0 pour l'indice, et 0 pour la période). Vérifiez que la période est 50 pour la liste obtenue par la commande précédente. Aide : utiliser deux boucles imbriquées.

4) Créer une fonction `extract_periode` qui prend en entrée une liste et renvoie une liste contenant le premier cycle complet, sans l'extrémité à droite (par exemple dans l'exemple précédent on renverra la liste [505, 1081, 1145, 41, 585, 201, 825, 1561, 1865, 121, 1705, 1881, 345, 841, 1785, 1001, 25, 361, 1065, 921, 905, 681, 1545, 1641, 985, 1801, 1225, 1161, 265, 1721, 105, 1481, 745, 441, 185, 601, 425, 1961, 1465, 521, 1305, 281, 1945, 1241, 1385, 1401, 1625, 761, 665, 1321]).

5) Sur l'exemple précédent, tracer l'histogramme des nombres contenus dans une période et normaliser comme dans l'équation (2), en divisant par m (pour obtenir des tirages sur l'intervalle $[0,1]$). Visualiser et commenter l'impact d'augmenter n et m.

Exercice 3. (Racine n^e de l'unité / graphisme)

Dans cette exercice on va s'intéresser aux racines de l'unité. Pour un enter $n \in \mathbb{N}^*$, on appelle racine n^e de l'unité les nombres

$$z_k = e^{2\pi i k/n} = \cos\frac{2k\pi}{n} + i\sin\frac{2k\pi}{n}, \qquad \text{pour } k \in \{0,1,\dots,n-1\} \ .$$

1) En utilisant le module `cmath`, créer une fonction `racines` qui prend n en entrée et renvoie la liste des n racines n^e de l'unité.

2) Afficher le chemin $[z_0,z_1],\dots,[z_{n-1},z_n],[z_n,z_0]$, avec `matplotlib.lines.line2D` : *cf.* https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.lines.Line2D.html, en utilisant parties réelles et parties imaginaires pour l'affichage en 2D.

3) Créer une fonction `compute_midpoints` qui prend en entrée une liste de complexes `points` (*i.e.*, points $= [x_0,\dots,x_{n-1}]$) et un nombre réel $t \in [0,1]$ et qui renvoie une liste de la même taille $[t\cdot x_0 + (1-t)\cdot x_1, \dots, t\cdot x_{n-2} + (1-t)\cdot x_{n-1}, t\cdot x_{n-1} + (1-t)\cdot x_0]$.

4) On reprend le cas des racines 6^e de l'unité et d'un paramètre $t = 0.2$. Afficher sur un même graphique le chemin $[z_0,z_1],\dots,[z_{n-1},z_n],[z_n,z_0]$, puis le chemin obtenus par `compute_midpoints`. Refaire récursivement cette construction pour afficher 12 chemins sur la même figure et obtenir un graphique similaire à la Figure3.

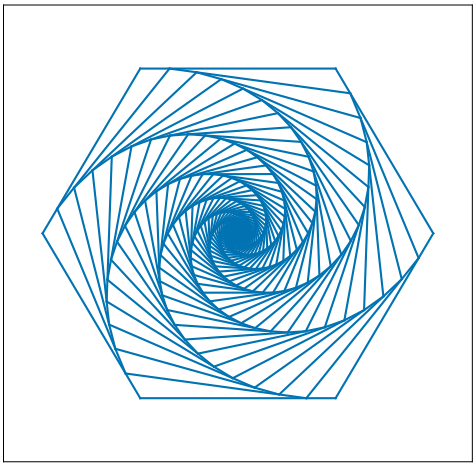

Figure 3 – Exemple avec les racines 6^e de l'unité.

5) Proposer un widget qui contrôle n (le nombre de racines de l'unité / le polygone de départ), t (le coefficient de convexité, introduit dans la question précédente), m le nombre de niveaux de récursion de la figure.