1 introduction

1.1 Objectifs de la feuille

- Savoir simuler une loi discrète usuelle à l'aide des fonctions de la librairie `numpy.random`, ou uniquement à partir de la fonction `rd.random()`.
- Simuler une loi donnée, c'est-à-dire créer une fonction ou un programme Python qui, à chaque appel, renvoie une réalisation d'une variable aléatoire X suivant cette loi.
  Par exemple, simuler le lancers d'un dé à six faces c'est créer une fonction Python qui renvoie les valeurs 1,2,3,4,5 et 6, tel que la fréquence d'apparition de chaque face est égale à 1/6, lorsque le dé est équilibré.
- Créer programme ou une fonction Python qui renvoie un échantillon observé de taille N de la loi, c'est-à-dire un vecteur de taille N contenant les réalisations de variables aléatoires $X_1, ..., X_N$ mutuellement indépendantes et suivant toutes cette loi.
- Vérifier la pertinence et/ ou interpréter graphiquement des simulations obtenues.

1.2 Bibliothèques et librairies utilisées

Nous avons besoin, pour toute la suite de cette feuille, à importer les bibliothèques et libraires nécessaires à chaque situation :

`numpy , numpy.random , matplotlib.pyplot :`

On les importent de la façon suivante :

```
from numpy.random import *
import numpy.random as rd
import matplotlib.pyplot as plt
```

Dans ces librairie on trouve la liste des commandes Python **exigibles aux concours pour ce chapitre :**

- Dans la librairie `numpy.random` : `rd.random, rd.binomial, rd.randint, rd.geometric, rd.poisson`
- Dans la librairie `matplotlib.pyplot : plt.hist,plt.bar, plt.show`.

2 Simulation d'une variable aléatoire

2.1 La fonction rd.random( ) ou rd.rand( )

2.1.1 Définition

1. La commande `rd.random()`ou `rd.rand()` simule la loi uniforme continue $\mathscr{U}([0,1[)$:
   elle renvoie donc un nombre aléatoire de $[0,1[$. Plus précisément, si $p \in [0;1]$, la probabilité que le nombre renvoyé soit inférieure ou égale à p (ou dans un intervalle de longueur$p$) vaut exactement p.
2. La commande `rd.random(r)` ou `rd.rand(r)` simule r réalisations de la loi $\mathscr{U}([0,1[)$ et renvoie le résultat sous la forme d'un **vecteur ligne de taille** r.
3. La commande `rd.random([r,s])` ou `rd.rand([r,s])`simule $r \times s$ réalisations de la loi $\mathscr{U}([0,1[)$ sous la forme d'une **matrice de**$\mathscr{M}_{r,s}(\mathbb{R})$.

2.1.2 Propriété

1. Pour tout $(a, b) \in [0, 1]^2$ tel que $a < b$, la probabilité que le réel renvoyé par `rd.random()` soit compris (strictement ou non) entre a et b vaut $b - a$.
2. Pour tout $p \in [0, 1]$, la probabilité que le réel renvoyé par `rd.random()` soit inférieur (strictement ou non) à p vaut p.

2.2 Simulation d'un évènement de probabilité p.

2.2.1 Définition

Pour tout $p \in [0, 1]$, l'instruction `rd.random()<=p` renvoie un booléen qui prend la valeur True avec la probabilité p et la valeur False avec la probabilité $1 - p$. Cette instruction permet de simuler un événement de probabilité p.

2.2.2 Remarque.

Pour simuler un événement de probabilité p, on pourra indifféremment utiliser l'une ou l'autre des instructions suivantes : `rd.random()<p` ou `rd.random()<=p`.

2.2.3 Propriété

Si u est un vecteur dont les composantes sont des booléens, alors :

1. la commande `np.sum(u)` renvoie le nombre de booléens qui ont pris la valeur `True` ;
2. la commande `np.mean(u)` renvoie la proportion de booléens qui ont pris la valeur `True` .

2.3 Exercice

Exercice 1

1. Ecrire une fonction qui simuler la variable X : le lancer d'une pièce de monnaie tel que $P(X = 1) = \frac{1}{5}$ et $P(X = 0) = \frac{4}{5}$.
2. Simuler un échantillon de taille N = 10000 de la loi X, puis vérifier que la valeur moyenne de cet échantillon est cohérente avec ce qu'on attend.

Exercice 2

On dispose de deux lots identiques de $n \geqslant 2$ boules distinctes. On désigne par U_n une urne contenant ces $2n$ boule. On veut simuler l'expérience suivante :

on tire simultanément deux boules, si les boules ne sont pas identiques, elles sont remises dans l'urne, et on recommence jusqu'à deux boules identiques.

On note T_n le nombre de double-tirage effectué. Les deux boules sont alors enlevées, il reste une urne U_{n-1} contenant deux lots de $n - 1$ boules distinctes. On recommence, jusqu'à vider l'urne.

1. Ecrire une fonction qui simule des tirages dans U_m jusqu'à l'obtention de deux boules identiques et affecte à Y le nombre de doubles-tirages effectués, c'est-à-dire la valeur de T_m .
2. Il est clair que le nombre de tirage de deux boules effectués jusqu'à vider l'urne est $Z = T_n + T_{n-1} + ... + T_1$.
   En utilisant la question 1 écrire une fonction qui simule, une réalisation de la variable aléatoire Z et affiche sa valeur.

Exercice 3

Le score d'un joueur lors d'un lancer de fléchette est modélisé par une variable aléatoire Xtelle que $X(\Omega) = \{0, 2, 5, 10\}$ et $P(X = 0) = \frac{1}{5}, P(X = 2) = \frac{1}{2}, P(X = 5) = \frac{1}{5},$ et $P(X = 10) = \frac{1}{10}$

1. Calculer l'espérance et la variance de X.
2. Écrire alors une fonction `simul_X()` qui renvoie une simulation de la variable X.
3. Simuler un échantillon de taille N = 10000 de la loi X, puis vérifier que la valeur moyenne de cet échantillon est cohérente avec ce qu'on attend.

3 Simulation des lois discrètes usuelles

3.1 Loi uniforme

Soient n et m deux entiers tels que n < m. On a :

$$\text{si } U \hookrightarrow U([0,1[), \text{ alors } V = n + \lfloor (m-n)U \rfloor \hookrightarrow U(\llbracket n, m-1 \rrbracket).$$

Exercice 4

1. Écrire une fonction `uniforme(n,m)` simulant la loi $U(\llbracket n, m-1 \rrbracket)$.
2. Ecrire une fonction `uniforme(n,m,N)`afin qu'elle renvoie un vecteur contenant N réalisations indépendantes de la loi $U(\llbracket n, m-1 \rrbracket)$.

3.2 Loi de Bernoulli

Pour simuler une loi de Bernoulli, on procèdera comme évoqué plus haut : on tire au hasard un nombre dans l'intervalle [0, 1] avec la fonction `random()`. On a alors deux cas :

— si `rd.random()<=p`, ce qui arrive avec une probabilité de p, on renvoie la valeur 1 ;
— si `rd.random()>`p, ce qui arrive avec une probabilité de $1 - p$, on renvoie 0.

Exercice 5

1. Ecrire une fonction afin qu'elle simule une loi de Bernoulli de paramètre p.
2. Écrire une fonction `Bernoulli(p,N)` renvoyant un vecteur contenant N réalisations indépendantes de la loi `B(p)`.

3.3 Loi binomiale

Soit $n \in \mathbb{N}$. Soit $p \in [0, 1]$. Si $X_1, ..., X_n$ sont n variables aléatoires mutuellement indépendantes suivant toutes la loi n $\mathscr{B}(p)$, alors $\sum_{i=1}^{n} X_i \hookrightarrow \mathscr{B}(n, p)$.

Exercice 6

1. Écrire une fonction `binomiale(n,p)` simulant la loi $\mathscr{B}(n, p)$.
2. Écrire une fonction `Binomiale(n,p,N)` donnant un échantillon de taille Nde la loi $\mathscr{B}(n, p)$.

3.4 Loi géométrique

Soit $p \in [0,1]$. Si X est le rang du premier succès lors de la répétition d'épreuves de Bernoulli indépendantes, toutes de probabilité de succès p, alors $X \hookrightarrow \mathscr{G}(p)$.

Exercice 7

1. Écrire une fonction `geom(p)` simulant la loi G (p).
2. Écrire une fonction `GEOM(P,N)` afin de simuler un échantillon de taille N de la loi G (p).
3. Simuler un échantillon de taille N = 10000 de la loi G (0.2), puis vérifier que la valeur moyenne de cet échantillon est cohérente avec ce qu'on attend.

4 Les simulations prédéfinit par Python

On importe la bibliothèque `numpy.random` en écrivant l'une ou l'autre des instructions suivantes (on privilégiera la deuxième) :

```
from numpy.random import *
import numpy.random as rd
```

Les instructions pour simuler les lois usuelles sont :

1. La commande `rd.randint(n)` simule la loi uniforme sur $[\![0, n-1]\!]$ avec $n \in \mathbb{N}$.
2. La commande `rd.randint(a,b)` simule la loi uniforme sur $[\![a, b-1]\!]$ avec $a < b$.
3. La commande `rd.binomial(n,p)` simule la loi binomiale $\mathcal{B}(n,p)$.
4. La commande `rd.geometric(p)` simule la loi géométrique $\mathcal{G}(p)$.
5. La commande `rd.poisson(lambda)` simule la loi de Poisson $\mathcal{P}(\lambda)$.

Remarques.

1. On peut simuler la loi de Bernoulli de paramètre$p$ en écrivant l'instruction `rd.binomial(1,p)`.
2. On peut obtenir r simulations d'une loi usuelle sous la forme d'un vecteur ligne de $r$composantes.
3. On peut obtenir $r \times s$ simulations sous la forme d'une matrice de $\mathscr{M}r, s(\mathbb{R})$. :
   — `rd.geometric(p,r)` renvoie un vecteur contenant r simulations de la loi géométrique $\mathscr{G}(p)$;
   `rd.poisson(lambda,[r,s])` renvoie $r \times s$ simulations de la loi de Poisson de paramètre λ.

On peut tester les deux instruction suivantes :

```
In[1]: rd.randint(1,7,10)
Out[1]:
In[2]: rd.poisson(5,[3,3])
Out[2]:
```

5 Représentations graphiques

Soit X une variable aléatoire discrète. Supposons qu'on dispose d'une fonction Loi permettant de simuler la loi de X. Pour interpréter les simulations, on peut utiliser des représentations graphiques. Pour cela, on procèdera comme suit :

1. on crée un échantillon de taille N c'est-à-dire un vecteur ligne x contenant N réalisations de la fonction Loi.
2. on compare graphiquement les fréquences empiriques obtenues (c'est-à-dire grâce à l'échantillon) avec les probabilités théoriques pour vérifier la cohérence de la simulation.

Dans le cas de simulation de variables aléatoires discrètes, on va comparer plus particulièrement :

1. le diagramme en bâtons des fréquences de notre échantillon de taille N ;
2. le diagramme en bâtons des probabilités théoriques $P(X = k)$ pour $k \in X(\Omega)$.

Dans ce cas on aura le théorème suivant :

Théorème d'Or de Bernoulli

Pour N« Assez grand », le diagramme en bâtons des fréquences de l'échantillon doit être proche de celui des probabilités théoriques.

5.1 Commandes Python

On suppose avoir importé la bibliothèque `matplotlib.pyplot` à l'aide de l'instruction :

```
import matplotlib.pyplot as plt
```

Diagramme en bâtons des probabilités théoriques

Pour dessiner le diagramme en bâtons des probabilités théoriques, on définit x le vecteur contenant (une partie de) $X(\Omega)$ et y le vecteur contenant les probabilités théoriques correspondantes. On utilise alors la commande suivante .

5.1.1 Définition.

Si x et y sont des vecteurs de même taille, `plt.bar(x,y)` trace le diagramme en bâtons d'abscisse x et d'ordonnée y.

Histogramme

Pour dessiner le diagramme en bâtons des fréquences de l'échantillon x, il nous faut trier notre série par modalités/fréquences, puis tracer le diagramme en bâtons associé.

5.1.2 Définition.

Si x est un vecteur contenant une série statistique et c un vecteur contenant les classes choisies, la commande `plt.hist(x,c)` dessine l'histogramme associé à la série statistique x triée selon les classes définies par c.

5.1.3 Méthode. Comment tracer le diagramme en bâtons des fréquences ?

Pour tracer le diagramme des fréquences d'un échantillon x (qu'on suppose à valeurs entières), on procède ainsi :

(i) on décide des modalités $m_1 < m_2 < \cdots < m_k$ qu'on souhaite représenter ;

(ii) on définit les classes $c = (m_1 - 0,5 < m_1 + 0,5 < m_2 - 0,5 < m_2 + 0,5 < \cdots < m_k - 0,5 < m_k + 0,5)$;

(iii) on dessine l'histogramme (le « diagramme en bâtons des fréquences ») à l'aide de la commande :

```
plt.hist(x,c,density='True',edgecolor='k',color='...', label="...")
```

où l'on a ajouté les options de tracé suivantes **(non exigibles)** :

- normalisation des rectangles (la surface totale vaut 1) : density='True'
- contours des rectangles en noir : edgecolor='k'
- couleur des rectangles : color='...' (mettre le nom de la couleur en anglais)
- légende associée à chaque histogramme : label="..." (mettre la légende choisie)

Remarque. Pour réaliser plusieurs graphiques dans une même fenêtre et ainsi pouvoir mieux les comparer, on peut utiliser l'instruction `plt.subplot(n,m,k)` avant chaque instruction de tracé de graphique, qui découpe la fenêtre graphique en n ligne et m colonnes, k indiquant le numéro de la colonne souhaitée pour chaque graphique.

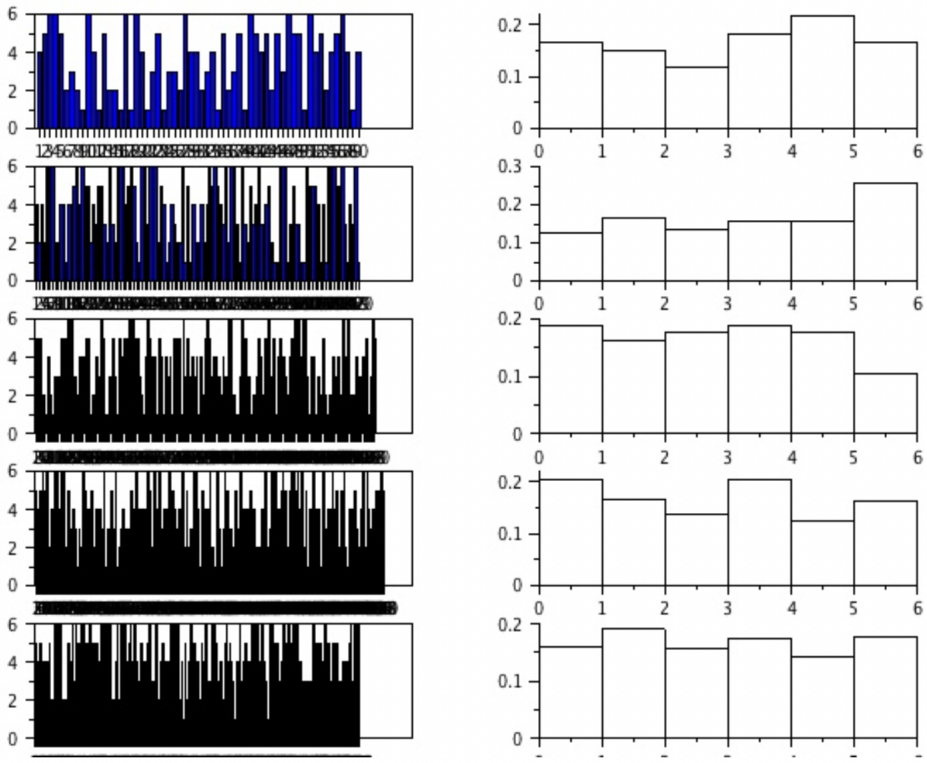
Exercice 8

Dans l'éditeur de Python, on a écrit le programme suivant :

```
nb_faces=int(input("Nombre de faces : "))
nb_lignes=5
nb_colonnes=2
for i in range(1, nb_lignes+1):
    nb_lancers=20*i
    x=np.arange(1,nb_lancers+1 )
    y=np.floor(6*rd.random(nb_lancers))+1
    plt.subplot ( nb_lignes , nb_colonnes , nb_colonnes*(i-1)+1)
    plt.bar(x,y)
    plt.subplot ( nb_lignes , nb_colonnes , nb_colonnes*(i-1)+2)
    plt.hist(y, range = (0, 7), bins = 7, color = 'yellow',edgecolor = 'red',density = T
```

1) Expliquer ce que fait ce programme ?

2) Le programme ci-dessus donne en sortie les graphes suivants. Interpréter les graphiques.



3) Modifier le programme afin qu'il n'affiche plus que les histogrammes des fréquences (et plus les résultats des lancers) pour un nombre de lancers donné successivement par l'utilisateur et répartis sur un tableau 3 lignes et 3 colonnes.