

TP5 : Simulation de v.a.r. discrètes

Commencer par importer les bibliothèques suivantes dans chaque fichier **Python** utilisé :

```
import numpy as np
import numpy.random as rd
import matplotlib.pyplot as plt
```

On rappelle que l'on a alors accès aux commandes suivantes :

- `rd.random()`
 \hookrightarrow génère un nombre aléatoire dans $[0, 1]$, *i.e.* simule une v.a.r. $X \hookrightarrow \mathcal{U}([0, 1])$
- `rd.binomial(n,p,d)`
 \hookrightarrow simule d fois de manière indépendante une v.a.r. $X \hookrightarrow \mathcal{B}(n, p)$ (le paramètre d est optionnel)
 \hookrightarrow en choisissant $n = 1$, simule d fois de manière indépendante une v.a.r. $X \hookrightarrow \mathcal{B}(p)$
- `rd.randint(a,b,d)`
 \hookrightarrow simule d fois de manière indépendante une v.a.r. $X \hookrightarrow \mathcal{U}(\llbracket a, b - 1 \rrbracket)$ (le paramètre d est optionnel)
- `rd.geometric(p,d)`
 \hookrightarrow simule d fois de manière indépendante une v.a.r. $X \hookrightarrow \mathcal{G}(p)$ (le paramètre d est optionnel)
- `rd.poisson(l,p)`
 \hookrightarrow simule d fois de manière indépendante une v.a.r. $X \hookrightarrow \mathcal{P}(l)$ (le paramètre d est optionnel)
- `rd.choice(E,d)`
 \hookrightarrow simule un d -tirage équiprobable avec remise sur l'ensemble E où E est un tableau ou une liste (le paramètre d est optionnel)
- `rd.choice(E,d,replace=False)`
 \hookrightarrow simule un d -tirage équiprobable sans remise sur l'ensemble E où E est un tableau ou une liste (le paramètre d est optionnel)

I. Simulations de v.a.r. discrètes finies à l'aide de la fonction `rd.random()`

I.1. Un exemple où l'ensemble image $X(\Omega)$ est « petit »

Le score d'un joueur lors d'un lancer de fléchettes est modélisé par une v.a.r. X dont la loi est donnée par :

- $X(\Omega) = \{0, 2, 5, 10\}$
- $\mathbb{P}([X = 0]) = \frac{1}{5}$, $\mathbb{P}([X = 2]) = \frac{1}{2}$, $\mathbb{P}([X = 5]) = \frac{1}{5}$, $\mathbb{P}([X = 10]) = \frac{1}{10}$
- Vérifier que cela définit bien une loi de probabilité.

- Calculer $\mathbb{E}(X)$.

- Découper l'intervalle $[0, 1]$ en quatre intervalles dont les longueurs sont (dans cet ordre) : $\frac{1}{5}, \frac{1}{2}, \frac{1}{5}, \frac{1}{10}$. On notera ces intervalles I_0, I_2, I_5 et I_{10} .

- Soit $k \in \{0, 2, 5, 10\}$. Quelle est la probabilité que la commande `rd.random()` renvoie un nombre appartenant à l'intervalle I_k ?

- Compléter la fonction suivante pour qu'elle renvoie une simulation de la v.a.r. X .

```
1 def simul_X():
2     r = _____
3     if _____:
4         return 0
5     elif _____:
6         return 2
7     elif _____:
8         return 5
9     else:
10        return 10
```

I.2. Un exemple de n -lancer d'une pièce de monnaie

On désigne par n un entier naturel supérieur ou égal à 2. On lance n fois une pièce équilibrée (c'est-à-dire donnant **Pile** avec probabilité $\frac{1}{2}$ et **Face** également avec probabilité $\frac{1}{2}$), les lancers étant supposés indépendants. On note Z la v.a.r. égale à 0 si l'on n'obtient aucun **Pile** pendant ces n lancers et qui, dans le cas contraire, prend pour valeur le rang du premier **Pile**.

On a vu au TP précédent que la réalisation de l'événement P_i : « la pièce tombe sur **Pile** au i^e lancer » peut être simulée par la commande `rd.random() < 1/2`. Nous allons utiliser cette commande pour simuler de deux manières différentes la v.a.r. Z .

- Dans la fonction **Python** suivante, on simule l'expérience en procédant à une suite constituée, *a priori*, de n lancers. A chaque lancer, si on tombe sur **Pile**, alors on arrête l'expérience et on renvoie le numéro du lancer, sinon on continue les lancers. Si, à la fin des n lancers, on n'est jamais tombé sur **Pile**, alors on renvoie 0. Compléter cette fonction.

```

1 def simul_Z(n):
2     for i in range(1,n+1):
3         if _____:
4             return _____
5     return _____

```

- Dans la fonction **Python** suivante, on simule l'expérience en procédant à une suite constituée, *a priori*, d'une infinité de lancers. On effectue cette suite infinie de lancers tant qu'on obtient **Face** et on compte alors le nombre de lancers effectués pour obtenir le premier **Pile**. Si ce nombre est inférieur ou égal à n , Z prend cette valeur, sinon elle prend la valeur 0. Compléter cette fonction.

```

1 def simul2_Z(n):
2     Z = 1
3     while _____:
4         Z = Z + 1
5     if _____:
6         return _____
7     return _____

```

- A la maison : calculer, pour tout $k \in \llbracket 0, n \rrbracket$, $\mathbb{P}([Z = k])$. On traitera le cas $k = 0$ à part.

I.3. Un exemple de suite de tirages dans une urne

Une urne contient initialement deux boules rouges et une boule bleue indiscernables au toucher. On appelle « tirage » la séquence suivante :

On pioche, au hasard, une boule de l'urne, puis :

- Si la boule piochée est bleue, on la remet dans l'urne.
- Si la boule piochée est rouge, on ne la remet pas dans l'urne mais on rajoute une boule bleue.

Ainsi, le nombre de boules dans l'urne ne change pas à l'issue d'un tirage.

Pour tout entier naturel n non nul, on note Y_n la v.a.r. égale au nombre de boules rouges présentes dans l'urne à l'issue du n^{e} tirage.

- Soit $n \in \mathbb{N}^*$. Donner $Y_n(\Omega)$.

- Compléter la fonction suivante pour que, prenant en argument un entier $n \geq 1$, elle renvoie une simulation de la v.a.r. Y_n .

```

1 def simul_Y(n):
2     r = 2 # nombre de boules rouges dans l'urne
3     for k in range(n):
4         if _____:
5             if _____:
6                 r = r - 1
7     return _____

```

- On introduit, pour tout $n \geq 1$, la v.a.r. indicatrice de l'événement $[Y_n = 0]$, que l'on note T_n . Autrement dit,

$$T_n = \begin{cases} 1 & \text{si l'événement } [Y_n = 0] \text{ est réalisé} \\ 0 & \text{sinon} \end{cases}$$

- × Quelle est la loi de la v.a.r. T_n ?

- × Ecrire une fonction `simul_T` qui

- prend en argument un entier $n \geq 1$
- renvoie une simulation de la v.a.r. T_n

- On note Z la v.a.r. égale au numéro du premier tirage à l'issue duquel il n'y a plus de boules rouges dans l'urne.

× Donner $Z(\Omega)$.

× Ecrire une fonction qui simule la v.a.r. Z .

I.4. Simulation d'une v.a.r. discrète finie : le cas général

Soit X une v.a.r. discrète finie telle que $X(\Omega) = \llbracket 1, n \rrbracket$ où $n \in \mathbb{N}^*$.

Pour tout $k \in \llbracket 1, n \rrbracket$, on note $p_k = \mathbb{P}([X = k])$.

- Compléter la fonction **Python** suivante pour qu'elle prenne en paramètre la liste $L = [p_1, \dots, p_n]$ et qu'elle simule la variable aléatoire X .

```
1 def simulX(L):
2     r = rd.random()
3     S = L[0]
4     k = 1
5     while _____ :
6         S = _____
7         k = _____
8     return _____
```

II. Simulation d'une v.a.r. discrète à l'aide des lois usuelles

II.1. Une loi inattendue ?

Soit $N \geq 3$ un entier. On considère une urne qui contient $(N - 1)$ boules blanches et une seule boule noire. On effectue des tirages sans remise jusqu'à l'obtention de la boule noire et on note X la v.a.r. égale au rang du tirage où l'on obtient la boule noire.

- Donner $X(\Omega)$. La variable X est-elle finie ou infinie ?

- Ecrire une fonction `simul_X` qui
- prend en argument un entier $N \geq 3$
 - renvoie une simulation de X

- Recopier et exécuter les instructions ci-contre. Expliquer ce que fait ce script.

```
1 N = 5 # On fera varier la valeur de N
2 comptage_val_X = np.zeros(N)
3 for k in range(10000):
4     i = simul_X(N)
5     comptage_val_X[i-1] += 1
6 freq_val_X = comptage_val_X / 10000
7 plt.bar(range(1,N+1), freq_val_X)
8 plt.show()
```

- En interprétant le tracé obtenu à la question précédente, faire une conjecture sur la loi de X .

- A la maison : démontrer la conjecture précédente.

II.2. Un jeu de fête foraine

A une fête foraine, un jeu est proposé, dont la partie jouée coûte 10 euros. Le jeu consiste à lancer une pièce de monnaie jusqu'à tomber sur **Pile**. On note alors n le rang d'obtention de ce premier **Pile**. Si $n \leq 2$, le joueur ne gagne rien, tandis que si $n \geq 3$, le joueur gagne n^2 euros. On note X la v.a.r. égale au gain algébrique (positif ou négatif) du joueur.

- Donner $X(\Omega)$. La variable X est-elle finie ou infinie?

- Ecrire une fonction `simul_X` qui

- prend en argument un réel $p \in]0, 1[$ tel que la pièce tombe sur **Pile** avec probabilité p
- renvoie une simulation de X