



## TP1

### SIMULATIONS DE VARIABLES ALÉATOIRES DISCRÈTES.

#### TABLE DES MATIÈRES

1. Simulation d'une variable aléatoire avec <code>rd.rand()</code>	2
2. (Simulation de) Lois usuelles finies et discrètes	3
2.1. Loi uniforme $\mathcal{U}(\llbracket m; n \rrbracket)$	3
2.2. Loi de Bernoulli $\mathcal{B}(p)$	4
2.3. Loi binomiale $\mathcal{B}(n, p)$	4
2.4. Loi géométrique $\mathcal{G}(p)$	5
2.5. Loi de Poisson $\mathcal{P}(\lambda)$	6

## 1. SIMULATION D'UNE VARIABLE ALÉATOIRE AVEC RD.RAND( )

On commence, pour toute la suite de ce TP à importer les bibliothèques et libraires nécessaires à chaque situation :

```
1 import numpy as np
2 import numpy.random as rd
```

La commande `rd.rand( )` renvoie un nombre réel aléatoire entre 0 et 1 *suivant la loi uniforme (continue)*  $\mathcal{U}([0; 1])$ . 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  $p$ .

On **identifie** et décide de représenter donc un évènement de probabilité  $p$  par le fait que le nombre aléatoire ainsi renvoyé soit inférieur ou égal à  $p$  (ou dans un intervalle de longueur  $p$ ).

**Exercice 1.-**

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

$$P(X = 0) = \frac{1}{5}, \quad P(X = 2) = \frac{1}{2}, \quad P(X = 5) = \frac{1}{5}, \quad \text{et} \quad P(X = 10) = \frac{1}{10}.$$

1. Calculer l'espérance et la variance de  $X$ .
2. Découper l'intervalle  $[0; 1]$  en quatre intervalles où pourrait *tomber* un nombre aléatoire généré avec `rd.rand()` dans le but de simuler  $X$ . Écrire alors une fonction `simul_X()` qui renvoie une simulation de la variable  $X$ .
3. On ajoute les commandes suivantes. Les exécuter et commenter l'affichage.

```
1 def simul_X(n):
2     S=[]
3     for k in range(n):
4         S.append(simul_X())
5     return S
6
7 print(np.mean(simul_X(1000)))
```

**Exercice 2.-**

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 la probabilité  $1/2$  et *face* également avec la probabilité  $1/2$ ), les lancers étant supposés indépendants.

On note  $Z$  la variable aléatoire qui vaut 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.

1. Recopier et compléter la fonction suivante permettant de simuler la variable aléatoire  $Z$ .

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

2. Réécrire cette fonction en utilisant une boucle `while`.
3. Déterminer  $P(X = k)$  pour  $k \in \llbracket 0; n \rrbracket$ . (On pourra différencier deux cas selon que  $1 \leq k \leq n$  et  $k = 0$ .)

**Exercice 3.-**

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

- (a) On pioche, au hasard, une boule de l'urne, puis :
- (b) • 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 remet une boule bleue dans l'urne à sa place.

Pour tout entier naturel  $n$  non nul, on note  $Y_n$  la variable aléatoire discrète égale au nombre de boules rouges présentes dans l'urne à l'issue du  $n$ -ième tirage.

1. Compléter la fonction ci-dessous afin qu'elle simule la variable  $Y_n$ .

```

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                 ...
7     return ....

```

2. On cherche à estimer  $\lim_{n \rightarrow +\infty} P(Y_n = 0)$ . On introduit la variable  $T_n$  définie par

$$T_n = \begin{cases} 1, & \text{si } [Y_n = 0] \\ 0, & \text{sinon} \end{cases}$$

- a. Quelle est la loi de la variable  $T_n$  ?
- b. Écrire une fonction qui simule  $T_n$ .
- c. On admet que la *moyenne empirique* d'un échantillon de 1000 réalisations de  $T_n$  renvoie une estimation de  $P(Y_n = 0)$ . Écrire une suite d'instruction permettant de visualiser graphiquement l'évolution de (l'estimation de)  $P(Y_n = 0)$  en fonction de  $n$ . Que peut-on conjecturer quant à la limite ci-dessus ?
- d. Justifier que  $(Y_n = 0) \subset (Y_{n+1} = 0)$ . En déduire que la conjecture précédente se reformule comme

$$P\left(\bigcup_{k=2}^{+\infty} (Y_k = 0)\right) = 1.$$

Interpréter.

3. On note  $Z$  la variable aléatoire égale au numéro de l'épreuve amenant la dernière boule rouge.
  - a. Donner  $Z(\Omega)$ .
  - b. Écrire une fonction qui simule la variable  $Z$ .

**Remarque.** L'utilisation de la moyenne empirique pour estimer le paramètre d'une Bernoulli prendra plus de sens après avoir traité le chapitre *Estimation*.

La variable  $T_n$  s'appelle aussi variable *indicatrice* de l'évènement  $Y_n = 0$ . On aura l'occasion de travailler avec ce type de variable, dont les sujets ESSEC sont bien friands.

## 2. (SIMULATION DE) LOIS USUELLES FINIES ET DISCRÈTES

2.1. **Loi uniforme**  $\mathcal{U}(\llbracket m; n \rrbracket)$ . La commande `rd.randint(m, n+1)` renvoie une simulation d'une variable  $X \leftrightarrow \mathcal{U}(\llbracket m; n \rrbracket)$ .

### Exercice 4.-

Représenter le diagramme à bâtons dont les hauteurs des bâtons sont les valeurs **théoriques** de la loi uniforme  $\mathcal{U}(\llbracket 1; N \rrbracket)$ .

Avoir en tête les diagrammes à bâtons théoriques des lois usuelles permet d'émettre des conjectures sur la loi suivie par une variable aléatoire inconnue, que l'on sait simuler, à partir du diagramme à bâtons (ou de l'histogramme) des fréquences obtenues sur un grand échantillon de la variable.

**Exercice 5.-**

On dispose de  $n$  urnes, numérotées de 1 à  $n$ . Pour chaque  $k \in \llbracket 1, k \rrbracket$ , l'urne  $k$  est composée de boules numérotées de 1 à  $k$ . On choisit une urne au hasard puis on tire une boule dans cette urne et on note  $X_n$  la variable aléatoire qui prend la valeur du numéro de la boule piochée.

Écrire une fonction `simul_X(n)` qui simule  $X_n$ .

**Exercice 6.-**

Soit  $N$  un entier supérieur ou égal à 3. 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 variable aléatoire qui prend pour valeur le nombre de tirages nécessaires.

1. Écrire une fonction `simul_X(N)` qui renvoie une simulation de la variable  $X$ .
2. Recopier et exécuter les instructions ci-contre. Comparer avec le diagramme théorique de l'Exercice 4. Commenter, conjecturer, démontrer.

```

1 import matplotlib.pyplot as plt
2
3 N=5 # on fera varier les valeurs de N
4 freq_val_X=np.zeros(N)
5 for k in range(10000):
6     i=simul_X(N)
7     freq_val_X[i-1]+=1
8 freq_val_X=freq_val_X/10000
9 plt.bar(range(1, N+1), freq_val_X)
10 plt.show()

```

**2.2. Loi de Bernoulli  $\mathcal{B}(p)$ .****Exercice 7.-**

Écrire, à l'aide de la commande `np.rand()` une fonction `Bernoulli(p)` permettant de simuler une variable  $X \leftrightarrow \mathcal{B}(p)$ .

**2.3. Loi binomiale  $\mathcal{B}(n, p)$ .** La commande `rd.binomial(n,p)` renvoie une simulation d'une variable  $X \leftrightarrow \mathcal{B}(n, p)$ .

On peut alors très rapidement simuler une variable de Bernoulli de paramètre  $p$  avec la commande `rd.binomial(1,p)`.

**Exercice 8.-**

Écrire, à l'aide de la commande `np.rand()` et d'une boucle `for`, une fonction `Binomiale(n,p)` permettant de simuler une variable  $X \leftrightarrow \mathcal{B}(n, p)$ .

**Exercice 9.-**

On donne le code suivant. Le recopier et exécuter `fig_mystere` pour plusieurs valeurs de  $n$ . Expliquer, commenter.

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def mystere(n,k):
5     return np.prod([range(n-k+1, n+1)])/np.
6     prod(range(1, k+1))
7
8 def fig_mystere(n):
9     Y=[mystere(n,k) for k in range(0, n+1)]
10    X=[k for k in range(n+1)]
11    plt.bar(X,Y)
12    plt.show()

```

**Exercice 10.-** *Poolage sanguin.*

On étudie une méthode de détection des porteurs d'un parasite au sein d'un ensemble donné de  $N$  individus tirés au sort. La probabilité d'être porteur du parasite dans la population est  $p \in ]0; 1[$ . Les personnes sont atteintes indépendamment les unes des autres.

On dispose d'un test permettant d'établir de façon certaine qu'un échantillon de sang contient ou non le parasite, le résultat de ce test étant dit positif dans le premier cas et négatif dans le second.

Pour chacun des  $N$  individus, on possède un prélèvement sanguin. On envisage alors deux méthodes de détection :

- Première méthode : on teste un à un les  $N$  prélèvements, effectuant ainsi  $N$  tests.
- Seconde méthode (*poolage*) :
  - On fixe un entier naturel non nul  $\ell$ . On suppose que  $N$  est un multiple de  $\ell$  et on pose  $N = n \times \ell$ . On répartit alors les  $N$  prélèvements en  $n$  groupes  $G_1, G_2, \dots, G_n$ , chaque groupe  $G_i$  contenant le même nombre  $\ell$  de prélèvements. Pour chacun des groupes  $G_i$ , on extrait une quantité de sang de chacun des  $\ell$  prélèvements qu'il contient, puis on mélange ces extraits, obtenant ainsi un échantillon de sang  $H_i$ , caractéristique du groupe  $G_i$ .
  - On teste alors  $H_i$  :
    - si le test de  $H_i$  est négatif, aucun des individus au sein du groupe  $G_i$  n'est porteur du parasite. Le travail sur le groupe  $G_i$  est alors terminé ;
    - si le test de  $H_i$  est positif, on teste un à un les prélèvements de  $G_i$  pour détecter les porteurs du parasite au sein du groupe  $G_i$ .

Soient  $X$  la variable aléatoire égale au nombre de groupes  $G_i$  pour lesquels le test de  $H_i$  a été positif et  $T$  la variable aléatoire égale au nombre total de tests effectués dans la réalisation de la méthode du *poolage*.

1. Écrire une fonction `test( $\ell$ , p)` simulant le test sur un groupe de taille  $\ell$  (avec probabilité  $p$  de contamination de chaque individu). (La fonction renverra 0 ou 1 selon que le test est négatif ou non).
2. Écrire alors une fonction `poolage_sanguin(N,p, $\ell$ )` renvoyant le résultat simultané de la simulation de  $X$  et  $T$ .
3. (\*) Déterminer, de manière théorique et rigoureuse  $E(T)$ . Établir une condition entre  $\ell$  et  $p$  pour que la méthode de *poolage* soit avantageuse (en moyenne).

2.4. **Loi géométrique  $\mathcal{G}(p)$ .** La commande `rd.geometric(p)` renvoie une simulation d'une variable  $X \hookrightarrow \mathcal{G}(p)$ .

**Exercice 11.-**

Écrire, à l'aide de la commande `rd.rand( )` et d'une boucle `while` une fonction `Geometrique(p)` permettant de simuler une variable  $X \hookrightarrow \mathcal{G}(p)$ .

**Exercice 12.-**

On effectue une succession infinie de lancers indépendants d'une pièce donnant `Pile` avec probabilité  $p$ . On note  $q = 1 - p$ .

On dit que la première série est de longueur  $n \geq 1$  si les  $n$  premiers lancers ont amené le même côté de la pièce et le  $(n + 1)$ -ième l'autre côté. De même la deuxième série commence au lancer suivant la fin de la première série et se termine au lancer précédant un changement de côté.

Par exemple si les lancers donnent les résultats `FFPPPPPPFFFP`... alors la première série est de longueur 2 et la deuxième est de longueur 6.

Soient  $X_1$  et  $X_2$  les variables aléatoires égales aux longueurs de la première et deuxième série.

1. Recopier et compléter la fonction suivante, prenant en argument la probabilité  $p$  d'obtenir `Pile` et permettant de simuler la variable aléatoire  $X_1$ .

```

1 def simul_X1(p):
2     X=1
3     t_old=rd.binomial(1,p)
4     t_new=rd.binomial(1,p)
5     while .... :
6         X= ...
7         t_old= ...
8         t_new= ...
9     return X

```

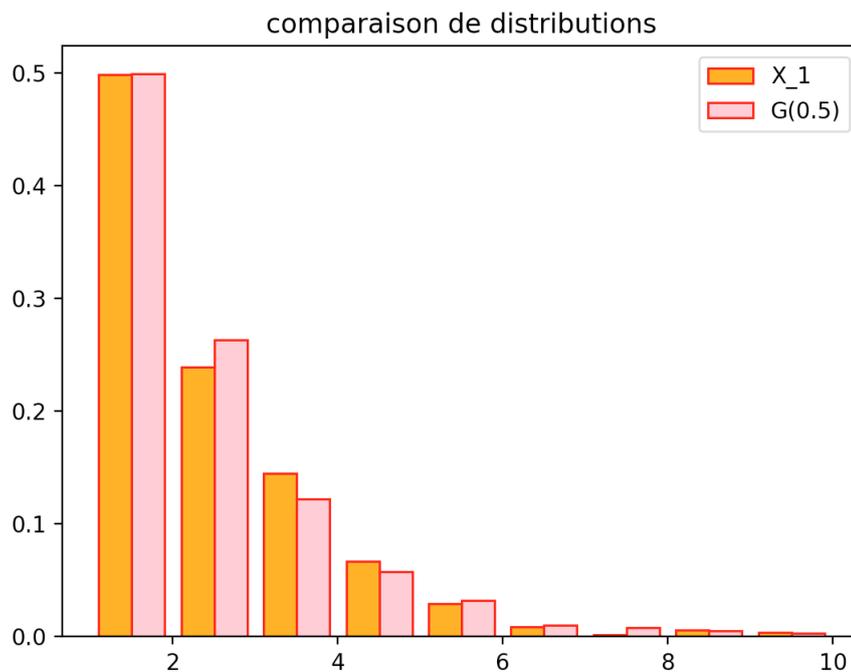
2. On ajoute le code suivant qui affiche la figure reproduite ci-après. Interpréter.

```

1 p=0.5
2 N=1000
3 x1=[simul_X1(p) for k in range(N)]
4 x2=[rd.geometric(p) for k in range(N)]
5 plt.hist([x1, x2], color = ['orange', 'pink'], density=True
6         ,
7         label = ['X_1', 'G(0.5)'])
8 plt.title('comparaison de distributions')
9 plt.legend()
10 plt.show()

```

Affichage Python : Histogramme



- Déterminer la loi de  $X_1$ . Retrouver le résultat conjecturé à la question précédente. Montrer qu'elle admet une espérance que l'on explicitera.
- Écrire une fonction d'entête  $\text{simul\_X2}(p)$  permettant de simuler la variable  $X_2$ .
- Déterminer, pour  $(k, j) \in \mathbb{N}^* \times \mathbb{N}^*$ ,  $P(X_1 = k \cap X_2 = j)$ . En déduire, à l'aide de la formule des probabilités totales, la loi de  $X_2$ .

2.5. **Loi de Poisson**  $\mathcal{P}(\lambda)$ . La commande  $\text{rd.poisson}(m)$  renvoie une simulation d'une variable  $X \leftrightarrow P(m)$ .

**Exercice 13.-**

Chaque jour, le nombre de personnes subissant un test de dépistage dans une certaine pharmacie est modélisé par une variable  $X$  suivant une loi de Poisson de paramètre  $m = 5$ . Chaque test une certaine probabilité  $p = 1/10$  d'être positif et les tests sont indépendants les uns des autres. On note  $N$  le nombre de tests positifs un jour donné.

1. Écrire une fonction permettant simuler la variable  $N$ .
2. Créer un échantillon de taille 1000 de cette variable aléatoire et le comparer, avec un histogramme, à un échantillon de même taille d'une loi de Poisson de paramètre  $mp = 0,5$ . Conjecturer.
3. (\*) En commençant par expliciter  $P_{X=n}(N = k)$  (en faisant attention sur le couplage des indices  $n$  et  $k$ ), démontrer le résultat précédemment conjecturé.