TP 11 – Simulations Variables Aléatoires

I Simulation de lois usuelles

Les lois usuelles peuvent être directement simulées grâces aux commandes suivantes :

- `rd.random()` : nombre aléatoire de [0,1[
- `rd.randint(a,b)` : loi uniforme sur [a,b[avec a et b des entiers (b exclu)
- `rd.binomial(n,p)` : loi binomiale de paramètres n et p
- `rd.geometric(p)` : loi géométrique de paramètre p
- `rd.poisson(`ℓ`)` : loi de Poisson de paramètre ℓ

Pour simuler plusieurs variables aléatoires d'un coup, on peut ajouter aux fonctions un autre argument (à la fin) qui correspond au nombre de simulations voulu.

Exercice 1 Dans chacune des situations ci-dessous, identifier la loi de la variable aléatoire considérée (parmi toutes les lois usuelles) et simuler un tirage d'une telle variable aléatoire grâce à Python.

1. On lance un dé équilibré. On note X la variable aléatoire égale au numéro obtenu. Faire une simulation de X_1.

```
rd.randint(1,7)
```

2. On réalise 10 tirages avec remise dans une urne contenant 2 boules rouges et 8 boules bleues. On note X_2 la variable aléatoire égale au nombre de boules rouges obtenues. Faire une simulation de X_2.

```
rd.binomial(10, 1/5)
```

3. On réalise une infinité de tirages avec remise dans une urne contenant 2 boules rouges et 8 boules bleues. On note X_3 la variable aléatoire égale au rang de la première boule rouge obtenue. Faire une simulation de X_3.

```
rd.geometric(1/5)
```

4. Une urne contient 10 boules numérotées de 1 à 10, indiscernables au toucher. On effectue 100 tirages dans cette urne avec remise. On note X_4 le nombre de boules numérotées 1 obtenues. Faire une simulation de X_4.

Entrée [1]:
```
rd.binomial(100,1/10)
```

5. Soit X_5 une loi de Poisson de paramètre 10. Faire une simulation de X_5.

```
rd.poisson(10)
```

6. On considère la variable aléatoire X_6 dont la loi est donnée par

$$X_6(\Omega) = \mathbb{N}^* \quad \text{et} \quad \forall k \in \mathbb{N}^*,\ \mathbb{P}([X_6 = k]) = \frac{1}{3} \times \left(\frac{2}{3}\right)^{k-1}$$

Faire une simulation de X_6.

Entrée [2]:
```
rd.geometric(1/3)
```

7. On considère la variable aléatoire X_7 dont la loi est donnée par

$$X_7(\Omega) = \{1, \dots, 10\} \text{ et } \forall k \in \{1, \dots, 10\},\ \mathbb{P}([X_7 = k]) = \frac{1}{10}$$

Faire une simulation de X_7.

Entrée [3]:
```
rd.randint(1, 11)
```

8. On considère la variable aléatoire X_8 dont la loi est donnée par

$$X_8(\Omega) = \mathbb{N} \text{ et } \forall k \in \mathbb{N},\ \mathbb{P}([X_8 = k]) = \frac{2^k \times \mathrm{e}^{-2}}{k!}$$

Faire une simulation de X_8.

Entrée [4]:
```
rd.poisson(2)
```

9. On considère la variable aléatoire X_9 dont la loi est donnée par

$$X_9(\Omega) = \{0, \dots, 50\} \text{ et } \forall k \in \{0, \dots, 50\},\ \mathbb{P}([X_9 = k]) = \binom{50}{k} \times \left(\frac{1}{4}\right)^k \times \left(\frac{3}{4}\right)^{50-k}$$

Faire une simulation de X_9.

Entrée [5]:
```
rd.binomial(50,1/4)
```

II Espérance/variance empirique d'une variable aléatoire

L'espérance **empirique** d'une variable aléatoire désigne la moyenne d'un *grand nombre* de simulations/tirages de la variable aléatoire. Ainsi, pour déterminer l'espérance empirique, on commence par réaliser plusieurs simulations d'une variable aléatoire

```
#Ici X contient 100 tirages
#d'une v.a géométrique de paramètre 100
X = rd.geometric(1/3, 100)
```

puis on utilise la commande suivante :

```
#On calcule son espérance empirique
np.mean(X)
```

Le résultat donné alors par Python doit être proche de l'espérance théorique de la variable aléatoire (cf. formules du cours).

De même, pour calculer la variance empirique d'une variable aléatoire, on pourra utiliser la commande suivante.

```
#Commande pour la variance empirique
np.mean(X)
```

Exercice 2 Reprenons les variables aléatoires $X_1, \dots, X_9$ de l'Exercice 1.

1. Donner la valeur théorique de l'espérance et de la variance des variables aléatoires X_1, X_2, X_3 et X_5 en utilisant les formules du cours.

   a) $\mathbb{E}(X_1) = \frac{6+1}{2} = \frac{7}{2}$
   b) $V(X_1) = \frac{6^2-1}{12} = \frac{35}{12}$
   c) $\mathbb{E}(X_2) = 10 \times \frac{1}{5} = 2$
   d) $V(X_2) = 10 \times \frac{1}{5} \times \left(1 - \frac{1}{5}\right) = \frac{8}{5}$
   e) $\mathbb{E}(X_3) = \frac{1}{\frac{1}{5}} = 5$
   f) $V(X_3) = \frac{1-\frac{1}{5}}{\left(\frac{1}{5}\right)^2} = 20$
   g) $\mathbb{E}(X_5) = 10$
   h) $V(X_5) = 10$

2. Écrire pour chaque variable aléatoire X_1 et X_3 un programme Python permettant de calculer et d'afficher la valeur de son espérance et variance empirique. Vérifier que les valeurs trouvées sont proches des valeurs théoriques calculées à la question précédente.

Entrée [6]:
```
#Bcp de simulations de X1
X1 = rd.randint(1, 7, 10000)

#Calcul de l'espérance empirique
espemp = np.mean(X1)
print('La moyenne empirique de X1 est', moyemp1)

#Calcul de la variance empirique
espemp = np.var(X1)
print('La variance empirique de X1 est', moyemp1)
```

Out [6]:

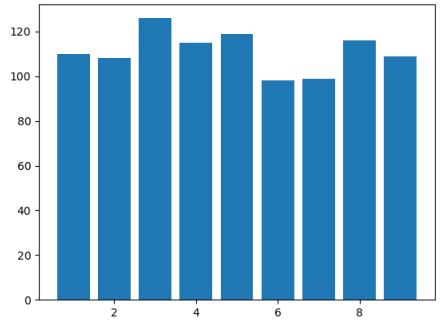
Entrée [7]:
```
#Bcp de simulations de X3
X3 = rd.geometric(1/5, 10000)

#Calcul de l'espérance empirique
espemp = np.mean(X3)
print('La moyenne empirique de X3 est', moyemp1)

#Calcul de la variance empirique
espemp = np.var(X3)
print('La variance empirique de X3 est', moyemp1)
```

Out [7]:

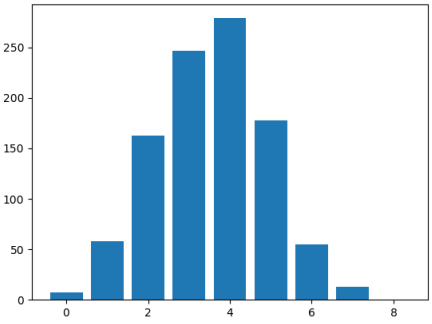
III Diagramme d'une variable aléatoire

Une variable aléatoire peut être représentée sous forme d'un diagramme en bâtons. Pour son diagramme théorique, chaque bâton correspond à une valeur possible de la variable aléatoire et la hauteur du bâton représente la probabilité associée à cette valeur. Cette distribution de probabilité peut se retrouver aussi empiriquement. Dans ce cas, on va réaliser un grand nombre de tirages de la variable aléatoire. Alors, chaque bâton correspond toujours à une valeur possible de la variable aléatoire mais cette fois-ci la hauteur correspond au nombre de fois où la valeur a été obtenue lors des tirages. Bien sur, ce diagramme empirique ressemble fortement au diagramme théorique.

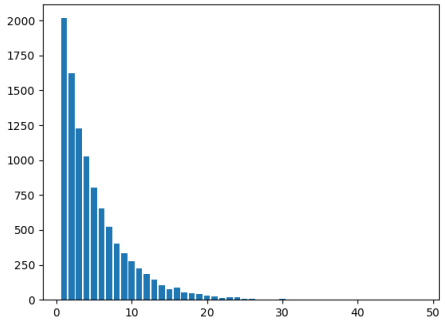
Exercice 3 Reconnaître à partir de son diagramme la loi usuelle simulée.



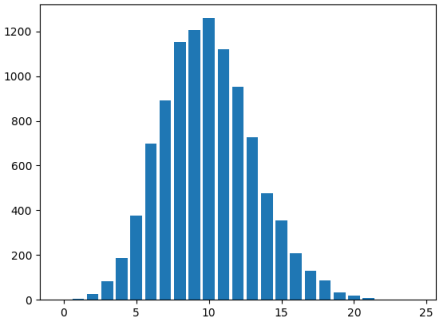
(a) Diagramme d'une loi uniforme



(b) Diagramme d'une loi binomiale



(a) Diagramme d'une loi géométrique



(b) Diagramme d'une loi de Poisson

IV Simulation d'une expérience aléatoire (à la main)

Exercice 4 On considère une urne contenant 5 boules rouges (numérotées 1,2,3,4,5) et 3 boules jaunes (numérotées 6,7,8). On réalise 7 tirages **avec remise** dans cette urne. On note X la variable aléatoire qui vaut 0 si aucune boule jaune n'est tirée et, sinon, est égale au **numéro du tirage** où l'on a obtenu la première boule jaune. Écrire une fonction `SimulationX()` qui réalise une simulation de X.

```
def SimulationX():
    #Au debut, la v.a est initialisée à 0
    X = 0
    #On effectue 7 tirages dans l'urne avec remise
    for k in range(1, 8):
        tirage = rd.randint(1,9)
        #si la boule est jaune alors...
        if tirage >= 6 :
            #...on recupère le num du tirage
            X = k
            #... et on s'arrête
            return X
```

V Pour aller plus loin

Exercice 5 **Réalisation d'un diagramme pour une variable aléatoire finie** – Soit X_1 une variable aléatoire suivant une loi uniforme sur $\{1, \dots, 9\}$.

1. Réaliser 1000 simulations de X_1. On appelera la liste créée `X1`.

```
X1 = rd.randint(1,10, 1000)
```

2. Définir la liste x des valeurs de X_1 (abscisses du diagramme).

```
x = range(1,10)
```

3. Écrire une fonction `nombre(L,k)` qui, étant donnée une liste L et un nombre k, renvoie le nombre de fois où k apparaît dans L.

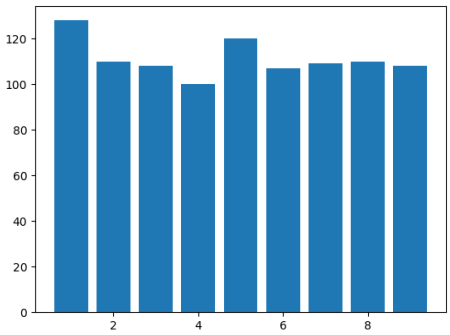
```
#Version 1
def nombre(L,k):
    c = 0
    for x in L:
        if x==k:
            c=c+1
    return c

#Version 2
def nombre(L,k):
    c=0
    for i in range(len(L)):
        if L[i] == k :
            c=c+1
    return c
```

4. Compléter alors le programme suivant pour définir la liste y des hauteurs des barres.

```
y = [0]*len(x) # liste nulle de même taille que x

for i in range(len(y)):
    y[i] = nombre(X1, i+1) #attention au decalage
print(y)
```

5. Enfin, on trace le diagramme en barres.

```
plt.bar(x,y)
plt.show()
```



Exercice 6 **Réalisation d'un diagramme pour une variable aléatoire discrète à support infini** – On va réaliser des simulations d'une variable aléatoire discrète puis représenter le diagramme en barres des valeurs obtenues :

- en abscisses, les valeurs : intervalle [mininum théorique ; maximum observé]
- en ordonnées, le nombre d'occurrences de chaque valeur lors des simulations.

1. (a) Soit X une variable aléatoire de loi géométrique de paramètre p. Réaliser N simulations de X. On commencera par $p = 0,2$ et $N = 10000$ et on pourra ensuite faire varier ces valeurs.

```
p = 0.2
N = 10000
# simulations
X = rd.geometric(p, 10000)
```

   (b) Les abscisses sont les entiers du minimum théorique au maximum observé. La liste des abscisses n'est pas définie en Python comme une liste mais comme un tableau (ce qui permet des manipulations plus souples). Pour cela, on utilise la commande `np.arrange(a,b)` qui permet de construire une liste de nombres compris entre `a` et `b` (exclu). On pourra aussi utiliser la commande `max(L)` qui permet de récupérer la valeur maximale de la liste `L`.

```
abscisses = np.arange(1,max(X)+1)
```

   (c) Passons aux barres.

barres = [nombre de 1, nombre de 2, nombre de 3, ...]

Ainsi,

barres[i] = nombre de i+1

On commence par créer une liste de taille correcte [0,0,...,0] puis on va compter le nombre d'occurrences de chaque valeur.

```
barres = abscisses * 0 # liste de 0 de même taille que abscisses
# on parcourt les simulations X
for valeur in X :
    # on ajoute une occurrence de cette valeur au bon endroit
    # dans la liste des barres
    i = valeur
    barres[i-1] = barres[i-1] + 1
print(barres)
```

   (d) On termine par le diagramme en barres avec matplotlib.

```
import matplotlib.pyplot as plt
plt.bar(abscisses,barres)
plt.show()
```

2. Faire de même avec une loi de Poisson de paramètre 10. On commencera par un schéma.

```
Y = rd.poisson(10, 10000)
abscisses = np.arange(0,max(Y)+1)
barres = abscisses * 0 # liste de 0 de même taille que abscisses
for valeur in Y :
    # on ajoute une occurrence de cette valeur au bon endroit
    # dans la liste des barres
    i = valeur
    barres[i] = barres[i] + 1
import matplotlib.pyplot as plt
plt.bar(abscisses,barres)
plt.show()
```

Exercice 7 **Ecricome 2023** – Soit n un entier naturel non nul. Une urne contient n boules indiscernables au toucher et numérotées de 1 à n. On tire une boule au hasard dans l'urne. Si cette boule tirée porte le numéro k, on place alors dans une seconde urne toutes les boules suivantes : une boule numérotée 1, deux boules numérotées 2, ..., jusqu'à k boules numérotées k. Les boules de cette deuxième urne sont aussi indiscernables au toucher. On effectue alors un second tirage d'une boule dans la seconde urne. Et on note X la variable aléatoire égale au numéro de la première boule tirée et on note Y la variable aléatoire égale au numéro de la deuxième boule tirée.

1. Écrire une fonction en langage Python, nommée `seconde_urne`, prenant en entrée un entier naturel `k` non nul, et renvoyant une liste contenant 1 élément valant 1, 2 éléments valant 2,..., j éléments valant j,..., jusqu'à k éléments valant k. Par exemple, l'appel de `seconde_urne(4)` renverra `[1,2,2,3,3,3,4,4,4,4]`.

Entrée [8]:
```
def seconde_urne(k):
    L = [1]
    for j in range(2,k+1):
        for i in range(j):
            L.append(j)
    return L
```

2. Recopier et compléter la fonction en langage Python suivante pour qu'elle prenne en entrée un entier naturel `n` non nul, et qu'elle renvoie une réalisation du couple de variables aléatoires (X,Y).

Entrée [9]:
```
import numpy.random as rd

def simul_XY(n):
    X = rd.randint(1,n+1)
    urne2 = seconde_urne(X)
    nb = len(urne2)
    i = rd.randint(0, nb)
    Y = urne2[i]
    return X, Y
```

Exercice 8 **Ecricome 2016** – Dans tout l'exercice, X et Y sont deux variables aléatoires définies sur le même espace probabilisé et à valeurs dans $\mathbb{N}$. On dit que les deux variables X et Y sont échangeables si :

$$\forall (i,j) \in \mathbb{N}^2, \qquad \mathbb{P}([X=i] \cap [Y=j]) = \mathbb{P}([X=j] \cap [Y=i])$$

Résultats préliminaires.

1. On suppose que X et Y sont deux variables indépendantes et de même loi. Montrer que X et Y sont échangeables.

2. On suppose que X et Y sont échangeables. Montrer, à l'aide de la formule des probabilités totales que :
$$\forall i \in \mathbb{N}, \qquad \mathbb{P}(X=i) = \mathbb{P}(Y=i)$$
On pourra utiliser le système complet d'évènement donné par $(Y=j)_{j\in\mathbb{N}}$.

Étude d'un exemple. Soient n, b et c trois entiers strictement positifs. Une urne contient initialement n boules noires et b boules blanches. On effectue l'expérience suivante, en distinguant trois variantes.

- On pioche une boule dans l'urne. On définit X la variable aléatoire qui vaut 1 si cette boule est noire et 2 si elle est blanche.
- On replace la boule dans l'urne et :
  - Variante 1 : on ajoute dans l'urne c boules de la même couleur que la boule qui vient d'être piochée.
  - Variante 2 : on ajoute dans l'urne c boules de la couleur opposée à celle de la boule qui vient d'être piochée.
  - Variante 3 : on n'ajoute pas de boule supplémentaire dans l'urne.
- On pioche à nouveau une boule dans l'urne. On définit Y la variable aléatoire qui vaut 1 si cette seconde boule piochée est noire et 2 si elle est blanche.

3. (a) Compléter la fonction Python suivante, qui simule le tirage d'une boule dans une urne contenant b boules blanches et n boules noires et qui retourne 1 si la boule tirée est noire, et 2 si la boule tirée est blanche. *On pourra introduire une numérotation des boules si nécessaire.*

> Pour faciliter le codage, dans l'urne contenant n boules noires et b boules blanches, on numérote les n boules noires de 1 à n et les b boules blanches de $n+1$ à $n+b$.

Entrée [10]:
```
import numpy.random as rd

def tirage(b,n):
    boule = rd.randint(1, n+b+1)
    if boule >= n+1 :
        res = 2
    else :
        res = 1
    return(res)
```

(b) Compléter la fonction suivante, qui effectue l'expérience étudiée avec une urne contenant initialement b boules blanches, n boules noires et qui ajoute éventuellement c boules après le premier tirage, selon le choix de la variante dont le numéro est `variante`. Les paramètres de sortie sont : `x`, une simulation de la variable aléatoire X et `y`, une simulation de la variable aléatoire Y.

Entrée [11]:
```
def experience(b,n,c, variante):
    x = tirage(b,n)
    if variante ==1 :
        if x == 1 : #la 1ere boule tirée est noire
            n=n+c #variante1 : on rajoute c boules noires
        else : #la 1ere boule tirée est blanche
            b=b+c  #variante1 : on rajoute c boules blanches
    elif variante == 2 :
        if x == 1 : #la 1ere boule tirée est noire
            b=b+c #variante2 : on rajoute c boules blanches
        else : #la 1ere boule tirée est blanche
            n=n+c  #variante1 : on rajoute c boules noires
    #variante3: le nbre de boules ne change pas
    y = tirage(b,n)
    return(x,y)
```

4. On se place dans cette question dans le cadre de la variante 1.

   (a) Donner la loi de X.

   (b) Déterminer les probabilités suivantes :

   $$\forall (i,j) \in \{1,2\}^2, \qquad \mathbb{P}([X=i] \cap [Y=j])$$

   (c) Déterminer la loi de Y.

   (d) Montrer que X et Y sont échangeables mais ne sont pas indépendantes.