

TP n° 4 : Simulations d'expériences aléatoires

On révisé dans ce TP les techniques de simulation d'expériences aléatoires classiques. Rappelons que le module `numpy.random` permet de simuler des expériences aléatoires. On l'importera par la commande `import numpy.random as rd`.

Les principales commandes dont nous aurons besoin sont :

- `rd.random()`, qui renvoie un nombre aléatoire entre 0 et 1 (selon une loi uniforme). Si $p \in]0, 1[$, `rd.random() < p` est vrai avec une probabilité p et faux avec une probabilité $1 - p$.
- `rd.randint(a,b)`, qui renvoie un nombre entier aléatoire entre a inclus et b exclu suivant une loi uniforme.

Il existe également des commandes permettant de simuler des variables aléatoires suivant les lois usuelles (binomiale, géométrique, de Poisson), mais le but de ce TP est de simuler certaines de ces lois à partir de commandes élémentaires.

1 Pile ou Face

1.1 Loi de Bernoulli

Compléter la fonction suivante afin qu'elle renvoie 1 avec une probabilité p et 0 avec une probabilité $1 - p$ (simulant ainsi une variable aléatoire suivant une loi de Bernoulli) :

```
def tirageBer(p):  
    a = rd.random()  
    if a < p :  
        res = ...  
    else :  
        ...  
    return res
```

On voit aussi la variante suivante (après avoir exécuter la commande `import numpy as np`) :

```
def tirageBer2(p):  
    res = np.floor(rd.random()+p)  
    return res
```

1.2 Loi géométrique

1. Compléter le programme suivant pour que N soit égal au premier rang où on obtient Pile, sachant que la probabilité d'obtenir Pile à chaque tirage est $p = 1/3$.

```
p=1/3  
tir = tirageBer(p) # premier tirage  
N=1 # nombre de tirages effectués  
while tir .....  
    N = .....  
    tir = ... # valeur du N-ème tirage  
  
print(N)
```

On verra la variante suivante, où l'on ne se sert pas de la fonction `tirageBer` :

```
p=1/3
N=1 # nombre de tirages effectués
while rd.random()>p :
    N = N+1

print(N)
```

2. Ecrire une fonction `tirageGeom` en adaptant le programme précédente, prenant en entrée p et donnant comme résultat N .
3. Compléter le programme suivant pour qu'il détermine une valeur approchée de l'espérance de la variable aléatoire de loi géométrique simulée :

```
p = 1/3
S=0
for i in range(1000):
    Ni = ..... # valeur de la v.a. simulée au i-ème tirage
    S = S + .....

moy = ...
print( moy )
```

1.3 Loi binomiale

1. Compléter la fonction `tirageBinom` suivante pour qu'elle simule cette expérience aléatoire et renvoie le nombre de Piles obtenus, avec comme paramètres la probabilité p d'obtenir Pile à chaque tirage et le nombre de tirages est n .

```
def tirageBinom(n,p):
    NP=0
    for i ...
        tir = tirageBer(p)
        NP = .....

    return NP
```

2. Ecrire un programme déterminant une valeur approchée de l'espérance de la variable aléatoire simulée avec $n = 10$ et $p = 1/3$.

1.4 Variante

Exercice 1. – On tire à Pile ou Face jusqu'à obtenir la combinaison PF. On cherche à simuler cette expérience et à renvoyer le nombre de lancers nécessaires. On représentera Pile par 1 et Face par 0, la probabilité d'obtenir pile étant notée p .

1. Compléter la fonction suivante, simulant une réalisation de l'expérience et renvoyant le nombre de tirages nécessaires.

```
def pileFace(p):
    tir_prec = tirageBer(p) # (n-1)-ème tirage
    tir = tirageBer(p) # n-ème tirage
    n=2 # nombre de tirages effectués
    while not(tir_prec == ... and ... )
        n = ...
        tir_prec = ...
        tir = ...
    return ...
```

2. Déterminer une valeur approchée de l'espérance du nombre de tirages effectués (avec $p = 0,4$).

2 Tirage dans des urnes

Exercice 2. – On dispose d'une urne contenant deux boules noires et deux boules blanches. On pioche deux boules successivement sans remise. On note X le nombre de boules noires piochées.

1. Compléter la fonction suivante afin qu'elle simule cette expérience aléatoire en renvoie la valeur de X .

```
def simulX() :
    nb = 2 # nombre de boules noires dans l'urne
    a1 = rd.random() # premier tirage
    if a1 < ... : # tirage d'une boule noire
        nb = ...
        a2 = rd.random() # deuxième tirage
        if a2 < ... :
            nb = ...
    else : # cas où le premier tirage a donné une boule blanche
        a2 = rd.random()
        if a2 < ... :
            nb = ...
    X =
    return X
```

2. Écrire un programme qui permet de calculer une valeur approchée de l'espérance de X . Vérifier cette estimation par le calcul (on déterminera la loi de X)
3. Pour vérifier la correction de notre fonction simulant cette variable aléatoire, on peut également tracer le diagramme en bâton représentant la loi empirique obtenue en faisant un

nombre important de simulations. Pour cela, au lieu de calculer la moyenne empirique, on va calculer les fréquences empiriques.

On donne tout d'abord la méthode pour tracer un diagramme en bâtons :

```
import matplotlib.pyplot as plt
X=[x for x in range(5)]
Y=[1,2,1,2,1]
plt.bar(X,Y,width=0.1,color='b')
```

Le programme suivant (à compléter) est destiné à tracer le diagramme en bâton de la loi empirique obtenue lors d'une simulation.

```
N = 1000 # nombre de simulations
liste_valeursX = [0,1,2]
liste_occ = [0,0,0] #liste_occ[k] doit contenir le nombre
                    #de fois que k apparaît lors des simulations
for i in range(N):
    Xi = ..... # valeur de la v.a. simulée au i-ème tirage
    liste_occ[Xi] = ...

liste_freq = [a/N for a in liste_occ]
plt.bar(liste_valeursX,liste_freq,width=0.1,color='b')
```

3 Compléments

Exercice 3. – On considère l'expérience suivante : on dispose d'une urne dans laquelle on a placé une boule verte, une boule blanche et une boule noire. On fait des tirages avec remise dans l'urne en remettant à chaque fois la boule tirée plus une autre de la même couleur. On arrête l'expérience lorsque l'on a 5 boules d'une couleur dans l'urne. On note V , B et N les nombres de boules de chaque couleur dans l'urne à la fin de l'expérience.

1. Compléter le programme suivant pour qu'il simule l'expérience :

```
V=1
B=1
N=1
while ..... and ..... and .....
    Tirage=rd.random()
    if Tirage < .....
        V=V+1
    elif Tirage < .....
        B = B+1
    else :
        N=N+1

print('On a V=',V)
print('On a B=',B)
print('On a N=',N)
```

2. Modifier le programme pour que l'on réalise l'expérience 100 fois et que l'on affiche la moyenne obtenue pour V , B et N .

Exercice 4. – On considère une urne contenant une boule blanche et une boule noire. On effectue des tirages successifs de deux boules dans l'urne suivant la règle suivante : si on pioche deux fois la même couleur, on enlève les deux boules piochées, sinon on les remet en ajoutant une boule blanche et une boule noire. On répète l'expérience jusqu'à avoir fait dix tirages, ou jusqu'à avoir vidé l'urne ou qu'il n'y ait plus qu'une boule dans l'urne.

1. Ecrire un programme simulant l'expérience et affichant N et B les nombres de boules blanches et noires dans l'urne.
2. En faisant répéter l'expérience mille fois, déterminer une valeur approchée de l'espérance de B ainsi que de l'espérance de N .

Exercice 5. – On considère deux urnes contenant chacune 10 boules blanches et une boule noire. On commence par tirer dans l'urne 1. On fait des tirages sans remise si on pioche une boule blanche. Si on pioche une boule noire, on la remet et on change d'urne. On s'arrête dès qu'une urne ne contient plus de boules blanches. On note N le nombre de tirages effectués et U le numéro de l'urne dans laquelle on a pioché en dernier.

1. Ecrire un programme simulant l'expérience et affichant N et U .
2. En faisant répéter l'expérience mille fois, déterminer une valeur approchée de l'espérance de U ainsi que de l'espérance de N .

Exercice 6. – On considère l'expérience suivante : deux joueurs A et B jouent à Pile ou Face en parallèle avec une pièce équilibrée. On note N le rang du premier tirage pour lequel les deux joueurs ont obtenus autant de Pile l'un que l'autre.

1. Ecrire un programme simulant cette expérience et donnant la valeur N .
2. Modifier votre programme pour que l'on puisse choisir la probabilité p de faire Pile et en faire une fonction `NBTirages` prenant p en entrée.