Feuille info_25 : Simulations et couples de variables aléatoires discrètes

Ex 1 : Etude d'un couple de variables aléatoires discrètes. *Premier exemple de la feuille_Cours_13*

On lance une pièce équilibrée, jusqu'à obtenir 3 piles ou 3 faces *(pas nécessairement l'un derrière l'autre)*. On note X le nombre de lancers effectués et Y le nombre de piles obtenus.

$X \backslash Y$	0	1	2	3
3	$\frac{1}{8}$	0	0	$\frac{1}{8}$
4	0	$\frac{3}{16}$	0	$\frac{3}{16}$
5	0	0	$\frac{3}{16}$	$\frac{3}{16}$

x	3	4	5
$P(X=x)$	$\frac{1}{4}$	$\frac{3}{8}$	$\frac{3}{8}$

y	0	1	2	3
$P(Y=y)$	$\frac{1}{8}$	$\frac{3}{16}$	$\frac{3}{16}$	$\frac{1}{2}$

1) Compléter les fonctions et les explications associées suivantes :

Explication : .....

```
def nombre(a,L):                                # Renvoie le nombre de a dans L
    s =
    for ........
        ..........
        ..........
    return s
```

Explication : .....

```
def simul_exp():                                # simule l'expérience aléatoire
    L = []
    while ...................                   #  A compléter
        L.append(rd.randint(0, 1))              #  0 : pour face  et  1 : pour pile
    return L
```

Explication : .....

```
def simul_X():                                  # simule la réalisation de X
    L = simul_exp()
    return  ...................                 #  A compléter
```

Explication : .....

```
def simul_Y():                                  # simule la réalisation de Y
    L = simul_exp()
    return  ...................                 #  A compléter
```

Explication : .....

```
def simul_X_Y():                                # simule la réalisation de (X,Y)
    L = simul_exp()
    return  ...................                 #  A compléter
```

2) Ecrire un programme permettant d'estimer $E(X)$, $E(Y)$ et $E(XY)$ à partir d'une liste de simulations.
On comparera avec le résultat vu en mathématiques : $E(X) = \dfrac{33}{8}$, $E(Y) = \dfrac{33}{16}$ et $E(XY) = \dfrac{141}{16}$

3) Un candidat à l'oral propose pour estimer $E(XY)$ le programme suivant :

```
N, S = 1000, 0
for k in range(N):
    x, y = simul_X(), simul_Y()
    S += x*y
print(S/N)
```

Expliquez pourquoi il se trompe. *(C'est une erreur que beaucoup d'entre-vous ont faite dans le DS8).*

Conseil :

Pour bien simuler une expérience aléatoire, rester au plus près de ce qui est décrit dans le modèle.

Ex 2 : Test du Khi_2

Le but de cet exercice est de répondre à la question suivante :

"On lance 1000 fois un dé et on obtient les résultats suivants :

faces	1	2	3	4	5	6
effectifs	160	180	163	150	166	181

Pensez-vous que ce dé est équilibré ?"

On étudie l'expérience aléatoire : On lance 1 000 fois un dé équilibré,

on note pour chaque $i \in [\![1, 6]\!]$, X_i le nombre de face i obtenues et $C = \sum_{i=1}^{6} \frac{(X_i - N/6)^2}{N/6}$

statistique du test du khi_2.

On prendra : $N = 1000$.

Pour ceux qui veulent, un programme est donné en annexe, vous pouvez vous contenter de le commenter.

1) **Question préliminaire sur la loi du Khi_2.**

On note $Z = \sum_{k=1}^{5} Y_k^2$ où $Y_1, \ldots, Y_5$ sont 5 gaussiennes centrées réduites indépendantes.

a. Ecrire une fonction qui permet de simuler la réalisation de Z.

b. Tracer la fonction de répartition de Z à l'aide d'une longue série de simulations.

c. Comparer *(graphiquement)* votre courbe avec celle de la fonction F suivante :

```
from scipy.stats import chi2
F = chi2(df = 5).cdf
```

d. Que peut-on en conclure sur F ?

e. Pour ceux qui veulent aller plus loin, vérifier que la comparaison est identique en changeant 5 par un autre entier.

2) **Fonction de répartition de la statistique du Khi_2.**

a. Ecrire une fonction qui permet de simuler la réalisation de C.

b. Tracer la fonction de répartition de C à l'aide d'un longue série de simulations.

c. Comparer *(graphiquement)* votre courbe avec celle de la fonction F de la question 1) c.

3) **Test du Khi_2.**

On approche, alors la fonction de répartition de C par F.

(Fonction de répartition de la loi du Khi^2 avec 5 degrés de liberté)

a. Quelle valeur c_{obs} prend C avec les résultats donnés en introduction ?

b. Quelle est la probabilité de $C \geqslant c_{\text{obs}}$? *(Culture : Comment appelle-t-on cette probabilité ?)*

c. Reprendre la question avec les résultats suivants :

faces	1	2	3	4	5	6
effectifs	195	145	158	179	149	174

Explication : .....

```
import random as rd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import chi2

plt.close('all')


N = 1000
```

Explication : .....

```
def gauss():
    return rd.gauss(0, 1)
```

Explication : .....

```
def simul_Z():
    s = 0
    for _ in range(5):
        y = gauss()
        s += y**2
    return s
```

Explication : .....

```
plt.figure('Question 1')
x = np.linspace(0, 20, 200)
F_theo = chi2(df=5).cdf
plt.plot(x, F_theo(x), label="Théorique")
```

Explication : .....

```
N_simul = 10000
x = [ simul_Z() for k in range(N_simul)]
x.sort()
y = [ (k+1)/N_simul for k in range(N_simul)]
plt.plot(x, y, label="Empirique")

plt.legend()
plt.title("Comparaison des fonctions de répartition")
plt.show()
```

Explication : .....

```
plt.figure('Question 2')
x = np.linspace(0, 20, 200)
F_theo = chi2(df=5).cdf
plt.plot(x, F_theo(x), label="Khi_2(5)")
```

Explication : .....

```
def calcul_C_obs(effectifs):
    C = 0
    for xi in effectifs:
        C += (xi - N/6)**2 / (N/6)
    return C
```

Explication : .....

```
def simul_C():
    effectifs = [0]*6
    for i in range(N):
        face = rd.randint(1, 6)
        effectifs[face-1] += 1
    return calcul_C_obs(effectifs)
```

Explication : .....

```
N_simul = 10000
x = [ simul_C() for k in range(N_simul)]
x.sort()
y = [ (k+1)/N_simul for k in range(N_simul)]
plt.plot(x, y, label="C empirique")

plt.legend()
plt.title("Comparaison C vs Khi_2")
plt.show()
```

Explication : .....

```
effectifs1 = [160, 180, 163, 150, 166, 181]
c_obs1 = calcul_C_obs(effectifs1)

N_simul = 10000
s = 0
for k in range(N_simul):
    if simul_C() > c_obs1:
        s += 1
print(s/N_simul)
```

Explication : .....

```
effectifs2 = [195, 145, 158, 179, 149, 174]
c_obs2 = calcul_C_obs(effectifs2)

N_simul = 10000
s = 0
for k in range(N_simul):
    if simul_C() > c_obs2:
        s += 1
print(s/N_simul)
```