

L'objectif est l'utilisation de Python dans la modélisation d'expériences et variables aléatoires. Nous aurons besoin de la bibliothèque `numpy.random` à importer ainsi :

```
import numpy.random as rd
```

Nous aurons également besoin, pour les graphiques, d'importer :

```
import matplotlib.pyplot as plt
```

Pour représenter un histogramme de données contenues dans la liste `Lvaleurs` en fonction de la listes de bornes `Lbornes`, on utilise :

```
plt.hist(Lvaleurs,Lbornes)
```

Après la création de l'histogramme, il faut rajouter la ligne `plt.show()` pour le représenter.

Nous aurons besoin de la commande `rd.random()` qui renvoie un réel aléatoire de $]0; 1[$

Remarque

Par défaut, il s'agit d'un histogramme d'effectifs. Pour obtenir un histogramme de fréquences, on ajoute `density=True` en option dans la commande `plt.hist()`.

| SIMULER LES LOIS DE BERNOULLI, BINOMIALE ET GÉOMÉTRIQUE AVEC RANDOM

I.1 LOI DE BERNOULLI

- Écrire une fonction prenant en argument un réel $p \in]0; 1[$ et renvoyant une réalisation d'une variable aléatoire suivant la loi $\mathcal{B}(p)$.

```
1 import numpy.random as rd
2
3 def bernoulli(p):
4     if rd.random()<p:
5         return 1
6     else :
7         return 0
```

I.2 LOI BINOMIALE

- Écrire une fonction prenant en arguments un entier naturel non nul n ainsi qu'un réel $p \in]0; 1[$ et renvoyant une réalisation d'une variable aléatoire suivant la loi $\mathcal{B}(n; p)$.

```
1 import numpy.random as rd
2
3 def binomiale(n,p):
4     nb_succes=0
5     for k in range(n): #ou range(0,n) ou range(1,n+1)
6         if rd.random()<p:
7             nb_succes=nb_succes+1
8     return nb_succes
```

I.3 LOI GÉOMÉTRIQUE

- Écrire une fonction prenant en arguments un réel $p \in]0; 1[$ et renvoyant une réalisation d'une variable aléatoire suivant la loi $\mathcal{G}(p)$.

```
1 import numpy.random as rd
2
3 def geometrique_bis(p):
4     rang=1
5     while rd.random()<1-p:
6         rang=rang+1
7     return rang
```

II SIMULER LES LOIS USUELLES AVEC LES COMMANDES EXISTANTES

Des commandes à connaître :

- `rd.random(N)` renvoie N réels aléatoires de $]0; 1[$ dans un tableau 1 ligne $\times N$ colonnes
- `rd.randint(a,b)` renvoie un entier aléatoire de $\llbracket a; b \rrbracket$
- `rd.randint(a,b,N)` renvoie N entiers aléatoires de $\llbracket a; b \rrbracket$ dans un tableau 1 ligne $\times N$ colonnes
- `rd.binomial(n,p)` renvoie une réalisation d'une variable aléatoire suivant la loi $\mathcal{B}(n;p)$
- `rd.binomial(n,p,N)` renvoie N réalisations d'une variable aléatoire suivant la loi $\mathcal{B}(n;p)$ dans un tableau 1 ligne $\times N$ colonnes
- `rd.geometric(p)` renvoie une réalisation d'une variable aléatoire suivant la loi $\mathcal{G}(p)$
- `rd.geometric(p,N)` renvoie N réalisations d'une variable aléatoire suivant la loi $\mathcal{G}(p)$ dans un tableau 1 ligne $\times N$ colonnes
- `rd.poisson(lam)` renvoie une réalisation d'une variable aléatoire suivant la loi $\mathcal{P}(\lambda)$, avec $\lambda = \text{lam}$
- `rd.poisson(lam,N)` renvoie N réalisations d'une variable aléatoire suivant la loi $\mathcal{P}(\lambda)$, avec $\lambda = \text{lam}$, dans un tableau 1 ligne $\times N$ colonnes

X Attention !
Comme pour la commande `range(a,b)`, la borne de droite est exclue !

4. 4.a. Représenter l'histogramme des fréquences de 10000 réalisations indépendantes d'une variable aléatoire suivant la loi binomiale de paramètres $n = 10$ et $p = 0,3$.
- 4.b. Sur le même graphique, faire apparaître les valeurs des probabilités d'une telle loi.

```
1 import numpy.random as rd
2 import matplotlib.pyplot as plt
3 import math

4
5 n=30
6 p=0.1
7 Labs=[-0.5+k for k in range(0,n+2)]
8 plt.hist(rd.binomial(n,p,10000),Labs,density=True,edgecolor='b')
9
10 Lproba=[math.comb(n,k)*p**k*(1-p)**(n-k) for k in range(0,n+1)]
11 plt.plot(range(0,n+1),Lproba,"r+")
12
13 plt.show()
```

Remarque

Le fait que la distribution empirique observée sur un échantillon de grande taille se rapproche de la distribution théorique est assez naturel. Nous le justifierons mathématiquement dans le chapitre 14.

5. 5.a. Représenter l'histogramme des fréquences de 10000 réalisations indépendantes d'une variable aléatoire suivant la loi géométrique de paramètre $p = 0,5$.
- 5.b. Sur le même graphique, faire apparaître les valeurs des probabilités d'une telle loi.

```
1 import numpy.random as rd
2 import matplotlib.pyplot as plt
3
4 p=0.5
5 L=rd.geometric(p,10000)
6 m=max(L)
7 Labs=[-0.5+k for k in range(1,m+2)]
8 plt.hist(L,Labs,density=True,edgecolor='b')
9
10
11 plt.show()
```

6. 6.a. Représenter l'histogramme des fréquences de 10000 réalisations indépendantes d'une variable aléatoire suivant la loi de Poisson de paramètre $\lambda = 3$.
- 6.b. Sur le même graphique, faire apparaître les valeurs des probabilités d'une telle loi.
- 6.c. Comparer le graphique obtenu avec celui de la question 4.

```
1 import numpy.random as rd
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import math
5
6 lam=3
```

```

7 L=rd.poisson(lam,10000)
8 m=max(L)
9 Labs=[-0.5+k for k in range(0,m+2)]
10 plt.hist(L,Labs,density=True,edgecolor='b')
11
12 #Lproba=[lam**k*np.exp(-lam)/math.factorial(k) for k in range(0,m+1)]
13 #plt.plot(range(0,m+1),Lproba,"r+")
14
15 plt.show()

```

III QUELQUES AUTRES LOIS...

III.1 GÉOMÉTRIQUE TRONQUÉE

On lance successivement et de façon indépendante une pièce donnant PILE avec la probabilité 0,3 jusqu'à obtenir le premier PILE, et on s'arrête dans tous les cas après 10 lancers. On note X la variable aléatoire égale au nombre de lancers effectués.

7. 7.a. Écrire une fonction telle que l'exécution de `simul_X()` renvoie une réalisation de la variable aléatoire X .

7.b. Représenter l'histogramme des fréquences de 10000 réalisations indépendantes de X .

```

1 import numpy.random as rd
2 import matplotlib.pyplot as plt
3
4 def simul_X():
5     n=1
6     while n<10:
7         if rd.random()<0.7:
8             n=n+1
9         else :
10            return n
11    return n
12
13 def simul_Xbis():
14     n=1
15     while n<10 and rd.random()<0.7:
16         n=n+1
17     return n
18
19 def simul_Xter():
20     for n in range(1,11):
21         if rd.random()<0.3:
22             return n
23     return 10
24
25 L=[simul_Xter() for k in range(10000)]
26 Labs=[-0.5+k for k in range(1,12)]
27 plt.hist(L,Labs,density=True,edgecolor='k')
28 plt.show()

```

7.c. Donner une valeur approchée de $\mathbb{P}([X = 10])$ et calculer sa valeur exacte.

- D'après l'histogramme obtenu, on peut penser que $\mathbb{P}([X = 10]) \approx 0,4$.
- Calculons cette probabilité.

Notons, pour tout $k \in \llbracket 1; 10 \rrbracket$, P_k l'événement "obtenir PILE au lancer k " et $F_k = \overline{P}_k$.

$[X = 10]$ est réalisé si, et seulement si, on a effectué 10 lancers

si, et seulement si, le premier PILE a été obtenu au dixième lancer ou aucun PILE
n'a été obtenu lors des 10 lancers

D'où :

$$[X = 10] = \left(\left(\bigcap_{k=1}^9 F_k \right) \cap P_{10} \right) \cup \left(\bigcap_{k=1}^{10} F_k \right)$$

Ainsi :

$$\begin{aligned} \mathbb{P}([X = 10]) &= \mathbb{P} \left(\left(\left(\bigcap_{k=1}^9 F_k \right) \cap P_{10} \right) \cup \left(\bigcap_{k=1}^{10} F_k \right) \right) \\ &= \mathbb{P} \left(\left(\bigcap_{k=1}^9 F_k \right) \cap P_{10} \right) + \mathbb{P} \left(\bigcap_{k=1}^{10} F_k \right) \end{aligned}$$

$\nwarrow F_{10}$ et P_{10} sont incompatibles, donc $\left(\left(\bigcap_{k=1}^9 F_k \right) \cap P_{10} \right)$ et $\left(\bigcap_{k=1}^{10} F_k \right)$ également indépendance des lancers

Remarque

Ici, on peut remarquer que

$$\left(\left(\bigcap_{k=1}^9 F_k \right) \cap P_{10} \right) \cup \left(\bigcap_{k=1}^{10} F_k \right) =$$

$$\left(\bigcap_{k=1}^9 F_k \right) \cap (P_{10} \cup F_{10}) = \bigcap_{k=1}^9 F_k \dots$$

$$\begin{aligned}
 &= 0,7^9 \times 0,3 + 0,7^{10} \\
 &= 0,7^9(0,3 + 0,7) \\
 &= 0,7^9
 \end{aligned}$$

III.2 MAXIMUM DE DEUX VA

Soit $n \in \llbracket 2; +\infty \rrbracket$. On considère une urne composée de n boules, numérotées de 1 à n , indiscernables au toucher. On tire simultanément deux boules dans cette urne. On note X_n la variable aléatoire égale au plus grand des deux nombres obtenus.

8. 8.a. Écrire une fonction telle que l'exécution de `simul_X(n)` renvoie une réalisation de la variable aléatoire X_n .

8.b. Écrire un programme permettant d'obtenir une valeur approchée de $\mathbb{E}(X_5)$.

```

1 import numpy.random as rd
2 def simul_X(n):
3     a=rd.randint(1,n+1)
4     b=rd.randint(1,n+1)
5     while b==a:
6         b=rd.randint(1,n+1)
7     return max(a,b)
8
9 n=5
10 L=[simul_X(n) for k in range(10000)]
11 E=sum(L)/len(L)
12 print(E)

```

III.3 LOI UNIFORME ?

Soit $n \in \llbracket 3; +\infty \rrbracket$. On dispose d'une urne contenant $(n - 1)$ balles blanches et une balle noire. On effectue des tirages sans remise dans l'urne jusqu'à l'obtention de la balle noire. On note X_n la variable aléatoire égale au rang d'apparition de la balle noire.

9. 9.a. Écrire une fonction Python telle que l'exécution de la commande `simul_X(n)` renvoie une réalisation de la variable aléatoire X_n .

9.b. Représenter l'histogramme des fréquences de 10000 réalisations indépendantes de X_n dans les cas $n = 5$, puis $n = 10$.

Que peut-on conjecturer ?

```

1 import numpy.random as rd
2 import matplotlib.pyplot as plt
3
4 def simul_X(n):
5     nb=n #nb de balles restantes
6     X=1
7     while rd.randint(1,nb+1)!=1: #on code par 1 la balle noire
8         nb=nb-1
9         X=X+1
10    return X
11
12 def simul_Xbis(n):
13     nb=n #nb de balles restantes
14     X=1
15     while rd.random()<(nb-1)/nb:
16         nb=nb-1
17         X=X+1
18     return X
19
20 def simul_Xter(n):
21     L=['N']+['B' for k in range(n-1)]
22     i=rd.randint(len(L))
23     X=1
24     while L[i]=='B':
25         del(L[i])
26         X=X+1
27         i=rd.randint(len(L))
28     return X
29
30 n=10
31 L=[simul_Xter(n) for k in range(100000)]

```

Remarque

On pourrait raisonner autrement... $[X = 10]$ est réalisé si, et seulement si, on a effectué 10 lancers ; si, et seulement si, on ne s'arrête pas avant le dixième lancer ; si, et seulement si, les lancers 1 à 9 ont tous donné FACE.
D'où : $[X = 10] = \bigcap_{k=1}^9 F_k$...

Remarque

L'interprétation de la moyenne empirique comme valeur approchée de l'espérance est basée sur la loi faible des grands nombres, que nous verrons dans le chapitre 14.

```

32| Ls=[-0.5+k for k in range(1,n+2)]
33| plt.hist(L,Ls,edgecolor='k',density=True)
34| plt.show()

```

Après visualisation de l'histogramme, on conjecture que X_n suit la loi uniforme sur $\llbracket 1; n \rrbracket$.

9.c. Démontrer cette conjecture.

Notons, pour tout $i \in \llbracket 1; n \rrbracket$: B_i l'évènement "tirer une balle blanche au tirage i " et $N_i = \overline{B_i}$.

- Ensuite, on a déjà $X_n(\Omega) = \llbracket 1; n \rrbracket$.
- Puis, soit $k \in \llbracket 1; n \rrbracket$. On a :

$$[X_n = k] = \left(\bigcap_{i=1}^{k-1} B_i \right) \cap N_k$$

D'où :

$$\begin{aligned} \mathbb{P}([X_n = k]) &= \mathbb{P}\left(\left(\bigcap_{i=1}^{k-1} B_i\right) \cap N_k\right) \\ &= \mathbb{P}(B_1) \times \dots \times \mathbb{P}_{B_1 \cap \dots \cap B_{k-2}}(B_{k-1}) \times \mathbb{P}_{B_1 \cap \dots \cap B_{k-1}}(N_k) \end{aligned}$$

formule des probabilités composées, licite car $\mathbb{P}(B_1 \cap \dots \cap B_{k-1}) \neq 0$

Soit ensuite $i \in \llbracket 1; k-1 \rrbracket$. Supposons l'évènement $B_1 \cap \dots \cap B_i$ réalisé. On a ainsi tiré i balles blanches dans l'urne.

Le prochain tirage s'effectue donc dans une urne composée de $n-i$ balles, dont 1 est noire et $n-i-1$ sont blanches.

Par équiprobabilité du choix de la balle, on a alors :

$$\begin{aligned} \mathbb{P}_{B_1 \cap \dots \cap B_i}(B_{i+1}) &= \frac{n-i-1}{n-i} \\ \mathbb{P}_{B_1 \cap \dots \cap B_i}(N_{i+1}) &= \frac{1}{n-i} \end{aligned}$$

D'où, en reprenant le calcul ci-dessus :

$$\begin{aligned} \mathbb{P}([X_n = k]) &= \frac{n-1}{n} \times \frac{n-2}{n-1} \times \dots \times \frac{n-1-(k-2)}{n-(k-2)} \times \frac{1}{n-(k-1)} \\ &= \frac{n-1}{n} \times \frac{n-2}{n-1} \times \dots \times \frac{n-k+1}{n-k+2} \times \frac{1}{n-k+1} \\ &= \frac{1}{n} \end{aligned}$$

) télescopage

Remarque

Avec comme convention $\bigcap_{i \in \emptyset} B_i = \Omega$ (dans le cas où $k=1$).

Important !

Il faut savoir rédiger correctement cette partie, qui peut faire l'objet d'une question séparée.

Conclusion : $X_n \hookrightarrow \mathcal{U}(\llbracket 1; n \rrbracket)$.