

Simulation de variables aléatoires

On dit que X est une variable aléatoire uniforme sur $[0;1]$ si pour tout intervalle I inclus dans $[0;1]$, la probabilité que X appartienne à I soit égale à la longueur de l'intervalle I . Ainsi, si X est une telle variable la probabilité que X prenne une valeur dans $[0;1/3]$ vaut $1/3$. Python permet de simuler une telle variable aléatoire grâce aux commandes suivantes :

```
import random as rd # À mettre en début de sujet
```

```
a = rd.random() # valeur aléatoire dans [0,1]
```

Loi de Bernoulli et loi binomiale

1. Complétez la fonction `VaBernoulli(p)` qui simule une variable aléatoire de Bernoulli de paramètre $p \in [0;1]$. Cette fonction renvoie donc 1 avec une probabilité de p et 0 avec une probabilité de $1 - p$.

```
def VaBernoulli(p):  
    a = rd.random()  
    if ...  
        return 1  
    else:  
        return ...
```

2. Écrire une fonction `VaBinomiale(n,p)` qui simule une variable aléatoire suivant une loi binomiale de paramètres $n \in \mathbb{N}^*$ et $p \in [0;1]$. On se rappellera que la somme de n variables de Bernoulli indépendantes et de paramètre p suit une loi binomiale de paramètres n et p .
3. Écrire une fonction `EchantillonBinomiale(n,p,Nb)` qui renvoie une liste contenant Nb éléments, où chaque élément prend une valeur suivant une loi binomiale de paramètres n et p .

On cherche maintenant à comparer l'échantillon de nos binomiales à la vraie loi binomiale, ainsi on aimerait afficher sur un même graphe la loi de probabilité de la binomiale avec l'histogramme normalisé, c'est-à-dire pour chaque valeur, le nombre d'échantillon ayant pris cette valeur divisé par la taille de l'échantillon.

4. Écrire une fonction `Factorielle(k)` qui renvoie $k!$.
5. Écrire une fonction `ProbaBinomiale(n,p,k)` qui renvoie $\mathbb{P}(X = k)$ pour $X \sim \mathcal{B}(n, p)$.

6. En déduire une fonction `LoiBinomiale(n,p)` qui renvoie une liste telle que, pour tout $k \in \llbracket 0; n \rrbracket$, l'élément d'indice k vaille $\mathbb{P}(X = k)$.
7. Recopiez le code suivant et comprenez-le :

```
n = 8  
p = 0.7  
Nb = 10  
L = LoiBinomiale(n,p)  
C = EchantillonBinomiale(n,p,Nb)
```

```
plt.figure()  
plt.plot(L,"or")  
plt.hist(C,density = True,bins = [k-0.5 for k in range(0,n+2)])  
plt.legend(["Loi","Histogramme normalisé des simulations"])  
plt.title("Comparaison entre loi binomiale et l'histogramme")  
plt.show()
```

Que se passe-t-il si Nb prend des valeurs grandes ?

Loi géométrique

8. Écrire une fonction `VaGéométrique(p)` qui simule une variable aléatoire suivant une loi géométrique. Pour cela, se rappeler qu'il suffit de simuler des Bernoulli de paramètre p jusqu'à obtenir un succès et compter le nombre d'essais jusqu'à obtenir un succès.
9. Écrire une fonction `EchantillonGéométrique(p,Nb)` qui renvoie un échantillon avec Nb éléments prenant des valeurs suivant une loi géométrique de paramètre p .

On cherche à comparer cet échantillon à la vraie loi, mais comme la vraie loi prend un nombre infini de valeurs, on va regarder la loi seulement pour des valeurs entre 1 et M le maximum des valeurs prises dans l'échantillon.

10. Écrire une fonction `Maximum(Liste)` qui, à une liste non vide de nombres, renvoie la valeur maximale.
11. Écrire une fonction `LoiGéoTronquée(p,N)` qui renvoie deux listes, la première contient tous les éléments de $\llbracket 1; N \rrbracket$ dans l'ordre la seconde contient $\mathbb{P}(X = k)$ pour¹ tout $k \in \llbracket 1; N \rrbracket$ si $X \sim \mathcal{G}(p)$.

¹ Si $X \sim \mathcal{G}(p)$, alors $X(\Omega) = \mathbb{N}^*$, seulement on ne pas stocker une liste infinie, c'est pour cela que l'on troque un entier $N \in \mathbb{N}^*$.

12. En vous inspirant de ce qui a été fait pour la loi binomiale, afficher l'histogramme normalisée d'un échantillon de la loi géométrique et de la loi tronquée à N , où N est le maximum des valeurs prises par l'échantillon.

Simulation d'une variable aléatoire ayant une loi donnée

13. Soit V une liste de nombres réels deux à deux distincts et P une liste de nombres positifs de somme valant 1 tel que P et V aient la même longueur. D'après le cours, il existe X une variable aléatoire telle que pour tout indice i , $\mathbb{P}(X = V[i]) = P[i]$. Écrire une fonction `VaLoiDonnée(V,P)` simulant une telle variable aléatoire. Pour cela, on pourra tirer un nombre au hasard dans $[0;1]$, puis si ce nombre est plus petit que $P[0]$ on renvoie $V[0]$, si ce nombre est entre $P[0]$ et $P[0] + P[1]$, renvoyer $V[1]$, si ce nombre est entre $P[0] + P[1]$ et $P[0] + P[1] + P[2]$ renvoyer $V[2]$ etc.

TIPE

14. Recopiez le code suivant :

```
p = rd.random()
Nb = 100
L = [VaBernoulli(p) for k in range(Nb)]
```

En effet, pour son TIPE, un élève a utilisé un échantillon de 100 variables aléatoires de Bernoulli de paramètre p , mais il a oublié de noter la valeur de p qu'il avait prise, il lui reste seulement la liste L . Le but est de l'aider à retrouver p , sachant qu'il souhaite retrouver cette valeur avec une approximation de 0.01 près, pour cela on utilise l'espérance empirique : la moyenne de l'échantillon.

15. Calculer la somme des éléments de L et diviser par Nb .
16. En utilisant l'inégalité de Bienaymé-Tchebychev, majorer la probabilité que l'écart entre l'approximation obtenue et p soit plus grande que 0.01. Est-ce satisfaisant ?
17. Vérifier en faisant la différence entre p et son approximation.

Les anniversaires en 2bio2

18. On attribue à chaque jour d'une année (non bissextile) un entier entre 0 et 364. On considère une classe de 47 élèves, attribuer à chacun de

ces 47 élèves un jour de naissance au hasard et vérifier s'il y a au moins deux personnes nées le même jour.

19. En refaisant cette expérience un grand nombre de fois, approximer la probabilité qu'il existe au moins deux personnes nées le même jour en 2bio2.

Histogramme revisité

Depuis le début, on utilise en «boîte noire» la commande `hist` de `matplotlib.pyplot`, on se propose de refaire les choses par nous-même (ce qui nous permettra d'afficher l'histogramme pour une variable aléatoire prenant un ensemble fini et quelconque de valeurs).

20. Écrire une fonction `DicoOccurrences(L)` qui, à une liste L , renvoie un dictionnaire tel que les clés de ce dictionnaire soit les éléments la liste et la valeur associée à une clé soit son nombre d'occurrences dans la liste.
21. Écrire une fonction `DicoDensité(L)` qui fait la même chose, mais le nombre d'occurrences est maintenant divisé par la longueur de la liste L .
22. Utiliser la fonction précédente pour afficher l'histogramme d'un échantillon d'une variable aléatoire simulée grâce à la fonction `VaLoiDonnée` ainsi que sa loi.

Simulation d'une loi de Poisson

23. Généraliser le principe de la fonction `VaLoiDonnée` (qui ne fonctionnait que pour une variable aléatoire finie) pour simuler une loi de Poisson de paramètre λ .