Math ECG 2. 2022-2023

Mathématiques Appliquées - F. Gaunard
http://frederic.gaunard.com
ENC Bessières, Paris 17e.

T.P. n°1

Rappels de cours sur les lois (discrètes) usuelles
Simulations de variables aléatoires (finies et) discrètes
Histogrammes. Diagrammes à bâtons.
Comparaison de distributions

Bibliothèques et librairies utilisées : `numpy`, `numpy.random`, `matplotlib.pyplot`.

1 Simulation d'une variable aléatoire avec `rd.rand( )`

☞ On commence, pour toute la suite de ce TP à importer les bibliothèques et libraires nécessaires à chaque situation:

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

À retenir!

La commande `rd.rand( )` renvoie un nombre réel aléatoire entre 0 et 1 *suivant la loi uniforme (continue)* $\mathcal{U}([0;1])$. Plus précisément, si $p \in [0;1]$, la probabilité que le nombre renvoyé soit inférieure ou égale à p (ou dans un intervalle de longueur p) vaut exactement p.

On **identifie** et décide de représenter donc un évènement de probabilité p par le fait que le nombre aléatoire ainsi renvoyé soit inférieur ou égal à p (ou dans un intervalle de longueur p).

Exercice 1. Le score d'un joueur lors d'un lancer de fléchette est modélisé par une variable aléatoire X telle que $X(\Omega) = \{0, 2, 5, 10\}$ et

$$P(X=0) = \frac{1}{5}, \qquad P(X=2) = \frac{1}{2}, \qquad P(X=5) = \frac{1}{5}, \quad \text{et} \quad P(X=10) = \frac{1}{10}.$$

(1) Calculer l'espérance et la variance de X.
(2) Découper l'intervalle $[0;1]$ en quatre intervalles où pourrait *tomber* un nombre aléatoire généré avec `rd.rand()` dans le but de simuler X. Écrire alors une fonction `simul_X()` qui renvoie une simulation de la variable X.
(3) On ajoute les commandes suivantes. Les exécuter et commenter l'affichage.

```
def sample_X(n):
    S=[]
    for k in range(n):
        S.append(simul_X())
    return S

print(np.mean(sample_X(1000)))
```

Exercice 2. On désigne par n un entier naturel supérieur ou égal à 2. On lance n fois une pièce équilibrée (c'est-à-dire donnant *pile* avec la probabilité 1/2 et *face* également avec la probabilité 1/2), les lancers étant supposés indépendants.
On note Z la variable aléatoire qui vaut 0 si l'on n'obtient aucun pile pendant ces n lancers et qui, dans le cas contraire, prend pour valeur le rang du premier pile.

(1) Recopier et compléter la fonction suivante permettant de simuler la variable aléatoire Z.

```
def simul_Z(n) :
    for i in range(1, n+1):
        if ........ :
            return ...
    return ...
```

(2) Réécrire cette fonction en utilisant une boucle `while`.
(3) Déterminer $P(X = k)$ pour $k \in [\![0; n]\!]$. (On pourra différencier deux cas selon que $1 \leq k \leq n$ et $k = 0$.)

Exercice 3. Une urne contient initialement deux boules rouges et une boule bleue indiscernables au toucher. On appelle "tirage" la séquence suivant:

On pioche, au hasard, une boule de l'urne, puis :

- Si la boule piochée est bleue, on la remet dans l'urne.
- Si la boule piochée est rouge, on ne la remet pas dans l'urne mais on remet une boule bleue dans l'urne à sa place.

Pour tout entier naturel n non nul, on note Y_n la variable aléatoire discrète égale au nombre de boules rouges présentes dans l'urne à l'issue du $n-$ième tirage.

(1) Compléter la fonction ci-dessous afin qu'elle simule la variable Y_n.

```
def simul_Y(n):
    r=2 # nombre de boules rouges dans l'urne
    for  k in range(n):
        if ... :
            if  ... :
                ...
    return ....
```

(2) On cherche à *estimer* $\lim\limits_{n\to+\infty} P(Y_n = 0)$. On introduit la variable T_n définie par

$$T_n = \begin{cases} 1, & \text{si } [Y_n = 0] \\ 0, & \text{sinon} \end{cases}$$

(a) Quelle est la loi de la variable T_n?
(b) Écrire une fonction qui simule T_n.
(c) On admet que la *moyenne empirique* d'un échantillon de 1000 réalisations de T_n renvoie une estimation de $P(Y_n = 0)$. Écrire une suite d'instruction permettant de visualiser graphiquement l'évolution de (l'estimation de) $P(Y_n = 0)$ en fonction de n. Que peut-on conjecturer quant à la limite ci-dessus?
(d) Justifier que $(Y_n = 0) \subset (Y_{n+1} = 0)$. En déduire que la conjecture précédente se reformule comme

$$P\left(\bigcup_{n=2}^{+\infty}(Y_n = 0)\right) = 1.$$

Interpréter.

(3) On note Z la variable aléatoire égale au numéro du "tirage" amenant la dernière boule rouge.

(a) Donner $Z(\Omega)$.
(b) Écrire une fonction qui simule la variable Z.

Remarque

☞ L'utilisation de la moyenne empirique pour estimer le paramètre d'une Bernoulli prendra plus de sens après avoir traité le chapitre *Estimation*.

☞ La variable T_n s'appelle aussi variable *indicatrice* de l'évènement $Y_n = 0$. On aura l'occasion de travailler avec ce type de variable, dont les sujets ESSEC sont bien friands.

2 (Simulation de) Lois usuelles finies et discrètes

2.1 Loi uniforme $\mathcal{U}(\llbracket m; n \rrbracket)$

Résultat du cours de première année

On dit que X suit une loi uniforme sur $\llbracket m; n \rrbracket$), ce qu'on note $X \hookrightarrow \mathcal{U}(\llbracket m; n \rrbracket)$ si $X(\Omega) = \llbracket m; n \rrbracket$ et

$$\forall k \in \llbracket m; n \rrbracket, \qquad P(X = k) = \frac{1}{n - m + 1}$$

De plus,

$$E(X) = \frac{n + m}{2}, \qquad V(X) = \frac{(n - m + 1)^2 - 1}{12}.$$

☞ On peut *reconnaître* une loi uniforme lors d'un tirage dont les issues sont équiprobables (lancer d'un dé non truqué, tirage de boules indiscernables ...).

☞ La commande `rd.randint(m, n+1)` renvoie une simulation d'une variable $X \hookrightarrow \mathcal{U}(\llbracket m; n \rrbracket)$.

Exercice 4. Représenter le diagramme à bâtons dont les hauteurs des bâtons sont les valeurs **théoriques** de la loi uniforme $\mathcal{U}(\llbracket 1; N \rrbracket)$.

À retenir!

☞ Avoir en tête les diagrammes à bâtons théoriques des lois usuelles permet d'émettre des conjectures sur la loi suivie par une variable aléatoire inconnue, que l'on sait simuler, à partir du diagramme à bâtons (ou de l'histogramme) des fréquences obtenues sur un grand échantillon de la variable.

Exercice 5. On dispose de n urnes, numérotées de 1 à n. Pour chaque $k \in \llbracket 1, n \rrbracket$, l'urne k est composée de boules numérotées de 1 à k. On choisit une urne au hasard puis on tire une boule dans cette urne et on note X_n la variable aléatoire qui prend la valeur du numéro de la boule piochée.
Écrire une fonction `simul_X(n)` qui simule X_n.

Exercice 6. Soit N un entier supérieur ou égal à 3. On considère une urne qui contient $(N-1)$ boules blanches et une seule boule noire.
On effectue des tirages **sans remise** jusqu'à l'obtention de la boule noire et on note X la variable aléatoire qui prend pour valeur le nombre de tirages nécessaires.

(1) Écrire une fonction `simul_X(N)` qui renvoie une simulation de la variable X.

(2) Recopier et exécuter les instructions ci-contre. Comparer avec le diagramme théorique de l'Exercice 4. Commenter, conjecturer, démontrer.

```
import matplotlib.pyplot as plt

N=5 # on fera varier les valeurs de N
freq_val_X=np.zeros(N)
for k in range(10000):
    i=simul_X(N)
    freq_val_X[i-1]+=1
freq_val_X=freq_val_X/10000
plt.bar(range(1, N+1), freq_val_X)
plt.show()
```

2.2 Loi de Bernoulli $\mathcal{B}(p)$

Résultat du cours de première année

On dit que X suit une loi de Bernoulli de paramètre p, ce qu'on note $X \hookrightarrow \mathcal{B}(p)$ si $X(\Omega) = \{0;1\}$ et

$$P(X=1) = p, \qquad P(X=0) = 1-p.$$

De plus,

$$E(X) = p, \qquad V(X) = p(1-p).$$

☞ La loi de Bernoulli correspond à une situation à deux alternatives, l'une qu'on appelle *succès* (et lors de laquelle la variable prend la valeur 1) et l'autre qu'on appelle *échec*.

Exercice 7. Écrire, à l'aide de la commande `np.rand( )` une fonction `Bernoulli(p)` permettant de simuler une variable $X \hookrightarrow \mathcal{B}(p)$.

2.3 Loi binomiale $\mathcal{B}(n,p)$

Résultat du cours de première année

On dit que X suit une loi binomiale de paramètres n et p, ce qu'on note $X \hookrightarrow \mathcal{B}(n,p)$ si $X(\Omega) = [\![0,n]\!]$ et

$$\forall k \in [\![0,n]\!], \qquad P(X=k) = \binom{n}{k} p^k (1-p)^{n-k}$$

De plus,

$$E(X) = np, \qquad V(X) = np(1-p).$$

☞ Si $n=1$, la loi binomiale $\mathcal{B}(1,p)$ n'est rien d'autre que la loi de Bernoulli $\mathcal{B}(p)$.

☞ On *reconnait* une loi binomiale de paramètre n et p lorsque l'on compte le nombre de succès (dont la probabilité de chaque est p) lors de n répétitions **indépendantes** d'épreuves de Bernoulli.

☞ La commande `rd.binomial(n,p)` renvoie une simulation d'une variable $X \hookrightarrow B(n,p)$.

☞ On peut alors très rapidement simuler une variable de Bernoulli de paramètre p avec la commande `rd.binomial(1,p)`.

Exercice 8. Écrire, à l'aide de la commande `np.rand( )` et d'une boucle `for`, une fonction `Binomiale(n,p)` permettant de simuler une variable $X \hookrightarrow \mathcal{B}(n,p)$.

Exercice 9. On donne le code suivant. Le recopier et exécuter `fig_mystere` pour plusieurs valeurs de n. Expliquer, commenter.

```
import numpy as np
import matplotlib.pyplot as plt

def mystere(n,k):
    return np.prod([range(n-k+1, n+1)])/np.prod(range(1, k+1))

def fig_mystere(n):
    Y=[mystere(n,k) for k in range(0, n+1)]
    X=[k for k in range(n+1)]
    plt.bar(X,Y)
    plt.show()
```

Exercice 10. (Poolage sanguin)
On étudie une méthode de détection des porteurs d'un parasite au sein d'un ensemble donné de N individus tirés au sort. La probabilité d'être porteur du parasite dans la population est $p \in]0;1[$. Les personnes sont atteintes indépendemment les unes des autres.

On dispose d'un test permettant d'établir de façon certaine qu'un échantillon de sang contient ou non le parasite, le résultat de ce test étant dit positif dans le premier cas et négatif dans le second.

Pour chacun des N individus, on possède un prélèvement sanguin. On envisage alors deux méthodes de détection:

- Première méthode: on teste un à un les N prélèvements, effectuant ainsi N tests.

- Seconde méthode (*poolage*):
  - On fixe un entier naturel non nul ℓ. On suppose que N est un multiple de ℓ et on pose $N = n \times \ell$. On répartit alors les N prélèvements en n groupes $G_1, G_2, \ldots, G_n$, chaque groupe G_i contenant le même nombre ℓ de prélèvements. Pour chacun des groupes G_i, on extrait une quantité de sang de chacun des l prélèvements qu'il contient, puis on mélange ces extraits, obtenant ainsi un échantillon de sang H_i, caractéristique du groupe G_i.

  - On teste alors H_i:
    - si le test de H_i est négatif, aucun des individus au sein du groupe G_i n'est porteur du parasite. Le travail sur le groupe G_i est alors terminé;
    - si le test de H_i est positif, on teste un à un les prélèvements de G_i pour détecter les porteurs du parasite au sein du groupe G_i.

Soient X la variable aléatoire égale au nombre de groupes G_i pour lesquels le test de H_i a été positif et T la variable aléatoire égale au nombre total de tests effectués dans la réalisation de la méthode du *poolage*.

(1) Écrire une fonction `test(`ℓ`, p)` simulant le test sur un groupe de taille ℓ (avec probabilité p de contamination de chaque individu). (La fonction renverra 0 ou 1 selon que le test est négatif ou non).
(2) Écrire alors une fonction `poolage_sanguin(N,p,`ℓ`)` renvoyant le résultat simultané de la simulation de X et T.
(3) (*) Déterminer, de manière théorique et rigoureuse $E(T)$. Établir une condition entre ℓ et p pour que la méthode de poolage soit avantageuse (en moyenne).

2.4 Loi géométrique $\mathcal{G}(p)$

Résultat du cours de première année

On dit que X suit une loi géométrique de paramètre p, ce qu'on note $X \hookrightarrow \mathcal{G}(p)$ si $X(\Omega) = \mathbb{N}^*$ et

$$\forall k \in \mathbb{N}^*, \qquad P(X = k) = (1-p)^{k-1}p$$

De plus, X admet espérance et variance et

$$E(X) = \frac{1}{p}, \qquad V(X) = \frac{1-p}{p^2}.$$

☞ Une loi géométrique correspond au *temps d'attente* du premier succès (de probabilité p) lors de répétitions (infinies) **indépendantes** d'une même épreuve de Bernoulli.

☞ La commande `rd.geometric(p)` renvoie une simulation d'une variable $X \hookrightarrow G(p)$.

Exercice 11. Écrire, à l'aide de la commande `rd.rand( )` et d'une boucle `while` une fonction `Geometrique(p)` permettant de simuler une variable $X \hookrightarrow \mathcal{G}(p)$.

Exercice 12. On effectue une succession infinie de lancers indépendants d'une pièce donnant `Pile` avec probabilité p. On note $q = 1-p$.
On dit que la première série est de longueur $n \geq 1$ si les n premiers lancers ont amené le même côté de la pièce et le $(n+1)$-ième l'autre côté. De même la deuxième série commence au lancer suivant la fin de la première série et se termine au lancer précédant un changement de côté.

Par exemple si les lancers donnent les résultats $FFPPPPPPFFFPP\ldots$ alors la première série est de longueur 2 et la deuxième est de longueur 6.
Soient X_1 et X_2 les variables aléatoires égales aux longueurs de la première et deuxième série.

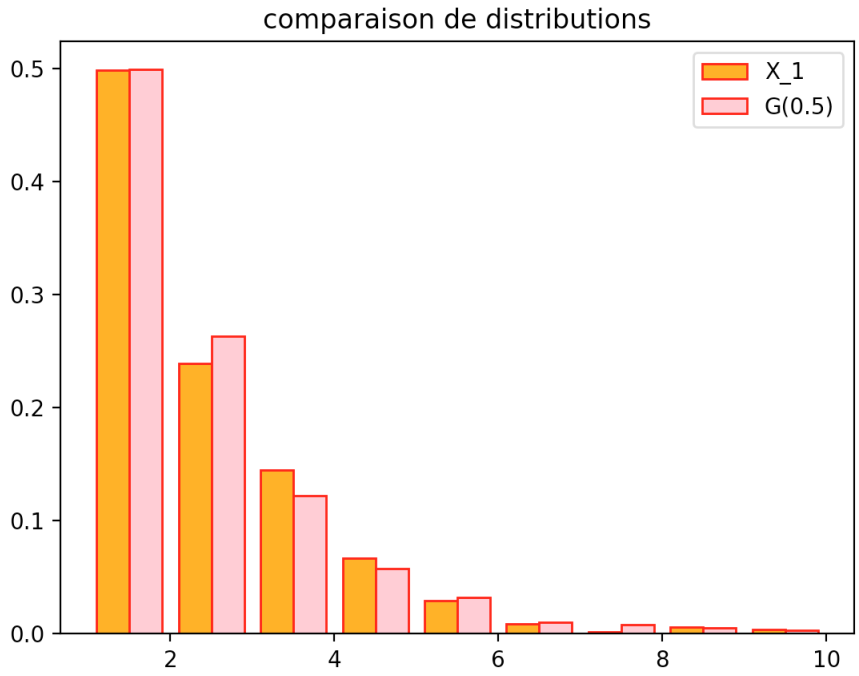
(1) Recopier et compléter la fonction suivante, prenant en argument la probabilité p d'obtenir `Pile` et permettant de simuler la variable aléatoire X_1.

```
def simul_X1(p):
    X=1
    t_old=rd.binomial(1,p)
    t_new=rd.binomial(1,p)
    while .... :
        X= ...
        t_old= ...
        t_new= ...
    return X
```

(2) On ajoute le code suivant qui affiche la figure reproduite ci-après. Interpréter.

```
p=0.5
N=1000
x1=[simul_X1(p) for k in range(N)]
x2=[rd.geometric(p) for k in range(N)]
plt.hist([x1, x2], color = ['orange', 'pink'], density=True,
         label = ['X_1', 'G(0.5)'])
plt.title('comparaison de distributions')
plt.legend()
plt.show()
```

Affichage Python



(3) Déterminer la loi de X_1. Retrouver le résultat conjecturé à la question précédente. Montrer qu'elle admet une espérance que l'on explicitera.

(4) Écrire une fonction d'entête `simul_X2(p)` permettant de simuler la variable X_2.
(5) Déterminer, pour $(k,j) \in \mathbb{N}^* \times \mathbb{N}^*$, $P(X_1 = k \cap X_2 = j)$. En déduire, à l'aide de la formule des probabilités totales, la loi de X_2.

2.5 Loi de Poisson $\mathcal{P}(\lambda)$

Résultat du cours de première année

On dit que X suit une loi de Poisson de paramètre $\lambda > 0$, ce qu'on note $X \hookrightarrow \mathcal{P}(\lambda)$ si $X(\Omega) = \mathbb{N}$ et

$$\forall k \in \mathbb{N}, \qquad P(X = k) = e^{-\lambda}\frac{\lambda^k}{k!}$$

De plus, X admet espérance et variance et

$$E(X) = \lambda, \qquad V(X) = \lambda^2.$$

☞ La commande `rd.poisson(m)` renvoie une simulation d'une variable $X \hookrightarrow P(m)$.

Exercice 13. Chaque jour, le nombre de personnes subissant un test de dépistage dans une certaine pharmacie est modélisé par une variable X suivant une loi de Poisson de paramètre $m = 5$. Chaque test a une certaine probabilité $p = 1/10$ d'être positif et les tests sont indépendants les uns des autres. On note N le nombre de tests positifs un jour donné.

(1) Écrire une fonction permettant simuler la variable N.
(2) Créer un échantillon de taille 1000 de cette variable aléatoire et le comparer, avec un histogramme, à un échantillon de même taille d'une loi de Poisson de paramètre $mp = 0,5$. Conjecturer.
(3) (*) En commençant par expliciter $P_{X=n}(N = k)$ (en faisant attention sur le couplage des indices n et k), démontrer le résultat précédemment conjecturé.