DS3

ECG 1

Réponses

Introduction

Le but est d'étudier une méthode pour créer des graphes aléatoires.

Dans toute la suite les graphes considérés seront des graphes non orientés sans boucle. Les sommets d'un graphe à n sommets seront toujours numérotés par l'ensemble $[\![0, n-1]\!] = \{0, 1, \ldots, n-1\}$

Tous les programmes et fonctions demandées doivent être écrits en python. Veuillez marquer de façon claire les indentations.

On considère que les modules `numpy, numpy.random` ont été importés avec les commandes.

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

Dans le module `numpy.random` on utilisera la fonction `random()` qui renvoie un réel entre 0 et 1 choisi de façon uniforme. Les graphes seront implémentés par leur matrice d'adjacence. Cette matrice sera représenté par une liste de liste ne contenant que des 0 ou des 1.

Lorsque l'on écrira des fonctions, on admettra, sans le verifier, que les matrices sont bien des matrices d'adjacence.

Même si vous n'avez pas réussi à écrire une fonction demandée dans une question, vous pouvez l'utiliser dans les questions suivantes

Préliminaires

1. Quelles sont les caractéristiques d'une matrice d'adjacence d'un graphe non orienté sans boucle.

RÉPONSE:

La matrice est symétrique, les coefficients diagonaux sont tous nuls et les autres coefficients ne peuvent avoir pour valeur que 0 ou 1.

*

2. Écrire une fonction `nulle(n)` qui pour un entier naturel non nul renvoie une matrice n lignes n colonnes dont tous les coefficients sont nuls.
   Par exemple

— `nulle(2)` renvoie [[0, 0], [0, 0]]
— `nulle(3)` renvoie[[0, 0, 0], [0, 0, 0], [0, 0, 0]]

RÉPONSE:

```
def nulle(n):
    '''matrice nulle de taille nxn'''
    L=[]
    M=[]
    for i in range(n):
        L=[]
        for j in range(n):
            L.append(0)
        M.append(L)
    return M
```

On peux aussi écrire

```
def nulle(n):
    M=[]
    for i in range(n):
        M.append([0]*n)
    return M
```

*

3. Écrire une fonction `verification(M)` : qui vérifie si la matrice M est bien symétrique.
   On ne demande pas de vérifier que la matrice M est une matrice d'adjacence.
   Par exemple
   — `verification([[0,1,0],[1,0,0],[0,0,0]])` renvoie la valeur `True`
   — `verification([[0,1,0],[1,0,0],[0,1,0]])` renvoie la valeur `False`

RÉPONSE:

```
def verification(M):
    '''verifie si la matrice M supposé carrée est bien
    ↪ symétrique'''
    n=len(M)
    for i in range(n):
        for j in range(i+1,n):
            if M[i][j]!=M[j][i]:
                return False
    return True
```

avec des boucles `while` et un drapeau.

```python
def verification(M):
    '''verifie si la matrice M supposée carrée est bien
    ↪ symétrique'''
    n=len(M)
    i=0
    drapeau=True
    while i<n and drapeau:
        j=0
        while j<n and drapeau:
            drapeau =(M[i][j]==M[j][i])## == et = sont des
            ↪ opérateurs distincts!
            j=j+1
        i=i+1

    return drapeau
```

*

4. Démontrez par récurrence que pour $n \in \mathbb{N}^*$ $\sum_{k=1}^{n-1} k = \frac{n(n-1)}{2}$

RÉPONSE:

— **Initialisation :**
Pour $n = 2$,$\sum_{k=1}^{n-1} k = \sum_{k=1}^{2-1} k = 2$ et $\frac{2 \times (2-1)}{2} = 1$

— **Hérédité :** Pour $n \in \mathbb{N}^* \setminus \{1\}$ supposons que

$$\sum_{k=1}^{n-1} k = \frac{n(n-1)}{2}$$

alors

$$\sum_{k=1}^{n+1-1} k = \sum_{k=1}^{n-1} k + n$$

$$\begin{aligned}
&= \frac{n(n-1)}{2} + n \\
&= \frac{n^2 - n + 2n)}{2} \\
&= \frac{n^2 + n)}{2} \\
&= \frac{(n+1)n)}{2} \\
&= \frac{(n+1)(n+1-1))}{2}
\end{aligned}$$

Da'près le principe de récurrence

$$\boxed{n \in \mathbb{N}^* \setminus \{1\},\ \sum_{k=1}^{n-1} k = \frac{n(n-1)}{2}}$$

*

5. Quelle est le nombre maximal et minimal d'arêtes d'un graphe à n sommets.

RÉPONSE:

Un graphe peut avoir 0 arête.

Une arête non orienté est définie par ses deux sommets. Il y a $\binom{n}{2} = \frac{n(n-1)}{2}$ façons de choisir une paire de deux sommets (sans répétition et sans ordre).

Le nombre d'arêtes maximal est $\binom{n}{2} = \frac{n(n-1)}{2}$

*

6. On fixe n le nombre de sommets du graphe. Combien de graphes différents peut on construire ? On pourra raisonner sur les coefficients de la matrice d'adjacence ,

RÉPONSE:

Pour déterminer un graphe il suffit de choisir les coefficients de la matrices d'adjacence. Comme la matrice doit être symétrique à diagonale nulle il faut chosir les coefficients $M_{i,j}$ pour $0 \leqslant i < j \leqslant n-1$. Il y a donc $n-1$ coefficients sur sur la première ligne $n-2$ coefficients sur la deuxième ligne,. . ., 1 coefficient sur la ligne $n-2$. Au total, il y a $n-1+n-2+\ldots 1 = \sum_{k=1}^{n-1} k = \frac{n(n-1)}{2}$
On retrouve le nombre d'arêtes possibles d'un graphe.
Chaque coefficient à deux valeurs possibles 0 ou 1. Le nombre de graphe possibles est donc

$$\underbrace{2 \times 2 \times \cdots \times 2}_{\frac{n(n-1)}{2} \text{ fois}}$$

Le nombre graphe est $\binom{=}{2}^{\frac{n(n-1)}{2}}$

*

7. Écrire une fonction `aretes(M)` qui, étant donné une matrice d'adjacence d'un graphe non orienté, renvoie le nombre d'arêtes du graphe.
   Par exemple

— `aretes([[0,1,0],[1,0,0],[0,0,0]])` renvoie 1
— `aretes([[0,1,0],[1,0,1],[0,1,0]])` renvoie 2

RÉPONSE:

```
def aretes(M):
    '''nombre d'aretes d'un graphe non orienté'''
    nb=0
    n=len(M)
    for i in range(n):
        for j in range(n):
            nb=nb+M[i][j]
    return nb//2
```

*

Dans toute la suite n est un entier naturel plus grand que 2 fixé, qui représente le nombre de sommet des graphes étudiés.

Graphe binomiale

Dans ce modèle on fixe un réel $p \in]0;\ 1[$. On regarde successivement tous les couples (i, j) de sommets tels que $i < j$ et on décide aléatoirement, avec une probabilité p, de relier les sommets i, j par une arête.

On note $G(n, p)$ le graphe alétoire obtenu avec le procédé précédent.

8. Recopier et compléter le programme suivant.

```
def graphe(n,p):
    '''retourne la matrice d'adjacence d'un graphe
    ↪ aléatoire à n sommets'''
    M=nulle(n)
    for i in range(n):
        for j in range(i+1,n):
            if rd.random()<p:
                M[i][j]=1
                M[j][i]=1
    return M
```

9. On note X la variable aléatoire du nombre d'arêtes d'un tel graphe. Quelle loi suit X ? Donner son espérance et sa variance.

RÉPONSE:

X compte le nombre de succès "l'arête (i, j) est présente" lors de la répétition de $\binom{n}{2}$ expérience de Bernouilli indépendantes de probabilité de succès p

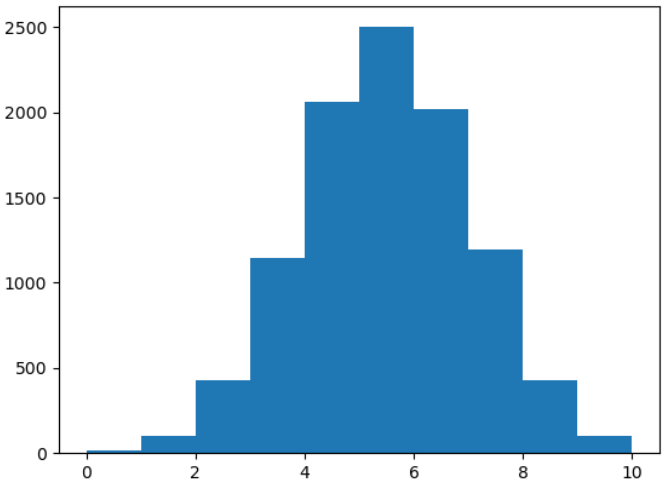
$$\boxed{X \hookrightarrow \mathcal{B}(n,p),\ E(X) = np,\ V(X) = np(1-p).}$$

*

10. On propose le programme suivant

```
n=5
p=0.5
N=10000
L=[]
for i in range(N):
    G=graphe(n,p)
    L.append(aretes(G))

plt.hist(L)
```

On obtient le graphique suivant



Expliquer ce que fait le programme et pourquoi le graphe obtenu est cohérent avec les questions précédentes.

RÉPONSE:

On construit des graphes à 5 sommets ou la probabilité de présence de chaque arête est $\frac{1}{2}$. Le nombre d'arête suit donc la loi $\mathcal{B}(10, 0.5)$
POur le vérifier on construit $N = 10000$ graphes, et on compte leur arètes, l'histogramme de la distribution du nombre d'arêtes ressemble bien à celui d'un variable aléatoire suivante cette loi. Il est notemment symétrique autour de 5 la moyenne.

*

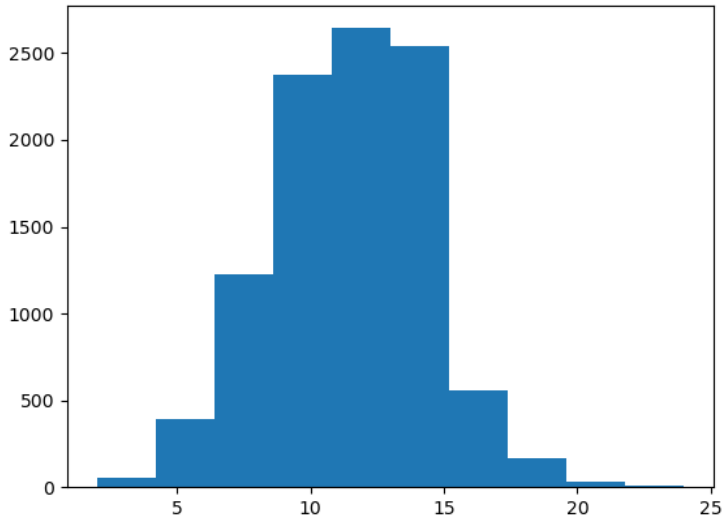
11. Dans le programme précédent on remplace les deux premières lignes par

```
n=10
p=0.25
```

Donner l'allure du graphique qui sera alors obtenu.

RÉPONSE:

Cette fois ci l'histogramme de la distribution du nombre d'arètes doit ressembler à celui de la loi $\mathcal{B}(10.025)$ il va être concentré autour de la moyenne $np = 0.25 \times \dfrac{10 \times 9}{2} = 11.25$ et s'étend de 0 à 45.



*

12. On note N_0 la variable donnant le nombre de sommets adjacents au sommet 0 dans le graphe $G(n,p)$. Donner la loi de N_0.

RÉPONSE:

Le sommet 0 à $n-1$ voisins possibles (tous les autres). Chaque arêts entre 0 et un autre sommet à une probabilité p d'être présente.
N_0 compte le nombre de succès "une arête est présente", de probabilité p lors de la répétition de $n-1$ expériences de Bernouilli indépendantes.

$$\boxed{N_0 \hookrightarrow \mathcal{B}(n-1,p)}$$

*

Sommets isolés

On dit qu'un sommet est isolé si il n'est relié à aucun autre sommet. Le but de cette partie est de calculer le nombre moyen de sommets isolé dans un graphe aléatoire.

Soit c un réel strictement positif. On note $p(n) = \dfrac{\ln n}{n} + \dfrac{c}{n}$, on s'intéresse au graphe aléatoire $G(n, p(n))$ (on utilise la définition de la partie précédente).
Pour $i \in [\![0;\ n-1]\!]$ on note Y_i la variable aléatoire qui vaut 1 si le sommet i est isolé 0 sinon. On note X_n l nombre de sommets isolés du graphe.

13. Écrire une fonction `isole(M)` qui pour un graphe donné par sa matrice M renvoie le nombre sommets isolés.
Par exemple
— `isoles([[0,0],[0,0]])` renvoie 2
— `isoles([[0,0,1],[0,0,0],[1,0,0]])` renvoie 1
— `isoles([[0,1,1],[1,0,0],[1,0,0]])` renvoie 0

RÉPONSE:

```
def isoles(M):
    n=len(M)
    nb=0
    for i in range(n):
        s=0
        for j in range(n):
            s=s+M[i][j]
        if s==0:
            nb=nb+1
    return nb
```

*

14. Donner une relation entre X_n et $Y_0, \dots Y_{n-1}$.

RÉPONSE:

$$\boxed{X_n = Y_0 + Y_1 + \cdots Y_{n-1}}$$

*

15. Donner une expression de $P(Y_0 = 1)$ en déduire $E(Y_1)$

RÉPONSE:

EN utilisant la question 12.
$P(Y_0 = 1) = P(N_0 = 1) = \binom{n-1}{0} p^0 (1-p)^{n-1} = (1-p)^{n-1}$

$$P(Y_0 = 1) = (1-p)^{n-1}$$

Comme Y_0 ne prend que deux valeurs elle suit une loi binomiale, $Y_0 \hookrightarrow \mathcal{B}((1-p)^{n-1})$.
D'après les résultats du cours

$$E(Y_0) = (1-p)^{n-1}$$

*

16. En déduire que $E(X_n) = n\left(1 - \dfrac{c + \ln n}{n}\right)^{n-1}$

RÉPONSE:

Par linéarité de l'espérance

$$E(X_n) = E(X_0) + E(X_1) + \cdots + E(X_{n-1})$$

$$E(X_n) = n\left(1 - \frac{c + \ln n}{n}\right)^{n-1}$$

*

17. Montrer que $\lim\limits_{n\to+\infty} E(X_n) = \mathrm{e}^{-c}$. *Indications :* $\lim_{x\to 0} \dfrac{\ln(1+x)}{x} = 1$. On a aussi pour a strictement positif et b réel $a^b = \mathrm{e}^{b \ln a}$.

RÉPONSE:

$$
\begin{aligned}
E(X_n) &= n\left(1-\frac{c+\ln n}{n}\right)^{n-1}\\
&= n\exp\left((n-1)\ln\left(1-\frac{c+\ln n}{n}\right)\right)\\
&= n\exp\left((n-1)\left(-\frac{c+\ln n}{n}\right)\frac{\ln\left(1-\frac{c+\ln n}{n}\right)}{-\frac{c+\ln n}{n}}\right)\\
&= n\exp\left((n-1)\left(-\frac{c}{n}\right)\frac{\ln\left(1-\frac{c+\ln n}{n}\right)}{-\frac{c+\ln n}{n}}\right)\exp\left((n-1)\left(-\frac{\ln n}{n}\right)\frac{\ln\left(1-\frac{c+\ln n}{n}\right)}{-\frac{c+\ln n}{n}}\right)\\
&= n\exp\left((n-1)\left(-\frac{c}{n}\right)\frac{\ln\left(1-\frac{c+\ln n}{n}\right)}{-\frac{c+\ln n}{n}}\right)\times\exp(-\ln n)\exp\left(\left(-\frac{n-1}{n}\right)\frac{\ln\left(1-\frac{c+\ln n}{n}\right)}{-\frac{c+\ln n}{n}}\right)\\
&= n\exp\left((n-1)\left(-\frac{c}{n}\right)\frac{\ln\left(1-\frac{c+\ln n}{n}\right)}{-\frac{c+\ln n}{n}}\right)\times\frac{1}{n}\exp\left(\left(-\frac{n-1}{n}\right)\frac{\ln\left(1-\frac{c+\ln n}{n}\right)}{-\frac{c+\ln n}{n}}\right)\\
&= \exp\left((n-1)\left(-\frac{c}{n}\right)\frac{\ln\left(1-\frac{c+\ln n}{n}\right)}{-\frac{c+\ln n}{n}}\right)\times\exp\left(\left(-\frac{n-1}{n}\right)\frac{\ln\left(1-\frac{c+\ln n}{n}\right)}{-\frac{c+\ln n}{n}}\right)
\end{aligned}
$$

Or $\lim_{n\to+\infty} -\dfrac{c+\ln n}{n} = 0$, théorème des croissances comparées. donc

$$\lim_{n\to+\infty}\frac{\ln\left(1-\frac{c+\ln n}{n}\right)}{-\frac{c+\ln n}{n}} = 1$$

et on trouve le résultat demandé en utilisant la continuité de la fonction exponentielle.

$$\boxed{\lim_{n\to+\infty} E(X_n) = \mathrm{e}^{-c}.}$$

Remarques Il est beaucoup plus simple d'utiliser un DL 1.

*

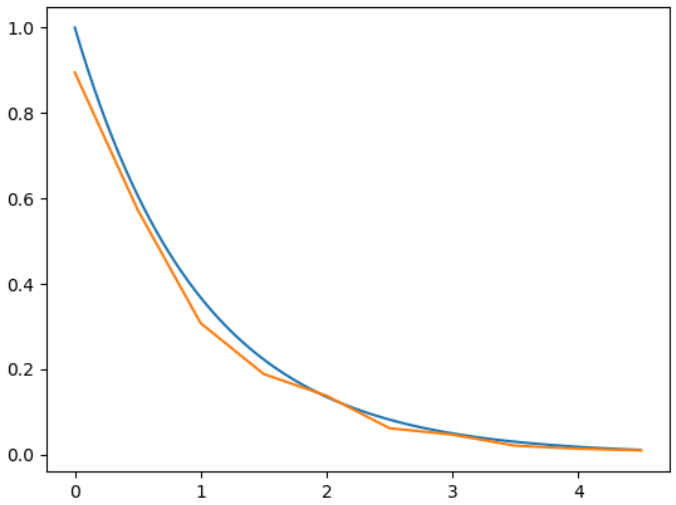
18. Proposer un programme illustrant le résultat précédent.

RÉPONSE:

On peut proposer deux approches.

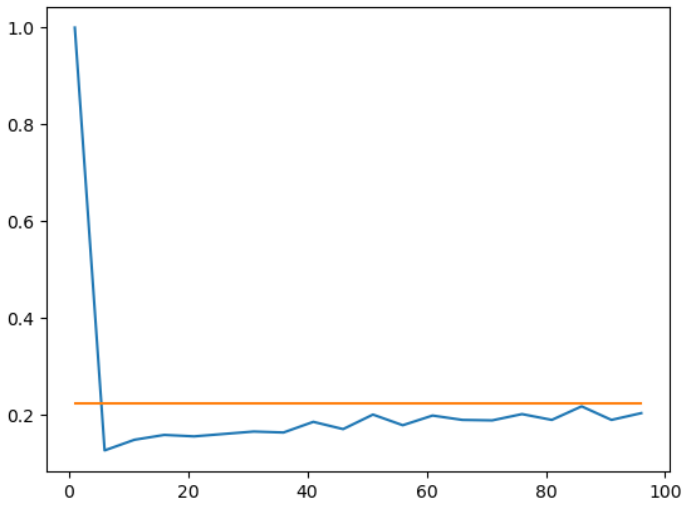
Faire varier c à n fixé On fixe le nombre de sommets à n assez grand. Puis pour plusieurs valeurs de c on construit N (grand) graphes $G(n, p(n))$ et on comptes leur nombre de sommets isolés pour faire une moyenne. On obtient un graphe en fonction de c. Pour comparaison on fait apparaitre le graphe de la fonction $c \mapsto \exp(-c)$

```
n=100  #nombre de sommets des graphes
N=1000 # nombre de graphes générés
C=[x/2 for x in range(10)] # valeurs de c étudiées
 ↪  0,0.5,1....,4.5

M=[]# liste des moyennes
for c in C:
    s=0
    for i in range(N):
        s=s+isoles(graphe(n,(np.log(n)+c)/n))
    M.append(s/N)
# comparaison avec la cour y=exp(-x)

x=np.arange(0,4.5,0.01)
y=np.exp(-x)
plt.plot(x,y)
plt.plot(C,M)
plt.show()
```



Faire varier n à c fixé. On fixe la paramètre c, puis on fait varier n. Pour chaque cas on construit N (grand) graphes pour compter leur nombre de sommets isolés. Le graphe obtenu montre une courbe s'approchant de $exp(-c)$ quand n devient grand.

```
c=1.5
N=1000 # nombre d'expériences
nmax=100
M=[]# liste des moyennes
for n in range(1,nmax,5)   :
    s=0
    for i in range(N):
        s=s+isoles(graphe(n,(np.log(n)+c)/n))
    M.append(s/N)

##
x= [n for n in range(1,nmax,5)]
plt.plot(x,M)
const= [np.exp(-c) for n in range(1,nmax,5)]
plt.plot(x,const)
plt.show()
```



*