⋆ Simulations de Variables Aléatoires Discrètes ⋆

Compétences attendues

- ✓ Savoir simuler une loi discrète usuelle avec les outils fournis par la bibliothèque Numpy, ou uniquement à partir d'une variable de loi uniforme
- ✓ Vérifier graphiquement la pertinence d'une simulation d'une loi (histogrammes, boxplot)

Liste des commandes Python exigibles aux concours

- Dans la librairie `numpy.random` : `random`, `binomial`, `randint`, `geometric`, `poisson`
- Dans la librairie `matplotlib.pyplot` : `hist`, `show`

Dans toute cette feuille technique, on supposera l'import de la bibliothèque `numpy.random` avec le préfixe `rd`.

1 Simulation de lois discrètes usuelles en Python

1.1 En Numpy

Ce qui suit sert de rappel sur les commandes permettant de simuler les principales lois de probabilités discrètes, sous la bibliothèque `numpy.random`.
Les commandes `rd.loi(args)` listées ci dessous permettent de simuler une unique variable aléatoire de loi donnée. Dans le cas où l'on souhaite simuler plusieurs fois cette même loi dans un vecteur/tableau numpy de taille N (respectivement $N \times M$) de cette , il suffit alors de rajouter l'argument de la sorte : `rd.loi(args,n)` (respectivement `rd.loi(args, (N,M))`).

Définition 1.1 [Loi Uniforme Discrète]

La fonction `rd.randint(a,b)` permet de modéliser une loi uniforme $\mathcal{U}([\![a, b-1]\!])$ à valeurs dans $[\![a; b-1]\!] \subset \mathbb{Z}$.

Exemple 1.1. La commande `rd.randint(0,10,(4,3))` permet de créer une matrice à 4 lignes et 3 colonnes dont les coefficients sont des simulations de la loi uniforme sur $[\![0, 9]\!]$.

Définition 1.2 [Loi de Bernoulli et loi Binomiale]

- La fonction `rd.binomial(1,p)` modélise la loi Bernoulli $\mathcal{B}(p)$.
- La fonction `rd.binomial(n,p)` modélise la loi binomiale $\mathcal{B}(n, p)$. Celle-ci renvoie le nombre de succès obtenus après la réalisation de n épreuves de Bernoulli indépendantes. La probabilité de succès à chaque épreuve est donnée par $p \in [0, 1]$.

Exemple 1.2. La commande `rd.binomial(50,0.6,[4,3])` permet de créer une matrice à 4 lignes et 3 colonnes dont les coefficients sont des réalisations de la loi binomiale $\mathcal{B}(50, 0.6)$.

Définition 1.3 [Loi géométrique $\mathcal{G}(p)$]

La fonction `rd.geometric(p)` modélise la loi géométrique $\mathcal{G}(p)$. Elle renvoie le numéro du tirage correspondant au premier succès lors de la répétition de n épreuves de Bernoulli indépendantes et de paramètre p.

Exemple 1.3. La commande `rd.geometric(0.001,[4,3])` permet de créer une matrice à 4 lignes et 3 colonnes dont les coefficients sont des réalisaitons de la loi géométrique $\mathcal{G}(0.001)$.

Définition 1.4 [Loi de Poisson $\mathcal{P}(p)$]

La fonction `rd.poisson(p)` permet de modéliser la loi de Poisson $\mathcal{P}(p)$. On peut présenter cette distribution comme étant une approximation d'une loi binomiale lorsque l'effectif n tend vers l'infini (en pratique, plusieurs dizaines) et la probabilité d'occurrence p tend vers zéro (en pratique, $p < 0, 1$).

Exemple 1.4. La commande `rd.poisson(2,10)` permet de simuler 10 fois l'expérience d'une $\mathcal{P}(2)$.

Cette section pourrait s'arrêter là, étant donné qu'on dispose de commandes pour simuler toutes les lois usuelles discrètes. Notre but dans la suite est de proposer des simulations à l'aide uniquement de la fonction `random` .

1.2 La fonction `random`

Il est alors possible de simuler les lois usuelles précédentes à partir d'une loi uniforme $\mathcal{U}([0; 1])$ c'est à dire en utilisant uniquement la fonction `rd.random()` .

Définition 1.5 [Loi uniforme $\mathcal{U}_{[0,1]}$]

`rd.random()` permet de simuler une variable aléatoire $\mathcal{U}([0; 1])$, elle tire donc un nombre flottant dans l'intervalle $[0; 1]$.

Proposition I.1

1. Pour tout $(a,b) \in [0;1]^2$ tel que $a < b$, la probabilité que le réel renvoyé par `rd.random()` soit compris (strictement ou non) entre a et b vaut $b-a$.
2. Pour tout $p \in [0;1]$, la probabilité que le réel renvoyé par `rd.random()` soit inférieur (strictement ou non) à p vaut p.

Méthode 1.5 [À retenir! Simulation d'un évènement de probabilité p.]

Pour tout $p \in [0;1]$, l'instruction `rd.random() <= p` renvoie le booléen qui prend la valeur `True` avec la probabilité p et la valeur `False` avec la probabilité $1-p$. Cette instruction permet de simuler un évènement de probabilité p.

Remarque 1.1. Pour simuler un évènement de probabilité p, on pourra indifféremment utiliser l'une ou l'autre des instructions suivantes : `rd.random()<p` ou `rd.random()<=p` .

Les commandes suivantes pourront être utiles dans la suite.

Définition 1.6

Si `u` est un vecteur dont les composantes sont des booléens, alors :
- la commande `np.sum(u)` renvoie le nombre de booléens qui ont pris la valeur `True` ;
- la commande `np.mean(u)` renvoie la proportion de booléens qui ont pris la valeur `True`.

Exemple 1.5. Après avoir importé la bibliothèque `numpy`, taper les instructions suivantes :

```
u = rd.random(15) ; u
v = (u <= 0.5) ; v
np.mean(v)
```

1.3 Exercices

Exercice 1 [Loi Uniforme]

Soient n et m deux entiers tels que $n \geq m$. On rappelle que si $U \sim \mathcal{U}([0;1])$ alors on a $V = n + \lfloor (m-n+1)U \rfloor \sim \mathcal{U}(\llbracket n, m \rrbracket)$.

1. Écrire une fonction `uniforme(n,m)` simulant la loi $\mathcal{U}(\llbracket n, m-1 \rrbracket)$.
2. Compléter la fonction suivante afin qu'elle renvoie un vecteur contenant N réalisations indépendantes de loi $\mathcal{U}(\llbracket n, m-1 \rrbracket)$.

```
def Uniforme(n,m,N):
    v = np.zeros(N)
    for k in range(N):
        v[k] = ...
    return v
```

Exercice 2 [Loi de Bernoulli]

1. Compléter la fonction suivante afin qu'elle simule une loi de Bernoulli de paramètre p.

```
def bernouli(p):
    u = 0
    if ...  :
        u = ...
    return u
```

2. Écrire une fonction `Bernoulli(p,N)` renvoyant un vecteur contenant N réalisations indépendantes de la loi $\mathcal{B}(p)$.

Exercice 3 [Loi Binomiale]

1. Écrire une fonction `binomiale(n,p)` simulant la loi $\mathcal{B}(n,p)$.
2. Écrire une fonction `Binomiale(n,p,N)` donnant un échantillon de taille N de la loi $\mathcal{B}(n,p)$.

Exercice 4 [Loi Géométrique]

1. Écrire une fonction `geom(p)` simulant la loi $\mathcal{G}(p)$.
2. Écrire une fonction `Geom(p,N)` donnant un échantillon de taille N de la loi $\mathcal{B}(n,p)$.
3. Simuler un échantillon de taille $N = 10000$ de la loi $\mathcal{G}(0.2)$, puis vérifier que la moyenne de cet échantillon est cohérente avec ce qu'on attend.

2 Représentation graphiques

2.1 Comparaison d'histogrammes et des probabilités théoriques

L'utilisation d'histogramme fait sens lorsque l'on souhaite vérifier la distribution (ou loi) d'un échantillon de valeurs par rapport à une loi théorique candidate. La superposition de la distribution théorique et du diagramme en bâton (en fréquence d'apparitions de valeurs) permet notamment de corroborer notre hypothèse.

Soit X une variable aléatoire discrète. Supposons qu'on dispose d'une fonction `Loi` permettant de simuler la loi de X. Pour juger de la pertinence des simulations, on peut utiliser des représentations graphiques. Pour cela, on procédera comme suit :

1. on crée un *échantillon* de taille N, c'est à dire un vecteur ligne `x` contenant N réalisations de la fonction `Loi` (qui permet de simuler une loi donnée) ;
2. on compare graphiquement les fréquences empiriques obtenues (c'est à dire à grâce à l'échantillon) avec les probabilités théoriques pour vérifier la pertinence de la simulation par `Loi`.

Dans le cas de simulation de variables aléatoires discrètes, on va comparer plus particulièrement :
- le diagramme en bâtons des fréquences de notre échantillon de taille N ;
- le diagramme en bâtons des probabilités théoriques $\mathbb{P}(X = k)$ pour $k \in X(\Omega)$.

Si notre simulation est bonne, on doit constater que :

Proposition II.1 [Théorème d'Or de Bernoulli]

Pour N "suffisamment grand", le diagramme en bâtons des fréquences de l'échantillon doit être proche de celui des probabilités théoriques.

2.2 Commandes Python

On suppose dans cette partie avoir importé `matplotlib.pyplot` comme `plt`.

2.2.1 Diagramme en bâtons des probabilités théoriques

Pour dessiner le diagramme en bâtons des probabilités théoriques, on définit `x` le vecteur contenant (une partie de) $X(\Omega)$ et `y` le vecteur contenant les probabilités théoriques correspondantes. On utilise alors la commande suivante (non exigible) :

Définition 2.1 [Tracer un diagramme en bâtons]

Si `x` et `y` sont des vecteurs de même taille, `plt.bar(x,y)` trace le diagramme en bâtons d'abscisses `x` et d'ordonnées `y`.

2.2.2 Diagramme en bâtons des probabilités théoriques

Pour dessiner le diagramme en bâtons des fréquences de l'échantillon `x`, il nous faut trier notre série par modalités/fréquences, puis tracer le diagramme en bâtons associé. On utilisera pour cela (de manière un peu détournée) la commande suivante :

Définition 2.2 [Tracer un histogramme d'une série statistique]

Si `x` est un vecteur contenant une série statistique et `c` un vecteur contenant les classes choisies, la commandes `plt.hist(x,c)` dessine l'histogramme associé à la série statistique `x` triée selon les classes définies par `c`.

Méthode 2.2 [Tracer le diagramme en bâtons des fréquences en pratique]

Pour tracer un histogramme ou diagramme des fréquences d'un échantillon `x` (qu'on suppose à valeurs entières), on procède de la manière suivante :

1. on décide des modalités $m_1 < m_2 < \cdots < m_k$ qu'on souhaite représenter (dans le cas où elles seraient en nombre infini) ;
2. on définit les classes $c = \{(m_1 - 0,5, m_1 + 0,5), (m_2 - 0,5, m_2 + 0,5), \ldots, (m_k - 0,5, m_k + 0,5)\}$;
3. on dessine l'histogramme (aka diagramme en bâtons des fréquences) à l'aide de la commande

```
plt.hist(x,c, density='True', edgecolor='k', color='...', label="...")
```

où l'on a rajouté les options de tracé suivantes (non exigibles) :
- normalisation des rectangles (la surface totale vaut 1) : `density='True'`
- contours des rectangles en noir : `edgecolor='k'`
- couleur des rectangles : `color='...'` (mettre le nom de la couleur en anglais)
- légende associée à chaque histogramme : `label="..."` (mettre la légende choisie)

Remarque 2.1. Pour réaliser plusieurs graphiques dans une même fenêtre et ainsi pouvoir mieux les comparer, on peut utiliser l'instruction `plt.subplot(1,m,k)` avant chaque instruction de tracé de graphique, qui découpe la fenêtre graphique en 1 ligne et `m` colonnes, `k` indiquant le numéro de la colonne souhaitée pour chaque graphique.

2.3 Exercices

Exercice 1 On se donne le code suivant :

```
# Echantillon
x = Uniforme(1,7,100000)

# Histo
c = np.arange(0.5,7)
plt.subplot(1,2,1)
plt.hist(x,c,density='True', edgecolor='k', color = 'blue', label="Diag. des freq.")

#Diag probas theoriques
u = np.arange(1,7)
v = (1/6)*np.ones(6)
plt.subplot(1,2,2)
plt.bar(u,v, label="Diag. des probas theoriques")

plt.show()
```

Exécuter ce code et interpréter le graphique ainsi obtenu. En particulier, à quoi correspond le vecteur `u` ?

Exercice 2 [Simulation de la loi Binomiale]

1. On considère les instructions suivantes :

   ```
   c=np.ones(n+1)
   c[1:(n+1)] = np.cumprod(np.arange(n,0,-1)/np.arange(1,n+1))
   ```

   Que contient le vecteur `c` à la suite de ces instructions ? Expliquer.
   On pourra commencer par exécuter ces instructions pour $n = 3$, $n = 4$.
2. À l'aide d'un diagramme en bâtons, tester la simulation de la loi $\mathcal{B}(10, 0.2)$ obtenue à l'aide de la fonction `Binomiale`.

Exercice 3 [Simulation d'une loi géométrique à partir d'une loi exponentielle]

Soit $\lambda \in]0; +\infty[$. Il est possible de montrer que si $X \sim \mathcal{E}(\lambda)$, alors $Y = \lfloor X \rfloor + 1 \sim \mathcal{G}(1 - \mathrm{e}^{-\lambda})$.

1. Écrire une fonction `Geom2(p,N)` simulant un échantillon de taille N de la loi $\mathcal{G}(p)$. On utilisera pour cela la fonction `rd.exponential(lamb,N)` pour simuler un échantillon de taille N de la loi exponentielle $\mathcal{E}(\lambda)$.
2. On souhaite comparer avec la loi théorique les simulations d'une variable Y suivant la loi $\mathcal{G}(0.3)$ obtenus à l'aide de `Geom` et `Geom2`. Pour ce faire, simuler $N = 10000$ réalisations d'une telle variable à l'aide de ces deux fonctions, puis tracer les histogrammes correspondants ainsi que le diagramme en bâtons des probabilités théoriques.

On fait remarquer que Y ne prend qu'exceptionnellement des valeurs au delà de 20, si bien qu'on pourra considérer des histogrammes pour des classes se limitant à la valeur 20. On pourra afficher les trois diagrammes de front (pour mieux les comparer), à l'aide de la commande `plt.subplot`.

3 Exercice concours

Exercice 1

On considère une pièce truquée de sorte que le côté "pile" apparaît avec la probabilité $p \in]0, 1[$. On effectue des lancers jusqu'à obtenir une première fois le côté "pile". X étant le nombre de lancers réalisés, on effectue ensuite X nouveaux lancers et on compte le nombre Y de "pile" obtenus au cours de cette nouvelle séquence de lancers.

1. Quelle est la loi de X ?
2. Écrire une fonction `X(p)` qui simule X et qui prend comme argument la probabilité p d'obtenir "pile".
3. Écrire une fonction `simul(p)` qui simule Y.
4. À partir d'un échantillon de 10000 simulations de Y (vous testerez pour plusieurs valeurs de p), représenter graphiquement la suite $(\bar{y}_n)_{1 \leq n \leq 10000}$ de terme général

$$\bar{y}_n = \frac{1}{n} \sum_{k=1}^{n} y_k$$

où $y_1, \ldots, y_{10000}$ sont les valeurs de l'échantillon. Que pouvez-vous en conclure ?