TP de probabilités : Autour de l'aléatoire

---

1 Modalités du TP

— Ce TP est réalisé sous Python et par binôme
— Chaque binôme dépose un compte-rendu sur Moodle sous forme vidéo (diaporama commenté, film...)
— Il est attendu que les deux membres du binôme prennent la parole
— L'évaluation portera sur : la qualité de la programmation, l'analyse des problèmes posés et la restitution (écrit et oral)
— La date limite de dépôt du compte-rendu est le **dimanche 31 mars 2024 à 23h59**.

2 Objectifs du TP

— Savoir générer un échantillon d'une variable aléatoire X de loi de probabilité connue ;
— Illustrer l'inégalité de Chebychev, la loi forte des grands nombres et le théorème de la limite centrale.

3 Générer des jeux de données

Soit X une variable aléatoire distribuée selon une loi de probabilité connue (loi binomiale, loi exponentielle, loi normale, etc). On souhaite développer une application qui permette de générer une suite d'observations $(X_1, X_2, \ldots, X_n)$ de X, telle que si n est suffisamment grand, la proportion d'observations dans l'intervalle $[a, b]$ soit proche de $P(X \in [a, b])$:

$$\frac{\#\{X_i \in [a, b]\}}{n} \approx P(X \in [a, b])$$

On procède pour cela en deux étapes :

1. on génère des observations d'une variable aléatoire U uniformément distribuée sur l'intervalle $[0, 1]$;
2. on utilise ensuite une transformation adéquate de U pour revenir à la variable aléatoire X de loi de probabilité souhaitée.

Pour simuler une loi de probabilités quelconque à partir de la loi uniforme, on dispose d'une multitude d'outils dont :

1. Méthode de la transformation inverse
2. Transformations simples
3. Méthode de rejet-acceptation

3.1 Méthode de la transformation inverse

Soit U une variable aléatoire de loi uniforme sur $[0,1]$ et F la fonction de répartition de la loi que l'on souhaite simuler. La variable aléatoire $X = F^{-1}(U)$ est distribuée selon la loi souhaitée, avec F^{-1} l'inverse à droite de F, c'est-à-dire :

$$F^{-1}(\alpha) = \inf\{x,\ F(x) > \alpha\}.$$

A) Quelques variables continues

Quelques lois continues dont les fonctions de répartition sont explicitement inversibles :

Loi	Densité f	Fonction de répartition F	Transformation
Exponentielle(λ)	$\lambda e^{-\lambda x}\ x > 0$	$1 - e^{-\lambda x}$	$-\frac{1}{\lambda}\log(U)$
Weibull(a), $a > 0$	$ax^{a-1}e^{-x^a},\ x > 0$	$1 - e^{-x^a}$	$(\log(1/U))^{1/a}$
Gumbel	$e^{-x}e^{-e^{-x}}$	$e^{-e^{-x}}$	$-\log(\log(1/U))$
Logistique	$\frac{1}{2+e^x+e^{-x}}$	$\frac{1}{1+e^{-x}}$	$-\log((1-U)/U)$
Cauchy	$\frac{1}{\pi(1+x^2)}$	$1/2 + (1/\pi)\arctan x$	$\tan(\pi U)$
Pareto(a), $a > 0$	$\frac{a}{x^{a+1}},\ x > 1$	$1 - 1/x^a$	$1/U^{1/a}$

B) Quelques variables discrètes

Nous allons utiliser ici la méthode de la transformation inverse dans trois contextes différents : celui d'une simple distribution de probabilités (cas 1), celui où l'inverse de la fonction de répartition est explicitement connue (cas 2) et celui où elle ne l'est pas (cas 3).

Cas 1 :

Pour générer une vad X de loi de distribution :

$$P(X = x_i) = p_i, \quad i = 1, \ldots, n.$$

on pourra utiliser l'algorithme suivant :

1. Générer un nombre aléatoire U
2. Transformer U en X comme suit

$$X = x_j \quad \text{si} \sum_{i=1}^{j-1} p_i \leq U < \sum_{i=1}^{j} p_i$$

Exemple : Simuler la v.a.d. X de loi de probabilité

$$p_1 = 0.2,\ p_2 = 0.15,\ p_3 = 0.25,\ p_4 = 0.40$$

Algorithme :

1. Générer un nombre aléatoire U
2. Si $U < p_1 = 0.2$, poser $X \leftarrow x_1$ et stop
3. Sinon si $U < p_1 + p_2 = 0.35$, poser $X \leftarrow x_2$ et stop
4. Sinon si $U < p_1 + p_2 + p_3 = 0.60$, poser $X \leftarrow x_3$ et stop
5. Sinon poser $X \leftarrow x_4$ et stop

Le coût de cet algorithme est proportionnel au nombre d'intervalles à parcourir ; aussi est-il plus avantageux de considérer les valeurs x_i dans l'ordre décroissant des p_i d'où l'algorithme suivant plus performant :

1. Générer un nombre aléatoire U
2. Si $U < p_4 = 0.4$, poser $X \leftarrow x_4$ et stop
3. Sinon si $U < p_4 + p_3 = 0.65$, poser $X \leftarrow x_3$ et stop
4. Sinon si $U < p_4 + p_3 + p_1 = 0.85$, poser $X \leftarrow x_1$ et stop
5. Sinon poser $X \leftarrow x_2$ et stop

Cas 2 : Loi géométrique

X suit une loi géométrique de paramètre p si sa distribution de probabilité s'écrit :

$$P(X = i) = q^{i-1}p, \quad i \geq 1,\ q = (1-p)$$

Cas modèle : X représente le nombre de prélèvements à effectuer pour trouver une pièce défectueuse, dans une population où la proportion de pièces défectueuses est p. La fonction de répartition de X s'écrit :

$$\begin{aligned} F(i) &= P(X \leq i) \\ &= 1 - P(X > i) \\ &= 1 - P(i \text{ premiers tirages soient des pièces non défectueuses}) \\ &= 1 - q^i \end{aligned}$$

Transformation : $X = j$ si

$$\begin{aligned} F(j-1) \leq U < F(j) &\Leftrightarrow 1 - q^{j-1} \leq U < 1 - q^j \\ &\Leftrightarrow q^j \leq 1 - U < q^{j-1} \end{aligned}$$

Ainsi

$$\begin{aligned} X &= \min\{j | q^j < 1 - U\} \\ &= \min\{j | j \log(q) < \log(1-U)\} \\ &= \min\{j | j > \tfrac{\log(1-U)}{\log(q)}\} \end{aligned}$$

c'est-à-dire

$$X = \left[\frac{\log(1-U)}{\log(1-p)}\right].$$

où $[z]$ désigne le plus petit entier supérieur ou égal à z et log le logarithme népérien.

Cas 3 : Loi binomiale

X suit une loi binomiale de paramètres n, p si sa distribution de probabilité s'écrit :

$$p_i = P(X = i) = \binom{n}{i} p^i (1-p)^i, \quad i = 0, \ldots, n$$

Cas modèle : X représente le nombre de pièces défectueuses dans un échantillon de taille n prélevé avec remise dans une population dont le taux de pièces défectueuses est p. Le tirage de l'échantillon peut se faire sans remise à condition que le taux de sondage soit inférieur à 10%.

3.2 Transformations simples :

Il s'agit d'utiliser les fonctions et opérations usuelles comme les fonctions trigonométriques, exponentielle, logarithmique pour se ramener de la loi de distribution uniforme à la loi souhaitée.

Loi normale et méthode de Box-Jenkins

Soient U_1 et U_2 deux variables aléatoires indépendantes de loi uniforme sur $[0,1]$. Les variables aléatoires

$$X = \cos(2\pi U_1)\sqrt{-2\log(U_2)}$$

et

$$Y = \sin(2\pi U_1)\sqrt{-2\log(U_2)}$$

sont indépendantes et distribuées selon la loi normale centrée-réduite $\mathcal{N}(0,1)$.

Loi de Student et méthode de Bailey

Soient U_1 et U_2 deux variables aléatoires indépendantes de loi uniforme sur $[0,1]$. La variable aléatoire

$$X = \sqrt{a\left(U_1^{-\frac{2}{a}} - 1\right)}\cos(2\pi U_2)$$

suit une loi de Student de paramètre a.

4 Travail à faire

4.1 Simulation de lois

1. Écrire sous Python des fonctions permettant de simuler les lois suivantes :
   (a) Loi uniforme sur un intervalle $[a,b]$ quelconque
   (b) Bonus : loi donnée dans l'exemple donné dans le cas 1, page 2 (cliquer ici)
   (c) Bonus : loi géométrique (cliquer ici)
   (d) loi binomiale (cliquer ici)
   (e) loi exponentielle (cliquer ici)
   (f) loi normale centrée-réduite (cliquer ici)
   (g) loi normale quelconque
2. Utiliser ces fonctions pour générer des jeux de données de taille n choisie et pour chacun, donner une synthèse graphique :
   — par histogramme pour les lois continues : utiliser la règle de Sturges pour le choix du nombre de classes $k = 1 + [10/3 * log_{10}(n)]$ où [.] désigne la partie entière ;
   — par diagramme en bâtons pour les lois discrètes.

4.2 Inégalité de Bienaymé-Chebychev

Tirer N réalisations indépendantes $X_1, \ldots, X_n$ d'une loi de votre choix. Vérifier l'inégalité de Bienaymé-Chebychev : pour tout réel $a > 0$,

$$P(|X - E(X)| \geq a) \leq \frac{\sigma^2}{a^2}$$

Commenter.

4.3 Loi forte des grands nombres

Tirer N réalisations indépendantes $X_1, \ldots, X_n$ d'une loi exponentielle et tracer le graphique donnant

$$\boxed{k \mapsto \frac{X_1 + \cdots + X_k}{k}}$$

Commenter.

4.4 Théorème de la limite centrale (Bonus)

Choisir un entier n assez grand. Tirer n réalisations indépendantes $X_1, \ldots, X_n$ d'une loi uniforme sur $[0, 1]$ et calculer

$$\boxed{Y = \frac{X_1 + \cdots + X_n - n/2}{\sqrt{n/12}}.}$$

Choisir un entier N assez grand. Recommencer N fois et tracer un histogramme de la loi de Y. Superposer la densité d'une loi normale centrée réduite. Jouer sur les paramètres n et N. Commenter.

4.5 Python et la génération de nombres aléatoires

Disposer d'un générateur de nombres aléatoires permet de réaliser des simulations ou d'accéder à des techniques basées sur le ré-échantillonnage (bootstrap, etc.). Tant SciPy que NumPy proposent des outils à cet effet.

Quelques modules :
import numpy as np
import scipy.stats as stat
import matplotlib.pyplot as plt
import random

Quelques fonctions :
np.random.sample, np.random.normal, np.random.exponential,stat.norm.rvs, stat.expon.rvs, plt.hist, plt.show()