Université Sorbonne Paris Nord, Institut Galilée
Année universitaire 2020–2021
mallein@math.univ-paris13.fr

TP de Probabilités – Aléa dans un programme, génération de variables aléatoires

1 Nombres pseudo-aléatoires

Un programme informatique est une suite d'instructions déterministes réalisées par un ordinateur. Il arrive qu'on ait besoin, ou que l'on souhaite, utiliser des entrée aléatoires. Nous explorons ici les nombres pseudo-aléatoires, qui sont une méthode utilisée pour répondre à ce besoin.

1.1 Loi uniforme

Comment un ordinateur simule-t-il une suite de variables aléatoires indépendantes et uniformes sur $[0,1]$? La question est loin d'être triviale puisqu'un ordinateur réalise facilement une suite d'instructions données, il n'existe pas de composant dans un ordinateur classique qui retourne un bit aléatoire lorsqu'on l'interroge.

Plutôt que de générer des variables aléatoires, on se satisfait alors de nombres *pseudo-aléatoires*, qui sont des suites de nombres déterministes qui imitent certaines propriétés des nombres aléatoires. Dans le cas de Python, plutôt que de générer des variables uniformes sur $[0,1]$, on génère des entiers pseudo-aléatoires dans $E := \{1, 2, \dots, M\}$, que l'on divise ensuite par M comme suit. On part d'un état $x_0 \in E$ (la graine, ou seed en anglais) qui est un nombre que l'on pense "choisi au hasard" (souvent initialisé en comptant le nombre de secondes écoulées depuis une date donnée). On applique ensuite successivement une même fonction $f : E \to E$ (par exemple $f(x) = ax + c$ modulo M, avec a et c de grands nombres choisis de façon adéquate). La suite définie par $x_{n+1} = f(x_n)$ et $U_n = x_n/M$ fourni les variables pseudo-aléatoires uniformes sur $[0,1]$ recherchées.

Cette transformation doit bien sûr vérifier plusieurs propriétés : on veut que la suite générée "ressemble" à une suite de variables aléatoires uniformes et indépendantes sur $[0,1]$. Dans quel sens cette suite "ressemble"-t-elle à des variables uniformes ? La suite doit pour ce faire réussir à passer une série de tests statistiques d'indépendance et d'adéquation à la loi uniforme. En particulier, la période (c'est-à-dire le plus petit entier k tel que $f^{(k)}(x) = x$ doit être très (très (très)) grande. Dans le cas de Python, on utilise l'algorithme appelé "Mersenne Twister" (développé par Makoto Matsumoto et Takuji Nishimura en 1997), qui possède une période de $2^{19937} - 1$.

Ainsi, existe-t-il des vrais générateurs de nombres aléatoires ?
NON, pas dans un ordinateur classique, où tout résultat est déterministe !
MAIS on a des générateurs de nombres construits pour passer les tests statistiques de nombres aléatoires.
CEPENDANT des générateurs certifiés de nombres aléatoires existent aussi (c.f. random.org)

Exercice 1. Exécuter plusieurs fois le code ci-dessous. Expliquer les résultats obtenus.

```
import numpy.random as npr

print(npr.rand())
npr.seed(seed=1)
print(npr.rand())
print(npr.rand())
npr.seed(seed=1)
print(npr.rand())
print(npr.rand())
```

La fonction `npr.rand` peut prendre des argument optionnels. Que renvoit `npr.rand(10)` ? `npr.rand(5,3)` ? `npr.rand(2,3,4)` ?

1.2 Histogramme et densité

Pour vérifier “à la main” que `npr.rand` retourne bien des variables aléatoires de loi uniforme, on pourra exécuter le code suivant :

```
import numpy.random as npr
import matplotlib.pyplot as plt

X = npr.rand(1000)
plt.hist(X, bins = 20, range = (0,1), density = True, color = 'blue')
plt.xlabel('Domaine')
plt.ylabel('Densité')
plt.title('Histogramme de npr.rand()')
plt.show()
```

À partir de maintenant, on consièrera toujours que l'on dispose d'une “boîte noire” (la fonction `npr.rand`) permettant de simuler des variables aléatoires i.i.d. de loi uniforme sur $[0, 1]$. Grâce à cette boîte noire, nous allons maintenant voir comment simuler des variables aléatoires de loi donnée.

2 Simulation de variables aléatoires

Plusieurs méthodes permettent de simuler des variables aléatoires. Certaines sont basées sur le calcul de la fonction de répartition d'une variable aléatoire, et de son inverse continu à droite (c.f. exercice 2.7.1 du poly de cours). D'autres utilisent la *méthode de rejet*, qui permet d'éviter ce calcul lorsqu'il est trop complexe. D'autres encore utilisent les propriétés particulières des variables aléatoires considérées.

2.1 Lois discrètes

On considère dans un premier temps des méthodes permettant de simuler des variables aléatoires de loi discrète.

2.2 Loi de Bernoulli

La loi de probabilité la plus élémentaire à simuler est la loi de Bernoulli de paramètre p. On montre sur cet exemple comment utiliser les nombres pseudo-aléatoires, de loi uniforme, pour simuler ces variables.

Exercice 2 (Loi de Bernoulli)**.** Soit U une variable aléatoire de loi uniforme sur $[0, 1]$ et $p \in [0, 1]$.

1. Quelle est la loi de $X = \mathbf{1}_{\{U<p\}}$?
2. Complétez le code ci-dessous pour que la fonction `bernoulli(p)` retourne à chaque appel une variable aléatoire de loi Bernoulli de paramètre p.

```
import numpy.random as npr

def bernoulli(p):
    if npr.rand() < p:
        #BLA
    else:
        #BLA

print(bernoulli(p))
print(bernoulli(p))
print(bernoulli(p))
```

On peut modifier la définition de la fonction `bernoulli` pour qu'elle tienne sur une ligne, avec des fonctions `lambda`, mais nous ne nous plongerons pas dans ces méthodes de définition alternative.

Exercice 3 (Suite de variables i.i.d.)**.**

1. Que retourne la commande `npr.rand(10) < 0.5` ? Et `(npr.rand(10)<0.5)*1` ? Expliquez.
2. Modifiez la fonction `bernoulli` pour qu'elle prenne en arguments deux nombres $p \in (0,1)$ et $n \in \mathbb{N}$ et retourne un tableau de n variables de loi de Bernoulli de paramètre p.
3. Utiliser la fonction `plt.hist` pour vérifier la loi de ces variables.

2.2.1 Loi à support fini

On généralise la méthode de simulation précédente à des variables aléatoires ayant un support fini.

Exercice 4 (Loi à support fini)**.** Soit U une variable aléatoire de loi uniforme sur $[0,1]$ et μ une distribution de probabilité sur $\{0,1,\dots N\}$.

1. Construisez une fonction mesurable f telle que $f(U)$ est une variable aléatoire de loi μ.
2. Créez une fonction `simulerVA` qui prend en entrée une liste `mu`, et qui retourne en sortie une variable aléatoire Z telle que $\mathbf{P}(Z=i) =$`mu[i]`.
3. Modifiez votre fonction pour que `simulerVA` permette de simuler un tableau de copies indépendantes de Z.
4. Vérifiez que votre fonction fonctionne correctement en utilisant la fonction `plt.hist`.
5. Exécutez le code suivant. Quel théorème du cours illustre-t-il ?

```
mu = [0.1,0.2,0.2.0.4,0.1]
N=10000

X = simulerVA(mu,N)
print(sum(X)/N)
```

6. Exécutez le code suivant. Quel théorème du cours illustre-t-il ? Essayez de modifier les paramètres utilisés.

```
from math import sqrt
import numpy as np

...

mu = [0.1 ,0.2 ,0.2,0.4 ,0.1]
esp = sum([i*mu[i] for i in range(len(mu))])
# L ' espérance d'une variable de loi mu est esp
N =10000
n=1000

Y = np.zeros(N)
for j in range(N):
    X = simulerVA(mu,n)
    Y[j] = (sum(X) - n*esp)/sqrt(n)

plt.hist (Y, bins = 20, density = True)
plt.xlabel ('Domaine')
plt.ylabel ('Densité')
plt.show ()
```

Exercice 5 (Loi binomiale)**.** On cherche à modéliser une variable aléatoire de loi binomiale de paramètres n et p.

1. Proposer une première fonction `binomiale1` basée sur la méthode décrite dans l'exercice 4 pour simuler une variable de loi binomiale de paramètres `n` et `p`.
2. En utilisant qu'une binomiale est une somme de variables de Bernoulli i.i.d. proposer une deuxième fonction `binomiale2` pour simuler cette loi.
3. Comparer la vitesse d'exécution de ces deux fonctions en utilisant la fonction `time` du module `time`.

4. En utilisant le théorème central limite, proposer une troisième façon de simuler une variable de loi binomiale, en utilisant la fonction `npr.randn` qui simule des variables aléatoires de loi normale centrée réduite, fonctionnant pour de grandes valeurs de n.

2.2.2 Loi à support infini

Dans les cas de lois discrète à support infini, on peut utiliser la méthode de la fonction de répartition détaillée plus haut, ou bien utiliser dans certain cas des propriétés ad-hoc de la loi pour permettre de la simuler plus simplement.

Exercice 6 (Loi géométrique)**.** On cherche à modéliser la loi géométrique de paramètre p.

1. En utilisant qu'une variable aléatoire de loi géométrique est le temps de première réussite d'une expérience ayant probabilité p de réussir, construire une première fonction `geometrique1` qui simule une variable aléatoire de loi géométrique.
2. Proposer une seconde méthode utilisant le calcul de la fonction de répartition de la loi géométrique.

Exercice 7 (Loi de Poisson)**.** Soit $(U_n, n \geq 0)$ une suite de variables aléatoires indépendantes de loi uniforme sur $[0,1]$ et $\lambda > 0$. On pose

$$T = \inf\{n \geq 0 : (-\log(U_0)) + \cdots + (-\log U_n) \geq \lambda\}.$$

1. Déterminer la loi de T.
2. Proposer une méthode de simulation d'une loi de Poisson de paramètre λ.

2.3 Lois à densité

On passe maintenant au cas des lois continues, ayant une densité par rapport à la mesure de Lebesgue. De la même façon qu'avec les lois discrètes, les trois méthodes pricipales de simulations de variables, par inversion de la fonction de répartition, méthode de rejet ou par des méthodes ad-hoc, peuvent être employées, en fonction de la situation.

Exercice 8 (Inversion de la fonction de répartition)**.**

1. Soit U une variable aléatoire de loi uniforme.
   (a) Rappeler la loi de $-\log U$.
   (b) Proposer une méthode de simulation d'une variable aléatoire de loi exponentielle de paramètre λ.
2. Soit $\alpha > 1$. Proposer une méthode permettant de simuler une variable aléatoire ayant pour densité $\alpha/x^{\alpha+1}\mathbf{1}_{\{x>1\}}$ par rapport à la mesure de Lebesgue.

Exercice 9 (Méthode de rejet)**.** On considère une loi de probabilité ayant une densité f par rapport à la mesure de Lebesgue, dont on ne sait pas calculer la fonction de répartition. On suppose qu'il existe une constante $c > 0$ et une fonction de densité g telle que $f \leq cg$, et telle qu'on sait simuler des variables aléatoires dont la loi a densité g.

1. Montrer que nécessairement, on a $c \geq 1$.
2. Soit $(Y_n, n \geq 1)$ une suite de v.a. i.i.d. ayant pour densité g, et $(U_n, n \geq 1)$ une suite de variables aléatoires i.i.d. de loi uniforme sur $[0,1]$. On pose $T = \inf\{n \geq 1 : U_n \leq f(Y_n)/cg(Y_n)\}$.
   (a) Déterminer la loi de T.
   (b) Montrer que la variable Y_T a pour densité la fonction f.
3. En déduire une méthode permettant de simuler des variables aléatoire ayant la densité f.
4. Application : construire une fonction `demicercle`, qui permet de simuler une variable aléatoire ayant pour densité la fonction $\frac{2}{\pi}\sqrt{1-x^2}\mathbf{1}_{\{x\in[-1,1]\}}$.

Exercice 10 (Loi normale, méthode ad-hoc)**.** Soit (X, Y) un couple de variables i.i.d. de loi $\mathcal{N}(0,1)$.

1. Déterminer la loi de $\sqrt{X^2+Y^2}$.
2. Montrer que $\arctan(Y/X)$, $\sqrt{X^2+Y^2}$ forme une paire de variables indépendantes, dont on déterminera les lois.
3. En déduire une méthode permettant de simuler une variable aléatoire de loi normale centrée réduite.
4. Application : construire une fonction `gaussienne`, qui permet de simuler une variable aléatoire de loi normale, dont on précise en paramètre la moyenne et la variance.