

Plaque 5 - Simulation aléatoire et problèmes de synthèse

Python et calcul scientifique — Licence 1 (L1)

Corrections détaillées

Boîte à outils

Module random : `random.random()` (uniforme sur $[0, 1[$), `random.randint(a, b)` (entier entre a et b inclus), `random.choice(L)`, `random.shuffle(L)`, `random.seed(k)`.

- **Fréquence** F_N d'un événement de probabilité p sur N essais indépendants : $\mathbb{E}(F_N) = p$, et

$$\sigma(F_N) = \sqrt{\frac{p(1-p)}{N}} \leq \frac{1}{2\sqrt{N}}$$

- **Moyenne empirique** d'une variable de variance σ^2 : écart type $\frac{\sigma}{\sqrt{N}}$; un écart de plus de 3 écarts types signale presque sûrement une erreur.
- **Inversion** : si F est la fonction de répartition et U uniforme, $F^{-1}(U)$ suit la loi de fonction de répartition F .
- **Mélange uniforme** : l'algorithme de Fisher-Yates produit chacune des $n!$ permutations avec probabilité $\frac{1}{n!}$.

Correction de l'exercice 1 - Tester un dé simulé

a) Chaque face a la probabilité $\frac{1}{6}$: on attend $\frac{6000}{6} = 1000$ occurrences de chaque face.

b) Le nombre de « 5 » suit $B(6000, \frac{1}{6})$:

$$\sigma = \sqrt{6000 \times \frac{1}{6} \times \frac{5}{6}} \approx 28,9$$

- $\frac{45}{28,9} \approx 1,6$ écart type : fluctuation tout à fait normale (le plus grand des six écarts dépasse souvent $1,5\sigma$).

c) Les indices vont de 0 à 6 ; on utilise les cases 1 à 6 pour que `compte[f]` corresponde à la face f (la case 0 reste vide).

d) Sept valeurs équiprobables : environ 857 par face et 857 « zéros » dans la case 0. L'écart $1000 - 857 = 143$ vaut près de 5σ : détecté sans ambiguïté.

Réponse. Environ 1000 par face, $\sigma \approx 29$; l'écart de 45 ($1,6\sigma$) est normal.

Correction de l'exercice 2 - Marche aléatoire : la loi en racine carrée

a)

```
import random
```

```
random.seed(2)
N, T = 10000, 100
s2 = 0
for k in range(N):
    S = 0
    for t in range(T):
        S = S + random.choice([-1, 1])
    s2 = s2 + S * S
```

```
print(s2 / N)
```

b) $S_T = X_1 + \dots + X_T$, $\mathbb{E}(S_T) = 0$ et, par indépendance, $\mathbb{E}(S_T^2) = V(S_T) = T = 100$ ✓ (écart relatif 0,5 %).

c) La distance quadratique moyenne vaut

$$\sqrt{\mathbb{E}(S_T^2)} = \sqrt{T}$$

— soit 10 pas pour $T = 100$.

d) $\sqrt{10\,000} = 100$ pas, contre 10 000 en ligne droite : c'est la lenteur caractéristique de la **diffusion** (une goutte d'encre s'étale en $\sqrt{t}$).

Réponse. $\mathbb{E}(S_T^2) = T = 100$ (simulé 99,5) ; distance typique $\sqrt{T}$.

Correction de l'exercice 3 - Anniversaires : simuler pour contrôler

a)

```
import random

random.seed(3)
N = 10000
succes = 0
for k in range(N):
    jours = [random.randint(1, 365) for i in range(23)]
    if len(set(jours)) < 23:
        succes = succes + 1
print(succes / N)
```

b) $\sigma = \sqrt{\frac{0,507 \times 0,493}{10\,000}} \approx 0,005$; écart observé 0,0002, soit $0,04\sigma$: excellent accord.

c) $2\sqrt{\frac{0,25}{N}} \leq 0,001$:

$$N \geq \frac{4 \times 0,25}{10^{-6}} = 10^6$$

— Cent fois plus de tirages pour une décimale de plus.

d) Un ensemble ne garde qu'un exemplaire de chaque valeur : il y a une coïncidence si et seulement si l'ensemble a **moins** de 23 éléments.

Réponse. Simulé 0,5071, exact 0,5073 ($\sigma \approx 0,005$) ; 10^6 tirages pour $\pm 0,001$.

Correction de l'exercice 4 - Estimer e avec des permutations

a)

```
import random, math

random.seed(4)
N, n = 100000, 10
sans = 0
for k in range(N):
    p = list(range(n))
    random.shuffle(p)
    if all(p[i] != i for i in range(n)):
        sans = sans + 1
print(sans / N, N / sans)
```

b) $\frac{D_{10}}{10!} = \sum_{k=0}^{10} \frac{(-1)^k}{k!} \approx 0,367879$, égal à $\frac{1}{e}$ à 10^{-7} près.

c) La fréquence estime $\frac{1}{e}$, son inverse estime $e \approx 2,71828$: on obtient 2,730.

d) $\sigma(F) = \sqrt{\frac{0,368 \times 0,632}{10^5}} \approx 1,5 \times 10^{-3}$; par la dérivée de $x \mapsto \frac{1}{x}$:

$$\sigma\left(\frac{1}{F}\right) \approx \frac{\sigma(F)}{F^2} \approx \frac{1,5 \times 10^{-3}}{0,135} \approx 0,011$$

— L'écart $2,730 - 2,718 = 0,012$ est de l'ordre d'un écart type : cohérent.

Réponse. Fréquence 0,366 pour $\frac{1}{e} \approx 0,368$; $e \approx 2,73$ à $\pm 0,01$.

Correction de l'exercice 5 - Volume des boules en grande dimension

a)

```
import random

random.seed(5)
def volume(d, N):
    dedans = 0
    for k in range(N):
        if sum(random.uniform(-1, 1) ** 2 for i in range(d)) <= 1:
            dedans = dedans + 1
    return 2 ** d * dedans / N
```

b) $d = 3$: 0,04 % ; $d = 5$: 0,33 % ; $d = 10$: 2,0 %.

c) La proportion $p = \frac{V_d}{2^d}$ s'effondre : $\frac{2,55}{1024} \approx 0,25$ % pour $d = 10$, soit environ 250 points dans la boule. L'erreur relative vaut

$$\frac{\sigma(F)}{p} \approx \frac{1}{\sqrt{Np}} \approx \frac{1}{\sqrt{250}} \approx 6 \%$$

— (l'erreur observée, 2 %, est dans cette marge).

d) Non : la boule inscrite n'occupe qu'une fraction infime du cube ; presque tout le volume est dans les « coins ». C'est un aspect de la **malédiction de la dimension**, qui oblige à des méthodes de tirage plus astucieuses.

Réponse. Erreurs 0,04 %, 0,3 %, 2 % ; en dimension 10 seuls 0,25 % des points tombent dans la boule.

Correction de l'exercice 6 - Simuler une loi par inversion

a) $F(x) = \int_0^x 2t dt = x^2$ sur $[0, 1]$, $F^{-1}(u) = \sqrt{u}$. Et

$$\mathbb{P}(\sqrt{U} \leq x) = \mathbb{P}(U \leq x^2) = x^2 = F(x)$$

b) $\mathbb{E}(X) = \int_0^1 2x^2 dx = \frac{2}{3} \approx 0,6667$; $F(0,5) = 0,25$. Écarts 0,0008 et 0,0011, de l'ordre de $\frac{1}{\sqrt{10^5}} \approx 0,003$: cohérent.

c) $\mathbb{P}(\max \leq x) = \mathbb{P}(U_1 \leq x)\mathbb{P}(U_2 \leq x) = x^2$: même fonction de répartition.

d)

```
import random, math

random.seed(6)
N = 100000
X = [math.sqrt(random.random()) for k in range(N)]
```

```
print(sum(X) / N, sum(1 for x in X if x <= 0.5) / N)
```

Réponse. $F(x) = x^2$, $X = \sqrt{U}$; $\mathbb{E}(X) = \frac{2}{3}$, $\mathbb{P}(X \leq \frac{1}{2}) = \frac{1}{4}$; aussi $\max(U_1, U_2)$.

Correction de l'exercice 7 - Collectionneur : théorie et simulation

a)

```
import random

random.seed(7)
def collection(n):
    vus = set()
    t = 0
    while len(vus) < n:
        vus.add(random.randint(1, n))
        t = t + 1
    return t

print(sum(collection(6) for k in range(10000)) / 10000)
```

b) $6\left(1 + \frac{1}{2} + \dots + \frac{1}{6}\right) = 14,7$ (plaque de probabilités) ; écart 0,04.

c) Écart type de la moyenne :

$$\frac{\sqrt{38,99}}{\sqrt{10\,000}} \approx 0,062$$

— L'écart 0,04 représente 0,7 écart type : normal.

d) Ajouter à un ensemble ignore les doublons et le test d'appartenance est quasi instantané ; avec une liste, il faudrait tester *if x not in L*, dont le coût croît avec la taille de la liste.

Réponse. Théorie 14,7, simulation 14,74 ; écart type de la moyenne $\approx 0,06$.

Correction de l'exercice 8 - Taux de couverture d'un intervalle

a)

```
import random, math

random.seed(9)
p, n, M = 0.3, 400, 1000
bons = 0
for k in range(M):
    f = sum(1 for i in range(n) if random.random() < p) / n
    if f - 1 / math.sqrt(n) <= p <= f + 1 / math.sqrt(n):
        bons = bons + 1
print(bons / M)
```

b) $\sigma(F_n) \leq \frac{1}{2\sqrt{n}}$, donc $\frac{1}{\sqrt{n}} \geq 2\sigma$; par l'approximation normale, $|F_n - p| \leq 2\sigma$ avec probabilité $\approx 95\%$ au moins.

c) $\sigma = \sqrt{\frac{0,3 \times 0,7}{400}} \approx 0,0229$ et

$$\frac{1/\sqrt{400}}{0,0229} = \frac{0,05}{0,0229} \approx 2,18 \text{ écarts types}$$

— ce qui donne un taux d'environ 97 % (loi normale) ; le résultat 98,1 % est compatible (écart type du taux sur 1000 répétitions : $\approx 0,5\%$).

d) C'est la proportion, **sur un grand nombre de répétitions**, des intervalles qui contiennent p . Une fois l'intervalle calculé, p (fixe) y est ou n'y est pas : la probabilité porte sur la **méthode**, pas sur un intervalle particulier.

Réponse. Taux simulé 98,1 % ; demi-largeur $\approx 2,18\sigma$ (taux théorique ≈ 97 %).

Correction de l'exercice 9 – Monty Hall par simulation

a)

```
import random

random.seed(10)
N = 30000
gagne_change = 0
for k in range(N):
    voiture = random.randint(1, 3)
    choix = random.randint(1, 3)
    ouvertes = [d for d in (1, 2, 3) if d != choix and d != voiture]
    ouverte = random.choice(ouvertes)
    autre = [d for d in (1, 2, 3) if d != choix and d != ouverte][0]
    if autre == voiture:
        qagne_change = qagne_change + 1
print(gagne_change / N)
```

b) Théorie $\frac{2}{3}$; $\sigma = \sqrt{\frac{2/9}{30000}} \approx 0,0027$; écart 0,003, soit $1,1\sigma$ ✓

c) Remplacer la liste *ouvertes* par les deux portes différentes du choix. Si l'animateur ignore l'emplacement, sachant qu'il a ouvert une porte perdante, les deux portes fermées restantes sont **équiprobables** :

$$\mathbb{P}(\text{gagner en changeant} \mid \text{porte ouverte perdante}) = \frac{1}{2}$$

— C'est l'**information** de l'animateur (il sait et évite la voiture) qui crée l'avantage.

d) Elle convainc sans calcul et **contrôle** le raisonnement : si la simulation donne $\frac{1}{2}$ dans le jeu original, c'est que le programme ne modélise pas correctement la règle de l'animateur.

Réponse. Simulé 0,664 pour $\frac{2}{3}$; si l'animateur ouvre au hasard, $\frac{1}{2}$.

Correction de l'exercice 10 – Attendre PP ou PF : vérification numérique

a)

```
import random

random.seed(11)
def attente(motif):
    derniers = ""
    t = 0
    while not derniers.endswith(motif):
        derniers = derniers[-1:] + random.choice("PF")
        t = t + 1
    return t

N = 100000
print(sum(attente("PP") for k in range(N)) / N)
print(sum(attente("PF") for k in range(N)) / N)
```

b) Écarts 0,006 et 0,012.

c)

$$\sigma_{PP} = \sqrt{\frac{22}{10^5}} \approx 0,015, \quad \sigma_{PF} = \sqrt{\frac{4}{10^5}} \approx 0,0063$$

— Écarts de $0,4\sigma$ et $1,9\sigma$: compatibles avec la théorie. La différence 6 contre 4 (des dizaines d'écart types) est, elle, **massivement** confirmée.

d) Seuls les deux derniers lancers comptent pour le test : garder une chaîne de longueur 2 au plus évite de construire une chaîne de plus en plus longue (mémoire et temps constants par lancer).

Réponse. Simulé 6,006 et 3,988 pour 6 et 4 ; écarts de $0,4\sigma$ et $1,9\sigma$.

Correction de l'exercice 11 - Ruine du joueur : simulation

a)

```
import random

random.seed(12)
def partie(k, p):
    while 0 < k < 3:
        if random.random() < p:
            k = k + 1
        else:
            k = k - 1
    return k == 3

N = 100000
print(sum(partie(1, 2 / 3) for i in range(N)) / N)
```

b) $\frac{4}{7} \approx 0,57143$; $\sigma = \sqrt{\frac{0,571 \times 0,429}{10^5}} \approx 0,0016$; écart 0,0022, soit $1,4\sigma$: compatible.

c) Il n'y a pas de variant décroissant (le capital monte et descend). Mais trois gains consécutifs terminent toujours le jeu : chaque bloc de 3 parties se termine avec probabilité au moins p^3 , et

$$\mathbb{P}(\text{jeu non fini après } 3m \text{ parties}) \leq (1 - p^3)^m \xrightarrow{m \rightarrow \infty} 0$$

d) Ajouter un compteur $t = t + 1$ dans la boucle et renvoyer t ; la moyenne des durées doit approcher $\frac{15}{7} \approx 2,14$.

Réponse. Simulé 0,569 pour $\frac{4}{7} \approx 0,571$ ($1,4\sigma$) ; terminaison presque sûre.

Correction de l'exercice 12 - Fabriquer du hasard : générateurs congruentiels

a)

```
def lcg(a, c, m, x0, n):
    L = []
    x = x0
    for k in range(n):
        x = (a * x + c) % m
        L.append(x)
    return L
```

b) Les 16 valeurs de 0 à 15 apparaissent toutes avant le retour à 3 : période $16 = m$

(maximale). Chaque terme ne dépend que du précédent et ne prend que m valeurs : dès qu'une valeur revient, la suite se répète.

c) (4, 1, 16) : $x = 5$ est un point fixe ($21 \equiv 5$), période 1 — inutilisable. (5, 0, 16) : période 4, seules les valeurs 1, 5, 9, 13 apparaissent. Le théorème de Hull-Dobell donne la période maximale m lorsque

$$\text{pgcd}(c, m) = 1, \quad a - 1 \text{ divisible par tout premier divisant } m \text{ (et par 4 si } 4 \mid m)$$

— conditions remplies par (5, 3, 16), pas par les deux autres.

d) Les termes alternent impair, pair, impair... : le bit de poids faible est parfaitement **prévisible**. Période maximale ne veut pas dire bon hasard ; Python utilise le Mersenne Twister, de période $2^{19937} - 1$.

Réponse. (5, 3, 16) : période 16 ; (4, 1, 16) : période 1 ; (5, 0, 16) : période 4 ; parité alternée : mauvais hasard.

Correction de l'exercice 13 - Mélanger correctement une liste

a) Uniforme : $\frac{60\,000}{6} = 10\,000$ chacune ($\sigma \approx 91$). Fisher-Yates est dans $\pm 2\sigma$; les écarts de ± 1100 du mélange naïf sont énormes (12σ).

b) j prend 3 valeurs pour $i = 2$, puis 2 valeurs pour $i = 1$: $3 \times 2 = 6$ suites de choix équiprobables, et chacune donne une permutation **différente** (la position 2 est fixée au premier tour, puis la position 1). Bijection :

$$\mathbb{P}(\sigma) = \frac{1}{3 \times 2} = \frac{1}{3!}$$

c) Chaque permutation serait obtenue par un nombre entier k de suites, avec $\frac{k}{27} = \frac{1}{6}$, soit $k = 4,5$: impossible. Plus généralement n^n n'est pas divisible par $n!$.

d) $\frac{4}{27} \times 60\,000 \approx 8889$ et $\frac{5}{27} \times 60\,000 \approx 11\,111$: exactement les ordres de grandeur observés (8850 à 8956, 11 015 à 11 164) ✓

Réponse. Fisher-Yates uniforme ($\frac{1}{n!}$) ; mélange naïf biaisé : probabilités $\frac{4}{27}$ ou $\frac{5}{27}$.

Correction de l'exercice 14 - Estimer π : comparaison générale des méthodes

a) $\sigma \approx \frac{1,6}{\sqrt{N}}$ (plaquelette 1) : $\frac{1,6}{\sqrt{N}} \leq 10^{-6}$ donne $N \geq 2,6 \times 10^{12}$ tirages, plusieurs heures de calcul pour un résultat seulement **probablement** juste.

b) $\frac{1}{n} \leq 10^{-6}$: $n = 10^6$ termes ; avec la moyenne, $\frac{1}{2n^2} \leq 10^{-6}$: $n \approx 710$.

c) Terme de rang k : $\frac{(1/5)^{2k+1}}{2k+1}$, multiplié par 16 ; il faut

$$\frac{16}{(2k+1)5^{2k+1}} \leq 10^{-6} \Leftrightarrow k \geq 4$$

— ($k = 4$: $\frac{16}{9 \times 5^9} \approx 9 \times 10^{-7}$) : cinq termes, plus deux pour $\arctan \frac{1}{239}$.

d) Pour la même précision : 10^{12} tirages, 10^6 termes, 700 termes, ou 7 termes. La vitesse de convergence (et non la puissance de l'ordinateur) fait toute la différence ; c'est avec des formules de type Machin que l'on a calculé des milliards de décimales.

Réponse. 6 décimales : $\approx 3 \times 10^{12}$ tirages, 10^6 termes de Leibniz (700 accéléré), 7 termes avec Machin.

Correction de l'exercice 15 - Synthèse : le problème de la secrétaire simulé

a)

```
import random

random.seed(15)
def strategie(n, k):
    v = list(range(n))
    random.shuffle(v)
    seuil = max(v[:k]) if k > 0 else -1
    for x in v[k:]:
        if x > seuil:
            return x == n - 1
    return v[-1] == n - 1

N = 20000
for k in (10, 37, 60):
    print(k, sum(strategie(100, k) for i in range(N)) / N)
```

b) $k \approx \frac{n}{e} \approx 37$, succès $\approx \frac{1}{e} \approx 0,368$: la simulation donne 0,367 ✓

c) $\sum_{j=k+1}^n \frac{1}{j-1} \approx \int_k^n \frac{dt}{t} = \ln \frac{n}{k}$, d'où une probabilité $\approx x \ln \frac{1}{x} = -x \ln x$. Dérivée :
 $-\ln x - 1 = 0$ pour $x = \frac{1}{e}$, où

$$-x \ln x = \frac{1}{e} \approx 0,368$$

d) $x = 0,1 : 0,230$; $x = 0,37 : 0,368$; $x = 0,6 : 0,306$. Accord avec les simulations (0,227 ; 0,367 ; 0,309) à moins de 0,004, soit moins d'un écart type ($\sigma \approx 0,0034$).

Réponse. Optimum $k \approx 37$, succès $\approx \frac{1}{e}$; théorie $-x \ln x : 0,230, 0,368, 0,306$ confirmée par la simulation.