CPGE BCPST 2
Année Académique: 2023 - 2024

Travaux Pratiques N°1

Méthodes numériques et statistiques: Simulation de variables aléatoires

Anicet E. T. Ebou

I. Simulation d'une v.a.r. suivant une loi quelconque

I.1. Méthode d'inversion de la fonction de répartition

La méthode d'inversion de la fonction de répartition a été proposée par Von Neumann en 1947. On rappelle un résultat du cours de probabilité:

Proposition 1: *Si F est la fonction de répartition d'une v.a.r. X, c'est-à-dire $F(t) = P(X \leq t)$ pour $t \in R$, alors en définissant son inverse généralisée $F^{-1}(u) = inf\{t|F(t) \geq u\}$, si U suit la loi uniforme sur [0,1], $F^{-1}(U)$ a même loi que X.*

Si on peut calculer F^{-1}, on peut donc simuler X à partir de U.

I.2. Loi géométrique

I.2.1. Simulation à l'aide de la fonction `random`

Exercice 1: Simulation de la loi géométrique avec `random`

A partir de la simulation de la loi de Bernoulli, écire une fonction `geom(p)` qui simule une variable aléatoire $G(p)$.

Note: Si $X \hookrightarrow \mathcal{G}(p)$, on sait que X représente le rang du premier succès dans un schéma de Bernoulli, donc on peut modéliser un résultat aléatoire suivant une loi géométrique en répétant une évaluation selon une loi de Bernoulli de paramètre p jusqu'à obtenir un premier succès : c'est à dire la valeur 1 avec probabilité p.

Remarque: La fonction `np.random.geometric(p)` fournit le même résultat.

I.2.2. Simulation à partir de l'inverse de la fonction de répartition

Pour la loi géométrique, pour une v.a.r. X, $q = 1 - p$ et on a $F(x) = 1 - q^x$ pour $x > 0$, d'où $F^-1(u) = \dfrac{ln(1-u)}{ln(1-p)}$.

Exercice 2: Simulation de la loi géométrique

Ecrire une fonction `geoinv(p)` qui permet de simuler la loi géométrique.

I.2.3. Diagrammes en bâtons

Exercice 3: Diagrammes en bâtons - loi géométrique

Ecrire un programme permettant de visualiser les données obtenues des deux fonctions de simulations de la loi géométrique et comparer à chaque fois avec l'allure de la fonction de masse de la loi géométrique.

I.3 Loi de Poisson

I.3.1. Simulation à l'aide de la fonction `random`

La loi de Poisson $\mathcal{P}(\lambda)$ est plus difficile à modéliser à partir de sa situation d'application. Par contre on peut la modéliser grâce à sa fonction de répartition $F(x)$.

En effet $\forall n \in X(\Omega) = N$, on a $P(X = n) = P(X \in]n-1, n]) = P(X \leq n) - P(x \leq n-1) = F_X(n) - F_X(n-1)$, c'est donc la probabilité qu'une évaluation de la fonction 'random' tombe dans l'intervalle $[F_X(n-1), F_X(n)[\subset [0, 1[$.

Donc pour modéliser un résultat qui suit une loi de Poisson, il suffit de choisir un nombre p aléatoirement avec la fonction `random()`, puis il reste à déterminer pour quelle valeur de n, p est dans l'intervalle $[F_X(n-1), F_X(n)[$.

Pour cela, il suffit de calculer successivement pour $n = 0, n = 1, n = 2, ...F_X(n) = \sum_{k=0}^{n} P(X = k) = \sum_{k=0}^{n} e^{-\lambda} \frac{\lambda^k}{k!}$ jusqu'à ce que l'ont ait $F_X(n) \geq p$.

Le premier entier n tel que $F_X(n) \geq p$ sera bel et bien un résultat d'une évaluation suivant une loi de Poisson.

Exercice 4: Simulation de la loi de Poisson

Ecrire une fonction `poisson(mu)` qui simule une variable aléatoire de loi $\mathcal{P}(\mu)$.

Remarque: la fonction 'numpy.random.poisson' conduit au même résultat.

I.3.2. Diagrammes en bâtons

Exercice 5: Diagrammes en bâtons - loi de Poisson

Ecrire un programme permettant de visualiser les données obtenues des simulations et comparer avec avec l'allure de la fonction de masse de la loi de Poisson.

I.3.3 Approximation de la loi de Poisson

On peut aussi approximer la loi de Poisson en utilisant la loi binomiale. En effet on sait que si $X \hookrightarrow \mathcal{B}(n, \frac{n}{p})$ alors sa loi se rapproche d'une variable $Y \hookrightarrow \mathcal{P}(p)$.

Exercice 6: Approximation de la loi de Poisson

Ecrire un code Python permettant de visualiser les données obtenues des simulations et comparer avec l'allure de la densité de probabilité de la loi normale.

I.4. Loi exponentielle

I.4.1. Simulation à partir de l'inverse de la fonction de répartition

Pour la loi $Exp(\lambda)$, on a $F(x) = 1 - e^{-\lambda x}$ pour $x > 0$, d'où $F^{-1}(u) = -\frac{1}{\lambda} ln(1-u)$.

Exercice 7: Simulation de la loi de exponentielle

Ecrire une fonction `expo(lambda)` qui simule une v.a.r. suivant la loi exponentielle de paramètre `lambda`.

Remarque: la fonction **numpy.random.exponential** conduit au même résultat.

I.4.2 Histogramme et densité de probabilité

Exercice 8: Diagrammes en bâtons - loi exponentielle

Ecrire un programme permettant de visualiser les données obtenues des simulations et comparer avec l'allure de la densité de probabilité de la loi exponentielle.

I.5 Loi normale

I.5.1 Simulation à partir de la fonction random

Il est possible de générer une variable aléatoire gausienne directement à l'aide de la transformation de Box-Muller. Elle s'appuie sur le résultat suivant.

Proposition 2: *Soient X et Y deux variables aléatoires gausiennes centrées réduites indépendantes. Définissons (R, θ) les coordonnées polaires de (X, Y): $X = Rcos(\theta), Y = Rsin(\theta)$ avec $R \geq 0$ et $\theta \in [0, 2\pi]$. Alors R^2 et θ sont deux variables aléatoires indépendantes, la première est de loi $Exp(\frac{1}{2})$, la seconde de loi uniforme sur $[0, 2\pi]$.*

Pour simuler une variable X aléatoire de $\mathcal{N}(0, 1)$, il suffit donc de prendre $U, V \sim \mathcal{U}([,])$ indépendantes et poser $X = \sqrt{-2ln(U)}cos(2\pi V)$. Pour simuler une variable X de loi $\mathcal{N}(\mu, \sigma^2)$, il suffit de prendre $X = \mu + \sigma Y$ avec $Y \sim \mathcal{N}(0, 1)$.

Exercice 9: Simulation de la loi normale

Ecrire une fonction `normale(mu, sigma)` qui simule une v.a.r. suivant la loi normale de paramètre `mu` et `sigma`.

Remarque: la fonction `numpy.random.normal` conduit au même résultat.

I.5.2 Histogramme et densité de probabilité

Exercice 10: Diagrammes en bâtons - loi normale

Ecrire un code Python permettant de visualiser les données obtenues des simulations et comparer avec l'allure de la densité de probabilité de la loi normale.

II. Illustration numérique de convergence en lois

II.1. Loi faible des grands nombres

Pour illustrer la loi des grands nombres, on peut créer un très grand échantillon de N réalisation d'une variable aléatoire comme précédemment et afficher la variation de l'espérance estimée sur les k premières valeurs de l'échantillon quand k augmente.

On peut ainsi à partir des fonctions de simulation de v.a.r réalisées dans les paragraphes précédents, visualiser l'évolution de la moyenne empirique des réalisations des lois précédentes.

Exercice 11: Visualisation de loi

Complétez le code suivant en utilisant une simulation de la loi géométrique et ensuite une de la loi normale:

```
from matplotlib import pyplot as plt

def simulation_grand_nombre_Y(n, N):
    L = ...
    # Cette variable servira à calculer la somme
    # cumulée des valeurs de l'échantillon
    val = 0
    xbar = []
    for i in range(N):
        val += L[i]
        # On enregistre dans xbar la moyenne des i premieres
        # valeurs de l'echantillon
        xbar.append(val/(i + 1))
    return xbar

n = 10
N = 100000
T = [i + 1 for i in range(N)]
x = simulation_grand_nombre_Y(n, N)
plt.plot(T, X)
plt.show()
```

On peut aussi mettre sur la même figure la simulation et l'espérance de la loi.

Exercice 12: Visualisation de loi

Afin d'illustrer la Loi des Grands Nombres, visualisons la suite $S_n = X_1 + X_2 + X_3 + ... + X_n$ pour X_i une suite de variables aléatoires indépendantes de loi uniforme sur [-1, 1].

```
from matplotlib import pyplot as plt
import numpy as np

n = 1000
x = 2 * np.random.rand(n) - 1
s = ...
plt.plot(range(1, n+1), s, 'r', label='simulation')
plt.plot((1, n), (0, 0), 'b--', label='espereance')
plt.xlabel('n')
plt.legend(loc='best')
plt.title('loi␣faible␣des␣grands␣nombres')
plt.show()
```

II.2. Théorème centrale limite

Le théorème central limite établit la convergence en loi de la **somme** d'une suite de variables aléatoires (indépendantes, et identiquements distribuées) vers **la loi normale**.

Nous pouvons visualiser cela en utilisant une suite de v.a.r suivant une loi de Poisson et vérifier sa convergence en loi vers la loi normale.

Exercice 13: Visualisation de loi

Complétez le code suivant:

```
import numpy as np
import matplotlib.pyplot as plt

# Paramètres pour la distribution
poisson_lambda = 5
normal_mean = 0
normal_std = 1
num_samples = 1000

# Générer les données
poisson_data = ...
normal_data = ...

# Création des sous figures
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 5))

# Plot Poisson histogram
axes[0].hist(..., bins=20, color='blue', alpha=0.7)
axes[0].set_title('Distribution de Poisson')
axes[0].set_xlabel('Valeur')
axes[0].set_ylabel('Fréquence')

# Plot Normal histogram
axes[1].hist(..., bins=20, color='orange', alpha=0.7)
axes[1].set_title('Distribution normale')
axes[1].set_xlabel('Valeur')
axes[1].set_ylabel('Fréquence')

# Ajuster le layout
plt.tight_layout()

# Affichage de la figure
plt.show()
```