⋆ Simulations de Variables Aléatoires à Densité ⋆

Compétences attendues

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

Liste des commandes Python exigibles aux concours

- Dans la librairie `numpy.random` :
  - Lois discrètes : `binomial`, `randint`, `geometric`, `poisson` ;
  - Lois continues : `random`, `uniform`, `exponential`, `normal`, `gamma`
- Dans la librairie `matplotlib.pyplot` : `hist`, `show`

Dans toute cette feuille, on supposera l'import des bibliothèques `numpy` utilisable avec l'alias `np` et `numpy.random` avec le préfixe `rd`.

1 Simulation des lois usuelles

1.1 Fonctions Python

Python et sa bibliothèque `numpy.random` dispose de fonctions simulant des lois continues usuelles.

Définition 1.1 [Lois à densité usuelles]

- Uniforme
  `rd.uniform(low, high, n)` retourne n réalisations d'une variable aléatoire uniforme $\mathcal{U}([\texttt{low}, \texttt{high}])$.
- Exponentielle
  `rd.exponential(1/lamb, n)` génère un tableau de n tirages à partir de la même loi exponentielle $\mathcal{E}(\texttt{lamb})$.
- Normale
  `rd.normal(mu, sigma, n)` génère n variables aléatoires de loi $\mathcal{N}(\mu, \sigma^2)$.
- Gamma
  `gamma(a)` génère une loi $\Gamma(a, 1)$ où a est le paramètre de forme. `gamma(a, theta, n)` génère n variables aléatoires de loi $\Gamma(a, \theta)$ où θ est un paramètre d'échelle.
- Beta
  `rd.beta(a,b,n)` génère n réalisation d'une loi $\beta(a, b)$.
- Khi-deux
  `rd.chisquare(v,n)` génère n réalisations d'une loi du Khi-deux $\mathcal{X}^2_\nu$, où ν est le dégré de liberté.
- Fisher
  `rd.f(v1, v2,n)` génère n réalisations d'une loi de Fisher de paramètres ν_1 et ν_2.
- Laplace
  `rd.laplace(loc, scale,n)` génère n variables aléatoires de loi de Laplace (double exponentielle) de paramètre de position `loc` et d'échelle `scale`.
- Student
  `standard_t(v, n)` génère n variables de loi de Student de paramètre ν.

1.2 Loi Uniforme

Exercice 1 Soient a et b deux réels tels que $a < b$. On rappelle que si $U \hookrightarrow \mathcal{U}([0;1])$, alors $(b-a)U + a \hookrightarrow \mathcal{U}([a;b])$.
Écrire une fonction `uniforme(a,b)` simulant la loi $\mathcal{U}([a;b])$ à l'aide de la fonction `rd.random` ou `rd.uniform`.

1.3 Loi normale

Proposition 1.1

- On suppose que $U_1, \dots, U_{12}$ sont des variables aléatoires mutuellement indépendantes, suivant toutes la loi $\mathcal{U}([0;1])$. On pose $X = \sum_{i=1}^{12} U_i - 6$.
  Par le théorème de Central Limite (TCL), on peut considérer que X suit approximativement la loi normale centrée réduite.
- Soit $m \in \mathbb{R}$ et $\sigma > 0$. Si $X \hookrightarrow \mathcal{N}(0,1)$, alors $\sigma X + m \hookrightarrow \mathcal{N}(m, \sigma^2)$.

Exercice 2

1. Écrire une fonction `norm_cen_red()` simulant la loi $\mathcal{N}(0,1)$ à l'aide de la fonction `rd.random`
2. Écrire une fonction `normale(m,s)` simulant la loi $\mathcal{N}(m,s^2)$ à partir de la fonction précédente.
3. Écrire une fonction `Normale(m,s,N)` qui renvoie un vecteur Numpy contenant N réalisations de la loi $\mathcal{N}(m,s^2)$.

2 Méthode d'inversion pour les variables aléatoires à densité

2.1 Principe

Théorème 2.1

On suppose que X est une variable aléatoire à densité dont la fonction de répartition F est strictement croissante de $]a;b[(-\infty \leq a < b \leq +\infty)$ sur $]0;1[$. Alors :
- F réalise une bijection de $]a;b[$ sur $]0;1[$;
- si $U \hookrightarrow \mathcal{U}(]0;1[)$, alors $F^{-1}(U)$ suit la même loi que X.

Exercice 3

1. Rappeler l'expression de la fonction de répartition de $U \hookrightarrow \mathcal{U}([0;1])$.
2. Démontrer le théorème précédent.

Méthode 2.1 [Simulation d'une variable aléatoire à densité par la méthode d'inversion]

Soit X une variable aléatoire à densité et F sa fonction de répartition. Supposons qu'on dispose d'une expression explicite de F^{-1}. Pour simuler la variable X, on procèdera comme suit :

1. on tire $u =$ `rd.random()` aléatoirement dans $[0;1]$;
2. on retourne $F^{-1}(u)$.

2.2 Simulation de la loi exponentielle

Exercice 4

1. Rappeler l'expression de la densité et de la fonction de répartition d'une loi exponentielle.
   Montrer que cette dernière réalise une bijection de $\mathbb{R}_+^*$ sur $]0;1[$ puis déterminer sa bijection réciproque.
2. (a) Écrire une fonction `exponentielle(lamb)` simulant une loi $\mathcal{E}(\lambda)$ à partir de la fonction `rd.random`.
   (b) Écrire une fonction `Exponentielle(lamb,N)` donnant un échantillon de taille N de la loi $\mathcal{E}(\lambda)$.
3. (a) Créer une vecteur de taille 10000 contenant 10000 simulations d'une variable aléatoire suivant la loi $\mathcal{E}\left(\frac{1}{2}\right)$.
   (b) En utilisant les commandes `np.mean` et `np.std` de Numpy, vérifier que la moyenne et l'écart-type empiriques (i.e. sur ce vecteur) sont bien conformes à ce qu'on attend.
4. (a) Écrire une fonction `gamma(n)` simulant la loi $\gamma(n)$ pour tout $n \in \mathbb{N}^*$.
   (b) Écrire une fonction `Gamma(n,N)` qui renvoie un vecteur contenant N réalisations de la loi $\gamma(n)$.

2.3 Simulation de la loi de Cauchy

Exercice 5 On rappelle qu'une variable X suit une loi de Cauchy si elle admet pour fonction de répartition la fonction :

$$F : x \in \mathbb{R} \mapsto \frac{1}{\pi}\left(\arctan(x) + \frac{\pi}{2}\right).$$

1. Donner la densité de X.
2. Montrer que F réalise une bijection de $\mathbb{R}$ sur $]0;1[$, et déterminer F^{-1}.
3. (a) Écrire une fonction `cauchy()` simulant une loi de Cauchy à partir de la fonction `rd.random`.
   (b) Écrire une fonction `Cauchy(N)` donnant un échantillon de taille N de la loi de Cauchy.
4. (a) Créer une vecteur de taille 10000 contenant 10000 simulation d'une variable aléatoire suivant la loi de Cauchy.
   (b) utiliser la commande `np.mean` avec ce vecteur pour évaluer l'espérance d'une loi de Cauchy. Recommencer avec plusieurs échantillons. Que constatez-vous ? Que conclure ?

3 Représentations Graphiques

Soit X une variable aléatoire à densité. Supposons que l'on dispose d'une fonction `Loi` permettant de simuler la loi de X. Pour juger de la qualité de cette simulation, on va utiliser des représentations graphiques, comme dans le cas des variables discrètes (cf TP7). Pour cela, on procède donc 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` ;
2. on compare graphiquement les fréquences empiriques obtenues grâce à l'échantillon avec les probabilités théoriques pour vérifier la pertinence de la simulation.

Remarque 3.1. Pour une variable discrète, afin de comparer nos résultats à la théorie, nous avions regroupé les simulations par modalités (ou classes) et tracé le diagramme en bâtons des fréquences. Nous n'allons pas pouvoir procéder de cette manière dans le cas d'une variable à densité X. En effet, on a pour tout $x \in \mathbb{R}\mathbb{P}(X = x) = 0$, et chaque modalité de notre échantillon risque d'avoir un effectif égal à 1 (pas très parlant...). Le tri par modalités n'est donc pas adapté ici, et la représentation à l'aide d'un diagramme en bâtons non plus.

On propose ici deux façons d'évaluer la qualité de nos simulations dans le cas de variables aléatoires à densité :
- en comparant l'*histogramme des fréquences d'un échantillon* à la densité théorique ;
- en comparant la *fonction de répartition empirique d'un échantillon* à la fonction de répartition théorique de la loi.

3.1 Comparaison histogramme des fréquences / densité

Le nombre de modalités de notre échantillon `x` étant a priori très grand, chacune avec un effectif de 1, nous allons les regrouper par classe. Afin de définir ces classes, on fixe une suite de réels strictement croissante :

$$c = (c_0 < c_1 < \cdots < c_p).$$

On va alors comparer :
- l'*histogramme des fréquences de l'échantillon*, défini par les rectangles ayant pour base les classes $[c_i; c_{i+1}]$ et d'aires les fréquences d'appartenances à ces classes pour notre échantillon `x` ;
- la courbe représentative d'une densité f de la loi de X.

Si notre simulation suit bien la loi attendue, on doit constater que :

Théorème 3.1 [Théorème d'or de Bernoulli]

Pour N "suffisamment grand", la fréquence observée pour la classe $[c_i; c_{i+1}]$ est proche de la probabilité théorique $\mathbb{P}(c_i \leq X \leq c_{i+1}) = \int_{c_i}^{c_{i+1}} f(t)\,\mathrm{d}t$, où f est la densité de notre variable aléatoire X.

Graphiquement, on devrait donc observer que l'aire du rectangle de base $[c_i; c_{i+1}]$, qui est égale à la fréquence d'appartenance à cette classe, est proche de l'aire sous la courbe représentative de f entre c_i et c_{i+1}.

On va pour cela avoir besoin des commandes suivantes.

Définition 3.1 [Commandes Graphiques]

- Si `x` et `c` sont des vecteurs, `c` à composantes strictement croissantes définissant les classes, alors `plt.hist(x,c, density='True')` trace l'histogramme des fréquences de `x` pour les classes définies par `c`.
- Si `x` est un vecteur et `n` un entier naturel non nul, alors `plt.hist(x,bins=n,density='True')` trace l'histogramme des fréquences de `x` à `n` classes (qui sont alors équi-réparties entre la plus petite et la plus grande valeur observée dans `x`).

Exercice 6

1. Simuler avec la fonction `Exponentielle` $N = 10000$ valeurs de la loi $\mathcal{E}\left(\frac{1}{2}\right)$.
2. Tracer la courbe représentative de la densité f de la loi $\mathcal{E}\left(\frac{1}{2}\right)$.
3. Tracer l'histogramme des fréquences de l'échantillon obtenu (on prendra pour cela une subdivision `c` de l'intervalle $[0; 10]$ en $p = 100$ intervalles de même longueur).
   Comparer l'histogramme des fréquences de l'échantillon à la courbe représentative de f. Qu'en pensez-vous ?
4. Procéder de même pour la loi $\gamma(3)$.

3.2 Comparaison de la fonction de répartition empirique / théorique

On propose dans cette section une deuxième méthode pour juger de la qualité de la simulation d'une loi de probabilité à densité. On va comparer :

- la *fonction de répartition empirique* : il s'agit de la fonction en escaliers qui à un réel x associe la fréquence d'apparition des nombres inférieurs ou égaux à ce x dans l'échantillon `x` ;
- la *fonction de répartition théorique.*

On doit alors observer que :

Théorème 3.2 [Théorème d'or de Bernoulli]

Pour N "suffisamment grand", la fréquence observée des modalités plus petites que x est proche de la probabilité $\mathbb{P}(X \leq x)$.

Graphiquement, la courbe de la fonction de répartition empirique doit donc être proche de la courbe de la fonction de répartition théorique si notre simulation correspond à la loi attendue.

Commandes utiles

Pour tracer la fonction de répartition empirique, nous aurons besoin des commandes suivantes :

Définition 3.2 [Commandes Tests]

- Si `u` et `v` sont deux vecteurs de même taille, l'instruction `u==v` renvoie un vecteur de même format que `u` et `v` dont les éléments sont `True` ou `False` selon que les coefficients correspondants de `u` et `v` à cette même place sont égaux ou non.
- Si tous les éléments de `v` sont égaux à un même réel `x`, on peut écrire simplement `u==x` et on teste alors si toutes les coordonnées de `u` sont égales à `x`.
- On définit de même les vecteurs booléens `u>v`, `u>=v`, `u<v`, `u<=v` et `u!=v`.

Définition 3.3 [Commande Moyenne]

Si `u` est uyn vecteur dont les composantes sont des booléens (`True` ou `False`), alors la commande `np.mean(u)` renvoie la proportion de booléens qui ont pris la valeur `True`.

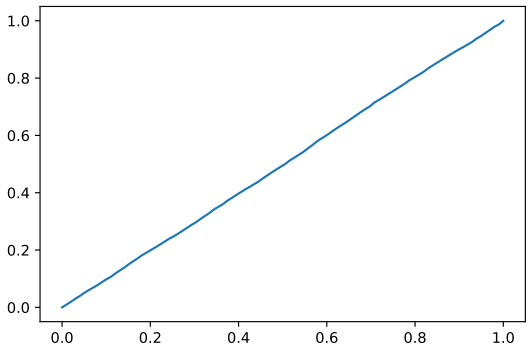
Méthode 3.1 [Tracé de la fonction de répartition empirique]

Pour tracer la fonction de répartition empirique d'un échantillon `x`, on procède comme suit :

```
n = 100
u = np.linspace(np.min(x), np.max(x),n)
v = np.zeros(n)
for k in range(n) :
    v[k] = np.mean(x <= u[k])
plt.plot(u,v)
plt.show()
```

Exemple 3.1. Prenons le cas où $X \hookrightarrow \mathcal{U}([0;1])$, avec un échantillon `x` de taille $N = 10000$ de la loi de X obtenu à l'aide de la fonction `rd.random` et traçons la fonction de répartition empirique associée.

```
N = 10000
n = 100
x = rd.random(N)
u = np.linspace(np.min(x), np.max(x),n)
v = np.zeros(n)
for k in range(n) :
    v[k] = np.mean(x <= u[k])
plt.plot(u,v)
plt.show()
```



Cette courbe étant proche de celle de la fonction de répartition théorique de la loi $\mathcal{U}([0;1])$, on peut ainsi en conclure que les simulations obtenues à l'aide de la fonction `rd.random` suivent bien la loi $\mathcal{U}([0;1])$.

Exercice 7

1. Expliquer les lignes de commandes proposées pour tracer la fonction de répartition empirique.
2. Simuler avec la fonction `Exponentielle`, $N = 10000$ réalisations de la loi $\mathcal{E}(1)$.
3. Tracer la fonction de répartition empirique de l'échantillon obtenu et la fonction de répartition théorique de cette loi. Comparer.

On va maintenant tester notre simulation de la loi $\mathcal{N}(m, \sigma^2)$. Il nous faut pour cela tracer la fonction de répartition théorique de cette loi. Cela nécessite d'importer la librairie `scipy.special` à l'aide de la commande suivante

```
import scipy.special as sp
```

On dispose ensuite de la commande suivante :

Définition 3.4 [Commande Fonction de Répartition Normale]

- Pour tout réel x, la commande `sp.ndtr(x)` renvoie la valeur de $\Phi(x)$, où Φ est la fonction de répartition de la loi $\mathcal{N}(0,1)$.
- Pour tout réel x, la commande `sp.ndtr((x-m)/s)` renvoie la valeur de $F(x) = \Phi\left(\frac{x-m}{s}\right)$, où F est la fonction de répartition de la loi $\mathcal{N}(m, s^2)$.

Exercice 8

1. Calculer $\Phi(0), \Phi(1), \Phi(1,96)$.
2. Soit X une variable aléatoire suivant la loi $\mathcal{N}(3,4)$. Calculer $\mathbb{P}(X > 10), \mathbb{P}(0 \leq X < 3)$.
3. Tester les simulations obtenues des lois $\mathcal{N}(0,1)$ et $\mathcal{N}(2,9)$ en traçant les fonctions de répartition théoriques et empiriques.
4. Comparer ces résultats avec ceux obtenus par la fonction `rd.normal`.

4 Exercices

Exercice 9 Soit $a \in \mathbb{R}_+^*$ et $(X_1, \dots, X_n)$ une famille de variables aléatoires indépendantes et identiquement distribuées suivant une loi uniforme sur $[0; a]$. On pose :

$$U = \min(X_1, \dots, X_n) \quad \text{et} \quad V = \max(X_1, \dots, X_n).$$

1. Déterminer les lois de U et de V.
2. On considère la fonction suivante :

```
def simulation(a,n):
    nb_sim = 10000
    R = rd.uniform(0,a,(nb_sim,n))
    return np.min(R, axis=1),np.max(R,axis=1)
```

   Expliquer la fonction `simulation` (on pourra consulter l'aides des fonctions `min` et `max` pour cela).
3. Prenons $n = 10$ et $a = 1$. Comparer alors graphiquement la qualité de cette simulation.

Exercice 10 [Simulation de la loi normale à l'aide du TCL] Soit $(X_k)_{k\geq 1}$ une suite de variables aléatoires mutuellement indépendantes et toutes de même loi. Notons $\bar{X}_n = \frac{1}{n}\sum_{k=1}^n X_k$. D'après le *Théorème Central Limite* (TCL), on a la variable centrée réduite

$$\bar{X}_n^* = \frac{\bar{X}_n - \mathbb{E}\left[\bar{X}_n\right]}{\sqrt{\mathrm{Var}\left(\bar{X}_n\right)}} \xrightarrow[n\to+\infty]{\mathcal{L}} \mathcal{N}(0,1).$$

1. On suppose dans cette question que les variables X_k suivent toutes la loi $\mathcal{U}(\llbracket 1; 6 \rrbracket)$.
   (a) Écrire une fonction `norm_cen_red2(n,N)` donnant N réalisations de la variable $\bar{X}_n^*$.
   (b) Évaluer la qualité de cette simulation de la loi $\mathcal{N}(0,1)$ selon les valeurs de n. Pour quelles valeurs de n cette simulation semble pertinente ?
2. Mêmes questions en supposant que les variables X_k suivent toutes la loi $\mathcal{E}(1)$.
3. Comparer les différentes méthodes de simulation de la loi $\mathcal{N}(0,1)$ ainsi obtenues.

Exercice 11 [Différentes simulations de la loi de Pareto] Soit $k \in \mathbb{N}^*$ et $\lambda > 0$.

1. Déterminer la valeur de r pour laquelle la fonction suivante est une densité de probabilité :

$$f_\lambda : x \mapsto \begin{cases} 0 & \text{si } x \leq \lambda, \\ \frac{r}{x^{k+1}} & \text{sinon.} \end{cases}$$

2. Déterminer la fonction de répartition d'une variable suivant la loi de Pareto de paramètres λ et k.
3. En utilisant la méthode d'inversion, simuler une variable aléatoire suivant une loi de Pareto de paramètres λ et k.
4. Soient $X_1, \dots, X_k$ k variables aléatoires indépendantes suivant toutes la loi uniforme sur $[0; 1]$. On pose alors

$$Y = \frac{\lambda}{\max(X_1, \dots, X_k)}.$$

   (a) Montrer que Y suit une loi de Pareto de paramètres λ et k.
   (b) En déduire une autre méthode pour simuler la loi de Pareto.
5. Comparons à présent les deux méthodes obtenues de simulation d'une loi de Pareto.
   (a) En simulant 100000 réalisations d'une loi de Pareto à l'aide de chacune de ces méthodes, déterminer laquelle est la plus rapide.
   On pourra à cet effet utiliser la fonction `time` *du module* `time`. *Pour cela on exécute la commande* `t1 = time.time()` *juste avant le début de la simulation, puis* `t2 = time.time()`. *La soustraction des deux temps permet alors de déduire le temps qui a été nécessaire à l'exécution de la portion de code encadrée par les chronomètres.*
   (b) Prenons $\lambda = 1$. Comparer graphiquement la qualité de chacune des deux simulations d'une loi de Pareto.