

Simulation de variables aléatoires à densité

1 Simulation des lois usuelles	2
1.1 Fonctions Python	2
1.2 Loi normale	2
2 Méthode d'inversion	3
2.1 Principe	3
2.2 Exercices	3
3 Représentations graphiques	4
3.1 Comparaison histogramme des fréquences / densité	4
3.2 Comparaison fonction de répartition empirique / théorique	5
4 Exercices	7

Compétences attendues.

- ✓ Savoir simuler une loi continue usuelle à l'aide des fonctions de la librairie `numpy.random`, ou uniquement à partir de la fonction `rd.random()`.
- ✓ Vérifier graphiquement la pertinence d'une simulation d'une loi.

Liste des commandes Python exigibles aux concours.

- Dans la librairie `numpy.random` : `rd.random`, `rd.exponential`, `rd.normal`.
- Dans la librairie `matplotlib.pyplot` : `plt.hist`, `plt.show`.

Anthony Mansuy

Professeur de Mathématiques en deuxième année de CPGE filière ECG au Lycée Clemenceau (Reims)

Page personnelle : <http://anthony-mansuy.fr>

E-mail : mansuy.anthony@hotmail.fr

1 Simulation des lois usuelles

1.1 Fonctions Python

Rappelons que le sous-module `numpy.random` est dédié aux simulations de variables aléatoires.

Définition.

On importe la bibliothèque `numpy.random` en écrivant l'une ou l'autre des instructions suivantes (on privilégiera la deuxième) :

```
from numpy.random import * ou import numpy.random as rd
```

Définition.

- `rd.random()` simule une réalisation de la loi uniforme sur $[0, 1]$.
- `rd.exponential(1/a)` simule une réalisation de la exponentielle $\mathcal{E}(a)$ de paramètre $a > 0$.
- `rd.normal(m, sigma)` simule une réalisation de la loi normale $\mathcal{N}(m, \sigma^2)$ de paramètres $m \in \mathbb{R}$ et $\sigma > 0$.

Remarques.

- On peut obtenir r simulations d'une loi usuelle sous la forme d'un vecteur de taille r , ou $r \times s$ simulations sous la forme d'une matrice de $\mathcal{M}_{r,s}(\mathbb{R})$. Par exemple :
 - `rd.exponential(1/a, r)` renvoie un vecteur contenant r simulations de la loi exponentielle $\mathcal{E}(a)$;
 - `rd.normal(m, sigma, [r,s])` renvoie $r \times s$ simulations de la loi normale de paramètres m et σ^2 .
- Attention à certains paramètres :
 - le paramètre de la loi exponentielle choisi par Python est l'inverse de celui du cours.
 - le second paramètre de la loi normale est σ et non σ^2 .
 - par défaut, `rd.normal()` simule la loi normale centrée réduite.
- La simulation d'une variable de loi uniforme sur $[a, b]$, où $a < b$, s'obtient à l'aide de la commande hors programme `rd.uniform(a, b)`, ou bien, en restant dans le cadre du programme, par la commande `(b-a)*rd.random()+a`.

Dans la suite de ce TP, on explique comment simuler les lois usuelles en utilisant uniquement la fonction `rd.random`.

1.2 Loi normale

Propriété 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 $X = \sum_{i=1}^{12} U_i - 6$.
D'après le *Théorème de la limite centrée*, on peut considérer que X suit approximativement la loi normale centrée réduite.
- Soit $m \in \mathbb{R}$ et $\sigma \in]0, +\infty[$. Si $X \hookrightarrow \mathcal{N}(0, 1)$, alors $\sigma X + m \hookrightarrow \mathcal{N}(m, \sigma^2)$.

Exercice 1 (★ - Simulation de la loi normale)

1. Écrire une fonction `norcentreereduite()` 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 `norcentreereduite()`.
3. Écrire une fonction `Normale(m,s,N)` donnant un vecteur contenant N réalisations de la loi $\mathcal{N}(m, s^2)$.

2 Méthode d'inversion

2.1 Principe

Théorème 2

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 .

Méthode.

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

- (i) on choisit un paramètre t de manière aléatoire dans $]0, 1[$ à l'aide de la fonction `rd.random()` ;
- (ii) on retourne $F^{-1}(t)$.

2.2 Exercices

Exercice 2 (★★ - Preuve de la méthode d'inversion)

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

Exercice 3 (★★ - Simulation de la loi exponentielle)

1. (a) Rappeler l'expression de la fonction de répartition d'une loi exponentielle, et montrer qu'elle réalise une bijection de \mathbb{R}_+^* sur $]0, 1[$.
(b) Déterminer sa bijection réciproque.
2. (a) Écrire une fonction `exponentielle(lambda)` simulant une loi $\mathcal{E}(\lambda)$ à partir de la fonction `rd.random()`.
(b) Écrire une fonction `Exponentielle(lambda,N)` donnant un échantillon de taille N de la loi $\mathcal{E}(\lambda)$.
3. (a) Créer un vecteur de taille 10000 contenant 10000 simulations d'une variable aléatoire suivant la loi $\mathcal{E}(1/2)$.
(b) En utilisant les commandes `np.mean` et `np.std` de la librairie `numpy`, vérifier que la moyenne et l'écart-type empiriques (c'est-à-dire de ce vecteur) sont bien conformes à ce qu'on attend.

3 Représentations graphiques

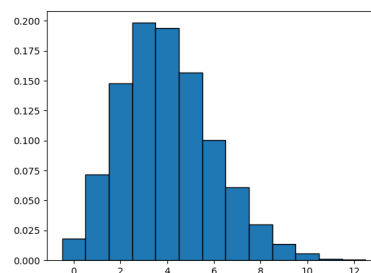
Soit X une variable aléatoire à densité. Supposons qu'on dispose d'une fonction `Loi` permettant de simuler la loi de X . Pour juger de la qualité de cette simulation, on va utiliser des représentations graphiques. Pour cela, on procédera comme suit :

- on crée un échantillon de taille N , c'est-à-dire un vecteur ligne \mathbf{x} contenant N réalisations de la fonction `Loi` ;
- on compare graphiquement les fréquences empiriques obtenues grâce à l'échantillon avec les probabilités théoriques pour vérifier la pertinence de la simulation.

Remarque. Pour une variable discrète, afin de comparer nos résultats à la théorie, nous avons regroupé les simulations par modalités, et tracé le diagramme en bâtons des fréquences. Par exemple, si \mathbf{x} est un vecteur contenant 10000 simulations d'une loi de Poisson de paramètre 4, on peut procéder ainsi :

```

1 | x = rd.poisson(4, 10000)
2 | c = np.arange(-0.5, 13)
3 | plt.hist(x, c, density = 'True', edgecolor = 'k')
4 | plt.show()
    
```



Nous n'allons pas pouvoir procéder de cette manière dans le cas d'une variable à densité X . En effet, on a $P(X = x) = 0$ pour tout $x \in \mathbb{R}$, et chaque modalité de notre échantillon risque donc d'avoir un effectif égal à 1. Le tri par modalités n'est donc pas adapté ici, et la représentation à l'aide d'un diagramme en bâtons non plus.

Nous proposons ici deux façons d'évaluer la qualité de nos simulations dans le cas de variables aléatoires continues :

- 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

Principe

Le nombre de modalités de notre échantillon \mathbf{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 :

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

On va alors comparer :

- l'*histogramme des fréquences de l'échantillon*, défini par les rectangles de base les classes $[c_i, c_{i+1}]$ et d'aires les fréquences d'appartenance à ces classes pour notre échantillon \mathbf{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 (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

$$P(c_i \leq X \leq c_{i+1}) = \int_{c_i}^{c_{i+1}} f(t) dt.$$

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} .

Commandes utiles

Nous allons avoir besoin des commandes suivantes :

Définition.

Soit x un vecteur.

- L'instruction `plt.hist(x, n)` trace l'histogramme associé à la série x en n classes équiréparties entre la plus petite valeur de x et la plus grande (par défaut, n vaut 10).
- L'instruction `plt.hist(x, c)` trace l'histogramme associé à la série x dont les classes sont définies par le vecteur aux composantes strictement croissantes c .
- On dispose pour la commande `plt.hist` des options de tracé suivantes (non exigibles) :
 - normalisation des rectangles (la surface totale vaut 1) : `density = 'True'`
Cette option permet d'obtenir l'histogramme des fréquences de l'échantillon.
 - contours des rectangles en noir : `edgecolor = 'k'`

Exercice 4 (★★)

1. Simuler avec la fonction `Exponentielle` $N = 10000$ valeurs de la loi $\mathcal{E}(0.5)$.
2. Tracer la courbe représentative de la densité f de la loi $\mathcal{E}(0.5)$.
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 ?

3.2 Comparaison fonction de répartition empirique / théorique

Principe

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 qui à un réel x associe la fréquence d'apparition des nombres inférieurs ou égaux à x dans l'échantillon x ;
- la fonction de répartition théorique.

On doit observer que :

Théorème 4 (Théorème d'or de Bernoulli)

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

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.

- Si u et v sont deux vecteurs de même format, l'instruction $u==v$ renvoie un vecteur de même format que u 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 $u==x$.
- On définit de même les vecteurs booléens $u>v$, $u>=v$, $u<v$, $u<=v$ et $u!=v$.

Définition.

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

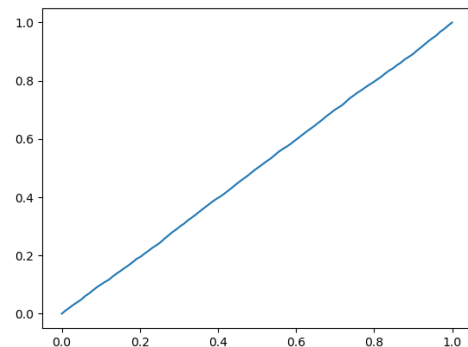
Méthode.

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

```
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. Prenons le cas où $X \leftrightarrow \mathcal{U}([0, 1])$, 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.

```
1 N = 10000 ; n = 100
2 x = rd.random(N)
3 u = np.linspace(np.min(x), np.max(x), n)
4 v = np.zeros(n)
5 for k in range(n) :
6     v[k] = np.mean(x <= u[k])
7 plt.plot(u, v)
8 plt.show()
```



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

Exercice 5 (★★)

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 alors de la commande (hors programme) suivante :

Définition.

- Pour tout réel x , la commande `sp.ndtr(x)` renvoie la valeur de $\Phi(x)$.
- 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 6 (★★)

1. Calculer $\Phi(0), \Phi(1), \Phi(1.96)$.
2. Soit X une variable aléatoire suivant la loi $\mathcal{N}(3, 2^2)$. Calculer $P(X > 10), P(0 \leq X < 3)$.
3. Tester les simulations obtenues des lois $\mathcal{N}(0, 1)$ et $\mathcal{N}(2, 3^2)$ en traçant les fonctions de répartition théoriques et empiriques.
4. Comparer ces résultats avec ceux obtenus par la fonction `rd.normal`.

4 Exercices

Exercice 7 (★★)

Soit X une variable aléatoire de fonction de répartition définie par :

$$F(x) = \begin{cases} 0 & \text{si } x \leq 0, \\ 2(1 - e^{-x}) & \text{si } 0 < x < \ln(2), \\ 1 & \text{si } x \geq \ln(2). \end{cases}$$

1. (a) Déterminer la réciproque de la restriction de F à $]0, \ln(2)[$.
 (b) A l'aide de la méthode d'inversion, écrire une fonction `simulX()` simulant la variable aléatoire X .
 (c) Écrire une fonction `SimulX(N)` donnant un échantillon de taille N de la loi de X .
2. (a) Tracer l'histogramme des fréquences d'un échantillon de taille $N = 10000$ de la loi de X (on prendra pour cela une subdivision c de l'intervalle $[-1, \ln(2) + 1]$ en 100 intervalles de même longueur).
 (b) Tracer sur le même graphique la densité (théorique) de X sur le même graphique. Comparer.
3. (a) Tracer la fonction empirique d'un échantillon de taille $N = 10000$ de la loi de X (on prendra toujours une subdivision c de l'intervalle $[-1, \ln(2) + 1]$ en 100 intervalles de même longueur).
 (b) Tracer sur le même graphique la fonction de répartition théorique de la loi de X . Comparer.

Exercice 8 (★★★★ - Différentes simulations de la loi de Pareto)

Soit $k \in \mathbb{N}^*$ et $\lambda > 0$.

1. Déterminer la valeur de r pour laquelle $f_\lambda : x \rightarrow \begin{cases} 0 & \text{si } x \leq \lambda \\ \frac{r}{x^{k+1}} & \text{sinon} \end{cases}$ est une densité de probabilité.

Si X admet pour densité f_λ , on dit que X suit la loi de Pareto de paramètre λ et k .

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 les deux méthodes obtenues de simulation d'une loi de Pareto. Prenons $\lambda = 1$ et $k = 2$.
- (a) En simulant $N = 100000$ réalisations d'une loi de Pareto à l'aide de chacune de ces méthodes, déterminer laquelle des deux est la plus rapide.
On pourra à cet effet utiliser la commande `time.time()` de la librairie `time`. Pour cela, on exécute la commande `tps1 = time.time()` juste avant le début de la simulation, puis la commande `tps2 = time.time()` juste après. La différence entre ces différents temps donnera le temps d'exécution de la portion de code encadrée (en secondes).
 - (b) Comparer graphiquement la qualité de chacune des deux simulations d'une loi de Pareto (histogramme des fréquences/densité théorique).
-