TP6

Simulation de variables aléatoires à densité

Compétences attendues

- Savoir simuler une loi usuelle à l'aide de fonctions de la bibliothèque `numpy.random`.
- Vérifier graphiquement la pertinence d'une simulation d'une loi.

6.1 Simulation des lois usuelles

Il est nécessaire d'importer le module `random` et le sous-module `numpy.random` pour réaliser des simulations de variables aléatoires (discrètes ou continues).

```
import random as rd      import numpy.random as nr
```

Python Loi uniforme $\mathscr{U}[0,1]$

- `rd.random()` simule une réalisation de la loi $\mathscr{U}[0,1]$.
- `nr.random(N)` simule un vecteur ligne contenant N réalisations indépendantes de la loi $\mathscr{U}[0,1]$
- `nr.random([r,s])` simule une matrice $r \times s$ dont les coefficients sont des réalisations indépendantes de la loi $\mathscr{U}[0,1]$.

Exercice 6.1 (Lois $\mathscr{U}[a,b]$ par transfert affine)**.**

1. Montrer que $X \hookrightarrow \mathscr{U}[0,1] \iff (a-b) * X + a \hookrightarrow \mathscr{U}[a,b]$
2. En déduire une fonction Python qui simule une réalisation de la loi $\mathscr{U}[a,b]$.

Lois exponentielles $\mathscr{E}(a)$ en Python

- `nr.exponential(1/a)` simule une réalisation de la loi exponentielle de paramètre a.
- `nr.exponential(1,a,N)` simule N réalisations de la exponentielle de paramètre a.
- `nr.exponential(1/a,[r,s])` simule une matrice $r \times s$ de réalisations de la loi $\mathscr{E}(a)$.

Exercice 6.2 (Lois exponentielles par inversion)**.**

1. Montrer que si $X \hookrightarrow \mathscr{U}[0,1]$ alors $-\frac{1}{\lambda}\ln(1-X) \hookrightarrow \mathscr{E}(\lambda)$
2. (a) Déduire du résultat précédent une fonction Python qui simule une réalisation de la loi $\mathscr{E}(\lambda)$ à partir de la fonction `rd.random()`.
   (b) Ecrire une fonction `exponentielle(lambda,N)` qui simule un échantillon de N réalisations d'une la loi $\mathscr{E}(\lambda)$.
3. (a) Créer un vecteur de taille 1000 contenant 1000 réalisations de la loi $\mathscr{E}(0.5)$.
   (b) Vérifier que la moyenne et l'écart-type empiriques sont conformes à ce que l'on attend.

De manière très générale le résultat suivant permet de simuler une variable à partir de la fonction `rd.random()`.

THÉORÈME 6.1 (Théorème d'inversion - Hors Programme)

Soit X est une variable aléatoire dont la fonction de répartition F est strictement croissante sur $]a,b[$.
Alors :
• F réalise une bijection de $]a,b[$ sur $]0,1[$,
• si $U \hookrightarrow \mathscr{U}[0,1]$, alors $F^{-1}(U)$ suit la même loi que X.

EXERCICE 6.3.

1. Montrer que la fonction de répartition de la loi exponentielle vérifie les hypothèses du théorème.
2. Retrouver directement que $-\frac{1}{\lambda}\ln(1-U) \hookrightarrow \mathscr{E}(\lambda)$ si $U \hookrightarrow \mathscr{U}[0,1]$.

EXERCICE 6.4. Démontrer le théorème d'inversion .

6.2 Représentations graphiques

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

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

REMARQUE **6.1.**
Pour une variable discrète, afin de comparer nos résultats à la théorie, nous avions regroupé les simulations par modalités, et tracé l'histogramme des fréquences. Par exemple, si `x` est un vecteur contenant 10000 simulations d'une loi de Poisson de paramètre 4, on peut procéder ainsi :

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

Nous n'allons pas pouvoir procéder de cette manière dans le cas d'une variable à densité X.
En effet, lorsque X est à densité 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.

6.2.1 Comparaison histogramme des fréquences / fonction de densité

PRINCIPE : 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 de base les classes $[c_i, c_{i+1}]$ et d'aires les fréquences d'appartenance 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 :

PROPRIÉTÉ 6.2 (Bernoulli)

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

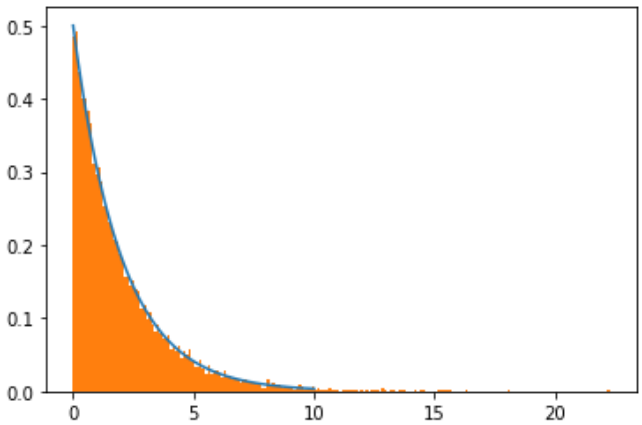
Graphiquement, on devrait donc observer que l'aire du rectangle de base $[c_i, c_{i+1}]$, qui est égale à la fréquence d'appartenance à cette classe, est proche de l'aire sous la courbe représentative de f entre c_i et c_{i+1}.

Commandes utiles

- L'instruction `plt.hist(x,n)` trace l'histogramme associé à la série `x` en `n` classes équi-réparties 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) :
  - `density='True'` : normalisation des rectangles (la surface totale vaut 1), cette option permet d'obtenir l'histogramme des fréquences de l'échantillon.
  - `edgecolor='k'` : contours des rectangles en noir :

EXERCICE 6.5 (Comparaison graphique de la loi $\mathscr{E}(0.5)$)**.**

1. Simuler avec la fonction `nr.exponential` un échantillon de taille $N = 10000$ de valeurs de la loi $\mathscr{E}(0.5)$.
2. Tracer la courbe représentative de la densité f de la loi $\mathscr{E}(0.5)$.
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).
4. Vous devriez obtenir le graphique suivant. Qu'en pensez vous?



6.2.2 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 :

PROPRIÉTÉ 6.3 (Bernoulli)

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

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.

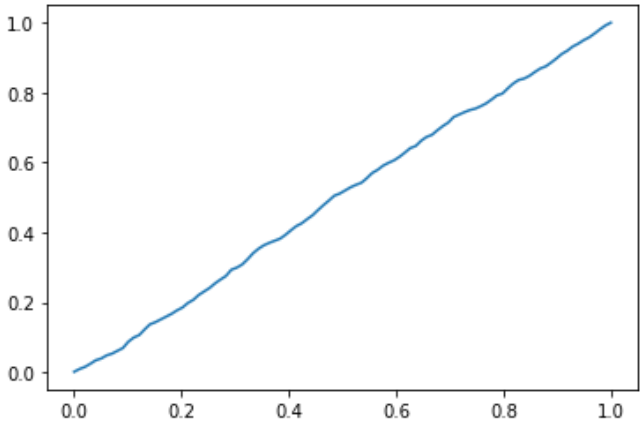
Représenter graphiquement la fonction de répartition empirique

Rappelons que la commande `np.mean(x<= a)` renvoie la proportion de valeurs de x inférieures ou égales à un réel a.
Voici comment tracer la fonction de répartition empirique :

```
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()
```

EXERCICE 6.6 (Fonction de répartition empirique de la loi $\mathscr{U}[0,1]$).

1. Rappeler l'expression de la fonction de répartition de la loi $\mathscr{U}[0,1]$.
2. Représenter graphiquement la fonction de répartition empirique associée à une échantillon de N réalisations de la loi $\mathscr{U}[0,1]$.
3. On obtient le graphique suivant :



Le résultat et-il conforme à vos attentes ?

EXERCICE 6.7 (Fonction de répartition empirique de la loi $\mathscr{E}(1)$).

Reprendre les mêmes questions pour la loi $\mathscr{E}(1)$

6.3 Exercices plus difficiles

EXERCICE 6.8 (Lois du min et du max)**.**

Soit $a \in \mathbb{R}_+^*$ et $(X_1, \dots, X_n)$ une famille de variables aléatoires indépendantes, 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 (mathématiquement) les lois de U et V.
2. On considère la fonction suivante :

```
def simule(a,n):
  nb_sim=10000
  R=a*nr.random([nb_sim,n])
  couple=np.array([np.min(R,0),np.max(R,0)])
  return couple
```

   Expliquer la fonction simulation.
3. Prenons $n = 2$ et $a = 1$. Comparer graphiquement la qualité de cette simulation en faisant apparaître dans un même graphique les fonctions de répartition empirique et théorique.

EXERCICE 6.9 (Différentes simulations de la loi de Pareto)**.**

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

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

   On dit que X suit la loi de Pareto de paramètre λ et k si X admet pour densité f_λ.
2. Déterminer la fonction de répartition (théorique) d'une variable suivant la loi de Pareto de paramètres λ et k.
3. En utilisant le théorème d'inversion, simuler une variable aléatoire suivant une loi de Pareto de paramètres λ et k.
4. Soient $X_1, \cdots, X_k$ des variables aléatoires indépendantes suivant toutes la loi uniforme sur $[0, 1]$.
   On pose alors $Y = \frac{\lambda}{\max(X_1, \cdots, X_k)}$.
   (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 $N = 100000$ 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.clock()` de la librairie `time`. Pour cela, on exécute la commande `tps1 = time.clock()` juste avant le début de la simulation, puis la commande `tps 2 time.clock ()` juste après. La différence entre ces différents temps donnera le temps d'exécution de la portion de code encadrée (en secondes).
   (b) Prenons $\lambda = 1$. Comparer graphiquement la qualité de chacune des deux simulations d'une loi de Pareto (histogramme des fréquences/densité théorique).