Télétrafic (TTR)

Laboratoire 1

Simulation de variables aléatoires

Octobre 2019

Stephan ROBERT - stephan.robert@heig-vd.ch

1 Une première fonction avec Python

Cette première partie vous permet de vous familiariser un petit peu avec Python (ensuite lisez le document disponible sur la page web du cours). On vous demande de dessiner une exponentielle mais au préalable il faut importer les librairies suivantes :

```
import numpy as np
import matplotlib as mpl
import pandas
import matplotlib.pyplot as plt
from matplotlib import rc
```

Il y a d'autres librairies que vous devrez importer par la suite . Le code est assez trivial pour dessiner notre exponentielle (avec Spyder) :

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import numpy as np
import matplotlib as mpl
import pandas
import matplotlib.pyplot as plt
from matplotlib import rc

X = np.linspace(-2, 2, 20)
Y = np.exp(X)
plt.figure()
plt.plot(X, Y)
plt.title('Exponential function')
```

```
plt.xlabel('x')
b plt.ylabel(r'$f(x) = e^x$')
```

et voici le résultat :

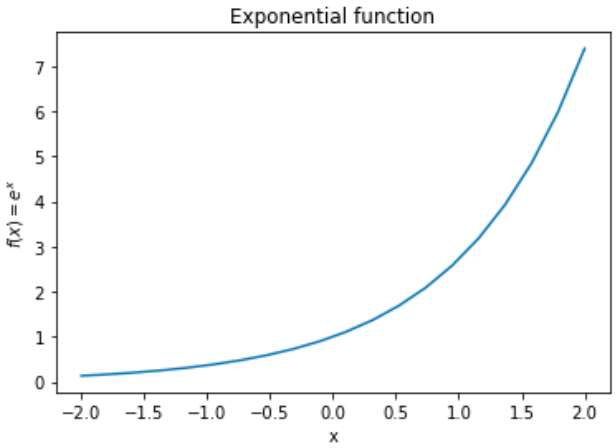


FIGURE 1 – Fonction exponentielle

2 Simulation de variables aléatoires

2.1 Loi uniforme

Comment Python simule-t-il une suite de variables aléatoires indépendantes et uniformes sur $[0,1]$? Cela revient a générer des entiers aléatoires indépendants uniformes dans un ensemble $E = \{1, 2, ..., M\}$, puis à les diviser par M. Pour cela, Python génère une suite de nombres pseudo-aleatoires en partant dun état $x_0 \in E$ (la graine, ou seed en anglais) et en appliquant successivement une meme fonction $f : E \to E$ (par exemple $f(x) = ax+c$ modulo M avec a et c bien choisis). Cette transformation doit bien sûr verifier plusieurs propriétés : on veut que la suite générée “ressemble” à une suite de variables aléatoires uniformes et indépendantes sur E. Que veut dire “ressemble”? La suite doit réussir à passer toute une batterie de tests statistiques d’indépendance et d’adéquation de loi.

Essayez ce code :

```
import numpy.random as npr
print npr.rand()
npr.seed(seed=1)
print npr.rand()
```

```
print npr.rand()
npr.seed(seed=1)
print npr.rand()
print npr.rand()
```

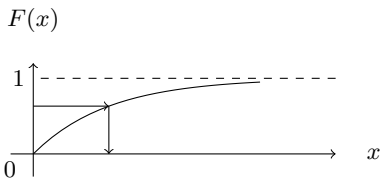
Si $a < b$ et U est une variable aleatoire uniforme sur $[0, 1]$, on peut demontrer que $a + (b - a)U$ suit une loi uniforme sur $[a, b]$. Ainsi, pour simuler une variable aleatoire uniforme sur $[a, b]$, on simule une variable aleatoire U uniforme sur $[0, 1]$ et on renvoie $a + (b - a)U$.

Exercice : simulez une variable uniforme sur un segment quelconque. Programmez ceci en Python

3 Simulation par inversion de la fonction de repartition

Si U est uniforme sur $[0, 1]$ et $\lambda > 0$, alors $V = -\frac{1}{\lambda}\ln(U)$ est une loi λ exponentielle de paramètre λ. Plus généralement, on a le résultat suivant : Si X une variable aléatoire reelle. On suppose que sa fonction de répartition F est strictement croissante (F est donc bijective de R sur $]0, 1[$ et on peut noter F^{-1} son inverse). Soit U une variable aléatoire uniforme sur $[0, 1]$. Alors $F^{-1}(U)$ a la même loi que X.

Génération d'un nombre pseudo-aléatoire dans $[0, 1]$



Exercices

- En utilisant la méthode de la fonction de répartition inverse, écrire une fonction en Python renvoyant une simulation d'une variable aléatoire suivant la loi en question (exponentielle, cauchy, pareto)
- Simuler un échantillon de taille 1000 de la variable aléatoire
- En tracer un histogramme que lon comparera avec la densité de la loi

3.1 Loi exponentielle

La loi exponentielle de paramètre $\lambda > 0$ a pour densité $\mathbf{1}_{x>0}\lambda e^{-\lambda x}$, et pour fonction de répartition $\mathbf{1}_{x>0}(1 - e^{-\lambda x})$, où $\lambda > 0$ et où $\mathbf{1}$ est la fonction indicatrice. Les simulations pourront être faites avec la valeur $\lambda = 1/2$.

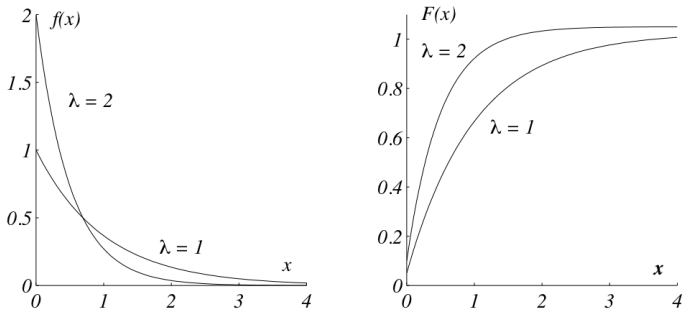


FIGURE 2 – Loi exponentielle

3.2 Loi de Cauchy

La loi de Cauchy a pour densité

$$\frac{1}{\pi(1+x^2)}$$

et pour fonction de répartition

$$\frac{\arctan(x) + \pi/2}{\pi}$$

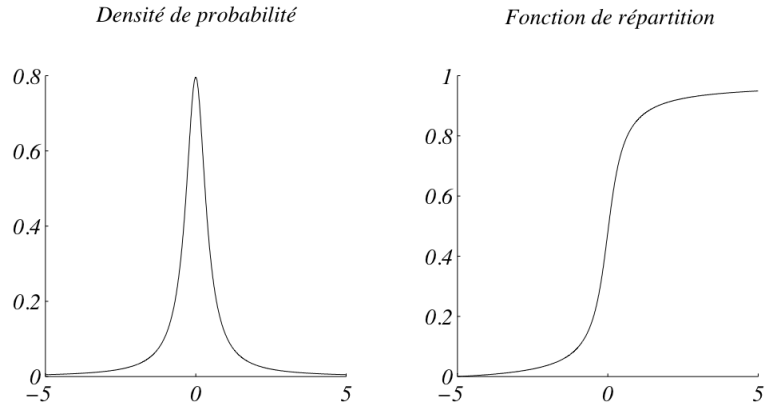


FIGURE 3 – Loi de Cauchy

3.3 Loi de Pareto

La loi de Pareto a pour densité $\mathbf{1}_{x>0} k x^{-1-k}$ où $k > 0$ et pour fonction de répartition $\mathbf{1}_{x>1}(1 - x^{-k})$. Les simulations pourront être faites avec $k = 1$ dans un premier temps mais essayez d'autres valeurs et tirez vos conclusions !

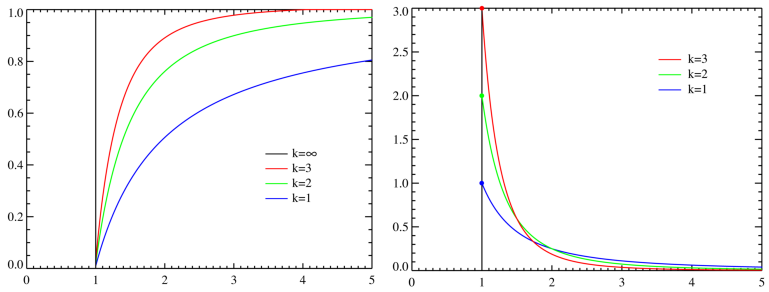


FIGURE 4 – Loi de Pareto

3.4 Code à compléter

Voici à quoi devraient ressembler vos codes

```
from math import *
import numpy.random as npr

def Cauchy():

YOUR CODE HERE
        U=... # Tirer un nombre uniforme sur [0,1]
        return... # Evaluer la fonction inverse de U

print Cauchy():
```

4 Simulation de la Loi géométrique par plusieurs méthodes

On va comparer plusieurs manières de simuler une variable géometrique de paramètre $p \in (0,1)$ a partir dune variable aléatoire uniforme sur $[0,1]$ en se fondant sur les résultats théoriques suivants :

- De maniere générale, pour simuler une variable aléatoire X a valeurs entières telle que $P(X = i) = \pi$ pour tout $i \geq 0$, on tire une variable uniforme U sur $[0,1]$ et on renvoie lentier k tel que $p_0 + + p_{k1} < U < p_0 + \ldots + p_k$.

- On tire des variables aleatoires de Bernoulli de parametre p (on renvoie 1 si une variable aleatoire uniforme U sur $[0,1]$ est plus petite que p, 0 sinon) et on sarrete a la première fois quon tombe sur 1.
- On pose $\lambda = -\frac{1}{\ln(1-p)}$ et on renvoie $\lceil \lambda X \rceil$($\lceil x \rceil$ est le plus petit entier k tel que $k \geq x$, `ceil` en Python) où X est une variable aléatoire exponentielle de paramètre 1.
- Faire appel a une fonction intégrée de Python.

Exercice : Comparez l'efficacité de ces trois méthodes pour simuler N variables aléatoires géométrique de paramètre p en complétant le code ci-dessous. Commentez l'influence des paramètres.

4.1 Code à compléter

```
from _future_import division
from math import *
import numpy as np
import numpy.random as npr
from time import time

p=0.1 #paramètre de la VA géometrique

N=10000 #nombre de fois quon simule la VA

def methode1():
    k=1
    tmp=p
    U=npr.rand()
    while U>tmp:
          tmp=tmp+YOUR CODE HERE
          k=k+1
    return YOUR CODE HERE


def methode2():
    tmp=0
    k=0
    while tmp==0:
          k=k+1
          if YOUR CODE HERE
:
```

```
            tmp=1
    return k

def methode3():
    X=-log(npr.rand())
    return int(ceil(YOUR CODE HERE))

def methode4():
    return npr.YOUR CODE HERE(p)

t1 = time()
[methode1() for i in range(N)]
t2 = time()
temps1 = t2 -t1
print "La methode 1 a pris ", temps1, " secondes"

t1 = time()
[methode2() for i in range(N)]
t2 = time()
temps1 = t2 -t1
print "La methode 2 a pris ", temps1, " secondes"

t1 = time()
[methode3() for i in range(N)]
t2 = time()
temps1 = t2 -t1
print "La methode 3 a pris ", temps1, " secondes"

t1 = time()
[methode4() for i in range(N)]
t2 = time()
temps1=t2-t1
print "La methode 4 a pris ", temps1, " secondes"
```

5 La loi Gaussienne par la méthode Box-Miller

La méthode de la fonction de répartition inverse ne s'applique pas facilement à la loi normale, car sa fonction de répartion et son inverse n'ont pas d'expression analytique simple et efficacement calculable. On emploie donc une autre méthode pour simuler des variables de loi normale. On peut montrer que les deux variables aléatoires ci-dessous sont iid et normalement distribuées :

$$X_1 = \sqrt{-2\ln(U_1)}\sin(2\pi U_2) \sim \mathcal{N}(0,1)$$
$$X_2 = \sqrt{-2\ln(U_1)}\cos(2\pi U_2) \sim \mathcal{N}(0,1)$$

avec U_1, U_2 générés dans l'intervalle $[0,1]$. Rappelons que la densité d'une loi normale réduite est $\frac{1}{\sqrt{2\pi}}e^{-x^2/2}$. Pour obtenir une distribution de $Y \sim \mathcal{N}(\mu, \sigma^2)$ il suffit de voir que $Y = \mu + \sigma X$ avec $X \sim N(0,1)$.

Exercice

- Ecrire une fonction qui renvoie une simulation du couple (X_1, X_2).
- Afficher 5000 réalisations du couple (X_1, X_2) sous forme de points dans le plan.
- Afficher un histogramme de 5000 réalisations de X, et le comparer avec la densité Gaussienne.

6 Convergence de variables aléatoires

6.1 Approximation de la loi de Poisson par une loi binômiale

On peut montrer que si $\lambda > 0$ et X_n est une variable aléatoire binomiale de paramètre $(n, \lambda/n)$, alors X_n converge en loi vers une loi de Poisson de paramètre λ lorsque $n \to \infty$ (c'est-à-dire que pour tout $k \geq 0$, $P(X_n = k) \to e^{-\lambda}\lambda^k/k!$ lorsque $n \to \infty$).

Exercice

Montrez ce que nous avons affirmé ci-dessus

6.2 Code à compléter

```
from _future_import division
import numpy as np
import numpy.random as npr
import scipy.stats as sps
import matplotlib.pyplot as plt

param=3
```

```
n=100
N=5000 nombre de fois quon simule la VA
YOUR CODE HERE
```

6.3 Approximation de la loi exponentielle par une loi géométrique

Soit $\lambda > 0$. Si X_n est une variable aléatoire de loi géométrique de paramètre λ/n, alors X_n converge en loi vers une variable exponentielle de paramètre λ lorsque $n \to \infty$ (c'est-à-dire que pour tout $x \geq 0$ on a $\mathbb{P}(X_n \leq x) \to \mathbb{P}(Z \leq x)$ où Z est une variable aléatoire exponentielle de paramètre λ).

Exercice

Montrez ce que nous avons affirmé ci-dessus

7 Méthode de Monte Carlo. Exemple

Considérons un couple (X, Y) de variables aléatoires indépendantes telles que chacune soit uniforme sur $[0, 1]$. Soit $((X_i, Y_i))_{i\geq 1}$ une suite de vecteurs aléatoires indépendants et de même loi que (X, Y). On admet que $\mathbb{P}(X2 + Y2 \leq 1) = \pi$. Ceci est intuitif au moins : (X, Y) représente un point "uniforme" dans le carré $[0, 1]^2$, et le probabilité qu'il appartienne à un quart de disque centré en 0 est laire de ce quart de disque (cest-à-dire $\pi/4$) divisé par l'aire totale du carré, qui vaut 1. Posons $Z_i = 1$ si $X_i^2 + Y_i^2 \leq 1$ et $Z_i = 0$ sinon. Ainsi $(Z_i)_{i\geq 1}$ sont des variables aléatoires indépendantes de Bernoulli de paramètre $\pi/4$. En posant

$$S_n = \frac{1}{4}.\sum_{i=1}^{n} Z_i$$

on en déduit que S_n converge presque sûrement vers π. On peut maintenant obtenir un intervalle de confiance sur la valeur de π. On remarque que

$$\mathbb{E} = \pi$$

$$\text{Var}(S_n) = \frac{12}{n^2}\sum_{i=1}^{n}\text{Var}(X_i) = \frac{16}{n}.\frac{\pi}{4}.\left(1 - \frac{\pi}{4}\right) \leq \frac{4}{n}$$

car $x(1 - x) \leq 1/4$ sur $[0, 1]$.

D'après linégalité de Bienaymé-Tchebychev, on a donc

$$\mathbb{P}(|S_n - \pi| \geq \alpha) \leq \frac{4}{\alpha^2 n}$$

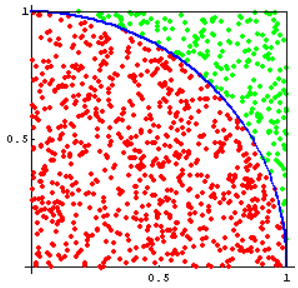


FIGURE 5 – Approximation de π avec la méthode de Monte Carlo

Exercice : Simuler un intervalle de confiance de π damplitude au plus 10^2 à 99% en complétant le code ci-dessous

7.1 Code à compléter

```
from _future_import division
import numpy as np
import numpy.random as npr
import scipy.stats as sps
import matplotlib.pyplot as plt

n= YOUR CODE HERE
X=npr.rand(n)
Y=npr.rand(n)
Sn=4/n*np.sum(YOUR CODE HERE)#sum compte le nombre d'élément non nuls dans u

YOUR CODE HERE
```

8 Référence

Ce laboratoire est tiré/adapté d'un "TP Python" de Igor Kortchemski, Ecole Polytechnique, Probabilités Approfondies, CPES2