TD : Simulation d'expériences aléatoires

1-2 heures

Rédigé par Pascal Delahaye

12 janvier 2018

Quelques fonctions utiles

1. Pour construire des listes :
   - `range(n+1)` : donne la liste des entiers de 0 à n
   - `range(p,q+1)` : donne la liste des entiers de p à q
   - `[f(i) for i in range(n)]` : donne la liste des valeurs f(i) pour i allant de 0 à (n-1)
   - `[i*j for i in L1 for j in L2]` : donne la liste des valeurs i*j pour i dans L1 et j dans L2
   - `L1 = L.copy()` : recopie le contenu de la liste L dans la liste L1

2. Fonctions de tirage aléatoire : dans la bibliothèque `random`, vous trouverez :
   - `choice(L)` : qui permet de choisir aléatoirement un élément de la liste `L` de façon équiproprable
   - `shuffle(L)` : qui effectue "en place" une permutation des éléments de la liste `L`
   - `sample(L,p)` : qui effectue un tirage successif et sans remise de p éléments de `L`
   - `random()` : qui tire de façon équiprobable un nombre flottant entre 0 et 1
   - `randint(a,b)` : qui tire de façon équiprobable un nombre entier entre a et b inclus
   - `uniform(a,b)` : qui tire de façon équiprobable un flottant entre a et b inclus
   - `e=L.pop(i)` : récupère la valeur de L[i] et l'élimine de la liste L

Exercice 1 : Deux petites expériences aléatoires

`Exemple 1.` **Somme des faces obtenues lors du lancer de 2 dés**

1. Faire un programme `S()` de simulation d'un lancer de deux dés à 4 faces et renvoyant la somme des faces obtenues. On appelera S cette variable aléatoire.

   On pourra utiliser la fonction `randint(a,b)`.

2. Fréquences et Espérances...

   (a) Estimation expérimentale de la loi de probabilité de X

   i. Faire un programme python `freqS(n)` qui permet d'observer sur un graphe les fréquences d'apparition des différentes valeurs possibles de S après avoir réalisé n expériences.
   ii. Appliquer votre programme à des valeurs n de plus en plus grandes.
   iii. L'observation est-elle conforme à la théorie ?

   (b) Estimation expérimentale de l'espérance de S

   i. Faire un programme python `esperanceS(N)` permettant d'observer sur un graphe l'évolution des moyennes des valeurs de S obtenues en fonction du nombre d'expériences n réalisées. N représente le nombre maximum d'expériences.
   ii. Appliquer votre programme à des valeurs de N de plus en plus grandes.
   iii. L'observation est-elle conforme à la théorie ?

`Exemple 2.` **Problème des rencontres.**

Dans une urne contenant 10 boules numérotées de 1 à 10, on effectue 10 tirages successifs sans remise.
On dit qu'il y a "rencontre au moment $i \in [\![1, N]\!]$ lorsque on tire la boule numérotée 'i' au ième tirage.
On appelle X la variable aléatoire représentant le nombre de rencontres obtenu lors de l'expérience.

1. Faire un programme `X()` de simulation de cette expérience et renvoyant donc la somme des faces obtenues.
   On pourra pour cela, modéliser l'urne à l'aide d'une liste L et utiliser la fonction `shuffle(L)` *pour le tirage.*

2. Fréquences et Espérances...

   (a) Estimation expérimentale de la loi de probabilité de X

   i. Faire un programme python `freqX(n)` qui permet d'observer sur un graphe les fréquences d'apparition des valeurs "0", "1", "2" ... "10" de X en fonction de n après avoir réalisé n expériences.
   ii. Appliquer votre programme à des valeurs n de plus en plus grandes.
   iii. L'observation est-elle conforme à la théorie ?

   (b) Estimation expérimentale de l'espérance de X

   i. Faire un programme python `esperanceX(N)` permettant d'observer sur un graphe l'évolution des moyennes des valeurs de X obtenues en fonction du nombre d'expériences n réalisées. N représente le nombre maximum d'expériences.
   ii. Appliquer votre programme à des valeurs de N de plus en plus grandes.
   iii. L'observation est-elle conforme à la théorie ?

Exercice 2 : Comment maximiser ses chances de gagner ?

A choisit secrètement deux entiers distincts. Il choisit alors un de ces nombres (de façon équiprobable) et le donne à B. B doit alors deviner s'il s'agit du nombre le plus grand ou le plus petit que A a choisi.

On fixe $N \in \mathbb{N}^*$. On supposera que A choisit ses deux entiers dans $[\![-N, N]\!]$ de manière équiprobable.
Nous allons comparer les 3 stratégies suivantes :

Stratégie 1 :

La première stratégie pour B est de toujours répondre qu'il s'agit du minimum quel que soit le nombre que lui annonce A.

Stratégie 2 : B décide de suivre la stratégie suivante :

- Si le nombre donné par A est strictement positif, il répond qu'il s'agit du maximum
- Si le nombre donné par A est strictement négatif, il répond qu'il s'agit du minimum
- Si le nombre donné par A est 0, il répond maximum ou minimum avec la probabilité $\frac{1}{2}$.

Stratégie 3 : B décide maintenant de suivre la stratégie suivante :

- Il choisit X dans $\{-N+\frac{1}{2},\ -N+\frac{3}{2},\ \ldots,\ -\frac{1}{2},\ \frac{1}{2},\ \ldots,\ N-\frac{3}{2},\ N-\frac{1}{2}\}$ selon une loi uniforme
- Si le nombre donné par A est strictement plus grand que X, il répond qu'il s'agit d'un maximum
- Si le nombre donné par A est strictement plus petit que X, il répond qu'il s'agit d'un minimum

Concevoir une simulation permettant de comparer les probabilités de gagner de B avec chacune des 3 stratégies précédentes.

1. Quelle stratégie semble la plus efficace ?
2. Conjecturer les probabilités pour B de gagner dans chacun des 3 cas.

On suppose maintenant que A choisit deux entiers distincts dans $[\![-N, N]\!]$ selon une loi non uniforme.
La meilleure stratégie est-elle toujours la même ?

Exercice 3 : Simulation d'une loi binômiale

Voici quelques commandes qui vous seront utiles dans le travail demandé :

Python
```
from matplotlib.pylab import plot, bar, show
from scipy.special import binom
from random import random

bar([1,2,3,4],[2,5,3,4])
plot([1,2,3,4],[3,4,2,5])
show()
```

1. Programmer à l'aide de la fonction `random()` une variable de bernoulli `X` de paramètre $p \in]0,\ 1[$.
2. En déduire une fonction `binomial` d'arguments n et p donnant la valeur de $X \hookrightarrow B(n, p)$.
3. En déduire une procédure d'arguments n, p et N qui représente sur un même graphe d'abscisse $k \in [\![0, n]\!]$:
   - la loi de probabilité de X sous forme de lignes brisées
   - les fréquences de chaque valeur de X obtenues en ré-itérant N fois la fonction `binomial` sous la forme d'un diagramme en bâtons.

   Comparer les deux graphes obtenus.