1 Estimation empirique d'une probabilité

Exercice 1. Durant une loterie, 100 billets sont mis en jeu, dont 10 billets gagnants. On achète 5 billets, et on souhaite estimer la probabilité qu'au moins un des billets soit gagnant.

1. Ecrire une fonction `recherche(L,e)` prenant en argument une liste `L` et une donnée `e` et qui renvoie `True` ou `False` selon si `e` apparaît comme élément dans `L` ou pas.
2. Simulation de l'expérience aléatoire.
   (a) Déclarer une liste `L` contenant 90 fois 0 et 10 fois 1 ; ils représentent les 100 billets et un 0 (respectivement) 1 représente un billet perdant (respectivement gagnant).
   (b) Simuler l'expérience aléatoire, en commençant par construire une liste `B` de 5 éléments obtenus aléatoirement par un tirage sans remise dans la liste `L`.
   La fonction `randint(0,n)` du module `random` permet d'obtenir un entier aléatoire situé entre 0 et n inclus.
   (c) Achever la simulation de l'expérience aléatoire en vérifiant l'obtention ou pas d'un billet gagnant à l'aide de la fonction `recherche()` écrite à la question 1.
3. Pour estimer empiriquement la probabilité de gain, on répète l'expérience aléatoire un grand nombre $N = 1000$ de fois, on compte le nombre de succès, que l'on divise par N.
   (a) Après avoir déclaré une variable globale `N=1000`, estimer empiriquement la probabilité de gain.
   (b) Augmenter la valeur de `N` pour préciser l'estimation.

2 Tracé d'un histogramme

Pour tracer un histogramme, on peut utiliser la fonction `bar()` du module `matplotlib.pyplot` :

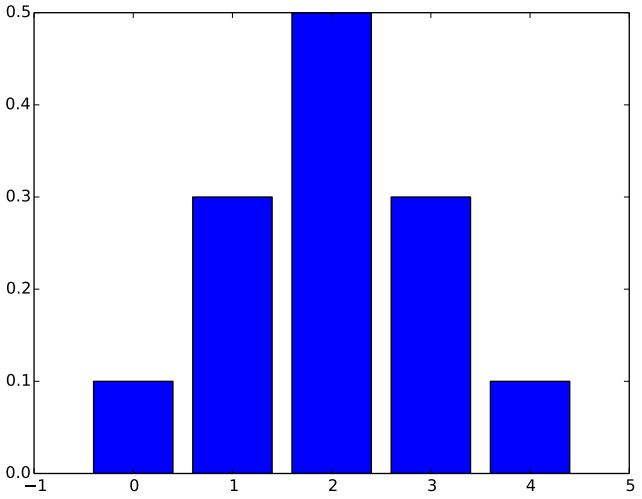
Soient `X` liste/tableau des abscisses et `Y` liste tableau des ordonnées.

• `bar(X,Y)` trace l'histogramme au dessus des abcisses dans `X` des valeurs dans `Y`.
Exemple.

```
from matplotlib.pyplot import bar, figure, show

X = [0,1,2,3,4]
Y = [0.1,0.3,0.5,0.3,0.1]
figure(1)
bar(X,Y)
show()
```

On obtient le graphique suivant :



3 Simulation d'une loi de Bernoulli

Exercice 1. *Simulation d'une loi de Bernoulli*

Soit X une v.a.r. de Bernoulli de paramètre p ($0 < p < 1$). On rappelle que son univers image est $X(\Omega) = \{0, 1\}$ et sa loi est $\mathbb{P}(X = 1) = p$, $\mathbb{P}(X = 0) = 1 - p$.

1. Ecrire une fonction `simulBernoulli(p)` prenant en paramètre un nombre $p \in]0, 1[$ et qui simule le résultat d'une expérience aléatoire suivant une loi de Bernoulli de paramètre p. Elle retournera 0 ou 1 et utilisera la fonction `random` du module `random`.
2. Ecrire une fonction `loiBernoulli(p,k)` prenant en paramètres p et un entier k et qui retourne une estimation de $\mathbb{P}(X = k)$. Pour celà elle appellera 1000 fois la fonction `simulBernoulli(p)` et retournera la proportion des résultats k obtenues.
3. Effectuer le tracé de l'histogramme de la loi de Bernoulli à l'aide de cette simulation.

4 Simulation d'une loi binomiale

On peut simuler la loi binomiale $\mathscr{B}(n,p)$ en faisant la somme de n variables aléatoires indépendantes suivant chacune la même loi de Bernoulli $\mathscr{B}(p)$.

C'est la première approche que nous utilisons dans l'exercice 2.

Exercice 2. *Simulation d'une loi binomiale.*

Soit X une V.A.R. qui suit la loi binomiale $\mathscr{B}(n,p)$.

1. Ecrire une fonction `simulBinomiale(n,p)` prenant en paramètre un nombre $0 < p < 1$ et un entier positif n, qui appelle n fois la fonction `simulBernoulli(p)` (écrite à l'exercice 1) et qui retourne la somme des résultats obtenus. Cette fonction simule la loi binomiale $\mathscr{B}(n,p)$.
2. Ecrire une fonction `loiBinomiale(n,p,k,N)` qui appelle N fois la fonction `simulBinomiale(n,p)` et qui retourne une estimation de la probabilité de l'évènement $(X = k)$.
3. Ecrire une fonction `freqBinomiale(n,p,N)` qui appelle N fois la fonction `simulBinomiale(n,p)` qui calcule la fréquence d'apparition des différents résultats obtenus que l'on stockera dans un tableau que la fonction retournera.
4. Tracer l'histogramme de la loi de X. On pourra prendre $n = 30$, $p = 1/2$ et $N = 10000$.
5. Tracer l'histogramme de la fonction de répartition de Y.

La simulation de la loi binomiale $\mathscr{B}(n,p)$ s'obtient aussi en simulant un tir avec remise de n boules dans une urne contenant une proportion p de boules blanches.

C'est l'approche effectuée dans l'exercice 3.

Exercice 3. *Simulation d'un tir dans une urne avec remise.*

Une urne contient 100 boules noires et blanches, dont 45 blanches.

1. Créer l'urne à l'aide du code :
```
L = [1] * 45 + [0] * 55 #Déclaration de L = [1, ..., 1, 0, ..., 0]
from random import shuffle
shuffle(L) #Mélange la liste
```
2. Ecrire une fonction `tir()` qui simule un tir aléatoire avec remise de 10 boules dans l'urne. On pourra utiliser la fonction `randint(0,99)` du module `random` (après l'avoir importé) qui retourne un entier aléatoire entre 0 et 99 (inclus).
3. Ecrire une fonction `frequence(m)` qui appelle m fois la fonction `tir()` et qui retourne la liste des 11 fréquences d'obtention de 0, 1, . . ., 10 boules blanches dans le tir.
4. Tracer sur une nouvelle figure l'histogramme des fréquences obtenues pour $m = 100000$.

5 Simulation d'une loi hypergéométrique

Rappel : Une urne contient un nombre N de boules blanches et noires, dont une proportion p de boules blanches (soit Np boules blanches et $Nq = N - Np$ boules noires).
On effectue un tir sans remise de n boules dans l'urne. Soit X la v.a.r. égale au nombre de boules blanches obtenues. La v.a.r. X suit **la loi hypergéométrique** $\mathscr{H}(N,n,p)$.

Son univers image est $X(\Omega) \subset [[0,n]]$,

et pour tout $k \in X(\Omega)$:

$$\mathbb{P}(X = k) = \frac{\binom{Np}{k}\binom{Nq}{n-k}}{\binom{N}{n}}$$

Exercice 4. *Simulation d'un tir dans une urne sans remise.*

Pour simuler la loi hypergéométrique $\mathscr{H}(N,n,p)$ on va simuler l'urne par une liste contenant Np boules blanches, représentées par des 1, et Nq boules noires, représentées par des 0 :

Créer l'urne puis la mélanger à l'aide de la fonction `shuffle()` :

```
Np = int(N*p)
Nq = N - Np
Urne = [1] * Np + [0] * Nq #Urne = [1, ..., 1, 0, ..., 0]
from random import shuffle
shuffle(Urne) #mélange la liste aleatoirement
```

1. Ecrire une fonction `hypergeometrique(N,n,p)` qui simule un tir aléatoire sans remise de n boules dans une urne contenant N boules dont une proportion p de blanches, et qui retourne le nombre de boules blanches obtenues.
   On pourra utiliser la fonction `randint(0,m)` du module `random` (après l'avoir importé) qui retourne un entier aléatoire entre 0 et m (inclus) et la méthode `pop()` des listes.
2. Ecrire une fonction `frequence(N,n,p,f=10000)` qui appelle `f` fois la fonction `hypergeometrique(N,n,p)` et qui retourne la liste des fréquences d'obtention de 0, 1, . . ., n boules blanches durant le tir.
3. Tracer sur une figure l'histogramme des fréquences obtenues. On prendra `N=100`, `n=30`, `p=0.45` et pour `f=10000`.
4. Tracer sur une nouvelle figure l'histogramme de la fonction de répartition.