TP # 24

Diagrammes en bâton

Bibliothèques `matplotlib.pyplot` et `numpy.random`

```
import matplotlib.pyplot as plt ; import numpy.random as rd
```

Lois usuelles : Binomiale, Uniforme, Géométrique, Poisson

- `rd.binomial(n,p)` génère un entier aléatoire selon la loi $\mathcal{B}(n,p)$.

On peut rajouter un paramètre à cette instruction pour générer plusieurs valeurs :

- `rd.binomial(n,p,m)` génère un vecteur ligne de taille `m` contenant des réalisations indépendantes.
- `rd.binomial(n,p,[m1,m2])` génère un tableau à `m1` lignes et `m2` colonnes contenant des réalisations indépendantes.

- `rd.randint(a,b+1)` génère un entier aléatoire selon la loi $\mathcal{U}(\llbracket a,b \rrbracket)$.

Même principe pour `rd.randint(a,b+1,m)` et `rd.randint(a,b+1,[m1,m2])`.

- `rd.geometric(p)` génère un entier aléatoire selon la loi $\mathcal{G}(p)$.

Même principe pour `rd.geometric(p,m)` et `rd.geometric(p,[m1,m2])`.

- `rd.poisson(lam)` génère un entier aléatoire selon la loi $\mathcal{P}(\lambda)$. (avec $\lambda =$`lam`)

Même principe pour `rd.poisson(lam,m)` et `rd.poisson(lam,[m1,m2])`.

Diagrammes en bâton avec `plt.bar`

Lorsque `X = [` $x_0, x_1, \dots, x_n$ `]` est une liste (ou un vecteur)
Lorsque `Y = [` $h_0, h_1, \dots, h_n$ `]` est une liste (ou un vecteur), les instructions :

```
plt.bar(X,Y); plt.show()
```

affichent un **diagramme en bâton** contenant, pour tout $i \in \llbracket 0,n \rrbracket$, une barre verticale à l'abscisse x_i, de hauteur h_i.

Les diagrammes en bâton seront souvent utilisés pour représenter graphiquement des lois de variables aléatoires discrètes.

Exercice 1

1. Tester le programme suivant :

```
X = [1,3,4] ; Y = [0.5, 0.1, 0.4]
plt.bar(X,Y) ; plt.show()
```

Reproduire la figure affichée :

2. Afficher un diagramme en bâton pour représenter la loi de probabilité suivante :

$$P(X=0) = \frac{1}{3}, \quad P(X=1) = \frac{1}{2}, \quad P(X=2) = \frac{1}{6}.$$

```
X = ......................... ; Y = ........................

plt.bar(X,Y) ; plt.show()
```

3. Afficher un diagramme en bâton pour représenter la loi de probabilité suivante : (somme de deux dés équilibrés)

$$\forall k \in \llbracket 2,7 \rrbracket, \quad P(X=k) = \frac{k-1}{36} \quad \text{et} \quad \forall k \in \llbracket 8,12 \rrbracket, \quad P(X=k) = \frac{13-k}{36}$$

```
X = ............................

Y = np.zeros( ..... )

for k in range(2,13) :
    if ............. :

        Y[ ..... ] = (k-1) / 36

    else :

        Y[ ..... ] = (13-k)/36

plt.bar(X,Y) ; plt.show()
```

✎ Exercice 2
Représentation de la loi de Poisson

1. Compléter le script suivant pour que la fonction `poissonbaton` prenne en entrée un réel `lam`$= \lambda > 0$ et renvoie la diagramme en bâton représentant la loi de Poisson :

$$P(X=k) = e^{-\lambda}\frac{\lambda^k}{k!}$$

On se limitera à afficher ces valeur pour $k \in [\![0, N]\!]$, avec $N = 2\lfloor \lambda \rfloor$.

```
def poisson_baton(lam) :

    N = 2 * int(lam)

    X = ............................

    Y = np.zeros( ..... ) ; Y[0] = .................

    for k in range(.............) :

      Y[k] = Y[k-1] * .......................

    plt.bar(X,Y)
```

Tester avec différentes valeurs ($\lambda = 5$, $\lambda = 10$, $\lambda = 20$, etc.)
On n'a volontairement pas inclus "`plt.show()`" dans la fonction afin de pouvoir afficher différents diagrammes sur une même figure :

```
poisson_baton(5) ; poisson_baton(10) ; poissonbaton(20) ; plt.show()
```

2. On souhaite retrouver, de manière "empirique", un diagramme représentant $\mathcal{P}(\lambda)$.

(a) On propose la fonction suivante (à rédiger à la suite de la fonction précédente) :

```
def poisson_baton_empirique(lam) :
    N = 2 * int(lam) ; X = np.arange(0,N+1) ; Z = np.zeros(N+1)

    M = 500 # nombre de simulations
    for k in range(M) :
        j = rd.poisson(lam)
        if j <= N :
            Z[j] = Z[j]+1

    Z = Z/np.sum(Z)

    plt.bar(X,Z)
```

A quoi correspondent les valeurs `Z[0]`, `Z[1]`, `Z[2]`, etc... ?

(b) Afficher, sur la même figure :

- Le diagramme en bâton produit par `poisson_baton(5)`
- Le diagramme en bâton produit par `poisson_baton_empirique(5)`

Comparer ces deux diagrammes : ils sont "très proches" !

(c) Remplacer `M =500` par `M = 50000`
et comparer de nouveau le diagramme empirique et le diagramme réel.

✎ Exercice 3
Représentation de la loi Binomiale

S'il vous reste du temps, reprendre l'exercice précédent avec la loi binomiale :

$$\forall k \in [\![0, n]\!],\ P(X=k) = \binom{n}{k} p^k (1-p)^{n-k}$$

```
def binomiale_baton(n,p) :

    X = ............................

    Y = np.zeros( ..... ) ; Y[0] = .................

    for k in range(.............) :

      Y[k] = Y[k-1] * .......................

    plt.bar(X,Y)
```

```
def binomiale_baton_empirique(n,p) :

   X = ........................ ; Z = ........................

    M = 500 # nombre de simulations
    for k in range(M) :






    Z = Z / np.sum(Z)
    plt.bar(X,Z)
```