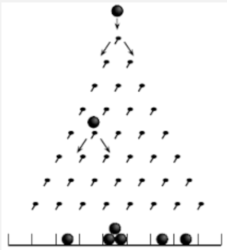
TP # 25

Révisions aléatoires : lois usuelles, représentations graphiques...

Exercice 1

On dispose d'une planche de Galton (dessin à suivre) composée de n lignes de clous formant un triangle, ainsi que de $n+1$ cases, numérotée de 0 à n, au bas de la structure (sur le dessin n = 8) :



N boules sont lachées une par une au sommet du triangle, et dégringolent la structure pour arriver dans une case. La case dans laquelle une boule atterit est modélisée par une variable aléatoire de loi $\mathcal{B}(n, 1/2)$.

On souhaite écrire une fonction `galton` qui prend en entrée les entiers n et N et renvoie un vecteur `V = [v0,v1, ..., vn]` contenant le nombre de boules contenues dans chaque case à l'issue de l'expérience.
(`v0` = nb de boules dans la case 0, `v1` = nb de boules dans la case 1, etc...)

1. *Première idée pour la fonction* ***`galton`*** *:*
- Initialement, chaque case contient 0 boules.
- Pour chaque lancer, on génère le numéro de la case atteint par la boule et on augmente de 1 la valeur correspondante dans le vecteur `V`.

```
import numpy as np ; import numpy.random as rd

def galton(n,N) :

    V = .........................

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




    return(X)
```

2. On choisit $N = 100$ et $n = 10$. Représenter le nombre de boules contenues dans chaque case sous la forme d'un diagramme en bâtons, avec l'instruction `plt.bar`.

```
N = 100 ; n = 10;
import matplotlib.pyplot as plt

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

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

On testera plusieurs fois, avec les valeurs suivantes :
- $n = 10$, $N = 100$ puis 1000 puis 10 000
- $n = 100$, $N = 500$ puis 1000 puis 10 000.

3. *Autre idée pour la fonction* ***`galton`*** *:*
- On génère directement une liste `B = [b1,b2, ..., bN]` contenant le numéro de la case atteinte par chaque boule.
- Pour une telle liste, l'instruction `B == k` correspond à la liste `[b1 == k, b2 == k, ..., bN == k]` qui contient `True` ou `False`. Par exemple :

Si `B` est la liste `[2,0,1,2,2]` alors `B == 2` est la liste `[True,False,False,True,True]`.

En Python, `True` compte comme "1" et `False` compte comme "0".
Ainsi, `np.sum(B == k)`, renvoie le nombre de valeurs de `B` égales à `k`.

```
def galton(n,N) :

    B = ........................

    V = ........................

    for k in range(n+1) :

        V[k] = ....................

    return(X)
```

Remplacer la fonction `galton` précédente par celle-ci, et tester à nouveau l'affichage des diagrammes en bâtons pour s'assurer de son fonctionnement.

✎ Exercice 2

Loi des évènements rares.

Soit $\lambda > 0$. Dans cet exercice, on souhaite démontrer à l'aide de Python que, lorsque n est très grand, $\boxed{\text{la loi binomiale } \mathcal{B}(n, \frac{\lambda}{n}) \text{ est "proche" de la loi de Poisson } \mathcal{P}(\lambda)}$.

Précisément, si $X_n \hookrightarrow \mathcal{B}(n, \frac{\lambda}{n})$, alors on a

$$\forall k \in \mathbb{N}, \quad P(X_n = k) \xrightarrow[n \to +\infty]{} P(X = k), \quad \text{où} \ \ X \hookrightarrow \mathcal{P}(\lambda).$$

1. Pour $n \in \mathbb{N}^*$, $\lambda > 0$ et $k \in \mathbb{N}$, rappeler les valeurs explicites des probabilités :

$P(X = k) = \ldots\ldots\ldots\ldots\ldots\ldots \qquad P(X_n = k) = \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$

2. Proposer une fonction `limite(lam,k)`qui renvoie la valeur de $P(X = k)$.

```
def limite(lam,k) :
```

3. On fixe $\lambda > 0$ et $k \in \mathbb{N}$, et on note $u_n = P(X_n = k)$.

(a) Compléter la fomule de récurrence : $\displaystyle\binom{n+1}{k} = \ldots\ldots\ldots\ldots\ldots\ldots \times \binom{n}{k}$

(b) Compléter le programme pour que `suite(lam,k,N)` renvoie un vecteur contenant la suite de valeurs $(u_n)_{n \in [\![k,N]\!]}$.

```
def suite(lam,k,N) :

    V = np.zeros( ...... );     c = 1;

    for n in range( ........ ) :

        V[ ..... ] = c * (lam/n)**k * (1-lam/n)**(n-k)

        c = ((n+1)/(n+1-k)) * c

    return V
```

(c) On choisit $\lambda = 10$, $k = 10$, $N = 100$. Représenter graphiquement la suite de valeurs $\Big(P(X_n = k)\Big)_{n \in [\![k,N]\!]}$ et constater la convergence vers la valeur $P(X = k)$.

On affichera en bleu (standard) la suite $\Big(P(X_n = k)\Big)_{n \in [\![k,N]\!]}$ et en pointillés rouges l'asymptote horizontale d'équation $y = P(X = k)$.

```
import numpy as np; import matplotlib.pyplot as plt
lam = 10; k = 10; N = 100;

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

plt.plot(X,Y)

plt.plot( [... , ...], [ ............... , ............... ], 'r--' )

plt.show()
```

Constater la convergence attendue.

Dessin :

On pourra tester d'autres valeurs de k : $\ k = 5, \ \ k = 7, \ \ k = 15$ et au delà
(on aura alors intérêt à augmenter la valeur de N car la convergence est plus lente...)