ECHANTILLONNAGE

I. Notion d'échantillon

1) Définition

Exemples :

1) Sur l'ensemble des cartes à puce produites par une entreprise en une semaine, on en prélève 200. On dit que cet ensemble de 200 cartes à puce constitue un **échantillon de taille 200** de la population de toutes les cartes à puce produites en une semaine.

2) On s'intéresse aux intentions de vote lors d'une élection. On sonde 1000 personnes en leur demandant leur intention de vote. L'ensemble de ces 1000 personnes constitue un **échantillon de taille 1000** de la population totale des électeurs.

3) On lance une pièce de monnaie 50 fois de suite et on note les résultats obtenus. L'ensemble de ces 50 lancers constitue un **échantillon de taille 50**.

Définition :
Un **échantillon de taille *n*** est constitué des résultats de n répétitions indépendantes de la même expérience sur l'ensemble des personnes ou objets sur lesquels porte l'étude statistique (la population).
Un échantillon issu d'une population est donc l'ensemble de quelques éléments de cette population.

2) Simulation d'une expérience aléatoire

On considère l'expérience aléatoire qui consiste à lancer un dé à 6 faces. Le programme Python suivant permet de simuler cette expérience.

Le fonction **randint** renvoie un nombre aléatoire entier de 1 à 6.

```
from random import*

def dé():
    r=randint(1,6)
    return(r)
```

```
>>> dé()
1
```

On exécute le programme et on obtient l'affichage ci-contre.
Cela signifie que le logiciel a simulé un lancer de dé et on a obtenu un « 1 ».

La règle du jeu veut que si le résultat est « 1 » ou « 6 », on gagne. Dans le cas contraire, on perd. On répète n fois de suite cette expérience à deux issues (gagner ou perdre) consistant à lancer le dé.

On modifie et complète le programme Python afin de simuler n lancers de dé. Le programme affiche le nombre de fois que l'on gagne.

La variable **n** désigne le nombre de lancers. La variable **s** permet de compter le nombre de fois que l'on gagne : le dé s'arrête sur «1 » ou sur « 6 ».

```
from random import*

def dé(n):
    s=0
    for k in range(n):
        r=randint(1,6)
        if r==1 or r==6:
            s=s+1
    return(s)
```

```
>>> dé(10)
3
```

On exécute le programme et on obtient l'affichage ci-contre. Cela signifie que sur 10 lancers, on a gagné 3 fois.

II. Loi des grands nombres

Modifions le programme afin d'afficher en sortie la fréquence de jeux gagnés sur un échantillon de n lancers de dé.

Il suffit de remplacer dans la dernière ligne **return(s)** (l'effectif) par **return(s/n)** (la fréquence).

```
from random import*

def dé(n):
    s=0
    for k in range(n):
        r=randint(1,6)
        if r==1 or r==6:
            s=s+1
    return(s/n)
```

```
>>> dé(10)
0.2
>>> dé(100)
0.32
>>> dé(1000)
0.328
>>> dé(5000)
0.3372
>>> dé(100000)
0.33353
```

On exécute le programme pour des valeurs de n de plus en plus grandes. Ci-contre les résultats obtenus à l'aide du logiciel.

On constate que, plus n devient grand, plus les fréquences observées semblent se rapprocher d'une valeur théorique égale à $\frac{1}{3}$.

En effet, la probabilité de gagner (obtenir un « 1 » ou un « 6 ») est égale à $\frac{2}{6} = \frac{1}{3}$.

Loi des grands nombres : Lorsque n devient grand, sauf exception, la fréquence observée est proche de la probabilité.

III. Estimation d'une probabilité

On se propose maintenant de répéter N fois la simulation de l'expérience aléatoire précédente. Dans chaque cas, pour n suffisamment grand, la fréquence observée f devrait être proche de la probabilité théorique $p = \frac{1}{3}$.
On veut calculer la proportion des cas pour lesquels l'écart entre f et p est inférieur ou égale à $\frac{1}{\sqrt{n}}$.

Après avoir importé le module **math**, nécessaire pour utiliser la fonction **abs** (valeur absolue), on complète le programme précédent avec la fonction **estim**.

abs(f-1/3) est l'écart entre f et 1/3.
$\sqrt{n}$ se note **sqrt(n)**.

On teste **N** fois si **abs(f-1/3)<=1/sqrt(n)**.
La variable **c** compte le nombre de fois où ce test est vérifié.

```
from math import*

def estim(N,n):
    c=0
    for k in range(N):
        f=dé(n)
        if abs(f-1/3)<=1/sqrt(n):
            c=c+1
    return(c/N)
```

Le programme complet au format texte se trouve sur la dernière page de ce document.

On exécute le programme pour différentes valeurs de N en choisissant n suffisamment grand, soit n = 10000.

On trouve des valeurs proches de 0,95 ce qui signifie que dans 95% des cas, l'écart entre la fréquence observée f et la probabilité p est inférieur ou égale à 0,01.
En effet : $\frac{1}{\sqrt{n}} = \frac{1}{\sqrt{10000}} = 0{,}01$.

```
>>> estim(10,10000)
0.9
>>> estim(50,10000)
0.96
>>> estim(100,10000)
0.96
>>> estim(100,10000)
0.94
```

Principe de l'estimation : Pour n assez grand, f donne une bonne estimation de p dans environ 95 % des cas.

Le programme complet :

```
from random import*
from math import*

def dé(n):
    s=0
    for k in range(n):
        r=randint(1,6)
        if r==1 or r==6:
            s=s+1
    return(s/n)

def estim(N,n):
    c=0
    for k in range(N):
        f=dé(n)
        if abs(f-1/3)<=1/sqrt(n):
            c=c+1
    return(c/N)
```