Loi binomiale
Simulation et Représentation graphique

1 Une simulation

On lance 100 fois une pièce de monnaie non équilibrée (la probabilité de faire Pile est $p = 0,7$). On s'intéresse au nombre de Pile obtenus.

1. *Recopier le programme Python ci-dessous dans l'environnement* AmiensPython.
2. *À quoi sert-il ?* Ce programme sert à simuler le lancer des 100 pièces de monnaie, et afiche le nombre de Pile obtenus.

```
from random import random

p = 0.7
n = 100

pile = 0
for i in range(n):
    if random() < p:
        pile = pile + 1
print pile
```

2 Plusieurs simulations

On aimerait maintenant savoir, sur nos 100 lancers, quelle est la probabilité d'obtenir 60 fois pile ? 3 fois pile ? 100 fois pile ?

L'expérience aléatoire que l'on considère est les 100 lancers de la pièce, et on appelle X la variable aléatoire correspondant au nombre de Pile obtenus. La question que l'on se pose est donc : Combien vaut $P(X = 60)$? $P(X = 3)$? $P(X = 100)$?

1. *Compléter le programme de la première question pour :* (a) *simuler 500 fois de suite l'expérience aléatoire ;* (b) *déterminer le nombre de simulations qui produisent exactement 50 fois Pile sur les 100 lancers.* Pour simuler les 500 expériences, on ajoute une boucle `for j in range(500):` ; pour compter le nombre de fois où le nombre de pile obtenus est exactement 50, on utilise un compteur `compteur`. Cela donne :

```
from random import random

p = 0.7
n = 100
compteur = 0

nombre = 500 # Nombre de simulations

for j in range(nombre):
    pile = 0
    for i in range(n):
        if random() < p:
            pile = pile + 1
    if pile == 50:
        compteur = compteur + 1

print compteur
```

L'exécution de ce programme affiche généralement 0. En effet, il est très rare d'obtenir exactement 50 pile avec cette expérience.

2. *Estimer $P(X = 60)$ (on pourra augmenter le nombre de simulations pour un résultat plus précis).* On remplace le 50 par 60, et en augmentant le nombre de simulations à 10000 (pour plus de précisions), on obtient généralement un nombre proche de 80. D'après la loi des grands nombres, cela donne une probabilité environ égale à $\frac{80}{10000} = 0,8\%$.
3. *Même question avec $P(X = 3)$.* On remplace le 60 par 3, et on obtient 0 à chaque fois. En effet, il est extrèmement rare d'obtenir 3 Pile sur 100 lancers de la pièce.
4. *On souhaite maintenant calculer tous les couples $(k, P(X = k))$, pour k allant de 0 à 100. Modifier l'algorithme précédent. On utilisera une liste X, où :*
   - *pour définir une liste de $n+1$ valeurs initialisées à 0, on utilise* `X=[0]*(n+1)` ;
   - *pour accéder ou modifier la valeur de X de rang i, on utilise* `X[i]`.

   Cela donne :

```
from random import random

p = 0.7
n = 100
X = [0]*(n+1)

nombre = 10000 # Nombre de simulations

for j in range(nombre):
```

```
    pile = 0
    for i in range(n):
        if random() < p:
            pile = pile + 1
    X[pile] = X[pile] + 1

print X
```

Remarque : La manipulation des listes (la variable `X` dans le programme, et toutes les occurences des crochets [ ]) est loin d'être au programme : pas de panique si vous ne comprenez pas, ou n'êtes pas capables de ré-utiliser cela. Par contre, l'analyse des résultats (ci-dessous) est au programme.

Une des exécutions (pour 10000 itérations) donne :

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 1, 3, 4, 6, 9, 15, 40, 50, 75,
153, 191, 291, 350, 480, 570, 680, 750, 798, 870,
858, 805, 740, 644, 493, 355, 306, 178, 105, 79,
48, 26, 14, 5, 5, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0]
```

C'est une liste résultats obtenus. Par exemple, le premier élément vaut 0, ce qui signifie que $P(X = 0) \approx 0$. L'élément d'indice 70 (en comptant à partir de 0) vaut 858, ce qui signifie que $P(X = 70) \approx \frac{858}{10000} \approx 8,58\%$.

3 Représentation graphique

1. *Copier la sortie du programme précédent dans un tableur, puis tracer le graphe des couples* $(k, P(X = k))$.

On peut remplacer la dernière ligne du programme `print X` par :

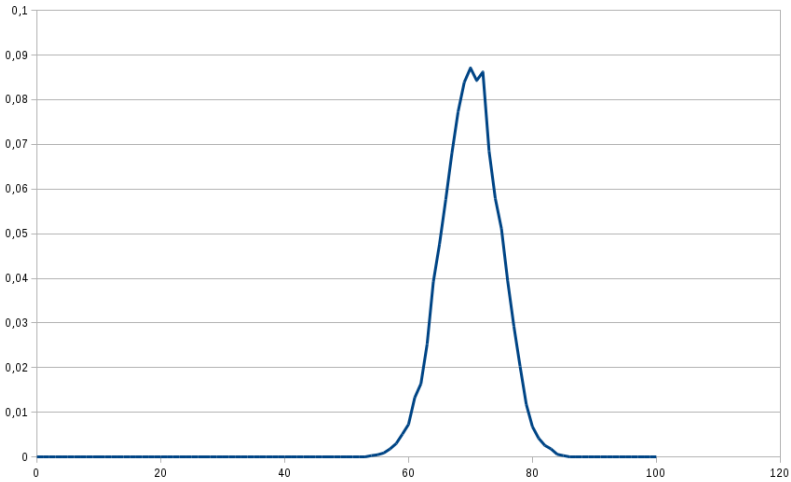
```
for k in range(n+1):
    print k, X[k]/10000.0
```

Cela affiche quelque chose comme les lignes qui suivent :

```
...
45  0.0
46  0.0
47  0.0
48  0.0
49  0.0
50  0.0001
51  0.0001
52  0.0
53  0.0005
54  0.0002
55  0.0004
56  0.001
57  0.0012
58  0.0045
59  0.0057
60  0.0086
...
```

La signification est (en prenant la dernière ligne) : $P(X = 60) \approx 0,0086 \approx 0,86\%$.

On peut copier les lignes dans LibreOffice, et tracer le graphique correspondant :



2. *Recommencer ce tracé en faisant varier les valeurs de p (la probabilité d'obtenir Pile). Quelle est l'influence de p sur cette représentation graphique ?* La correction de cette question a été faite dans le cours.
3. *Recommencer ce tracé en faisant varier les valeurs de n (le nombre de lancers de la pièce). Quelle est l'influence de n sur cette représentation graphique ?* La correction de cette question a été faite dans le cours.