Simulations échantillons variable aléatoire

Auteur : Thomas Léchenne

Niveaux scolaires :
Première

Mots clés :
Algorithme/Programmation
Suites

ENONCE

Simuler, avec Python, N échantillons de taille n d'une variable aléatoire, d'espérance μ et d'écart type σ. Si m désigne la moyenne d'un échantillon, calculer la proportion des cas où l'écart entre m et μ est inférieur ou égal à $\frac{2\sigma}{\sqrt{n}}$.

A la fête foraine se déroule le jeu suivant : pour une mise de 3 euros, on peut lancer un dé à 10 faces. Si le résultat obtenu est 10, le joueur gagne 10 euros. Si le résultat est un multiple de 3, le joueur gagne 3 euros. Si le résultat est un autre chiffre impair, le joueur gagne 2 euros. Et dans tous les autres cas, le joueur ne gagne rien.
Etablissons la loi de probabilité de la variable aléatoire X modélisant le gain (algébrique) à ce jeu.

$X = x_i$	-3	-1	0	7
$P(X = x_i)$	0,3	0,3	0,3	0,1

Calculons l'espérance du gain : $E(X) = -3 \times 0{,}3 - 1 \times 0{,}3 + 0 \times 0{,}3 + 7 \times 0{,}1 = -0{,}5$
En moyenne, le joueur perd 50 centimes par partie.

Calculons maintenant la variance de cette variable aléatoire, à l'aide de la formule de Koenig-Huygens :

$$V(X) = E(X^2) - [E(X)]^2$$

Or $E(X^2) = 9 \times 0{,}3 + 1 \times 0{,}3 + 0 \times 0{,}3 + 49 \times 0{,}1 = 7{,}9$ et $[E(X)]^2 = 0{,}25$
Donc : $V(X) = 7{,}9 - 0{,}25 = 7{,}65$.
L'écart-type vaut $\sigma = \sqrt{V(X)} = \sqrt{7{,}65}$.

Nous allons maintenant créer une fonction Python simulant un échantillon de n (nombre entier naturel) parties du jeu précédent et renvoyant la moyenne des gains.

Pour débuter ce programme, chargeons tout d'abord la bibliothèque "random". Pour cela, nous allons chercher la commande "from random import *" dans le catalogue. Cette bibliothèque est indispensable pour pouvoir générer un nombre au hasard entre 0 et 1 (avec la commande random()).

```
echantil.py    002/002
from random import *
FILE RUN SYMBOL CHAR A⇔a ▷
```

Appelons notre fonction JEU. Elle aura pour argument la valeur de n, le nombre de parties à effectuer (attention à ne pas oublier les deux points qui permettent de démarrer l'indentation, symbolisant ce qui appartient à cette fonction).
Nous allons nommer G la variable comptabilisant le gain cumulé lors de ces parties successives. Au départ, G vaut donc 0.

```
echantil.py    003/003
from random import *
def JEU(n):
  G=0
FILE RUN SYMBOL CHAR A⇔a ▷
```

Ensuite, simulons, à l'aide d'une boucle Pour, n fois successivement notre jeu. Comme l'on effectue n parties, on répétera la même expérience pour i allant de 0 à $n-1$. Ce qui donne, en langage Python "for i in range(n):".

```
echantil.py    004/004
from random import *
def JEU(n):
  G=0
  for i in range(n):
FILE RUN SYMBOL CHAR A⇔a ▷
```

Nous allons simuler directement le gain du jeu (ce qui est plus simple que de simuler le résultat du dé puis d'effectuer le gain correspondant). En effet, il suffit de s'occuper des probabilités de chacune des valeurs possibles du gain et de répartir ces valeurs sur l'intervalle $[0;1]$. Ainsi, grâce à la commande random(), on génère un nombre aléatoire X entre 0 et 1.

```
echantil.py    005/005
from random import *
def JEU(n):
  G=0
  for i in range(n):
    X=random()
FILE RUN SYMBOL CHAR A⇔a ▷
```

Si celui-ci est compris entre 0 et 0,3, on considère que le gain est de -3 $(P(X \leq 0{,}3) = 0{,}3)$, la correspond ainsi exactement à la probabilité que le gain soit égal à -3). On retranche alors 3 à la valeur de G (on peut utiliser la commande "$-=$ " pour effectuer cette opération).

```
echantil.py    006/007
def JEU(n):
  G=0
  for i in range(n):
    X=random()
    if X<=0.3:
      G-=3
```

La probabilité que le gain d'une partie soit égal à -1 est aussi de 0,3. On peut faire correspondre cela à la probabilité que notre valeur de X soit comprise dans l'intervalle $]0{,}3; 0{,}6]$. Pour traduire cette appartenance en langage Python, nous aurons besoin de la commande "and" puisque on ne peut tester directement un encadrement en langage Python (comme dans la plupart des langages d'ailleurs).
On utilise alors le fait que "$X \in]0{,}3; 0{,}6]$" est équivalent à "$X > 0{,}3$ et $X \leq 0{,}6$"

```
echantil.py    008/012
=0
or i in range(n):
 X=random()
 if X<=0.3:
   G-=3
 if X>0.3 and X<=0.6:
   G-=1
```

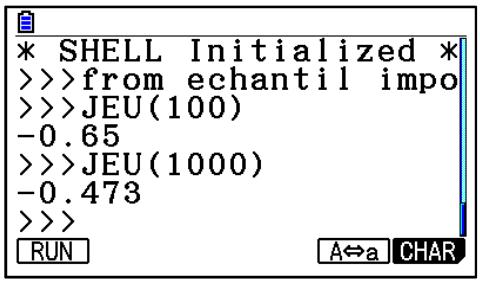
Lorsque le gain vaut 0, la valeur de notre variable G ne change pas. Il est donc inutile de programmer ce cas-là. Néanmoins on peut considérer que celui-ci se produit lorsque la variable X appartient à l'intervalle $]0{,}6; 0{,}9]$. Ainsi le dernier cas, lorsque le gain est égal à 7, se produit pour $X > 0{,}9$. On utilise la commande "$+=$ " pour ajouter 7 à la variable G.

```
echantil.py    011/012
 X=random()
 if X<=0.3:
   G-=3
 if X>0.3 and X<=0.6:
   G-=1
 if X>0.9:
   G+=7
```

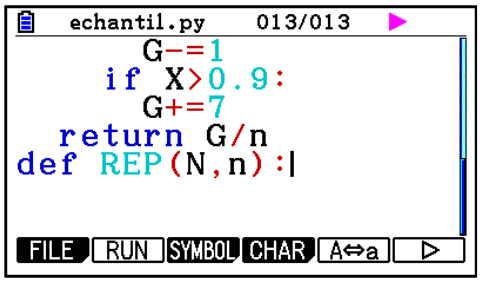
Il ne reste plus qu'à renvoyer la valeur de la moyenne des gains (cette valeur est G/n). Attention à sortir des indentations de la structure conditionnelle et de la boucle Pour (mais à rester dans celle de la fonction JEU).

```
echantil.py    011/012
     if X<=0.3:
       G-=3
     if X>0.3 and X<=0
       G-=1
     if X>0.9:
       G+=7
  return G/n
```

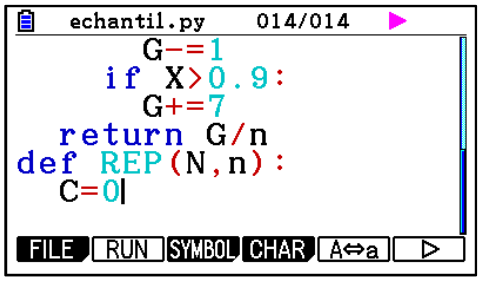
On peut maintenant exécuter cette simulation en allant dans l'onglet **{Run}**. On peut remarquer que plus la valeur de n est grande, plus la probabilité que la moyenne soit proche de l'espérance est importante. C'est ce que l'on appelle la loi des grands nombres.



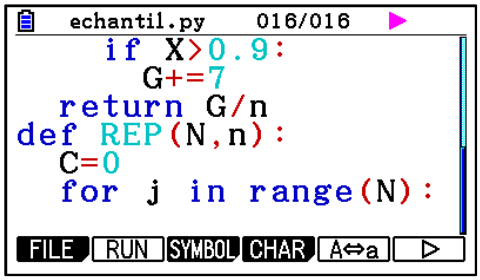
Il faut maintenant répéter N fois cet échantillon. Pour cela, dans le même fichier, nous allons créer une nouvelle fonction, dénommée REP, ayant pour argument le nombre N de répétitions de notre échantillon et la taille n de chaque échantillon. A noter que l'on revient à l'écriture du programme en appuyant sur la touche EXIT.



Nous allons initialiser (à 0) une variable C qui comptabilisera le nombre d'échantillons dont l'écart entre la moyenne et l'espérance est inférieure ou égale à $\frac{2\sigma}{\sqrt{n}}$.



Ensuite, nous allons simuler un à un chaque échantillon de taille n et tester l'écart entre la moyenne et l'espérance. On utilise pour cela une boucle Pour (pour j allant de 0 à $N-1$).



Si l'écart (c'est-à-dire la valeur absolue de la différence) entre la moyenne (valeur stockée dans JEU(n)) et l'espérance (qui vaut $-0{,}5$) est inférieure ou égal à $\frac{2\sigma}{\sqrt{n}}$. Alors, on ajoute 1 à notre variable C. A noter que l'on utilise "$**0.5$" (puissance un demi) en lieu et place de la racine carrée pour éviter de charger la bibliothèque Math.

Il ne reste plus qu'à renvoyer la proportion des échantillons où l'écart est inférieur ou égal à $\frac{2\sigma}{\sqrt{n}}$ c'est-à-dire C/N.

On peut enfin tester notre programme (onglet **{Run}**)

```
echantil.py    018/018
  return G/n
def REP(N,n):
  C=0
  for j in range(N):
    if abs(JEU(n)+0.5
      C+=1
  return C/N
FILE RUN SYMBOL CHAR A⇔a ▷
```

```
|CASIO COMPUTER CO.,
>>>from echantil impo
>>>REP(100,100)
0.95
>>>REP(1000,100)
0.9609999999999999
>>>
RUN                A⇔a CHAR
```

Le programme dans son entièreté :

```
echantil.py    001/018
from random import *
def JEU(n):
  G=0
  for i in range(n):
    X=random()
    if X<=0.3:
      G-=3
    if X>0.3 and X<=0.6:
      G-=1
    if X>0.9:
      G+=7
  return G/n
def REP(N,n):
  C=0
  for j in range(N):
    if abs(JEU(n)+0.5)<=2*(7.65/n)**0.5:
      C+=1
  return C/N
FILE RUN SYMBOL CHAR A⇔a ▷
```