Thème: simulation ; loi des grands nombres

Niveau: 2nde

TI-83 Premium CE

python

Une somme de hasards

F. GIROD

Compétences visées

- **modéliser**, faire une simulation, valider ou invalider un modèle ;
- **raisonner**, démontrer, trouver des résultats partiels et les mettre en perspective ;
- **calculer**, appliquer des techniques et mettre en œuvre des algorithmes.

« En classe de seconde, on formalise la notion de loi (ou distribution) de probabilité dans le cas fini en s'appuyant sur le langage des ensembles et on précise les premiers éléments de calcul de probabilités. On insiste sur le fait qu'une loi de probabilité (par exemple une équiprobabilité) est une hypothèse du modèle choisi, et ne se démontre pas. Le choix du modèle peut résulter d'hypothèses implicites d'équiprobabilité (par exemple, lancers de pièces ou dés équilibrés, tirage au hasard dans une population) qu'il est recommandable d'expliciter : **il peut aussi résulter d'une application d'une version vulgarisée de la loi des grands nombres, où un modèle est construit à partir de fréquences observées pour un phénomène réel** (par exemple : lancer de punaise, sexe d'un enfant à la naissance). Dans tous les cas, on distingue nettement le modèle probabiliste abstrait et la situation réelle. »

Il est précisé dans cet extrait de programme pour la classe de 2^{nde} que les fréquences sont observées sur un phénomène réel ; on l'élargit ici à une simulation numérique dans le cas où la probabilité de réalisation de l'événement attendu ne se calcule pas facilement. Si le fait de réaliser des essais 'physiques' est important, une simulation numérique est un complément intéressant comme illustration de « la loi des grands nombres ».

Situation déclenchante

Somme de quatre dés ...

On lance quatre dés à six faces (dés bien équilibrés) et on s'intéresse à la somme des valeurs des faces supérieures.

Quelle est la probabilité que cette somme soit inférieure ou égale à 18 ?

Problématique

Comment traiter cette situation : modèle probabiliste ou simulation ?

Thème: simulation ; loi des grands nombres

Niveau: 2nde

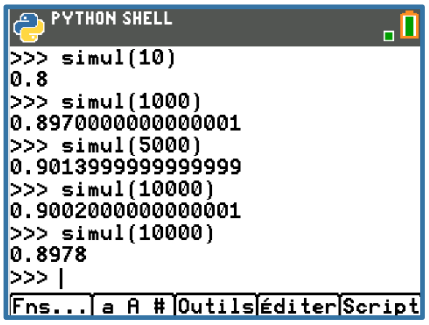
TI-83 Premium CE

python

F. GIROD

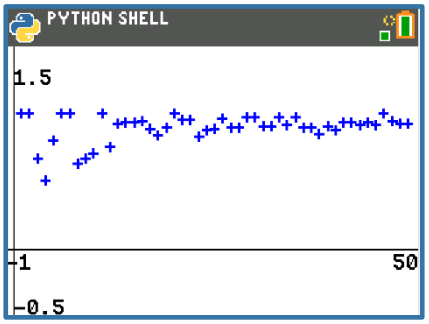
Fiche méthode

Proposition de résolution



On pourra effectuer des simulations grâce à un script en langage Python.

- Écrire une fonction **simul** de paramètre *n* (le nombre fois où on lance les quatre dés) qui renvoie la fréquence où la somme des valeurs des faces supérieures des quatre dés est inférieure ou égale à 18.
- Créer un compteur qui est incrémenté chaque fois que la somme des valeurs des faces supérieures des quatre dés est inférieure ou égale à 18.
- On peut d'une part observer l'évolution de la fréquence en fonction du nombre d'essais. On visualise alors une stabilisation de cette fréquence vers une certaine valeur, illustrant ainsi la « loi des grands nombres ».
- Cette observation peut être numérique (première copie d'écran) ou graphique : le script permettant de générer le graphique ci-contre est donné à la fin de cette fiche.
- Par ailleurs, on peut dès à présent observer la fluctuation d'échantillonnage en testant plusieurs fois la fonction **simul** avec la même valeur de *n*.



Remarque

Quelle que soit la méthode utilisée, il faudra un générateur de nombres (pseudo) aléatoires et donc importer la bibliothèque « random » par « **from random import ✶** »
Cette importation est en préambule du code.

On peut aussi choisir d'importer seulement la fonction randint issue de la bibliothèque random en saisissant « **from random import randint** ».

Lorsque l'on saisit « from random import ✶ », cela sous-entend que l'on importe toutes les fonctions de cette bibliothèque.

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Thème: simulation ; loi des grands nombres

Niveau: 2nde

TI-83 Premium CE

python

F. GIROD

Fiche méthode

Etapes de résolution

La fonction **simul** simule le lancer de quatre dés. Elle a pour paramètre *n* qui représente le nombre de répétitions et renvoie la fréquence de réussite de l'expérience aléatoire.

```
EDITEUR : LOIGDNBR
LIGNE DU SCRIPT 0009
from random import randint

def simul(n):
  c=0
  for i in range(n):
    if randint(1,6)+randint(1,6)
      +randint(1,6)+randint(1,6)<
      =18:
        c=c+1
  return c/n
Fns... | a A # | Outils | Exéc | Script
```

- On initialise un compteur (représenté par la variable *c*) à 0.
- Chaque fois que la somme des quatre nombres entiers aléatoires compris entre 1 et 6 est inférieure ou égale à 18, ce compteur est incrémenté.
- Au final, le compteur est égal au nombre de cas favorables.
- En le divisant par *n*, le nombre d'essais, la fonction retourne la fréquence de réussite de l'expérience aléatoire.

Remarque

Pour un élève de 2nde, il n'est peut-être pas évident que :

- randint(1,6) + randint(1,6) + randint(1,6) + randint(1,6)
- randint(1,6) x 4
- randint(4,24)

ne représentent pas la même simulation.

Il peut être intéressant pour convaincre les élèves de lancer chacune de ces fonctions et de comparer les résultats comme l'illustre la copie d'écran ci-contre.

On observe des valeurs significativement différentes qui confirment que ces simulations ne sont pas équivalentes.

```
ÉDITEUR : LOIGDNBR
LIGNE DU SCRIPT 0020
def sim2(n):
  c=0
  for i in range(n):
    if randint(1,6)*4<=18:
      c=c+1
  return c/n

def sim3(n):
  c=0
  for i in range(n):
    if randint(4,24)<=18:
Fns… | a A # | Outils | Exéc | Script
```

```
PYTHON SHELL
>>> simul(1000)
0.9
>>> sim2(1000)
0.64
>>> sim3(1000)
0.706
>>>
Fns... | a A # | Outils | éditer | Script
```

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Thème: simulation ; loi des grands nombres

Niveau: 2nde

TI-83 Premium CE

python™

Fiche méthode

F. GIROD

Approche probabiliste

On peut avoir une approche 'probabiliste' en dénombrant le nombre de cas favorables.

Lorsqu'on lance quatre dés à six faces, dans combien de cas la somme des valeurs des faces supérieures des dés sortis est-elle inférieure ou égale à 18 ?

Si on se lance dans un dénombrement « à la main », cela risque d'être long et fastidieux, et le risque de se tromper n'est pas négligeable.

Un programme peut dénombrer le nombre de cas favorables ; la calculatrice est en effet adaptée pour effectuer ce type de tâche longue et fastidieuse.

On va donc tester toutes les sommes de quatre entiers compris entre 1 et 6 et comptabiliser le nombre de cas où la somme est inférieure ou égale à 18.

Ceci permet d'obtenir la probabilité de réalisation de l'événement voulu, et de comparer a posteriori les fréquences obtenues avec cette probabilité.

La fonction **test** ci-contre retourne le résultat sous la forme du couple (numérateur, dénominateur) soit ici (1170,1296) et on pourra comparer cette valeur à celles obtenues par simulations.

On obtient $p = \frac{1170}{1296} \approx 0{,}90278$.

```
ÉDITEUR : LOIGDNBR
LIGNE DU SCRIPT 0033
def test():
  c=0
  for i in range(1,7):
    for j in range(1,7):
      for k in range(1,7):
        for l in range(1,7):
          if i+j+k+l<=18:
            c=c+1
  return c,6**4
Fns… | a A # | Outils | Exéc | Script
```

```
PYTHON SHELL
>>> sim(1000)
0.8859999999999999
>>> sim(1000)
0.9140000000000001
>>> sim(1000)
0.8890000000000001
>>> test()
(1170, 1296)
>>> 1170/1296
0.9027777777777779
>>> |
Fns… | a A # | Outils | Éditer | Script
```

Complément : graphique par ti_plotlib

On peut réaliser le graphique présenté au début de cette fiche (fréquence de réussite en fonction du nombre d'essais), ce qui permet d'illustrer la stabilisation de la fréquence autour d'une valeur lorsque *n* augmente.

Pour cela, on utilise des fonctions de la bibliothèque **`ti_plotlib`** comme présenté ci-contre.

```
ÉDITEUR : LOIGDNBR
LIGNE DU SCRIPT 0045
import ti_plotlib as plt

def g(n):
  x,y=[],[]
  for i in range(1,n):
    x.append(i)
    y.append(sim(i))
  plt.cls()
  plt.window(-1,n,-0.5,1.5)
  plt.axes("on")
  plt.color(0,0,255)
  plt.scatter(x,y,"+")
  plt.color(0,0,0)
  plt.show_plot()
Fns… | a A # | Outils | Exéc | Script
```

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus