# Probabilités: quand un tableur ne peut rien pour vous

publié le 19/09/2022

## Python pour simuler le jeux de "Monty Hall"

*Descriptif :*

Le jeu "let's make a deal" présenté par Monty Hall à la TV nord américaine peut être simulé en Python sans que la probabilité de gagner apparaisse explicitement.

*Sommaire :*

- Capacités et connaissances
- Let's make a deal : le problème de Monty Hall
- Activités
- Documents
- Questions ouvertes

Fluctuations d'une fréquence selon les échantillons, probabilités

### Capacités et connaissances

| Capacités | Connaissances |
|---|---|
| Expérimenter pour observer la fluctuation des fréquences (jets de dés, lancers de pièces de monnaie…). | Vocabulaire des probabilités : expérience aléatoire, ensemble des issues (univers), événement, probabilité. |
| Estimer la probabilité d'un événement à partir des fréquences | Stabilisation relative des fréquences vers la probabilité de l'événement quand n augmente. |

### Let's make a deal : le problème de Monty Hall

Let's make a deal est un jeu de la télévision américaine des années 1970, présenté par Monty Hall. Il est évoqué dans le film Las vegas 21.

**La règle du jeux est la suivante :**

> *Le jeu oppose un présentateur à un candidat (le joueur). Ce joueur est placé devant trois portes fermées. Derrière l'une d'elles se trouve une voiture et derrière chacune des deux autres se trouve une chèvre. Il doit tout d'abord désigner une porte. Puis le présentateur doit ouvrir une porte qui n'est ni celle choisie par le candidat, ni celle cachant la voiture (le présentateur sait quelle est la bonne porte dès le début). Le candidat a alors le droit d'ouvrir la porte qu'il a choisie initialement, ou d'ouvrir la troisième porte. Quelle est la meilleure stratégie pour remporter la voiture ?[1] :*

- *Conserver son choix initial ?*
- *Choisir l'autre porte restée fermée ?*

Dans ce jeux la détermination de la probabilité de gagner si l'on change son choix n'est pas immédiate, voire contre intuitive. Un candidat qui change d'avis modifie-t-il ses chances de remporter la voiture ?

Dans la mesure ou cette probabilité n'est pas connue, la modélisation avec un tableur n'est pas si évidente à réaliser, mais cela reste possible. L'expressivité du langage Python permet de rester proche de la conception mathématique du problème avec des ensembles ou des listes sans connaître la probabilité de gain de chaque stratégie.

## Activités

On propose :

- Une activité déconnectée pour identifier la bonne stratégie de gain.
- Un premier programme python simulant une unique partie. Jouer une vingtaine de parties, valide la stratégie.
- Un deuxième programme permet, en le modifiant, d'augmenter le nombre de répétitions des parties pour voir la stabilisation de la fréquence de gain des deux stratégies se stabiliser. Cette stabilisation de la fréquence estime la probabilité de gain de chacune des stratégies.

On peut vérifier ainsi que la fréquence de gain d'un joueur qui ne changerait pas d'avis, tend vers P(gain)=1/3.

### Activité déconnectée

Avant de joueur avec les simulation Python, il est sans doute préférable de commencer par une activité déconnectée. Un document papier est disponible pour les élèves :

.

montyhall-debranche (PDF de 139 ko)

En faisant quelques parties

### Une première simulation en Python

**Code Python simulant une unique partie du jeu des trois portes** :

Pour s'approprier la simulation, on peut commencer par proposer d'analyser une sortie du programme :

`import time`
Le module time permettra d'afficher l'heure d'exécution du programme
`import random`
Sans ce module, pas de simulation aléatoire
`PORTES_FERMEES = ["porte1", "porte2", "porte3"]`
Ceci est la liste des trois portes.
`PORTE_VOITURE = set([random.choice(PORTES_FERMEES)])`
On a choisi aléatoirement la porte dissimulant la voiture.
`CHOIX1_JOUEUR = set([random.choice(PORTES_FERMEES)])`
Le joueur choisit la porte dissimulant la voiture. Comme le jeux n'est pas interactif, le choix est aléatoire.
`PORTES_FERMEES = set(PORTES_FERMEES)`
On convertit la liste des portes en ensemble des portes de manière à pouvoir utiliser les opération ensemblistes.
On place une chèvre derrière une porte où il n'y a pas de voiture, c'est l'ensemble des portes dissimulant une chèvre privé de l'ensemble des portes dissimulant une voiture (ici un seul élément, "la bonne porte")
`PORTES_CHEVRES = PORTES_FERMEES - PORTE_VOITURE`
Le présentateur sait où est la voiture. Il choisit donc une porte où se trouve une chèvre. On choisit aléatoirement un élément (porte) de l'ensemble des portes privée de voiture.
Malheureusement il faut ici retransformer l'ensemble "chevres" en liste pour faire un tirage aléatoire entre les deux

portes.
Mont Hall doit choisir une porte qui ne cache pas de voiture et qui n'est pas celle du joueur. On réalise un tirage aléatoire dans cet ensemble avec un tour de passe-passe exploitant une liste :

```
CHOIX_PRESENTATEUR = random.choice(list(PORTES_CHEVRES - CHOIX1_JOUEUR))
CHOIX_PRESENTATEUR = set([CHOIX_PRESENTATEUR])
PORTES_FERMEES = PORTES_FERMEES - CHOIX_PRESENTATEUR
```

On simule un joueur qui ne change pas son choix :

```
print("Monthy Hall demande au joueur s'il veut changer son choix:")
print("      Non, le joueur ne change pas son choix, on ouvre la ", CHOIX1_JOUEUR)
```

Monty Hall ouvre une porte, Il y a deux issues :

- La porte du joueur et celle de la voiture sont identique (les deux ensembles de portes sont égaux, ce qui en écriture ensembliste s'exprime par chaque ensemble étant sous ensemble de l'autre).
- sinon le joueur perd.

Il faut donc introduire un test *if ... else ...*

```
if CHOIX1_JOUEUR.issubset(PORTE_VOITURE) and PORTE_VOITURE.issubset(CHOIX1_JOUEUR):
  print('      le joueur gagne la voiture')
else:
  print('      le joueur perd')
```

On simule maintenant un joueur qui change d'avis :

```
NOUVEAU_CHOIX = PORTES_FERMEES - CHOIX1_JOUEUR
print("      Oui: le joueur change son choix, on ouvre la ",NOUVEAU_CHOIX)
```

Si un ensemble E1 est ss ensemble de E2 et E2 ss ensemble de E1 alors E1=E2

```
if NOUVEAU_CHOIX.issubset(PORTE_VOITURE) and PORTE_VOITURE.issubset(NOUVEAU_CHOIX):
  print('      le joueur gagne la voiture')
else:
  print('      le joueur perd')an
```

L'ensemble du programme est disponible ci-dessous :



```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 23 18:44:50 2021

@author: jeanpat
"""

#On importe le module random pour faire un tirage aléatoire dans une liste
import time
import random

print(time.strftime(' %d /%m/%Y %H:%M:%S'))
print()
print('----------Simulation d\'une partie du jeux des trois portes----------')
print('      Jeu de la télévision américaine présenté par Monty Hall')
```

```
print()
#Il y a trois portes, enregistrée dans une liste
PORTES_FERMEES = ["porte1", "porte2", "porte3"]#ceci est une liste
#On place une voiture derrière une porte
#      convertion de la liste en ensemble (pour la suite)
PORTE_VOITURE = set([random.choice(PORTES_FERMEES)])
print("La voiture est cachée derrière la ", PORTE_VOITURE)

CHOIX1_JOUEUR = set([random.choice(PORTES_FERMEES)])
PORTES_FERMEES = set(PORTES_FERMEES)
print("Le joueur choisit la ", CHOIX1_JOUEUR)
print()
#On place une chèvre derrière une porte où il n'y a pas de voiture:
PORTES_CHEVRES = PORTES_FERMEES - PORTE_VOITURE
#Le présentateur sait où est la voiture. Il choisit donc une porte où se trouve une chèvre
#Malheureusement il faut ici retransformer l'ensemble "chevres" en liste pour faire un tirage aléatoire #entre les deux portes
#print("Le présentateur ne choisit pas la porte du joueur, ni celle de la voiture")

CHOIX_PRESENTATEUR = random.choice(list(PORTES_CHEVRES - CHOIX1_JOUEUR))
CHOIX_PRESENTATEUR = set([CHOIX_PRESENTATEUR])

print("Monty Hall ouvre la ", CHOIX_PRESENTATEUR)
# Le joueur ne change pas son choix
PORTES_FERMEES = PORTES_FERMEES - CHOIX_PRESENTATEUR
print("Les ", set(PORTES_FERMEES), "sont fermées.")
print()
print("Monthy Hall demande au joueur s'il veut changer son choix:")
print("      Non, le joueur ne change pas son choix, on ouvre la ", CHOIX1_JOUEUR)


if CHOIX1_JOUEUR.issubset(PORTE_VOITURE) and PORTE_VOITURE.issubset(CHOIX1_JOUEUR):
    #print("Le joueur",CHOIX1_JOUEUR, '--', PORTE_VOITURE, 'le joueur gagne la voiture')
    print('         le joueur gagne la voiture')
else:
    #print("----- ",CHOIX1_JOUEUR, '--', PORTE_VOITURE, ' le joueur gagne la chèvre')
    print('         le joueur perd')
# le joueur change d'avis
NOUVEAU_CHOIX = PORTES_FERMEES - CHOIX1_JOUEUR
print("      Oui: le joueur change son choix, on ouvre la ",NOUVEAU_CHOIX)
# Si un ensemble E1 est ss ensemble de E2
# et E2 ss ensemble de E1 alors E1=E2
if NOUVEAU_CHOIX.issubset(PORTE_VOITURE) and PORTE_VOITURE.issubset(NOUVEAU_CHOIX):
    #print(NOUVEAU_CHOIX, '--', PORTE_VOITURE, 'le joueur aurait gagné la voiture')
    print('         le joueur gagne la voiture')
else:
    #print(NOUVEAU_CHOIX, '--', PORTE_VOITURE, ' le joueur aurait gagné la chèvre')
    print('         le joueur perd')
print()
```

MonthyHall_PartieSimple.py hosted with ❤ by GitHub view raw

## Simulation Python : observer la fluctuation de fréquence de gain

La répétition automatisée d'une partie du jeux des trois portes montre la diminution de la fluctuation de la fréquence des gains avec le nombre croissant de répétitions.
Cette deuxième partie de la simulation est disponible dans un notebook Basthon.

Une sortie du programme donne par exemple :

```
23 /06/2022 18:15:27
Un joueur joue 100 fois au jeu de Monthy Hall :
le joueur ne change pas son choix
le joueur gagne : 34 fois sur 100
le joueur change son choix
En changeant son choix, le joueur gagne : 68 fois sur 100
```

Il faut déterminer la variable du programme à modifier pour augmenter le nombre de parties. Dans cet exemple, il est normal que le total 34+68 soit différent de 100. Le programme teste séquentiellement les deux stratégies (ne pas changer de choix ou changer de choix).
Ce deuxième programme, n'utilise que des listes et pas d'ensemble (set) :

⚠ This file contains bidirectional Unicode text that may be interpreted or compiled differently than what appears below. To review, open the file in an editor that reveals hidden Unicode characters. Learn more about bidirectional Unicode characters
Show hidden characters

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Jun 17 17:56:29 2019
Simulation du jeux MONTY HALL
@author: jean-patrick Pommier
"""
# random : module permettant de faire des tirages aléatoire
import time
import random

print(time.strftime(' %d /%m/%Y %H:%M:%S'))
gain = 0
#Nombre de répétitions du jeux
N_JEUX = 100

print("Un joueur joue ",N_JEUX," fois au jeu de Monthy Hall:")
print('le joueur ne change pas son choix')
for jeux in range(N_JEUX):
    #liste contenant le numéro de chacub=ne des trois portes
    PORTES = [1, 2, 3]
#   On choisit une des trois portes
    voiture = random.choice(PORTES)
    choix1_joueur = random.choice(PORTES)
#   Le présentateur choisit une porte au hasard sauf celle
#   qui cache la voiture
    PORTES.remove(voiture)
    if voiture != choix1_joueur:
        PORTES.remove(choix1_joueur)

    presentateur = random.choice(PORTES)
    if choix1_joueur == voiture:
        gain = gain + 1

print('   le joueur gagne :', gain,' fois sur ',N_JEUX)


print('le joueur change son choix')
gain = 0
for jeux in range(N_JEUX):
    PORTES = [1, 2, 3]
    voiture = random.choice(PORTES)
```

```
43      choix1_joueur = random.choice(PORTES)
44      #le présentateur ne choisit pas la porte de la voiture
45      PORTES.remove(voiture)
46      if voiture != choix1_joueur:
47          PORTES.remove(choix1_joueur)
48      presentateur = random.choice(PORTES)
49
50      #Le joueur change d'avis
51      #On a besoin de la liste des trois portes
52      PORTES = [1, 2, 3]
53      #le joueur ne choisit pas la porte ouverte(donc on supprime la porte du presentateur)
54      PORTES.remove(presentateur)
55      #le joueur ne choisit pas son premier choix(donc on supprime la porte du premier choix)
56      PORTES.remove(choix1_joueur)
57      #astuce pour avoir le numéro de la porte et pas une liste
58      choix2_joueur = PORTES.pop()
59
60      if choix2_joueur == voiture:
61          gain = gain + 1
62  print('   En changeant son choix, le joueur gagne :', gain,' fois sur ',N_JEUX)
```

MonthyHall_PartiesRépétées.py hosted with ❤ by GitHub view raw

## ● Documents

- Un deuxième document élève permet de noter les résultats des expériences

  TP probabilités : le jeux des trois portes (PDF de 1010.6 ko)
  Utilisation d'une simulation informatique en langage python

- La version pdf du notebook est également téléchargeable

  Le jeux "let's make a deal" ou jeux des trois portes, présenté par Monty Hall (PDF de 133.2 ko)

▸ notebook Basthon

## ● Questions ouvertes

- que se passe-t-il si on ajoute des portes ?
- comment modifier le premier programme pour répéter des parties ?

(1) https://fr.wikipedia.org/wiki/Problème_de_Monty_Hall