TD4 : Simulations probabilistes

Fonctions définies précédemment

On pourra réutiliser dans ce TD la fonction `affiche_tableau` définie lors du TD2.

Probabilités lors d'un lancer de dés

On lance deux dés à six faces, et on calcule la somme X des chiffres indiqués.

Déterminer la loi de X, c'est à dire calculez chacune des probabilités $\mathbb{P}[X=0]$, $\mathbb{P}[X=1]$,$\mathbb{P}[X=2]$, $\mathbb{P}[X=3]$, etc.

Écrire une fonction `proba_des` telle que `proba_des(k)` calcule $\mathbb{P}[X=$ `k`$]$

Simulation d'un lancer de dés

Le module `random` contient une fonction `choice` pour tirer un élément au hasard parmi une liste d'éléments. Ainsi, `choice(range(1,7))` choisit au hasard un entier entre 1 et 6 (selon la loi uniforme).

Pour simuler un lancer de deux dés il suffit d'addition deux nombres tirés au hasard de cette façon, comme ci-dessous :

```
from random import choice
def simul_deux_des():
    return choice(range(1,7))+choice(range(1,7))
```

On peut ensuite calculer, lorsqu'on répète `n` fois l'expérience, la proportion de fois où X a pris chaque valeur, et on peut le comparer aux probabilités obtenues précédemment

```
def repete_des(n):
    """Cette fonction répète n fois un lancer de dés et calcule
    la proportion de fois où chaque valeur est tombée"""
    nbs=[0]*15
    for i in range(n):
        v=simul_deux_des()
        nbs[v]=nbs[v]+1
    return [["","Proportion","Probabilite"]]+[['X='+str(i),
            (float(nbs[i])/n),proba_des(i)] for i in range(15)]

affiche_tableau(repete_des(200000))
```

On constate, lorsque le nombre de lancers de dés est élevé, que les proportions ainsi obtenues sont très proches des probabilités calculées précédemment.

Lancer de fléchettes

On considère un rectangle de $2m \times 2m$, au centre duquel se trouve un cible de $1m$ de rayon. On suppose que l'on lance une fléchette dans le rectangle, selon une loi uniforme.

Quelle est la probabilité que la fléchette arrive dans le cercle ?

Enregistrez là dans la variable `proba_cercle`.

```
proba_cercle= ....
```

Écrire une fonction qui, étant donné deux nombres x et y, détermine si le point de coordonnées (x,y) est à l'intérieur du cercle de centre 0 et de rayon 1.

Par exemple `f(0,0.5)` devra renvoyer `True` tandis que `f(-1,0.5)` devra renvoyer `False`.

Le module `random` contient aussi une fonction `uniform` qui permet de simuler une loi sur un intervalle arbitraire. Ainsi, la position aléatoire de la fléchette s'obtient en choisissant une abscisse et une ordonnée selon la loi `uniform(-1,1)`

On peut maintenant simuler de nombreuse fois ce "lancer de fléchettes" et comparer la proportion expérimentale à la probabilité théorique :

```
from random import uniform
def repete_cercle(n):
    p=0
    for i in range(n):
        if dans_cercle(uniform(-1,1),uniform(-1,1)):
```

```
            p=p+1
    return float(p)/n

for n in [10**k for k in range(1,7)]:
    print("En répétant "+str(n)+""" fois, on a obtenu la
     proportion expérimentale """+str(repete_cercle(n))+""" que
     l'on peut comparer à la valeur théorique """+str(proba_cercle))
    print("")
```

Le Monty Hall

Monty Hall est le nom du présentateur d'un jeu télévisé où le candidat pouvait gagner une voiture. Il présentait au candidat trois boites dont deux contenaient des chèvres et une contenait la voiture (le présentateur savait laquelle mais pas le candidat). Après que le candidat ait choisit une boite, le présentateur ouvrait une autre boite que celle désignée par le candidat, mais qui contenait une chèvre. Le candidat se voyait alors proposer de choisir la dernière boite ou de conserver son choix initial. Si la boite qu'il choisissait à la fin contenait la voiture, le candidat gagnait la voiture, sinon il perdait.

À l'aide de simulations, on souhaite estimer la probabilité de gagner pour un candidat qui change de boite après qu'on ait ouvert une des boites perdantes.

De même on souhaite estimer cette probabilité pour un candidat qui ne changerait pas de boite.

Pour cela vous compléterez (puis exécuterez) le code ci-dessous.

```
def candidat_qui_change():
    bv=choice(range(3)) #boite de la voiture
    bc=choice(range(3)) #boite du candidat
    lb=[]
    for b in range(3):
        if b!=bv and b!=bc:
            lb=lb+[b]
    bo=choice(lb) #boite ouverte par le présentateur
    lb=[]
    for b in range(3):
        if .....:
            lb=lb+[b]
    bc=lb[0] #changement d'avis du candidat
    return bc==bv

def candidat_qui_ne_change_pas():
    ............
    ............
    return bc==bv

def repete_monty_hall(n):
    n1,n2=0,0
    for i in range(n):
        if candidat_qui_change():n1=n1+1
        if ............
    return (float(n1)/n,float(n2)/n)

print(repete_monty_hall(5000))
```

Enfin, déterminer par le calcul la probabilité exacte de gain dans chacune des deux situations

Test de dépistage

On considère un test de dépistage d'une maladie.

On suppose que la maladie touche une proportion p_0 de la population.

On suppose aussi que lors du test, un individu sein a une probabilité e_1 d'être diagnostiqué à tort comme malade, et qu'un individu malade a une probabilité e_2 de ne pas être diagnostiqué comme malade.

Si un individu est diagnostiqué malade, quelle est la probabilité qu'il le soit réellement ?

Vous la déterminerez par le calcul et comparerez avec des simulations, comme précédemment.

En particulier commentez le cas d'une maladie rare, pour laquelle $p_0 = 10^{-5}$, $e_1 = 10^{-3}$, et $e_2 = 10^{-3}$.