L3 E - S5 2018-19 --- TD ordinateurs — séance 6 — solutions

Probabilités

1 Exercice 1 : lancers de dés **1**

2 Exercice 2 : tirages dans une urne **2**

3 Exercice 3 : la loi binomiale **3**

4 Exercice 4 : le jeu de Saint Petersbourg **4**

5 Exercice 5 : le jeu de Monty Hall **6**

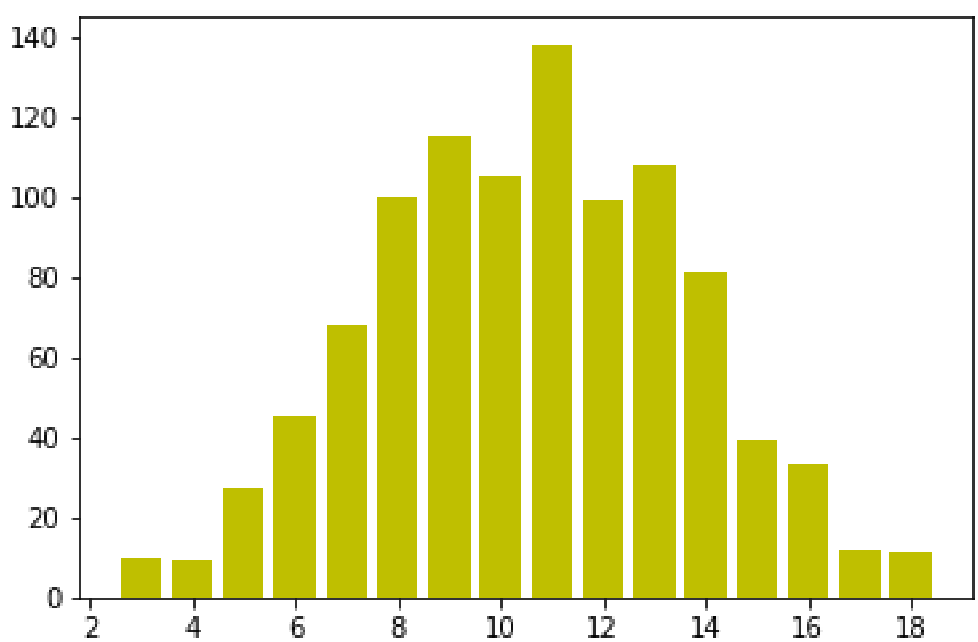
5.1 Une 1ère résolution 6

5.2 Une 2ème résolution 7

1 Exercice 1 : lancers de dés

On lance un dé à 6 faces 3 fois d'affilée et on note la somme des nombres obtenus. Ecrire un programme qui trace l'histogramme ds différentes sommes possibles pour un nombre N de séries de 3 lancers donné à l'avance.

```
In [3]: lancers=[randint(1,6)+randint(1,6)+randint(1,6) for k in range(1000)]

        pl.hist(lancers,range = (2.5, 18.5), bins = 16,facecolor='y',rwidth = 0.8)
        pl.show()
```



2 Exercice 2 : tirages dans une urne

Trois urnes U_1, U_2 et U_3 contiennent des boules rouges et noires. L'urne U_1 (resp. U_2, resp. U_3) contient 4 (resp.6, resp.3) boules rouges et 6 (resp.7, resp.9) boules noires.

On tire une boule dans chaque urne.

1. A partir de simulations, estimer la probabilité p_k d'obtenir k boules rouges avec $k \in \{0, 1, 2, 3\}$.
2. Représenter le résultat à l'aide d'un camembert.
3. Comparer avec les valeurs données par la théorie.

```
In [ ]: from numpy.random import choice

        U1 = ["Rouge","Noir"]
        proba1 = [0.4,0.6]
        U2 = ["Rouge","Noir"]
        proba2 = [6/13,7/13]
        U3 = ["Rouge","Noir"]
        proba3 = [0.25,0.75]

        NombreDeRouge=[]
        n=300
        for k in range (n):
            Resultat=[]
            Resultat.append(choice(U1, p=proba1))
            Resultat.append(choice(U2, p=proba2))
            Resultat.append(choice(U3, p=proba3))
            NombreDeRouge.append(Resultat.count("Rouge"))
        RES=[(NombreDeRouge.count(i)/float(3)) for i in range(4)]
        print(RES)

        name = ['0', '1', '2', '3']
        data = RES


        explode=(0, 0.15 , 0 , 0.2 )
        pl.pie(data, explode=explode, labels=name, autopct='%1.1f%%', startangle=-30)
        pl.axis('equal')
        title("nombre de Rouge pour %g tirages"%n)
        savefig("nb-de-Rouges-pour-%g-tirages.pdf"%n)


        pl.show()
```

```
In [6]: probas=[]
        probas.append((6*7*9)/(10*13*12))
        probas.append((4*7*9+6*6*9+3*6*7)/(10*13*12))
        probas.append((4*6*9+4*3*7+3*6*6)/(10*13*12))
        probas.append((4*6*3)/(10*13*12))
        print(probas)
        print(sum(probas))
```

```
[0.2423076923076923, 0.45, 0.26153846153846155, 0.046153846153846156]
1.0
```

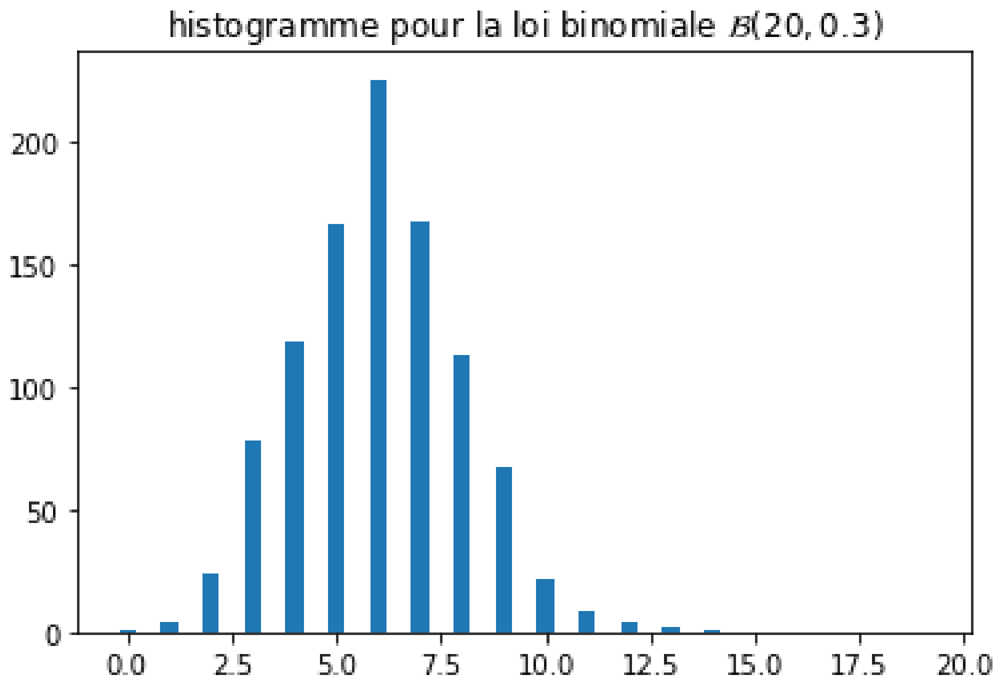
3 Exercice 3 : la loi binomiale

Définir une fonction `LoiBinomiale(n,p)` qui simule la loi binomiale $\mathcal{B}(n, p)$.

Représenter l'histogramme de cette simulation pour différentes valeurs de n et p (on rappelle que `random()` renvoie un nombre pris au hasard entre 0 et 1).

```
In [1]: from matplotlib.pyplot import *
        from random import *
        import numpy as np
        import pylab as pl

        def loi_binomiale(n,p):
            k=0
            for i in range(n):
                if random()<=p:
                    k=k+1
            return k

        n=20
        p=0.3

        N=10000
        B=[loi_binomiale(n,p) for k in range(1000)]

        bornes=np.arange(0,n+1)-0.5
        pl.hist(B,bornes, rwidth=0.4)
        pl.title("histogramme pour la loi binomiale $\mathcal{B}(%g,%g)$"%(n,p))

        savefig("histogramme_loi_binomiale B(%g,%g).pdf"%(n,p))

        pl.show()

        distribution=[B.count(i) for i in range(0,20)]
        print(distribution)
        total=sum([k for k in distribution])
        print(total)
```

```
frequences=[(B.count(i)/float(N)) for i in range(1,20)]
print(frequences)
```




```
[1, 4, 24, 78, 118, 166, 225, 167, 113, 67, 21, 9, 4, 2, 1, 0, 0, 0, 0, 0]
1000
[0.0004, 0.0024, 0.0078, 0.0118, 0.0166, 0.0225, 0.0167, 0.0113, 0.0067, 0.0021, 0.0009, 0.0004,
```

4 Exercice 4 : le jeu de Saint Petersbourg

Pierre et Paul joue au jeu suivant : Pierre lance une pi'ece jusqu''a ce qu'il obtienne un "Face" (et le jeu s'arrˆete au premier "Face" obtenu). Alors, si ce premier "Face" a 'et'e au n-'eme lancer, Pierre donne 2^n euros 'a Paul.

1. Simuler ce jeu répété un tr'es grand nombre de fois.
2. Combien Pierre donnera-t'il en moyenne à Paul selon cette simulation~?
3. Combien Pierre donnera-t'il en moyenne à Paul selon la théorie~?
4. Combien Paul doit-il donner à Pierre pour que Pierre accepte de jouer à ce jeu~?

```
In [8]: from matplotlib.pyplot import *
        from random import *
        import numpy as np
```

```python
import pylab as pl



def peter(n):
    res_peter=[]
    for i in range(n):
        k=1
        a=randint(0,1)
        while a<1:
            k=k+1
            a=randint(0,1)
        res_peter.append(2**k)
    return res_peter

print (peter(100),'\n')

def gain_peter(n):
    A=peter(n)
    #print(A)
    E=0
    for k in A:
        E += k
    return E/float(n)

for k in range(25):
    print (gain_peter(10000))
```

```
[2, 2, 8, 2, 4, 2, 4, 2, 8, 2, 16, 4, 8, 4, 4, 8, 16, 2, 64, 2, 4, 2, 2, 4, 16, 2, 2, 2, 2, 4, 2

15.2946
23.8936
16.2838
27.2168
12.2898
13.4282
15.6972
26.7548
11.7424
15.338
21.958
15.15
12.036
15.7688
35.7892
18.1084
27.8842
```

```
13.0546
14.1836
21.846
11.046
15.1378
11.5954
21.384
13.5938
```

5 Exercice 5 : le jeu de Monty Hall

Extrait de la page wikipedia (titre : "Problème de Monty Hall") :

Le problème de Monty Hall est un casse-tête probabiliste librement inspiré du jeu télévisé américain "Let's Make a Deal". Il est simple dans son énoncé, mais non intuitif dans sa résolution et c'est pourquoi on parle parfois à son sujet de "paradoxe de Monty Hall". Il porte le nom de celui qui a présenté ce jeu aux Etats-Unis pendant treize ans, Monty Hall. (...)

«Supposez que vous êtes sur le plateau d'un jeu télévisé, face à trois portes et que vous devez choisir d'en ouvrir une seule, en sachant que derrière l'une d'elles se trouve une voiture et derrière les deux autres des chèvres. Vous choisissez une porte, disons la numéro 1, et le présentateur, qui sait, lui, ce qu'il y a derrière chaque porte, ouvre une autre porte, disons la numéro 3, porte qui une fois ouverte découvre une chèvre. Il vous demande alors : "désirez-vous ouvrir la porte numéro 2 ?". A votre avis, est-ce à votre avantage de changer de choix et d'ouvrir la porte 2 plutôt que la porte 1 initialement choisie ?»

Simulez cette expérience pour vous aider à trouver la réponse.

5.1 Une 1ère résolution

```
In [9]: from random import *


        je_change_et_je_gagne= 0
        je_ne_change_pas_et_je_gagne = 0
        portes = ["voiture", "chèvre", "chèvre"]

        ## on choisy la congiguration avec la voiture derrière la porte 0
        ## cela ne réduit pas la généralité du résultat...

        for i in range(100000):     ### On fait 10000 parties du jeu


            porte_choisie = portes[randint(0,2)]   ### je choisis une porte au hasard

            # Le présentateur ouvre une des deux autres portes
            # derrière laquelle se trouve une chèvre.
```

```
        if porte_choisie == "voiture":
            autre_porte = 'chèvre'

            ### Si j'ai choisi la voiture, l'autre porte cache une chèvre

        else:
            autre_porte = 'voiture'

            ### sinon, la voiture est derrière l'autre porte

        # NOTA BENE : int(False) == 0 et int(True) == 1 en Python

        je_change_et_je_gagne += int(autre_porte == 'voiture')
        je_ne_change_pas_et_je_gagne += int(porte_choisie == 'voiture')


    print("Nombre de victoires quand on reste :",je_ne_change_pas_et_je_gagne)
    print("Nombre de victoires quand on change :",je_change_et_je_gagne)
```

```
Nombre de victoires quand on reste : 33217
Nombre de victoires quand on change : 66783
```

5.2 Une 2ème résolution

```
In [10]: from random import *



         def monty_hall(n):
             je_ne_change_pas_et_je_gagne = 0
             je_change_et_je_gagne=0
             for k in range(n):
                 porte_voiture=randint(0,2)
                         ## la voiture est derrière une porte choisie au hasard
                 premier_choix=randint(0,2)
                         ## la 1ère porte désignée est choisie au hasard
                 if premier_choix==porte_voiture:
                     je_ne_change_pas_et_je_gagne += 1
                 else:
                     je_change_et_je_gagne += 1
             SC=je_ne_change_pas_et_je_gagne
             EC=je_change_et_je_gagne
             print("Sur %g essais,"%n,'\n')
             print("il y a %g chances de gagner en changeant"%EC,"\n")
             print("contre %g chances de gagner sans changer"%SC)

In [11]: monty_hall(1000)
```

```
Sur 1000 essais,

il y a 658 chances de gagner en changeant

contre 342 chances de gagner sans changer
```

```
In [12]: monty_hall(10000)
```

```
Sur 10000 essais,

il y a 6654 chances de gagner en changeant

contre 3346 chances de gagner sans changer
```

```
In [13]: monty_hall(100000)
```

```
Sur 100000 essais,

il y a 66663 chances de gagner en changeant

contre 33337 chances de gagner sans changer
```