
TP6 Corrigé- Intervalle de confiance pour le paramètre d'une loi de Bernoulli et test de conformité à la moyenne

1 Simuler une variable aléatoire de loi de Bernoulli

1. La commande `random()` permet de tirer aléatoirement un nombre réel dans l'intervalle $]0, 1[$. Pour simuler une Bernoulli de paramètre p à partir de cette commande, il faut considérer que si `random()` donne un nombre dans $]0, p]$, on a un succès, sinon on a un échec.

2. On propose le code suivant :

```
1 import random as rd
2 def Bern(p):
3     if rd.random() < p:
4         return 1
5     else:
6         return 0
```

3. On pourrait estimer l'espérance de la variable aléatoire en calculant la moyenne des réalisations.

2 La moyenne empirique

Définition 2.0.1. Soit $n \in \mathbb{N}^*$ fixé et soient $(X_1, X_2, \dots, X_n)$ des variables aléatoires indépendantes qui suivent une même loi de Bernoulli de paramètre p .

La **moyenne empirique** associée est :

$$M_n = \frac{1}{n} \sum_{k=1}^n X_k.$$

4. Par linéarité de l'espérance, on a :

$$\begin{aligned} E(M_n) &= E\left(\frac{1}{n} \sum_{k=1}^n X_k\right) \\ &= \frac{1}{n} \sum_{k=1}^n E(X_k) \\ &= \frac{1}{n} \sum_{k=1}^n p \text{ car } X_k \text{ suit une loi de Bernoulli de paramètre } p \\ &= p \end{aligned}$$

Par propriété de la variance, on a :

$$\begin{aligned} V(M_n) &= \frac{1}{n^2} V\left(\sum_{k=1}^n X_k\right) \\ &= \frac{1}{n^2} \sum_{k=1}^n V(X_k) \text{ par indépendance des variables aléatoires} \\ &= \frac{p(1-p)}{n} \end{aligned}$$

5. On propose la fonction suivante :

```
1 def M(n):  
2     S=0  
3     for k in range(n):  
4         S+=Bern(1/3)  
5     return S/n
```

6. On remarque que pour les grandes valeurs de n , le résultat est proche du $\frac{1}{3}$ attendu.

```
>>> M(100)  
0.44  
  
>>> M(1000)  
0.311  
  
>>> M(10000)  
0.3282
```

3 Intervalle de confiance

On cherche maintenant à donner non plus simplement une approximation de p mais un intervalle dans lequel on est sûr à 95% que p se trouve. C'est ce qu'on appelle un **intervalle de confiance** du paramètre p . Pour ça, on a besoin d'avoir une valeur approchée de la variance de X .

Définition 3.0.1. Soient $n \in \mathbb{N}^*$ et $(X_1, X_2, \dots, X_n)$ des variables aléatoires mutuellement indépendantes qui suivent une même loi de Bernoulli de paramètre p . La **variance empirique** S_n^2 associée est la variable aléatoire :

$$S_n^2 = \frac{1}{n} \sum_{k=1}^n (X_k - M_n)^2 \quad \text{où} \quad M_n = \frac{1}{n} \sum_{k=1}^n X_k$$

7. Montrer que pour tout $n \in \mathbb{N}^*$, on a $S_n^2 = \frac{1}{n} \sum_{k=1}^n X_k^2 - (M_n)^2$. On part de la définition de S_n^2 :

$$\begin{aligned} S_n^2 &= \frac{1}{n} \sum_{k=1}^n (X_k - M_n)^2 \\ &= \frac{1}{n} \sum_{k=1}^n (X_k^2 - 2X_k M_n + M_n^2) \\ &= \frac{1}{n} \left(\sum_{k=1}^n X_k^2 - 2M_n \underbrace{\sum_{k=1}^n X_k}_{nM_n} + nM_n^2 \right) \\ &= \frac{1}{n} \left(\sum_{k=1}^n X_k^2 - nM_n^2 \right) \\ &= \frac{1}{n} \left(\sum_{k=1}^n X_k^2 \right) - M_n^2 \end{aligned}$$

8. On a par linéarité de l'espérance :

$$\begin{aligned}
 E(S_n^2) &= E\left(\frac{1}{n} \left(\sum_{k=1}^n X_k^2\right) - M_n^2\right) \\
 &= \frac{1}{n} \sum_{k=1}^n E(X_k^2) - E(M_n^2) \\
 &= \frac{1}{n} \sum_{k=1}^n (V(X_k) + E(X_k)^2) - (V(M_n) + E(M_n)^2) \text{ par formule de König-Huygens} \\
 &= \frac{1}{n} \sum_{k=1}^n p - \frac{p(1-p)}{n} - p^2 \\
 &= p(1-p) - \frac{p(1-p)}{n}
 \end{aligned}$$

On observe bien que $\lim_{n \rightarrow +\infty} E(S_n^2) = \text{Var}(X)$.

9. On propose la fonction suivante :

```

1 def S2(n):
2     L=[]
3     for k in range(n):
4         L.append(Bern(1/3))
5     m=sum(L)/n
6     Lcarre=[x**2 for x in L]
7     espxcaire=sum(Lcarre)/n
8     return espxcaire-m**2

```

Théorème 3.0.1. Soit $(X_n)_{n \geq 1}$ une suite de variables aléatoires mutuellement indépendantes qui suivent une même loi de Bernoulli de paramètre p . Pour tout $n \in \mathbb{N}^*$, on pose :

$$M_n = \frac{1}{n} \sum_{k=1}^n X_k \quad \text{et} \quad S_n = \sqrt{\frac{1}{n} \sum_{k=1}^n (X_k - M_n)^2}$$

On a :

$$\lim_{n \rightarrow +\infty} \mathbf{P}\left(M_n - 1.96 \frac{S_n}{\sqrt{n}} < p < M_n + 1.96 \frac{S_n}{\sqrt{n}}\right) = 0.95.$$

Ainsi, l'intervalle de confiance du paramètre p est :

$$\left] M_n - 1.96 \frac{S_n}{\sqrt{n}}; M_n + 1.96 \frac{S_n}{\sqrt{n}} \right[.$$

10. On modifie la fonction S2 pour renvoyer moyenne et variance empirique :

```

1 def MS2(n):
2     L=[]
3     for k in range(n):
4         L.append(Bern(1/3))
5     m=sum(L)/n
6     Lcarre=[x**2 for x in L]
7     espxcaire=sum(Lcarre)/n
8     return m, espxcaire-m**2

```

On propose la fonction suivante :

```

1 from math import sqrt
2 def Intervalle(n):
3     m,s2=MS2(n)
4     s=sqrt(s2)
5     a=m-1.96*s/sqrt(n)
6     b=m+1.96*s/sqrt(n)
7     return [a,b]

```

11. On obtient les tests suivants :

```

>>> Intervalle(100)
[0.3010626377724188, 0.47893736222758
12]

>>> Intervalle(100)
[0.24632922334046767, 0.4336707766595
3235]

>>> Intervalle(100)
[0.29781746803217, 0.48218253196783]

```

12. La largeur de l'intervalle dépend de $\sqrt{n}$ donc il faut multiplier n par 4 pour augmenter la largeur de l'intervalle par 2, d'où le choix de $n = 400$.
13. On veut que la largeur de l'intervalle soit de 10^{-3} d'où :

$$2 \times 1.96 \times \frac{s}{\sqrt{n}} = 10^{-3} \iff n \approx (2 \times 1.96 \times s \times 10^3)^2$$

$$\iff n \approx 3,4 \times 10^6$$

4 Test de conformité à la moyenne

Soient X une variable aléatoire d'espérance μ (inconnue) et $\mu_0 \in \mathbb{R}$. On voudrait savoir si l'espérance de X (*ie.* μ) est égale à μ_0 ou non. Pour cela, on fait ce qu'on appelle un **test de conformité sur la moyenne**. On émet pour cela l'hypothèse suivante :

$$(H_0) : \mu = \mu_0$$

et on travaille sous cette hypothèse. On se demande comment faire pour accepter ou rejeter cette hypothèse avec un certain degré de certitude.

Le protocole du test est le suivant :

1. On émet l'hypothèse $(H_0) : \mu = \mu_0$
2. On pose $\alpha = 0,05$: on prend un risque de 5% de rejeter l'hypothèse alors qu'elle était vraie.
3. On calcule la valeur de $\frac{M_n - \mu_0}{\frac{S_n}{\sqrt{n}}}$ pour l'échantillon choisi.
4. Si la valeur obtenue n'appartient pas à $[-1.96, 1.96]$, on rejette l'hypothèse.
Sinon on l'accepte.

Il se base sur le théorème suivant :

Proposition 4.0.1. Soit $(X_n)_{n \geq 1}$ une suite de variables aléatoires indépendantes et de même loi, admettant une espérance μ et une variance σ^2 non nulle. Pour tout $n \in \mathbb{N}^*$, on pose :

$$M_n = \frac{1}{n} \sum_{k=1}^n X_k \quad \text{et} \quad S_n = \sqrt{\frac{1}{n} \sum_{k=1}^n (X_k - M_n)^2}$$

On a alors :

$$\lim_{n \rightarrow +\infty} P \left(\left| \frac{M_n - \mu}{\frac{S_n}{\sqrt{n}}} \right| > 1.96 \right) = 0.05.$$

Dans un centre avicole, des études antérieures ont montré que la masse d'un œuf choisi au hasard peut être considérée comme la réalisation d'une variable aléatoire gaussienne X de moyenne m et de variance σ^2 . On admet que les masses des œufs sont indépendantes les unes des autres. On prend un échantillon de $n = 36$ œufs que l'on pèse. Les mesures sont données dans le tableau suivant :

50,34	52,62	53,79	54,99	55,82	57,67
51,41	53,13	53,89	55,04	55,91	57,99
51,51	53,28	54,63	55,12	55,95	58,10
52,07	53,30	54,76	55,24	57,05	59,30
52,22	53,32	54,78	55,28	57,18	60,58
52,38	53,39	54,93	55,56	57,31	63,15

Vous trouverez la liste correspondante dans le fichier Liste.py sur CDP.

14. On les calcule via le script suivant :

```

1 L=[50.34 , 52.62 , 53.79 ,54.99 , 55.82 ,57.67,51.41 , 53.13 ,53.89 ,55.04 , 55.91 , 57.99
2
3 M=0
4 n=len(L)
5 for k in range(n):
6     M+=L[k]
7 M=M/n
8
9 S=0
10 for k in range(n):
11     S+=(L[k]-M)**2
12 S=1/sqrt(n)*sqrt(S)
13

```

15. On crée le script suivant :

```

1 m0=55.95
2 total=sqrt(n)*(M-m0)/S
3 u=1.96
4
5 if abs(total)<u:
6     print('Hypothèse acceptée')
7 else:
8     print('Hypothèse rejetée')

```

L'hypothèse est ici rejetée.

5 Test de Student de la moyenne

Soient X une variable aléatoire suivant une loi normale d'espérance μ (inconnue) et $\mu_0 \in \mathbb{R}$. On voudrait savoir si l'espérance de X (*ie.* μ) est égale à μ_0 ou non. Pour cela, on fait ce qu'on appelle un **test de conformité Student**. On émet l'hypothèse suivante :

$$(H_0) : \mu = \mu_0$$

et on travaille sous cette hypothèse. On se demande comment faire pour accepter ou rejeter cette hypothèse avec un certain degré de certitude.

Le protocole du test est le suivant :

1. On émet l'hypothèse $(H_0) : \mu = \mu_0$
2. On se donne un réel $\alpha \in]0, 1[$ assez petit (en pratique, on a souvent $\alpha = 0,05$).
3. On calcule la valeur de $\frac{M_n - \mu_0}{\frac{S_n^*}{\sqrt{n}}}$ pour l'échantillon choisi avec

$$M_n = \frac{1}{n} \sum_{k=1}^n X_k \quad \text{et} \quad S_n^* = \sqrt{\frac{1}{n-1} \sum_{k=1}^n (X_k - M_n)^2}$$

4. Si la valeur obtenue n'appartient pas à $[-u, u]$, on rejette l'hypothèse.
Sinon on l'accepte. (u est calculé à partir de α et de n)

On reprend l'exemple précédent sur le poids des œufs.

16. On calcule S_n^* via le script suivant :

```
1 from math import *
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 L=[50.34 , 52.62 , 53.79 ,54.99 , 55.82 ,57.67,51.41 , 53.13 ,53.89 ,55.04 , 55.91 , 57.
6
7 M=0
8 n=len(L)
9 for k in range(n):
10     M+=L[k]
11 M=M/n
12
13 S=0
14 for k in range(n):
15     S+=(L[k]-M)**2
16 S=1/sqrt(n-1)*sqrt(S)
```

17. On teste via le script suivant :

```
1 from scipy import stats
2
3 m0=55.95
4 total=sqrt(n)*(M-m0)/S
5
6 t=2.03
7
8 if abs(total)<t:
```

```

9     print('Hypothèse acceptée')
10 else:
11     print('Hypothèse rejetée')

```

L'hypothèse est ici acceptée.

18. On se rend compte que la même hypothèse, liée à la même série statistique, peut être acceptée ou rejetée selon le test choisi. Le choix du test est donc essentiel.

6 Pour aller plus loin

On veut comprendre d'où vient de lien entre le 1.96 annoncé dans la partie précédente et la certitude à 95%.

19. Montrer que $P\left(M_n - 1.96 \frac{S_n}{\sqrt{n}} < \mu < M_n + 1.96 \frac{S_n}{\sqrt{n}}\right) = P(-1.96 \leq M_n^* \leq 1.96)$ où $M_n^* = \sqrt{n} \frac{M_n - \frac{1}{3}}{S_n}$.

On cherche à déterminer la valeur de u pour laquelle

$$P(-u \leq M_n^* \leq u) = 0.95.$$

Notre objectif est de montrer que $u \approx 1.96$.

20. On propose la fonction suivante :

```

1 def Mstar(n):
2     m=M(n)
3     s2=S2(n)
4     return (m-1/3)/sqrt(s2/n)

```

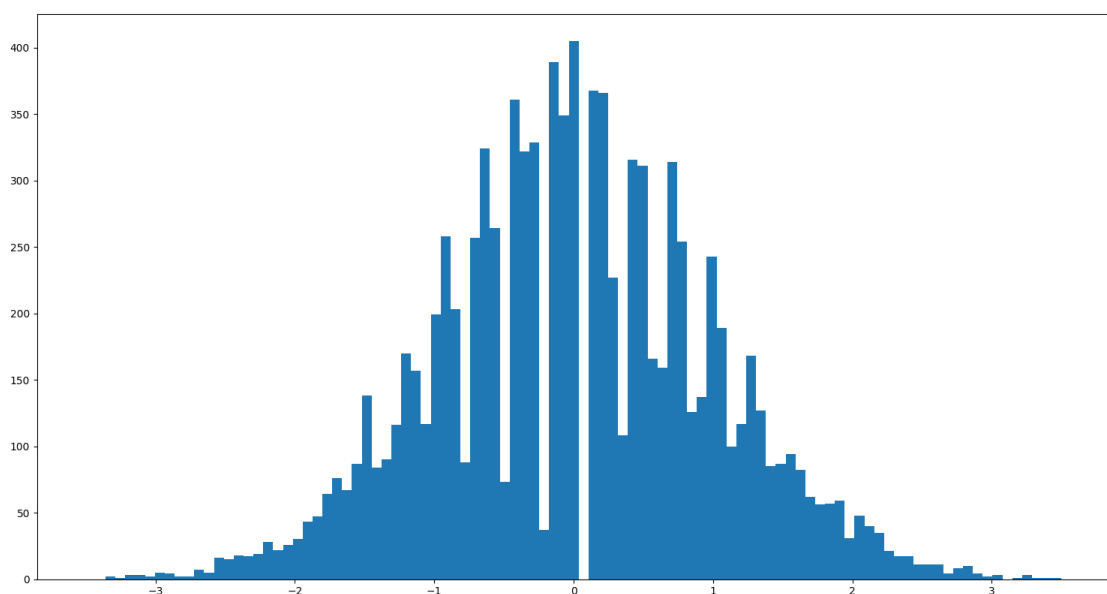
21. On propose la fonction suivante :

```

1 def Liste(n):
2     L=[]
3     for k in range(10000):
4         L.append(Mstar(n))
5     return L

```

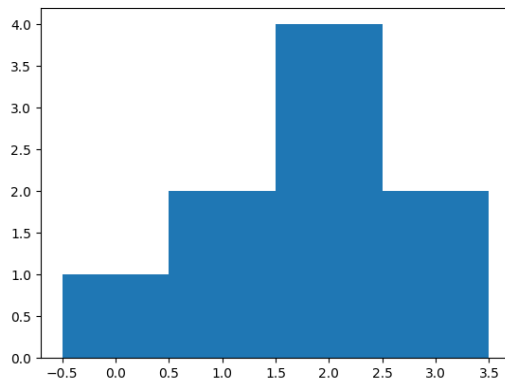
22. On obtient l'histogramme suivant :



On constate que la variable aléatoire M_n^* est symétrique.

23. On obtient ce résultat par symétrie de M_n^* .

Dans la suite, nous aurons besoin des informations contenues dans **res**. Le vecteur **res** contient dans sa première composante les hauteurs des rectangles, autrement dit les effectifs, et dans sa deuxième composante les coordonnées des bases des rectangles. Pour l'histogramme suivant, on aurait :



```
>>> res[0]
array([1., 2., 4., 2.])

>>> res[1]
array([-0.5,  0.5,  1.5,  2.5,  3.5])
```

24. On complète comme suit :

```
1     S=0
2     k=49
3     while S<5000*0.95:
4         S+=res[0][k] #On part de 0 et on additionne les effectifs cumulés jusqu'à atte
5         k+=1
6     print(res[1][k])
```

25. On crée le programme suivant :

```
1     T=[]
2     for k in range(20):
3         n=500
4         L=Liste(n)
5         b=np.linspace(-3.5,3.5,100)
6         res=plt.hist(L,b,density=False)
7         S=0
8         k=49
9         while S<5000*0.95:
10            S+=res[0][k]
11            k+=1
12        T.append(res[1][k])
13    print(sum(T)/20)
```

On obtient des valeurs proches de 1.96 :

```
>>> (executing file "Bern.py")
1.9691919191919183

>>> (executing file "Bern.py")
1.9196969696969695
```