12 - Variables+aleatoires-corrige

April 20, 2020

1 1. Moyenne et écart type d'une liste de nombres

On va écrire dans un premier temps les fonctions *moyenne* et *ecarttype* que nous utiliserons ensuite.

- La moyenne d'une série statistique $(x_i)_{1\leqslant i\leqslant n}$ s'obtient par la formule $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$.
- Sa variance est $V(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$ (moyenne des carrés des écarts des valeurs à la moyenne).
- Son écart type est alors $\sigma(x) = \sqrt{V(x)}$.

Q1. Rappeler les formules de calculs de moyenne et de variance dans le cas d'une série donnée avec effectifs $(x_i; n_i)$ puis avec fréquences $(x_i; f_i)$.

- La moyenne d'une série statistique donnée avec effectifs $(x_i; n_i)_{1\leqslant i\leqslant k}$ s'obtient par la formule $\bar{x} = \frac{1}{n}\sum_{i=1}^{k} n_i x_i$ où n est l'effectif total : $n = \sum_{i=1}^{k} n_i$.
- Sa variance est $V(x) = \frac{1}{n}\sum_{i=1}^{k} n_i(x_i - \bar{x})^2$.
- La moyenne d'une série statistique donnée avec fréquences $(x_i; f_i)_{1\leqslant i\leqslant k}$ s'obtient par la formule $\bar{x} = \sum_{i=1}^{k} f_i x_i$.
- Sa variance est $V(x) = \sum_{i=1}^{k} f_i(x_i - \bar{x})^2$.

Q2. Écrire une fonction $moyenne(L)$ renvoyant la moyenne d'une liste de nombres L.

```
In [106]: def moyenne(L):
              s = 0
              for x in L:
                  s = s + x
              return s/len(L)
```

La fonction $ecarttype(L)$ renvoie l'écart type d'une liste de nombres L.

```
In [107]: from math import sqrt

          def ecarttype(L):
              v = 0
              moy = moyenne(L)
              for x in L:
                  v += (x-moy)**2
              return sqrt(v/len(L))
```

Q3. Tester dans la cellule ci-dessous les deux fonctions précédentes avec la série 15, 12, 10, 16.

```
In [145]: moyenne([15, 12, 10, 16]), ecarttype([15, 12, 10, 16])
```

```
Out[145]: (13.25, 2.384848003542364)
```

2 2. Simulations d'expériences aléatoires

Plusieurs fonctions de la bibliothèque `random` permettent de simuler l'aléatoire en Python. Si n et p sont des entiers :

- $randrange(n)$ renvoie un entier entre 0 et $n-1$.
- $randrange(n,p)$ renvoie un entier entre n inclus et p exclu.
- $randint(n,p)$ renvoie un entier entre n et p inclus.
- $random()$ renvoie un flottant de l'intervalle $[0;1[$

2.1 2.1. Lancers de dés

$randint(1,6)$ permet de simuler le lancer d'un dé ordinaire à 6 faces.

Q4. Quel est le rôle de la fonction suivante ?

```
In [109]: from random import randint

          def lancerdes(n):
              res = []
              for i in range(n):
                  res.append(randint(1,6))
              return res
```

Cette fonction simule un lancer de n dés (ou d'un dé n fois) et renvoie les n valeurs dans une liste.

Q5. Dans la cellule ci-dessous, on simule une série de 10 000 lancers de dés. Déterminer sa moyenne et son écart type.

```
In [110]: serie = lancerdes(10000)
          moy = moyenne(serie)
          et = ecarttype(serie)
          print("moyenne :", moy, "ecart-type: ",et)
```

```
moyenne : 3.4949 ecart-type:  1.701756148806283
```

Q6. Décrire le fonctionnement du code suivant.

```
In [111]: def frequence(serie):
              freq = [0]*6
              for x in serie:
                  freq[x-1] += 1    # même effet que freq[x-1] = freq[x-1] + 1
              for i in range(len(freq)):
                  freq[i] = freq[i]/len(serie)
              return freq
```

La fonction *frequence* renvoie une liste contenant la fréquence d'apparition de chaque résultat, dans l'ordre croissant. Par exemple, [0.2, 0.25, 0.1, 0.15, 0.3, 0] signifie qu'il y a 20% de 1, 25% de 2, 10% de 3, 15% de 4, 30% de 5 et 0% de 6.

Pour cela, on crée une liste freq contenant six 0 : [0, 0, 0, 0, 0, 0]. Dans un premier temps, cette liste va indiquer le nombre d'apparitions de chaque valeur. La deuxième boucle for divise ces effectifs par le nombre de valeurs (len(serie)) pour calculer les fréquences correspondantes.

On rappelle qu'on accède aux éléments d'une liste avec le nom de la liste suivi de l'indice de l'élément entre crochet. Ainsi freq[0] correspond au premier élément de la liste, donc au nombre de 1; freq[1] au deuxième élément de la liste, donc au nombre de 2, ...

Dans la première boucle for, la variable x parcourt la série. x est un nombre entre 1 et 6 et les indices de la liste *freq* vont de 0 à 5.

Ainsi, si x prend la valeur 3 par exemple, freq[2] qui correspond au nombre de 3, doit augmenter de 1. Si x prend la valeur 5 par exemple, freq[4] qui correspond au nombre de 5 doit augmenter de 1. D'où la ligne freq[x-1] += 1.

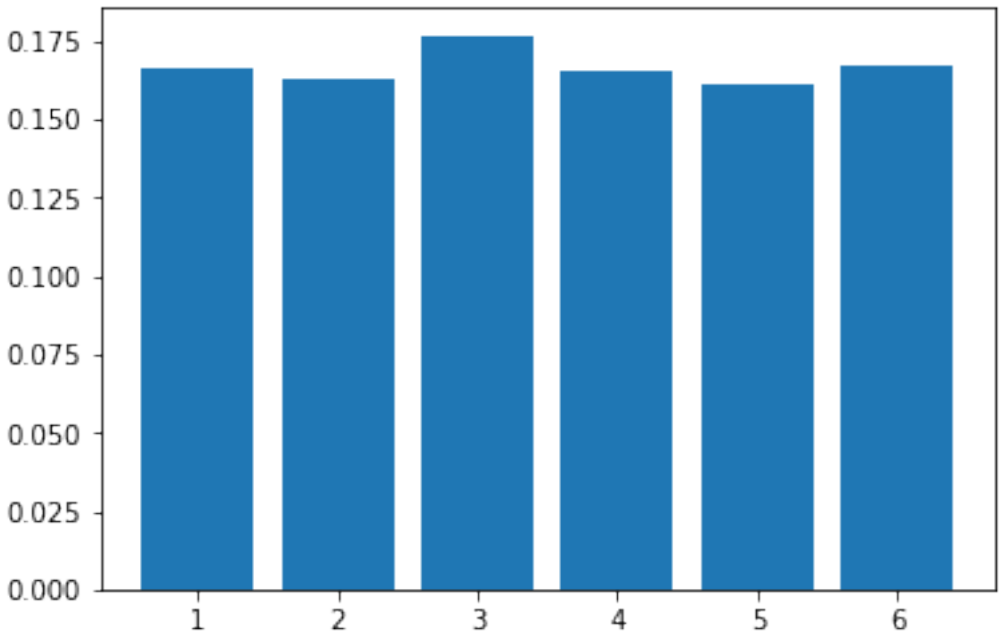
```
In [112]: def frequence(serie):
              freq = [0]*6                      # On crée une liste freq contenant six 0.
              for x in serie:                   # On parcourt la serie de nombres en comptant
                  freq[x-1] += 1                #   le nombre d'apparitions de chaque valeur.
              for i in range(len(freq)):        # On parcourt la liste freq en divisant les ef
                  freq[i] = freq[i]/len(serie)  #   par l'effectif total pour calculer les fré
              return freq
```

On peut alors représenter cette série sous la forme d'un diagramme en barres.

```
In [113]: import matplotlib.pyplot as plt

          def diagramme(freq):
              u = plt.bar(range(1,7),freq)
              plt.show()

          diagramme(frequence(serie))
```



2.2 2.2. Lancers d'une pièce de monnaie

$random() < 0.5$ permet de simuler le lancer d'une pièce de monnaie.

Q7. Que renvoie l'expression $random() < 0.5$? Expliquer comment l'utiliser pour simuler le lancer d'une pièce.

random() < 0.5 est un test. Il renvoie donc un booléen True ou False avec équiprobabilité. On peut considérer que True correspond à pile et False à face et simuler ainsi une pièce de monnaie. On aurait pu créer une fonction piece qui lance une pièce et renvoie True pour pile et False pour face. Je n'ai pas utilisé cette fonction dans la suite.

```
In [3]: from random import random

        def piece():
            return random()<0.5
```

Q8. Écrire une fonction effectuant une simulation de n lancers de pièces de monnaie et renvoyant la fréquence d'apparition de pile (ou face). On pourra la tester avec $n = 10\,000$ par exemple.

```
In [4]: def simulationLancers(n):
            nbr_pile = 0
            for i in range(n):
                if random() < 0.5:    # Condition vérifiée une fois sur deux environ (probabili
                    nbr_pile += 1     # Rajoute un pile
            return nbr_pile/n

        simulationLancers(10000)

Out[4]: 0.4994
```

2.3 2.3. Évènement de probabilité p

$random() < p$ permet de simuler un évènement de probabilité p.

2.3.1 Tir à l'arc V1

Loic tire 10 flèches sur une cible. A chaque tir, il a une probabilité de 0,7 de toucher la cible.

Q9. Écrire une fonction permettant de simuler cette expérience, renvoyant le nombre de fois où Loic a touché la cible lors de ses 10 tirs.

```
In [6]: from random import random

        def simulation10TirV1():
            nbr_touche = 0
            for i in range(10):
                if random() < 0.7:
                    nbr_touche += 1
            return nbr_touche
```

2.3.2 Tir à l'arc V2

Lors du premier tir, Loic a une probabilité de 0,7 de toucher la cible. Ensuite, si Loic touche la cible, il a une probabilité de 0,8 de retoucher la cible au tir suivant, et s'il manque, il a une probabilité de 0,5 de toucher la cible au tir suivant.

Q10. Écrire une fonction permettant de simuler cette expérience, renvoyant le nombre de fois où Loic a touché la cible en 10 tirs.

```
In [7]: def simulation10TirV2():
            # Premier tir
            if random() < 0.7:
                nbr_touche = 1
                touche = True
            else:
                nbr_touche = 0
                touche = False
            # 9 tirs suivants
            for i in range(9):
                if touche:
                    proba = 0.8
                else:
                    proba = 0.5
                if random() < proba:
                    nbr_touche += 1
                    touche = True
                else:
                    touche = False
            return nbr_touche
```

Q11. Simuler n séries de 10 tirs, en retenant dans chaque série le nombre de fois où Loic touche la cible. En prenant n = 1 000, afficher la moyenne et l'écart type de cette série dans les deux situations précédentes (V1 et V2).

```
In [118]: def simulationTirsV1(n):
              res_series = []
              for i in range(n):
                  nbr_touche = simulation10TirV1()
                  res_series.append(nbr_touche)
              return moyenne(res_series), ecarttype(res_series)

          def simulationTirsV2(n):
              res_series = []
              for i in range(n):
                  nbr_touche = simulation10TirV2()
                  res_series.append(nbr_touche)
              return moyenne(res_series), ecarttype(res_series)

          print(simulationTirsV1(1000))
          print(simulationTirsV2(1000))
```

```
(7.004, 1.4394387795248524)
(7.012, 1.904693151140105)
```

2.4 2.4. Déplacement aléatoire d'un robot

Un robot se déplace dans une direction aléatoire définie par un angle entre 0ř et 360ř d'une distance aléatoire entre 20 et 100 centimètres.

Le script suivant permet de simuler un tel déplacement.

```
In [4]: from random import random

        def deplacement():
            angle = random()*360
            distance = random()*80 + 20
            return angle,distance
```

On souhaite effectuer une simulation de n déplacements consécutifs. Le robot démarre au point de coordonnées (0; 0).

Q12. Compléter le script suivant, qui affiche la trajectoire du robot, afin qu'il renvoie les coordonnées du robot après les déplacements.

```
In [5]: # Création des fonctions cosinus et sinus en degrés
        import numpy as np
        def cos(x):
            return np.cos(np.pi*x/180)

        def sin(x):
```

```
        return np.sin(np.pi*x/180)

    # Fonctions permettant d'afficher les déplacements du robot
    import turtle

    def simulationDeplacements(n):
        x,y = 0,0
        for i in range(n):
            angle,distance = deplacement()
            turtle.setheading(angle)
            turtle.forward(distance)
            x = x + distance * np.cos(angle)
            y = y + distance * np.sin(angle)
        turtle.exitonclick()
        return x,y

    # Les lignes suivantes exécutent la fonction simulationDeplacements.
    # Les instructions try et except ne présentent pas d'intérêt pour nous mais permettent

    try:
        print(simulationDeplacements(10))
    except:
        print(simulationDeplacements(10))
```

```
(248.17643787133858, 67.648701146900805)
```

3 3. Variables aléatoires réelles

Définition

On considère une expérience dont l'univers est un ensemble fini Ω. Une variable aléatoire X est une fonction définie sur Ω et à valeurs dans $\mathbb{R}$.

Exemples - La variable X qui, à tout lancer d'un dé à six faces, associe le nombre obtenu sur la face supérieure est une variable aléatoire à valeurs dans $\{1; 2; 3; 4; 5; 6\}$. - La variable Y qui, à toute série de 10 tirs de Loic, associe le nombre de fois où il touche la pile est une variable aléatoire à valeurs dans $\{0; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10\}$. - La variable qui, à tout déplacement du robot, associe la distance parcourue en centimètres est une variable aléatoire à valeurs dans [20 ; 100]. Cet exemple dépasse le cadre du programme de première car l'ensemble des déplacements est infini.

Définition

La loi de probabilité d'une variable aléatoire X est la donnée de la probabilité $P(X = x_i)$ de l'évènement $\{X = x_i\}$ pour chaque valeur x_i prise par X.

Exemples - La loi de probabilité de la variable X est donnée par : $\forall n \in \{1; 2; 3; 4; 5; 6\}$, $P(X = n) = \frac{1}{6}$ - La loi de probabilité de la variable Y est plus complexe à déterminer mais peut être estimée statistiquement. On rappelle que la fréquence de réalisation d'un évènement lors de n expériences se rapproche de la probabilité de réalisation de cet évènement quand n est très grand.

Q13. Ecrire la fonction *simulationFreq*(n) renvoyant la fréquence d'apparition de chaque résultat, les résultats étant $\{Y = 0\}$, $\{Y = 1\}$, ..., $\{Y = 10\}$. On utilisera, sans chercher à comprendre

son fonctionnement, la fonction afficheResultat qui permet d'afficher les fréquences sous forme de tableau.

```
In [8]: from IPython.display import HTML, display

        def simulationFreq(n):
            """appelle n fois simulation10TirV2 et renvoie une liste freq contenant la fréquen
            d'obtiention de chaque résultat"""
            freq = [0]*11
            for i in range(n):
                nbr_touche = simulation10TirV2()
                freq[nbr_touche] += 1
            for i in range(11):
                freq[i] = freq[i]/n
            return freq

        def afficheResultat(freq):
            data = [["n",0,1,2,3,4,5,6,7,8,9,10],
                ["P(Y=n)"]+freq]
            display(HTML(
               '<table><tr>{}</tr></table>'.format(
                   '</tr><tr>'.join(
                       '<td>{}</td>'.format('</td><td>'.join(str(_) for _ in row)) for row in d
                   )
            ))

        freq = simulationFreq(10000)
        afficheResultat(freq)
```

<IPython.core.display.HTML object>

Définition

On étend les notions de moyenne, de variance et d'écart type des séries statistiques aux variables aléatoires.

X étant une variable aléatoire prenant les valeurs x_i avec des probabilités p_i, on définit :

- L'espérance de X : $E(X) = \sum_{i=1}^{n} p_i x_i$
- La variance de X : $V(X) = \sum_{i=1}^{n} p_i (x_i - \bar{x})^2$ (moyenne des carrés des écarts des valeurs à la moyenne)
- L'écart type de X : $\sigma(X) = \sqrt{V(X)}$

Exemples

$E(X) = \sum_{i=1}^{6} p_i x_i = \frac{1}{6} \times 1 + \frac{1}{6} \times 2 + \frac{1}{6} \times 3 + \frac{1}{6} \times 4 + \frac{1}{6} \times 5 + \frac{1}{6} \times 6 = \frac{21}{6} = 3,5$

$V(X) = \sum_{i=1}^{6} p_i (x_i - \bar{x})^2 = \frac{1}{6} \times (1-3,5)^2 + \frac{1}{6} \times (2-3,5)^2 + \frac{1}{6} \times (3-3,5)^2 + \frac{1}{6} \times (4-3,5)^2 + \frac{1}{6} \times (5-3,5)^2 + \frac{1}{6} \times (6-3,5)^2 = \frac{35}{12}$

$\sigma(X) = \sqrt{V(X)} = \sqrt{\frac{35}{12}} \approx 1,7$

4 Exercices

4.1 Exercice 1

La fonction *frequenceLettres* ci-dessous calcule pour l'instant le nombre d'apparitions de chaque lettre dans un texte. Compléter cette fonction pour qu'elle renvoie dans la liste *freq* la fréquence d'apparition de chaque lettre.

Les plus motivés pourront rajouter quelques lignes pour gérer les lettres accentuées (cette partie est plus proche de la NSI que des mathématiques).

```
In [122]: def frequenceLettres(texte):
              texmin = texte.casefold()  # convertit le texte en lettres minuscules
              freq = [0]*26
              for lettre in texmin:
                  num = ord(lettre) - 97  # renvoie un nombre associé à chaque caractère entre
                  if 0 <= num <= 25:
                      freq[num] += 1
              # compléter ici la fonction par une ou plusieurs lignes
              n = len(texte)
              for i in range(len(freq)):
                  freq[i] = freq[i]/n
              return freq
```

Tester la fonction précédente sur le petit chaperon rouge en français.

```
In [123]: text = """
          Il était une fois une petite fille de village, la plus éveillée quon eût su voir : sa
          Un jour, sa mère ayant cuit et fait des galettes, lui dit : ń Va voir comment se port
          Le petit Chaperon rouge partit aussitôt pour aller chez sa mère-grand, qui demeurait
          Le Loup se mit à courir de toute sa force par le chemin qui était le plus court, et l
          Le Loup ne fut pas longtemps à arriver à la maison de la mère-grand ; il heurte : toc
          Ensuite il ferma la porte, et salla coucher dans le lit de la mère-grand, en attendan
          Le Loup, la voyant entrer, lui dit en se cachant dans le lit, sous la couverture : Me
          """
          t2=text.casefold()  # convertit les majuscules en minuscules
          lstfreq = frequenceLettres(t2)
          for i in range(len(lstfreq)):
              print("Fréquence de",chr(i+97)," : ",lstfreq[i])
```

```
Fréquence de a  :  0.060323553605703316
Fréquence de b  :  0.007403345215245407
Fréquence de c  :  0.02111324376199616
Fréquence de d  :  0.021935837674801208
Fréquence de e  :  0.11516314779270634
Fréquence de f  :  0.007129147244310392
Fréquence de g  :  0.012887304633945709
Fréquence de h  :  0.009871126953660542
Fréquence de i  :  0.05017822868110776
Fréquence de j  :  0.0021935837674801205
```

```
Fréquence de k  :  0.0
Fréquence de l  :  0.05182341650671785
Fréquence de m  :  0.021935837674801208
Fréquence de n  :  0.042774883465862355
Fréquence de o  :  0.04332327940773238
Fréquence de p  :  0.02604880723882643
Fréquence de q  :  0.009871126953660542
Fréquence de r  :  0.05209761447765286
Fréquence de s  :  0.03729092404716205
Fréquence de t  :  0.06580751302440362
Fréquence de u  :  0.048258842884562655
Fréquence de v  :  0.016451878256100904
Fréquence de w  :  0.0
Fréquence de x  :  0.002741979709350151
Fréquence de y  :  0.0024677817384151355
Fréquence de z  :  0.0019193857965451055
```

Tester sur une traduction anglaise.

```
In [124]: text = """
          Once upon a time there lived in a certain village a little country girl, the pretties
          One day her mother, having made some cakes, said to her, "Go, my dear, and see how yo
          Little Red Riding Hood set out immediately to go to her grandmother, who lived in ano
          As she was going through the wood, she met with a wolf, who had a very great mind to
          "Does she live far off?" said the wolf
          "Oh I say," answered Little Red Riding Hood; "it is beyond that mill you see there, a
          "Well," said the wolf, "and I'll go and see her too. I'll go this way and go you that
          The wolf ran as fast as he could, taking the shortest path, and the little girl took
          "Who's there?"
          "Your grandchild, Little Red Riding Hood," replied the wolf, counterfeiting her voice
          The good grandmother, who was in bed, because she was somewhat ill, cried out, "Pull
          The wolf pulled the bobbin, and the door opened, and then he immediately fell upon th
          "Who's there?"
          Little Red Riding Hood, hearing the big voice of the wolf, was at first afraid; but h
          The wolf cried out to her, softening his voice as much as he could, "Pull the bobbin
          Little Red Riding Hood pulled the bobbin, and the door opened.
          The wolf, seeing her come in, said to her, hiding himself under the bedclothes, "Put
          Little Red Riding Hood took off her clothes and got into bed. She was greatly amazed
          "All the better to hug you with, my dear." "Grandmother, what big legs you have!" "Al
          "Grandmother, what big ears you have!" "All the better to hear with, my child." "Gran
          "All the better to see with, my child." "Grandmother, what big teeth you have got!"
          "All the better to eat you up with."
          And, saying these words, this wicked wolf fell upon Little Red Riding Hood, and ate h
          t2=text.casefold()
          lstfreq = frequenceLettres(t2)
          for i in range(len(lstfreq)):
              print("Fréquence de",chr(i+97)," : ",lstfreq[i])
```

```
Fréquence de a  :  0.05495118549511855
Fréquence de b  :  0.014504881450488146
Fréquence de c  :  0.012552301255230125
Fréquence de d  :  0.044630404463040445
Fréquence de e  :  0.09679218967921897
Fréquence de f  :  0.013389121338912133
Fréquence de g  :  0.02398884239888424
Fréquence de h  :  0.05801952580195258
Fréquence de i  :  0.04686192468619247
Fréquence de j  :  0.0
Fréquence de k  :  0.005578800557880056
Fréquence de l  :  0.039888423988842396
Fréquence de m  :  0.016457461645746164
Fréquence de n  :  0.03933054393305439
Fréquence de o  :  0.06666666666666667
Fréquence de p  :  0.008647140864714086
Fréquence de q  :  0.0002789400278940028
Fréquence de r  :  0.049372384937238493
Fréquence de s  :  0.03096234309623431
Fréquence de t  :  0.07754532775453278
Fréquence de u  :  0.01896792189679219
Fréquence de v  :  0.006136680613668061
Fréquence de w  :  0.0198047419804742
Fréquence de x  :  0.0008368200836820083
Fréquence de y  :  0.012831241283124128
Fréquence de z  :  0.0002789400278940028
```

Expliquer comment un programme peut déterminer si un texte est en français ou en anglais.

Les lettres n'apparaissent pas avec la même fréquence en français et en anglais. En calculant la fréquence des lettres dans un texte, un programme peut ainsi déterminer la langue utilisée. Par exemple, les lettres w et y apparaissent beaucoup plus en anglais. Ainsi, si les fréquences de y et de w dans le texte sont supérieures à 0.01 ou 0.007 par exemple, on peut penser que le texte est en anglais.

4.2 Exercice 2

On considère la variable aléatoire X, qui à tout lancer de deux dés, associe $|a - b|$ où a et b sont les nombres obtenus sur chaque dé.
On rappelle que $|a - b|$, que l'on lit valeur absolue de $a - b$ est la distance entre les nombres a et b.
1. Déterminer la loi de probabilité de la variable X.
2. Calculer son espérance et son écart type σ au centième.
3. Écrire la fonction *unLancer* qui simule un lancer de deux dés et renvoie la valeur prise par X.

1. La technique la plus simple est peut-être d'utiliser un tableau pour obtenir tous les tirages possibles.

	1	2	3	4	5	6
1	0	1	2	3	4	5

	1	2	3	4	5	6
2	1	0	1	2	3	4
3	2	1	0	1	2	3
4	3	2	1	0	1	2
5	4	3	2	1	0	1
6	5	4	3	2	1	0

On peut ensuite donner la loi du probabilité (chaque résultat possible avec sa probabilité). Par exemple, 0 apparait 6 fois sur 36 donc $P(X = 0) = \frac{6}{36}$.

x_i	0	1	2	3	4	5
$P(X=x_i)$	$\frac{6}{36}$	$\frac{10}{36}$	$\frac{8}{36}$	$\frac{6}{36}$	$\frac{4}{36}$	$\frac{2}{36}$

En simplifiant les fractions, on obtient :

x_i	0	1	2	3	4	5
$P(X=x_i)$	$\frac{1}{6}$	$\frac{5}{18}$	$\frac{2}{9}$	$\frac{1}{6}$	$\frac{1}{9}$	$\frac{1}{18}$

2. L'espérance de X est $E(X) = \sum_{i=1}^{6} p_i x_i = \frac{1}{6} \times 0 + \frac{5}{18} \times 1 + \frac{2}{9} \times 2 + \frac{1}{6} \times 3 + \frac{1}{9} \times 4 + \frac{1}{18} \times 5 = \frac{5}{18} + \frac{8}{18} + \frac{9}{18} + \frac{8}{18} + \frac{5}{18} = \frac{35}{18}$

Pour calculer l'écart type de X, on commence par calculer sa variance : $V(X) = \sum_{i=1}^{6} p_i(x_i - \bar{x})^2 = \frac{1}{6}(0 - \frac{35}{18})^2 + \frac{5}{18}(1 - \frac{35}{18})^2 + \frac{2}{9}(2 - \frac{35}{18})^2 + \frac{1}{6}(3 - \frac{35}{18})^2 + \frac{1}{9}(4 - \frac{35}{18})^2 + \frac{1}{18}(5 - \frac{35}{18})^2 = \frac{665}{324}$

D'où l'écart type de X : $\sigma(X) = \sqrt{V(X)} = \frac{\sqrt{665}}{18} \approx 1,43$

3.

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

         def unLancer():
             de1 = randint(1,6)
             de2 = randint(1,6)
             x = abs(de1 - de2)
             return x
```

4. Écrire la fonction *moyenneEchantillon* qui simule n lancers de deux dés et renvoie la valeur moyenne de X obtenue sur ces n lancers. Le résultat obtenu avec 100 000 lancers est-il proche du résultat théorique ?

```
In [11]: def moyenneEchantillon(n):
             xmoy = 0
             for i in range(n):
                 xmoy += unLancer()
             xmoy = xmoy/n
             return xmoy

         moyenneEchantillon(100000)
```

```
Out[11]: 1.94306
```

On est bien proche du résultat théorique puisque $\frac{35}{18} \approx 1,94$.

5. Écrire une fonction permettant d'estimer $P(X < 3)$, la probabilité de l'évènement $\{X < 3\}$. Vérifier que cette estimation correspond au résultat attendu.

```
In [12]: def estime(n):
             c = 0
             for i in range(n):
                 if unLancer()<3:
                     c += 1
             return c/n

         estime(100000)
```

```
Out[12]: 0.66819
```

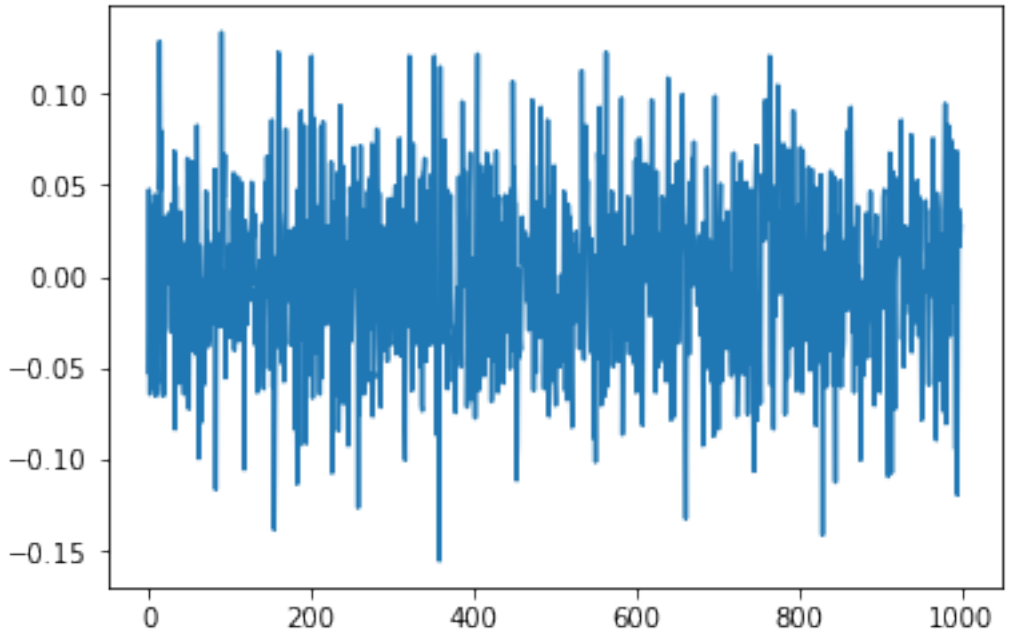
$P(X < 3) = P(X = 0) + P(X = 1) + P(X = 2) = \frac{6}{36} + \frac{8}{36} + \frac{10}{36} = \frac{24}{36} = \frac{2}{3} \approx 0,67$ donc l'estimation correspond bien au résultat attendu.

La fonction suivante simule N échantillons de taille n et renvoie les distances entre la moyenne de chaque échantillon et l'espérance de X.

```
In [14]: def distanceMoyenneEsperance(N, n, esp):
             distance = []
             for i in range(N):
                 dist = (moyenneEchantillon(n) - esp)
                 distance.append(dist)
             return distance
```

Le code suivant permet de représenter cet écart.

```
In [15]: import matplotlib.pyplot as plt
         esp = 35/18
         dist = distanceMoyenneEsperance(1000,1000,esp)
         plt.plot(dist)
         plt.show()
```



6. Compléter le code ci-dessous pour calculer la proportion des cas où l'écart entre la moyenne d'un échantillon et l'espérance de X est inférieur ou égal à $\frac{2\sigma}{\sqrt{n}}$ dans la liste *dist*.
7. Cette proportion varie-t-elle beaucoup lors de plusieurs simulations ?

```
In [131]: from math import sqrt

          dist = distanceMoyenneEsperance(1000,1000,esp)
          sigma = sqrt(665)/18
          n = 1000
          esp = 35/18
          c = 0        # variable comptant le nombre de cas vérifiant la condition
          for d in dist:
              if abs(d) < 2*sigma/sqrt(n):
                  c += 1
          print(c/len(dist))
```

```
0.962
```

7. Cette proportion ne varie pas beaucoup lors des différentes simulations. Elle est autour de 0,95. Il semblerait que $P(|X - E(X)| < \frac{2\sigma}{\sqrt{n}}) \approx 0,95$. En d'autres termes, dans 95% des simulations, la valeur de la variable aléatoire X tombe dans l'intervalle $[E(X) - \frac{2\sigma}{\sqrt{n}}; E(X) + \frac{2\sigma}{\sqrt{n}}]$.

4.3 Exercice 3

On s'intéresse au nombre de filles et de garçons dans des familles de n enfants.
On admet que la probabilité qu'un enfant soit un garçon est égale à 0,5.

1. Compléter la fonction $famille(n)$ simulant une famille à n enfants et renvoyant le nombre de garçons.

```
In [1]: from random import random

        def famille(n):
            nbg = 0
            for i in range(n):
                if random()<0.5:
                    nbg += 1
            return nbg
```

2. Écrire une fonction $echantillon(nb, n)$ simulant un échantillon de nb familles à n enfants et renvoyant une liste contenant le nombre de garçons dans chaque famille.

```
In [4]: def echantillon(nb, n):
            lstgarcons = []
            for i in range(nb):
                lstgarcons.append(famille(n))
            return lstgarcons

        # On peut aussi utiliser les listes en compréhension.
        # Les deux versions font exactement la même chose:
        # elles renvoient une liste contenant le nombre de garçons dans chaque famille.

        def echantillon(nb, n):
            return [famille(n) for i in range(nb)]


        echantillon(20,4)
```

```
Out[4]: [1, 2, 0, 2, 2, 1, 1, 1, 2, 2, 3, 1, 0, 3, 2, 2, 2, 0, 3, 3]
```

On prend maintenant $n = 4$.
On note X la variable aléatoire qui, à toute famille de n enfants, associe le nombre de garçons.

3. Écrire une fonction permettant de calculer la moyenne d'un échantillon de taille nb de valeurs prises par la variable aléatoire X. Le résultat obtenu avec $n = 10000$ vous semble-t-il cohérent ?

```
In [5]: def moyenneEchantillon(nb):
            lstgarcon = echantillon(nb, 4)
            moy = 0
            for x in lstgarcon:
                moy += x
            return moy/len(lstgarcon)

        moyenneEchantillon(100000)
```

```
Out[5]: 2.00215
```

Cette fonction permet de calculer la moyenne d'un échantillon de taille nb de valeurs prises par la variable aléatoire X. Le résultat obtenu semble cohérent puisque dans des familles de 4 enfants, on aura bien 2 garçons en moyenne.

4. A l'aide d'un arbre, déterminer la loi de probabilité de la variable aléatoire X.

5. Calculer l'espérance μ et l'écart type σ de la variable aléatoire X.

6. Dans le code ci-dessous :
- Expliquer la ligne `ecarts = [abs(moyenneEchantillon(nb)-esp) for i in range(N)]`.
- Que représente la variable c ?

7. Simuler 1 000 échantillons de taille 50 de valeurs prises par la variable aléatoire X et calculer les écarts entre la moyenne m de chaque échantillon et l'espérance μ de X. Déterminer la proportion des cas où cet écart est inférieur ou égal à $\frac{2\sigma}{\sqrt{n}}$.

4. On fait un arbre avec deux branches pour le premier enfant (F pour fille et G pour garçon), qui se divisent en deux pour le deuxième enfant (F et G), puis chaque branche se divise encore en deux pour le troisième enfant (on a alors 8 branches), en enfin, chaque branche se divise en deux pour la quatrième enfant. L'arbre a au total 16 branches. Les probabilités portées sur les branches sont toutes égales à $\frac{1}{2}$. Si à chaque division, on a écrit un F en haut et un G en bas, le chemin du haut est F-F-F-F et le chemin du bas est G-G-G-G. La variable X, égale au nombre de garçons peut prendre pour valeurs 0, 1, 2, 3 ou 4. - $P(X=0) = \left(\frac{1}{2}\right)^4 = \frac{1}{16}$ (correspond au chemin F-F-F-F) - $P(X=1) = 4 \times \left(\frac{1}{2}\right)^4 = \frac{1}{4}$ (correspond aux chemins G-F-F-F, F-G-F-F, F-F-G-F, F-F-F-G) - $P(X=2) = 6 \times \left(\frac{1}{2}\right)^4 = \frac{3}{8}$ (correspond aux chemins G-G-F-F, G-F-G-F, G-F-F-G, F-G-G-F, F-G-F-G, F-F-G-G) - $P(X=3) = 4 \times \left(\frac{1}{2}\right)^4 = \frac{1}{4}$ (correspond aux chemins G-G-G-F, G-G-F-G, G-F-G-G, F-G-G-G) - $P(X=4) = \left(\frac{1}{2}\right)^4 = \frac{1}{16}$ (correspond au chemin G-G-G-G)

On peut donner la loi de probabilité dans un tableau mais ce n'est pas obligatoire.

5. $\mu = E(X) = \sum_{i=1}^{5} p_i x_i = \frac{1}{16} \times 0 + \frac{1}{4} \times 1 + \frac{3}{8} \times 2 + \frac{1}{4} \times 3 + \frac{1}{16} \times 4 = \frac{1}{4} + \frac{3}{4} + \frac{3}{4} + \frac{1}{4} = 2$

$V(X) = \sum_{i=1}^{5} p_i (x_i - \bar{x})^2 = \frac{1}{16}(0-2)^2 + \frac{1}{4}(1-2)^2 + \frac{3}{8}(2-2)^2 + \frac{1}{4}(3-2)^2 + \frac{1}{16}(4-2)^2 = \frac{1}{16} \times 4 + \frac{1}{4} + \frac{1}{4} + \frac{1}{16} \times 4 = 1$

D'où $\sigma(X) = \sqrt{V(X)} = \sqrt{1} = 1$

6. - `ecarts = [abs(moyenneEchantillon(nb)-esp) for i in range(N)]` crée une liste en compréhension nommée ecarts contenant N valeurs. Chacune de ces valeurs est calculée ainsi : `moyenneEchantillon(nb)` renvoie le nombre moyen de garçons par famille dans un échantillon de nb familles aléatoires à 4 enfants. `abs(moyenneEchantillon(nb)-esp)` calcule pour chaque échantillon l'écart (positif, en valeur absolue) entre ce nombre moyen et l'espérance. - La variable c compte le nombre de valeurs de la liste ecart inférieures ou égales à $\frac{2\sigma}{\sqrt{n}}$.

7.

```
In [144]: from math import sqrt

          esp = 2
          sigma = 1

          def simulation(N, nb):
              ecarts = [abs(moyenneEchantillon(nb)-esp) for i in range(N)]
              c = 0
              for e in ecarts:
                  if e<=2*sigma/sqrt(nb):
                      c += 1
```

```
            return c/N

        print(simulation(1000, 10))
        print(simulation(1000, 50))
        print(simulation(1000, 500))
```

```
0.96
0.954
0.96
```

7. Reprendre la question précédente avec des échantillons de différentes tailles. Qu'observe-t-on ?

8. En remarquant que $|m - \mu| \leq \frac{2\sigma}{\sqrt{n}} \Leftrightarrow m \in [\mu - \frac{2\sigma}{\sqrt{n}}; \mu + \frac{2\sigma}{\sqrt{n}}]$, interpréter les observations précédentes en terme de fluctuation d'échantillonnage.

7. On teste avec des échantillons de tailles différentes (penser à prendre n grand comme 500, puis n petit, comme 10 ou même 5). Le résultat de la simulation ne change pas beaucoup. Il reste autour de 0,95 ou 0,96.

8. Pour 95% ou 96% des échantillons, le nombre moyen d'enfants par famille dans l'échantillon est dans l'intervalle $[\mu - \frac{2\sigma}{\sqrt{n}}; \mu + \frac{2\sigma}{\sqrt{n}}]$.

5 Quelques liens

Histoire des probabilités

Formule de König-Huygens : La formule de König-Huygens fournit une autre méthode pour calculer la variance.