Correction du TD 18 - Statistiques inférentielles

Exercice 6 : ✶ Intervalles de confiance

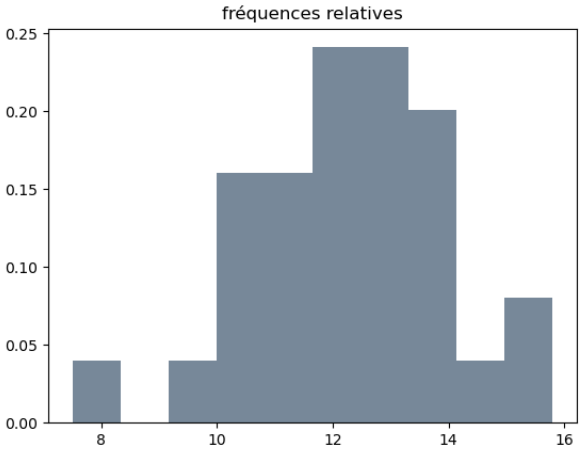
① *Calculons l'espérance et la variance de la série statistique proposée :*
On peut imaginer qu'on ne nous autorise pas à importer la bibliothèque « numpy »... On réécrira donc les fonctions utilisées...

```
S = [13.7,11.9,10.8,12.9,12.5,13.1,10.2,7.5,12.1,13.0,12.9,12.4,12.7,11.6,11.7,...
...,15.8,11.5,11.5,12.4,15.2,14.3,11.4,9.6,14.1,13.6,10.5,12.4,13.7,14.0,10.5,]

def moyenne(L):
    return sum(L)/len(L)

def variance(L):
    L_carres = [x**2 for x in L]
    return sum(L_carres)/len(L)-moyenne(L)**2
```

Un appel à ces fonctions donne : `moyenne(S) = 12.32` et `variance(S) = 2.84`

② On considère **dix** classes de même amplitude.
Pour l'histogramme des fréquences, on fera :

```
plt.figure('frequences relatives')
H = plt.hist(S,10,density = True)
```



On obtient :

```
H[0] = array([0.04016064, 0. , 0.04016064, 0.16064257, 0.16064257, 0.24096386, 0.24096386,
... ...,0.20080321, 0.04016064, 0.08032129])
```
et les classes dans lesquelles sont calculées ces fréquences :
```
H[1] = array([ 7.5 , 8.33, 9.16, 9.99, 10.82, 11.65, 12.48, 13.31, 14.14, 14.97, 15.8
])
```

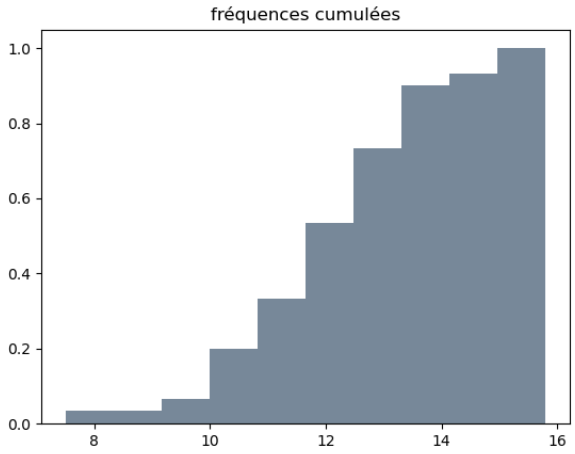
✍**Attention dans ce cas :** On rappelle que Python fait en sorte que la somme des **surfaces** de chaque barre de l'histogramme fasse 1... aussi on pourra vérifier que `sum(H[0])` ne retourne pas 1 !
Sachant que l'amplitude de chaque classe vaut `H[1][1]-H[1][0] = 0.83`, si on vous demande les fréquences relatives dans chaque classe, on exécutera `H[0]*ampl` et on obtiendra :

```
ampl*H[0] = array([0.033, 0. , 0.033, 0.133, 0.133, 0.2 , 0.2 , 0.166, 0.033,
                   0.066])
```

Pour les fréquences cumulées, le problème ne se pose pas et on fera :

```
1 plt.figure('frequences cumulees')
2 Result = plt.hist(S,10,cumulative=True,density=True)
```

On obtient :



`Result[0] = array([0.033, 0.033, 0.066, 0.2 , 0.333, 0.533, 0.733, 0.9 , 0.933, 1. ])`
et les classes dans lesquelles sont calculées ces fréquences :
`Result[1] = array([ 7.5 , 8.33, 9.16, 9.99, 10.82, 11.65, 12.48, 13.31, 14.14, 14.97, 15.8 ])`

On peut faire la synthèse de ces résultats dans le tableau suivant :

classes	[7.8 ;8.3[	[8.3 ;9.2[	[9.2 ;10[	[10 ;10.8[	$\cdots$	[14.4,14.97[	[14.97,15.8]
f_i	0.03	0	0.03	0.13	$\cdots$	0.03	0.06
F_i	0.03	0.03	0.06	0.2	$\cdots$	0.93	1

③ *Donnons des arguments permettant d'admettre que les observations proviennent d'une loi normale :*
Le premier argument est juste un argument de symétrie. Le fait que l'histogramme des fréquences relatives soit globalement symétrique et que les prises de tension soient indépendantes suffit, sans autre indication dans l'énoncé, pour conclure que chacun des X_i suit une loi normale.
Il existe pourtant une méthode qui permet de justifier ce choix de la loi normale pour modéliser les tensions artérielles des étudiants. Elle repose sur la « droite de Henry » dont voici le principe :

— On commence par estimer $\mathbb{P}(X \leq x_i)$ par les fréquences cumulées F_i. Par exemple :
$$\mathbb{P}(X \leq 8.3) \approx F_0 = 0.03,\ \mathbb{P}(X \leq 9.2) \approx F_1 = 0.03,\ \mathbb{P}(X \leq 10) \approx F_2 = 0.06,$$
$$\mathbb{P}(X \leq 10.8) \approx F_3 = 0.2 \text{ etc}$$

— On pose $t_i = \phi^{-1}(\mathbb{P}(X \leq x_i)) = \phi^{-1}(\mathbb{P}(X^* \leq \dfrac{x_i - m}{\sigma})) = \phi^{-1}\left(\phi\left(\dfrac{x_i - m}{\sigma}\right)\right) = \dfrac{x_i - m}{\sigma} = \dfrac{1}{\sigma}x_i - \dfrac{m}{\sigma}$

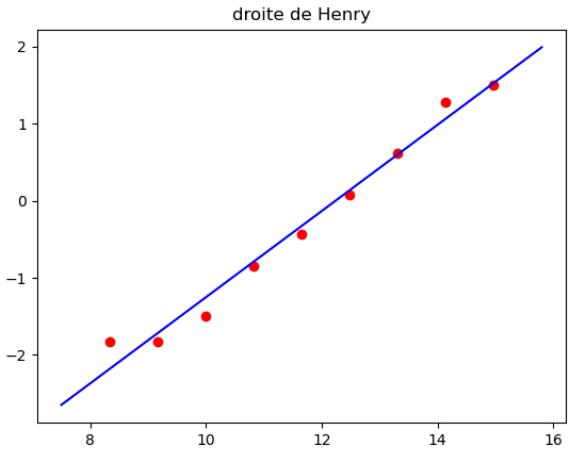
— On trace le nuage de points $(x_i, t_i), i \in [\![0, 10]\!]$. Si le graphe de t en fonction de x est linéaire (on peut le vérifier avec un coefficient de corrélation...), alors on pourra conclure que X suit une loi normale.
Quant à la droite de régression $t = ax + b$ du nuage de points, elle permettra d'obtenir :

$$a = \frac{1}{\sigma} \Leftrightarrow \sigma = \frac{1}{a} \text{ et } b = -\frac{m}{\sigma} \Leftrightarrow m = -b\sigma = -\frac{b}{a}$$

Allons-y avec Python :

```
Fc = Result[0][:-1] # ne pas prendre 1 car on va ensuite composer par norm.ppf()
                    # qui n'est pas définie pour cette valeur

T = norm.ppf(Fc) # Soit t = Phi^{-1}(Fc)

Lx = Result[1][1:-1] # borne droite des classes
plt.figure('droite de Henry')
plt.plot(Lx,T,'ro')
(a,b)=np.polyfit(Lx,T,1)
# Question : Sauriez-vous toujours écrire la fonction qui retourne a et b ?
X = np.linspace(np.min(S),np.max(S),100)
plt.plot(X,a*X+b,'b')

mu_2 = -b/a # on obtient mu_2 = 12.24
s2_2 = 1/a # on obtient s2_2 = 1.79
```

Le nuage de points $(x_i, t_i), i \in [\![0, 10]\!]$ est le suivant :



④ *Donnons pour μ l'intervalle de confiance au niveau de confiance de 95% :* Deux rédactions sont possibles dont l'uen utilise le résultat de la question précédente...

— *Rédaction 1 :* On suppose que les tensions artérielles X_i sont indépendantes et suivent une même loi normale de même espérance μ qu'il s'agit d'encadrer et de même écart-type connu qui vaut $\sigma = 1.79 * *2 = 3.2$. Dès lors :

$$S_n = \sum_{i=1}^{n} X_i \hookrightarrow \mathcal{N}(n \times \mu, n \times 3.2)$$

On rappelle en effet que si X_1 et X_2 sont deux variables aléatoires indépendantes telles que $X_1 \hookrightarrow \mathcal{N}(m_1, \sigma_1^2)$ et $X_2 \hookrightarrow \mathcal{N}(m_2, \sigma_2^2)$ alors $X_1 + X_2 \hookrightarrow \mathcal{N}(m_1 + m_2, \sigma_1^2 + \sigma_2^2)$...

et donc

$$M_n = \frac{S_n}{n} \hookrightarrow \mathcal{N}(\mu, \tfrac{3.2}{n}) \text{ avec } n = 30$$

On rappelle en effet que $\mathbb{E}(M_n) = \frac{1}{n}\mathbb{E}(S_n)$ et $\mathbb{V}(M_n) = \frac{1}{n^2}\mathbb{V}(S_n)$.
D'où :

$$M_n^* = \frac{M_n - \mu}{1.79/\sqrt{n}} \hookrightarrow \mathcal{N}(0,1)$$

ou encore :

$$\mathbb{P}\left(-1.96 \leq \frac{M_n - \mu}{1.79/\sqrt{n}} \leq 1.96\right) = 0.95$$

✐ **Important !** On retrouve un résultat conforme au Théorème central limite sous sa première forme, sauf que ce n'est pas un résultat asymptotique mais bien un résultat valable pour tout $n \in \mathbb{N}^*$... !

Conclusion : l'intervalle $\left]M_{30} - 1.96\frac{1.79}{\sqrt{30}}; M_{30} + 1.96\frac{1.79}{\sqrt{30}}\right[=]11.68; 12.96[$ est un intervalle de confiance au seuil de risque de 5%

— *Rédaction 2 :* On suppose ne pas savoir suivie par les variables X_i. On utilise cette fois le théorème central limite sous sa seconde forme en estimant σ grâce à l'écart-type statistique calculé à la question 1. :
Soit $M_{30} = 12.32$ et $S_{30} = 1.68$ et un 30-échantillon supposé suffisamment grand pour appliquer notre théorème... (c'est là ou le bât blesse car on ne nous dit rien à ce sujet !) :

$$\left]M_{30} - 1.96\frac{S_{30}}{\sqrt{30}}; M_{30} + 1.96\frac{S_{30}}{\sqrt{30}}\right[=]11.71; 12.92[$$

Exercice 7 : ✶✶ : Simulation et comparaison d'intervalles de confiance - Agro 2015

Dans une population d'individus amateurs de café, un proportion p (inconnue) préfère le robusta à l'arabica. On interroge n individus de cette population et on note $X_i = 1$ si l'individu i préfère le robusta et $X_i = 0$ sinon. On suppose que les X_i sont indépendantes. Soit Z_n le nombre d'individus interrogés préférant le robusta à l'arabica.

① *Quelle est la loi suivie par Z_n ? Donner son espérance et sa variance :* C'est immédiat. Z_n suit une loi binomiale de paramètres n et p car Z_n dénombre les succès au cours de n épreuves de Bernoulli indépendantes de même paramètre p.
On conclut avec les résultats de cours $\boxed{\mathbb{E}(Z_n) = np \text{ et } \mathbb{V}(Z_n) = np(1-p)}$

② On se propose de simuler informatiquement le tirage des X_i ainsi que les informations statistiques qu'on peut en tirer ; on rappelle à cet effet que la fonction `random()` de la bibliothèque Python `random` renvoie un nombre pseudo-aléatoire qu'on peut supposer uniformément distribué dans 0 et 1.

a) Proposer une fonction Python `observation()` qui, pour n et p donnés en entrée, renvoie une liste de 0 et de 1 correspondant aux valeurs prises par les X_i pour une observation d'un échantillon aléatoire de taille n répondant au schéma de Bernoulli de paramètre p.
Une écriture condensée est la suivante :

```
        def observation1(n, p):
    return [int(rdm.random()<p) for k in range(n)]
```

mais on peut faire plus long en écrivant :

```
        def observation2(n, p):
    L = [] # initialisation d'une liste vide
    for k in range(n):
        y = 0
        if rdm.random()<p: #succès
            y = 1
        L.append(y)
    return L
```

b) Proposer une fonction Python `moyempir()` fournissant la moyenne empirique (c'est-à-dire, la fréquence des 1) à partir de la donnée d'un échantillon sous la forme d'une liste de 0 et de 1. On l'a déjà fait dans la question précédente :

```
def moyenne(L):
    return sum(L)/len(L)
```

c) Proposer une fonction Python `varempir()` fournissant la variance empirique à partir de la donnée d'un échantillon sous la forme d'une liste de 0 et de 1.
On pourrait là aussi recopier la fonction écrite dans l'exercice précédent mais dans le cas d'une somme de lois de bernoulli il y a plus malin car $S_n^2 = \frac{1}{n}\sum_{i=1}^{n} X_i^2 - M_n^2$ avec X_i^2 et X_i qui ont même loi ! (en effet, $X_i^2(\Omega) = \{0,1\} = X_i(\Omega)$ et $\mathbb{P}(X_i^2 = 1) = \mathbb{P}(X_i = 1) = p$...).
D'où

$$S_n^2 = \frac{1}{n}\sum_{i=1}^{n} X_i - M_n^2 = M_n - M_n^2 = M_n(1 - M_n)$$

Ce qui donne la fonction plutôt concise pour l'écriture de la variance :

```
def varempir(L):
    M = moyempir(L)
    return M*(1-M)
```

Dans la suite, on souhaite proposer (par diverses méthodes) un intervalle de confiance pour p de niveau de confiance 99%, c'est-à-dire un intervalle que l'on peut calculer à partir des observations dont on dispose (c'est-à-dire Z_n) et auquel p appartient pour plus de 99% des échantillons utilisés.

③ On utilise ici le théorème central limite sous sa première forme puisque la loi des X_i est connue. On a $\mu = \mathbb{E}(X) = p$, valeur qu'on cherche à encadre et $\sigma = \sqrt{p(1-p)}$. Alors pour n suffisamment grand, $a, b \in \mathbb{R}$, $a < b$, on a :

$$\mathbb{P}\left(a < \frac{M_n - p}{\sqrt{\frac{p(1-p)}{n}}} < b\right) \approx \phi(b) - \phi(a)$$

et comme $Z_n = \dfrac{M_n}{n}$, on a en posant $b = u$ et $a = -u$:

$$\mathbb{P}\left(-u\sqrt{\frac{p(1-p)}{n}} \leq M_n - p \leq u\sqrt{\frac{p(1-p)}{n}}\right) \approx \phi(u) - \phi(-u) = 2\phi(u) - 1 = 0.99$$

soit :

$$\mathbb{P}\left(\frac{Z_n}{n} - u\sqrt{\frac{p(1-p)}{n}} \leq p \leq \frac{Z_n}{n} + u\sqrt{\frac{p(1-p)}{n}}\right) \approx 0.99$$

avec $\phi(u) = \dfrac{1.99}{2} \Rightarrow u = \phi^{-1}\left(\dfrac{1.99}{2}\right) \approx 2.57$

④ *Montrons que pour tout* $p \in [0,1]$, $p(1-p) \leq 1/4$ *:* C'est une question de cours qui repose sur l'étude de la fonction $f : x \longmapsto x(1-x)$ sur l'intervalle $[0,1]$. Dès lors :

$$0 < \frac{u\sqrt{p(1-p)}}{\sqrt{n}} \leq \frac{u}{2\sqrt{n}}$$

et donc :

$$\left]\frac{Z_n}{n} - u\sqrt{\frac{p(1-p)}{n}}; \frac{Z_n}{n} + u\sqrt{\frac{p(1-p)}{n}}\right[\subset \left]\frac{Z_n}{n} - \frac{u}{2\sqrt{n}}; \frac{Z_n}{n} + \frac{u}{2\sqrt{n}}\right[$$

Conclusion : L'intervalle $I_1 = \left]\dfrac{Z_n}{n} - \dfrac{2.57}{2\sqrt{n}}; \dfrac{Z_n}{n} + \dfrac{2.57}{2\sqrt{n}}\right[$ est l'intervalle de confiance cherché.

⑤ Proposer une fonction Python `ic1()` fournissant (sous forme d'une liste de deux valeurs) un intervalle de confiance pour p au niveau 99%, à partir d'un échantillon fourni sous la forme d'une liste de 0 et de 1.
C'est immédiat. On écrira par exemple :

```
1  def ic1(L):
2      M = moyempir(L)
3      n = len(L)
4      u = norm.ppf(1.99/2)
5      a,b = M-u/(2*np.sqrt(n)),M+u/(2*np.sqrt(n))
6      return [a,b]
```

⑥ En recourant à la seconde forme du TCL, proposer un autre intervalle de confiance de niveau 99% por p, qui sera noté I_2 dans la suite. On estime cette fois l'écart-type par l'ecart-type statistique. Soit :

```
1  def ic2(L):
2      M,S2 = moyempir(L),varempir(L)
3      n = len(L)
4      u = norm.ppf(1.99/2)
5      a,b = M-u*np.sqrt(S2/n),M+u*np.sqrt(S2/n)
6      return [a,b]
```

⑦ Comparer les intervalles de confiance I_1 et I_2 (lorsque n est suffisamment grand). On appelle plusieurs fois les fonctions précédentes et on note que $ic2(L) \subset ic1(L)$ ce qui est conforme au fait que $\mathbb{E}(S_n) < \sigma$ (biais de la variance empirique).

⑧ Le principe des intervalles de confiance est que, si l'on dispose d'un grand nombre d'échantillons issus d'un tirage de Bernoulli de paramètre p et de longueur n (n grand), alors p doit appartenir, dans 99% des cas, aux intervalles de confiance I_1 et I_2. Vérifier ce faite par simulation.
On rappelle que $\mathbb{P}(ic \text{ contient } p) \approx 0.99$.
Donc on répète 1000 fois la construction des intervalles de confiance et on compte le nombre de fois où ils ne contiennent pas p, à savoir :

$$p \notin]a,b[\Leftrightarrow p \leq a = ic[0] \text{ ou } p \geq b = ic[1]$$

```
1  def testIC(m,n,p):
2      cpt = 0
3      for k in range(m):
4          L = observation1(n,p)
5          I = ic1(L)
6          if (p < I[0]) or (p > I[1]):# p n'est pas dans I
7              cpt += 1
8      return cpt/m
```