Probas-Stats avec Python

Remarque • Avec les données en exemple qui suivent et les applications numériques (#AN), vous devriez voir quelles sont les commandes correspondantes sur vos calculatrices.

I Importations

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import linregress
```

II Données en exemple

$x=$	-1.61	-1.20	-0.97	-0.51	-0.42
$y=$	2.22	2.27	2.38	2.60	2.65

```
X = [-1.61,-1.20,-0.97,-0.51,-0.42]
Y = [2.22,2.27,2.38,2.60,2.65]
```

III Statistiques univariées

```
# Moyenne :
mx = np.mean(X)              #AN : -0.94146716215

# Variance des données :
s2x = np.var(X,ddof=0)       #AN : 0.195570714089

# Écart-type des données :
sx = np.std(X,ddof=0)        #AN : 0.442233777644

# Variance non biaisée :
np.var(X,ddof=1)             #AN : 0.244463392612

# Écart-type non biasé :
np.std(X,ddof=1)             #AN : 0.494432394379
```

Rem.1

a. Python calcule de la façon suivante :

$$\texttt{np.std(X,ddof=D)} = \frac{1}{n-\texttt{D}}\sum_{i=1}^{n}(\texttt{X[i]}-\bar{x})^2,$$

qui vaut bien s_x^2 quand `ddof`$=0$, et la variance non biaisée si `ddof`$=1$.

b. `ddof` signifie : `delta degrees of freedom`.

c. On voit encore sur cet exemple le fait que la variance (resp. l'écart-type) empririque sous-estime systématiquement la variance (resp. l'écart-type) théorique.

IV Statistiques bivariées

```
# Covariance de la série :
M = np.cov(X,Y,ddof=0)

#AN :
array([[ 0.19557071,  0.07506966],
       [ 0.07506966,  0.03029122]])

sxy = M[0,1]        #AN : 0.07506966

# Coefficient de corrélation : voir V. A) !
```

Rem.2 On obtient pour la covariance non pas un flottant, mais une matrice (symétrique réelle). C'est la matrice :

$$M = \begin{pmatrix} s_x & s_{xy} \\ s_{xy} & s_y \end{pmatrix},$$

dite *matrice de covariance*. Par lecture du tableau `M` :

```
# sx = 0.19557071
# sy = 0.03029122
# sxy = 0.07506966
```

V Régression linéaire

A) Coefficients de la droite de régression

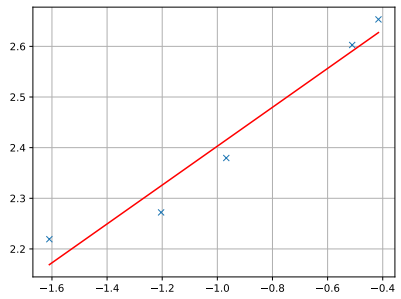
Les trois premiers arguments de sortie de la fonction `linregress` sont les coefficients a,b de la droite de régression, ainsi que le coefficient de corrélation r$=r_{x,y}$.

```
a,b,r = linregress(X,Y)[:3]

#AN :
# a = 0.38384918507505805
# b = 2.7867428337153797
# r = 0.9753357022963296
```

b) Tracés du nuage de points et de la droite de régression

```
# Nuage de points
plt.plot(X,Y,'x')
# Droite de régression
x = np.array(X)
y = a*x+b # avantage de np.array sur list !
plt.plot(x,y,'r')
plt.grid('on')
plt.show()
```



VI Autour de la loi $\mathcal{N}(m, \sigma^2)$

a) Exemple

Exple.1 | Dans une population, il y a 20% de malades. On prélève indépendamment un échantillon de 500 individus.

a. Calculer la probabilité p_1 que l'échantillon contienne 80 malades

b. Calculer la probabilité p_2 que l'échantillon contienne entre 50 et 150 malades.

b) Solution

La variable aléatoire S qui compte le nombre de malades dans l'échantillon de 500 individus suit une soit $\mathscr{B}(N, p)$ où $N = 500$, et $p = 0,2$. Les hypothèses du théorème de Moivre-Laplace sont vérifiées ici, puisque :

- $N = 500 \geq 30$
- $Np = 100 \geq 5$
- $Nq = 400 \geq 5$

On peut donc remplacer la loi de S par la loi d'une variable aléatoire Z de loi $\mathcal{N}(E(S), V(S))$. On obtient donc les réponses suivantes :

a. À l'aide d'une correction de continuité pour commencer :

$$\begin{aligned} p_1 = P(S = 80) &= P(S \in [79,5; 80,5]) \\ &\simeq P(Z \in [79,5; 80,5]) \\ &= F_Z(80,5) - F_Z(79,5), \end{aligned}$$

où F_Z est la fonction de répartition de la variable Z.

b. De même :

$$\begin{aligned} p_2 = P(S \in [100, 150]) &\simeq P(Z \in [100; 150]) \\ &= F_Z(150) - F_Z(100), \end{aligned}$$

c) Calculs sous Python

Ces deux dernières expressions approchées des probabilités p_1, p_2 se calculent facilement sous Python puisque la fonction F_Z y est implémentée :

a. La commande `norm.cdf(t,m,sigma)` renvoie $F_Z(t)$ pour une variable aléatoire Z de loi $\mathcal{N}\left(\texttt{m}, \texttt{sigma}^2\right)$:

$$F_Z(t) = \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{t} \exp - \frac{(x-m)^2}{2\sigma^2} dx.$$

b. ☠ Attention, c'est bien σ, et pas σ^2 qui est pris en argument de la fonction `cdf` !

c. Par défaut, `(m,sigma)=(0,1)` : ainsi `norm.cdf(t)` renvoie la valeur de $\Phi(t)$.

d. `cdf` signifie : `cumulative distribution function`.

```
N,p  = 500, 0.2  # Param. de la loi binomiale
q = 1-p
def Fz(t):
    m,s  = N*p,np.sqrt(N*p*q) # évite d'avoir à
    return norm.cdf(t,m,s)    # retaper m,s

# Réponse à la question a.
p1 = F_z(80.5)-Fz(79.5)
     #AN : 0.003668872638773692

# Réponse à la  question b.
p2 = F_z(150)-Fz(50)
     #AN : 0.9999999731525148
```

Rem.3 | Sur cet exemple, l'écart type de S est $\sigma_s \simeq 8.94$, et $\texttt{m} = E(S) = Np = 100$. Or $[100; 150] = [\texttt{m} - 50; \texttt{m} + 50]$, et $50 \simeq 5.6\sigma_S$. Il est donc tout à fait normal (!) que la valeur de p_2 soit aussi proche de 1 (en effet, d'après les propriétés de la loi normale, ce serait déjà le cas pour un intervalle de rayon $3\sigma_S$ centré sur m).