TD D'INFORMATIQUE : RÉGRESSION LINÉAIRE

On souhaite calculer la pente et l'ordonnée à l'origine d'une droite de régression linéaire, pour des données de mesures expérimentales et valider l'hypothèse d'une relation linéaire entre ces données :

On envisage ainsi ces données comme deux variables aléatoires Xi et Yi correspondant aux N mesures indexées par i de 0 à N-1. On veut également pondérer les différentes mesures en fonction de l'incertitude de chacune des mesures sur la variable Y mesurée. La question que l'on pose est donc de déterminer le degré de corrélation entre ces deux variables aléatoires.

PRÉSENTATION DU PROBLÈME [bien lire cette partie]

Les valeurs expérimentales nous sont données sous la forme de deux listes X et Y de N éléments, ainsi que la listes Sy [Sy pour sigma_y] des N incertitudes respectives aux N valeurs de Y mesurées.

On veut évaluer la relation linéaire entre les deux variables aléatoires : Y = a.X + b

Soit $y_i^{th} = ax_i + b \quad \forall i$: la valeur théorique attendue pour Yi en Xi, et soit y_i^{obs} la valeur réellement mesurée. On peut quantifier l'écart global de nos mesures à la théorie en calculant l'écart-quadratique moyen entre la théorie et l'expérience (divisé par l'incertitude) :

$$S = \sum_0^{N-1} \left(\frac{y_i^{th} - y_i^{obs}}{\sigma_i} \right)^2 \qquad \text{soit} \qquad S(a,b) = \sum_0^{N-1} \frac{\left(ax_i + b - y_i^{obs} \right)^2}{\sigma_i^{\,2}}$$

S(a, b) est l'écart quadratique cumulé entre la théorie et l'expérience, et pondéré par l'inverse de l'incertitude de mesure sur Yi : plus on est confiant sur un point de mesure, plus ce point pèsera sur S et inversement. **C'est la méthode dite du χ^2 de Pearson !**

Pour trouver la meilleure droite passant à travers les données, on cherche donc à minimiser cet écart : il faut trouver a et b qui rendent S(a, b) minimale. On écrit :

$$\frac{dS(a,b)}{da} = 0 \qquad \text{qui amène :} \qquad \sum_0^{N-1} \frac{x_i \left(ax_i + b - y_i^{obs} \right)}{\sigma_i^{\,2}} = 0 \qquad \text{(eq° a)}$$

$$\frac{dS(a,b)}{db} = 0 \qquad \text{qui amène :} \qquad \sum_0^{N-1} \frac{\left(ax_i + b - y_i^{obs} \right)}{\sigma_i^{\,2}} = 0 \qquad \text{(eq° b)}$$

On peut alléger les notations en introduisant l'opérateur de moyenne < . > qui donne la moyenne sur l'ensemble des valeurs prises par l'indice i d'une variable aléatoire :

$$\boxed{\langle f \rangle = \sum_0^{N-1} \frac{f_i}{\sigma_i^{\,2}} \Big/ \sum_0^{N-1} \frac{1}{\sigma_i^{\,2}}}$$ (eq° c) Cette moyenne étant pondérée par l'inverse de la variance.

En pratique on écrira une fonction en Python qui lit une liste de mesures et une liste d'incertitudes et qui renvoie la valeur moyenne [cf fonction bracket(x) au verso].
On peut ainsi reformuler nos équations (eq° a) et (eq° b) en développant et en divisant par la somme des inverses des variances. Soit :

$$\left\langle x_i\left(ax_i+b-y_i^{obs}\right)\right\rangle=0 \quad\Rightarrow\quad a\left\langle x_i^{\,2}\right\rangle+b\left\langle x_i\right\rangle-\left\langle x_i y_i^{obs}\right\rangle=0 \qquad \text{(eq° a')}$$

$$\left\langle ax_i+b-y_i^{obs}\right\rangle=0 \quad\Rightarrow\quad a\left\langle x_i\right\rangle+b-\left\langle y_i^{obs}\right\rangle=0 \qquad \text{(eq° b')}$$

Les inconnues étant a et b c'est un simple système de deux équations à deux inconnues. Soit :

$$a=\frac{\left\langle x_i y_i^{obs}\right\rangle-\left\langle x_i\right\rangle\left\langle y_i^{obs}\right\rangle}{\left\langle x_i^{\,2}\right\rangle-\left\langle x_i\right\rangle^2} \qquad \text{et} \qquad b=\left\langle y_i^{obs}\right\rangle-a\left\langle x_i\right\rangle$$

On note que pour calculer les termes $\left\langle x_i y_i^{obs}\right\rangle$ ou encore $\left\langle x_i^{\,2}\right\rangle=\left\langle x_i\cdot x_i\right\rangle$, il faut construire au préalable la liste des N produits Xi*Yi ou Xi*Xi. Il faudra donc créer une fonction qui prend en argument deux listes et renvoie la liste des N produits deux à deux [cf fonction ProdXY(x) au verso].

Calcul du coefficient de corrélation :
(En général, les calculatrices indiquent R au carré.)

$$\boxed{R=\frac{COV(x_i,y_i^{obs})}{\sigma_x\sigma_y}} \qquad \text{(eq° d)}$$

On définit pour cela :

- la covariance des deux variables aléatoires : $COV(x_i,y_i^{obs})=\dfrac{\left\langle\left(x_i-\mu_x\right)\left(y_i^{obs}-\mu_y\right)\right\rangle}{N}$

  => Celle-ci nécessite une fonction moyenne qui calcule la moyenne d'une liste [à utiliser sur X et Y].

- Les écarts types des listes X et Y : σ_x et σ_y

Comparaison au résultat obtenu avec votre machine à calculer :

Nous testerons nos résultats sur les listes de données suivantes :

```
##données
x=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y=[0.8, 2.4, 2.0, 2.2, 2.7, 3.5, 3.0, 4.0, 5.0, 4.8]

#incertitude sur Yi
Sy=[0.5, 1.0, 0.2, 0.5, 0.5, 0.5, 0.5, 1.0, 0.2, 0.5]
```

Rq : A priori votre calculatrice ne prendra pas en compte les incertitudes de mesure. Toutes les mesures ont alors le même poids qu'elles soient fiables ou non.
Pour comparer à votre machine, il suffit de poser Sy[i] = 1 pour toutes les mesures i. Il est bien sûr plus fiable de tenir compte des incertitudes de mesures.

*** Y = aX + b ***

a : 0.48 b : 0.50 R2 : 0.89366 R : 0.94534

IMPLÉMENTATION EN PYTHON

Listes de fonctions à créer :

Dans chaque cas on veillera à minimiser les calculs redondants donc inutiles.
Pb : les fonctions ci-dessous ne prennent pas en entrée la/les moyenne(s) des listes.

```
def Mean(x):          #moyenne d'une variable aléatoire
                      #retourne la moyenne d'une liste x      ex: mean([1, 2, 3]) retourne le réel 2.0

def Sigma(x):         #écart-type d'une variable aléatoire
                      #retourne l'écart type d'une liste x (soit un réel)

def Cov(x, y):        #Covariance des variables aléatoires X et Y
                      #retourne la covariance de deux listes x et y (soit un réel)

def ProdXY(x, y):     #Construction de la liste produit Xi.Yi
                      #exemple ProdXY([1, 2, 3], [4, 5, 6])  retourne [4, 10, 18] donc une liste !

def Bracket(x, Sy=[1.]*len(x)):     # cf (eq° c)
                                    # Sy=[1.]*len(x) permet de mettre par défaut toutes les
                                    # incertitudes à 1.0
#Attention par définition de notre opérateur de moyenne, Bracket prend en entrée :
#une liste x quelconque de taille N et les N incertitudes Sy d'indices i associées à Y.
# Rq : Sy=[1.]*len(x) ne marche pas si x n'est pas déjà en mémoire …

def Regression(x, y, Sy=[1.]*len(y)):     ## Calcul de la régression

    # Calcul de a
    # Calcul de b
    # Calcul de R
    # Affichage
```

FIN