

TP8 : Statistiques bivariées avec le module pandas

Commencer par importer les bibliothèques suivantes dans chaque fichier **Python** utilisé :

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

I. Importation d'un fichier csv en Python

Dans ce T.P, on utilise la bibliothèque **pandas** qui permet la lecture de fichiers *.csv* (*Comma Separated Values*) et la création/manipulation de tables. Si certaines commandes et instructions seront rappelées ci-après, on renvoie également au cours de première année.

On va utiliser comme document de travail, tout au long de ce TP, un [fichier](#) csv qui regroupe tout un tas de données publiques récupérées sur le site [World Bank Data](#). En particulier, pour la période (1960-2020) et dans le Monde entier :

- le taux de fertilité des jeunes femmes (nombre d'enfants pour 1000 jeunes femmes entre 15 et 19 ans),
- le pourcentage (du groupe concerné) de jeunes femmes étant scolarisé dans l'enseignement secondaire,
- l'espérance de vie,
- le pourcentage de population ayant accès à l'électricité,
- les émissions de CO₂ (en kT),
- la consommation électrique moyenne *per capita* (en KWh par habitant),
- la surface de forêt (en *km*²).

On commence donc par importer le fichier susmentionné dans Python avec la commande

```
1 donnees=pd.read_csv('https://tomdutilleul.com/wp-content/uploads/
2 2022/10/tp_stats_data.csv',sep=';')
```

Ici, on rajoute l'argument `sep=';'` car les données du fichier sont séparées par un point virgule.

La variable `donnees` est alors une *table de données* ou *Dataframe*. On rappelle que :

- La commande `donnees.head()` permet de n'afficher que les 5 premiers rangs du tableau.
- La commande `donnees.shape` renvoie un couple (n, p) où n est le nombre de lignes et p le nombre de colonnes.
- La commande `donnees.columns` permet d'afficher l'ensemble des colonnes du tableau.

Une colonne intitulée `index` est ajoutée par la bibliothèque **pandas** à la table de données lors de sa lecture afin de donner un numéro à chaque ligne de la table de données (la numérotation commençant comme toujours avec Python à 0).

Notre jeu de données manipulé ici est (relativement) grand. Il contient 8 colonnes et 61 lignes... On va donc n'en considérer qu'une sous-table.

```
1 table1=donnees[['Année', 'taux fertilité j. femmes','femmes scol. sec.']]
2 table1=table1.rename(columns={'taux fertilité j. femmes': 'TF',
3 'femmes scol. sec.': 'FSS'})
```

II. Commandes de statistiques descriptives avec pandas

On a vu au TP précédent plusieurs commandes pour calculer la moyenne empirique, la variance empirique ainsi que l'écart-type empirique. Ces commandes utilisaient le module `numpy` et s'appliquaient donc à des objets de type *vecteurs* en **Python**. Dans ce TP, on manipule des *tables de données*, il faut donc utiliser des commandes propres au module `pandas` :

- `table1.mean()` renvoie la liste des moyennes pour chaque colonne numérique et `table1['nom_de_la_colonne'].mean()` renvoie la moyenne des valeurs d'une colonne précise.
- `table1.std()` renvoie la liste des écarts-types pour chaque colonne numérique et `table1['nom_de_la_colonne'].std()` renvoie l'écart-type des valeurs d'une colonne précise.
- `table1.median()` renvoie la liste des médianes pour chaque colonne numérique et `table1['nom_de_la_colonne'].median()` renvoie la médiane des valeurs d'une colonne précise.
- `table1.min()` renvoie la liste des valeurs minimales pour chaque colonne numérique et `table1['nom_de_la_colonne'].min()` renvoie la valeur minimale d'une colonne précise.
- idem avec `table1.max()`

Définition Une médiane d'une série statistique est une valeur m telle que 50% des données sont inférieures à m et 50% supérieures à m . Si la série statistique comporte $2n + 1$ valeurs ordonnées, on choisit pour m la valeur de rang $n + 1$. Si la série statistique comporte $2n$ valeurs ordonnées, on choisit pour m la moyenne arithmétique entre les valeurs de rang n et $n + 1$. Intuitivement, la médiane est le point milieu des observations (à ne pas confondre avec le point moyen).

On s'intéresse aussi parfois à d'autres quantiles. On note q_α le quantile d'ordre α qui désigne le réel tel qu'une proportion α des observations est inférieure à α et une proportion $1 - \alpha$ est supérieure à q_α . La médiane est le quantile d'ordre $\frac{1}{2}$.

III. Tracer un nuage de points à partir d'une table de données

- Recopier le code suivant et le compiler. Commenter.

```

1 table1 = table1.dropna() # on supprime les lignes avec des données manquantes
2 X = table1['FSS']
3 Y = table1['TF']
4 plt.grid()
5 plt.plot(X,Y,'+')
6 plt.show()
```

On voit un nuage de points plutôt bien aligné, sous la forme d'une droite décroissante.

- Modifier le code précédent pour qu'il affiche également le point moyen du nuage.

```

1 table1 = table1.dropna()
2 X = table1['FSS']
3 Y = table1['TF']
4 plt.grid()
5 plt.plot(X,Y,'+')
6 plt.plot(X.mean(),Y.mean(),'+')
7 plt.show()
```

IV. Coefficient de corrélation linéaire et régression linéaire

- Donner les formules en **Python** pour calculer s la covariance empirique et r le coefficient de corrélation linéaire associés à la série statistique double (X, Y) considérée dans la partie précédente.

```
s = (X*Y).mean() - X.mean() * Y.mean()
r = s/(X.std()*Y.std())
```

- Calculer le coefficient de corrélation linéaire de la série statistique double (X, Y) . Commenter

On trouve $r = -0,94$. On a $|r| \geq 0,9$ donc un ajustement linéaire est justifié.

- Ecrire un programme permettant de représenter la droite de régression linéaire de Y en X au dessus du nuage de points de la série statistique double (X, Y) .

```
1 table1 = table1.dropna()
2 X = table1['FSS']
3 Y = table1['TF']
4 plt.grid()
5 plt.plot(X,Y,'+')
6 plt.plot(X.mean(),Y.mean(),'+')
7 s = (X*Y).mean() - X.mean() * Y.mean()
8 a = s/(X.std()**2)
9 b = Y.mean() - a*X.mean()
10 plt.plot(X,a*X+b)
11 plt.show()
```

V. Une régression linéaire avec transformation

Dans certains cas, qui seront pour nous complètement guidés par l'énoncé, on peut appliquer le principe de régression linéaire à un couple obtenu par transformées de X et/ou de Y .

On considère un exemple avec des données correspondant à l'évolution du PIB par habitant (en USD) et du pourcentage de la population en zone urbaine de la Norvège, de 1960 à 2020 (source : [World Bank Data](#)).

- Recopier et exécuter les instructions suivantes. Commenter le nuage de points.

```
1 data2=pd.read_csv('http://frederic.gaubard.com/2223/tp2_nor.csv',sep=';')
2 X = data2['PIB per capita']
3 Y = data2['Pop urbaine %']
4 plt.grid()
5 plt.plot(X,Y,'+')
6 plt.show()
```

Les points ne sont pas bien alignés, il y a une très forte croissance au début puis cette croissance diminue très fortement.

- Représenter le nuage de points $(\ln(X), Y)$. Commenter.

Les points sont plus alignés que pour le nuage de points précédent.

- Calculer le coefficient de corrélation linéaire de Y en $\ln(X)$.

$$r = 0.9473699208635016$$

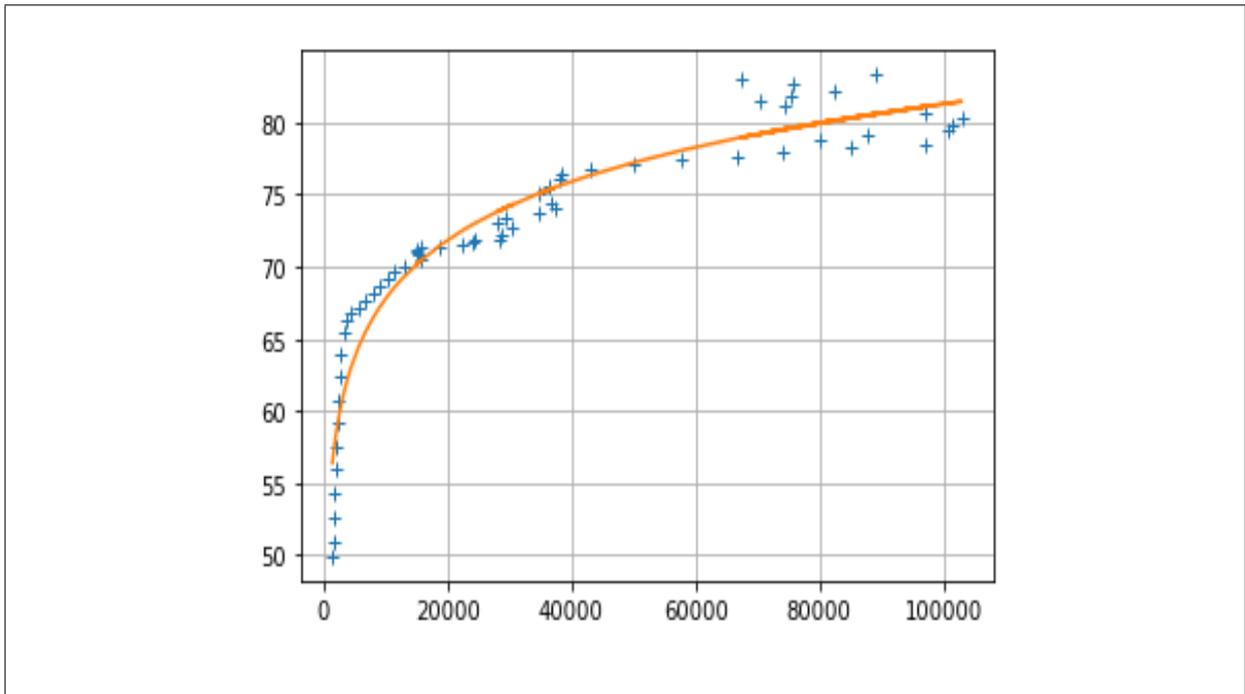
- Déterminer l'équation de la droite de régression linéaire de Y en $\ln(X)$.

Pour éviter les confusions, on note t l'abscisse et y l'ordonnée. La droite de régression linéaire a alors pour équation $y = at + b$ où $a = 5.87$ et $b = 13.69$.

- En déduire que l'on peut supposer que la relation de dépendance entre X et Y est de la forme : $Y = a \ln(X) + b$.

Le coefficient de corrélation linéaire de Y en $\ln(X)$ vérifie $|r| \geq 0,9$ donc un tel ajustement linéaire est justifié.

- Représenter à nouveau le nuage de points associé à la série statistique double (X, Y) en superposant la courbe d'équation $y = a \ln(x) + b$.



VI. L'utilisation de la bibliothèque pandas aux concours

VI.1. HEC/ESSEC I 2023

On considère X une variable aléatoire à densité de fonction de répartition F et de densité de probabilité f qui dépendent d'un paramètre inconnu θ , où $\theta \in \Theta$, Θ un intervalle de \mathbb{R} .

Soit a un point de continuité de f , fixé. On souhaite estimer $f(a)$.

Par exemple, si X suit la loi exponentielle de paramètre θ et $a > 0$, on souhaite estimer $\theta e^{-\theta a}$.

On dispose pour tout $\theta \in \Theta$, d'une suite de variables aléatoires $(X_i)_{i \geq 1}$ indépendantes de même loi que X .

On choisit une suite $(h_n)_{n \geq 1}$ de réels strictement positifs tels que :

$$\lim_{n \rightarrow +\infty} h_n = 0 \text{ et } \lim_{n \rightarrow +\infty} nh_n = +\infty$$

Pour tout $n \in \mathbb{N}^*$, et $\omega \in \Omega$, on définit :

$C_n(\omega)$ comme le nombre d'indices $i \in \llbracket 1, n \rrbracket$ tels que $X_i(\omega) \in]a - h_n, a + h_n]$

et $f_n(\omega) = \frac{1}{2nh_n} C_n(\omega)$.

On suppose que l'on dispose d'un fichier `stats.csv` qui comporte une colonne nommée `salaire`. On considère que les valeurs de cette colonne constituent la réalisation d'un échantillon de la loi de X dont la taille dépasse 10000.

- Après avoir exécuté `import pandas as pd`, quelle(s) instruction(s) permet(tent) de lire dans le fichier `stats.csv` les valeurs de la colonne `salaire` et d'affecter cette série `pandas` obtenue à une variable `échantillon` ?

On supposera que le fichier `stats.csv` se trouve dans le répertoire de travail.

```

1 donnees = pd.read_csv('stats.csv')
2 échantillon = donnees['salaire']

```

- On souhaite calculer et afficher $f_n(\omega)$ pour a donné, lorsque la réalisation d'un échantillon $(X_1(\omega), \dots, X_n(\omega))$ de la loi de X est représentée en **Python** par `échantillon` et, pour tout $n \in \mathbb{N}^*$, $h_n = \frac{1}{\sqrt{n}}$.

Compléter le script suivant pour qu'il réalise cette tâche.

```

1 a = float(input('a='))
2 n = échantillon.count()
3 h = 1 / np.sqrt(n)
4 C = 0
5 for i in range(n):
6     if ... and ...:
7         ... += 1
8 print(C / ...)

```

```
1 a = float(input('a='))
2 n = échantillon.count()
3 h = 1 / np.sqrt(n)
4 C = 0
5 for i in range(n):
6     if échantillon[i] > a - h and échantillon[i] <= a + h:
7         C += 1
8 print(C / (2 * n * h))
```