Université Claude Bernard - Lyon 1 **Semestre d'automne 2024–2025**

L2 - Informatique UE : Statistiques pour l'Informatique

Feuille TP 2: Résumés numériques, régression linéaire.

Dans ce TP, on utilisera les bibliothèques suivantes :

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

```
import pandas as pan
import scipy.stats.mstats as ms
```

On charge deux jeux de données[1], dont celui du TP1, en utilisant les commandes :

```
df=pan.read_csv("https://math.univ-lyon1.fr/~dabrowski/nutriage.csv",sep="\t")
Air=pan.read_csv("https://math.univ-lyon1.fr/~dabrowski/pollution.csv",sep="\t",
    na_values="-")
```

On peut utiliser une petite boucle pour nommer toutes les variables de `df` plus simplement (par exemple `poids` au lieu de `df['poids']`.)

```
for nom in df.keys():
    globals()[nom] = df[nom]
```

Exercice TP2.1 On considère de nouveau les vecteurs $x = (1, 8, 5, 1)$ et $y = (0, 1, 3, 5, 7, 9)$.

1. Pourquoi la commande `plt.plot(x,y)` retourne-t-elle une erreur ?
2. Ajouter $(3, 5)$ au vecteur x et représenter le nuage des points (x_i, y_i).
3. Ajouter le point $(\overline{x}, \overline{y})$ en rouge en utilisant la commande `plt.plot`.
4. Ajouter la droite de régression en utilisant `plt.axline` et `ms.linregress`.
5. Calculer la corrélation empirique `cor(x,y)` pour décider si une approximation par une droite est raisonnable.

Exercice TP2.2

1. Calculer la moyenne, variance, variance sans-biais, médiane, écart-type sans-biais des variables `poids` et `taille`. Indication : On pourra utiliser les fonctions `np.mean`, `np.var`,`np.std`, `np.median`.
2. En utilisant la commande `np.quantile`, calculer les 3 quartiles de `poids` selon la définition du cours.
3. En utilisant la commande `pan.DataFrame`, créez un jeu de donnée `tp` rassemblant poids et taille.
4. Que fait la fonction `tp.describe()` ?
5. Représenter un diagramme de points de $y = poids$ en fonction de $x = taille$.
6. Trouver la droite de régression et l'ajouter au nuage de point, avec un titre.
7. Discuter si cette approximation est raisonnable. Comment interprète-t-on un point qui se trouve nettement au-dessus/au-dessous de la droite de régression ?

Exercice TP2.3

1. Trouver la moyenne et l'écart-type de l'age des femmes de l'échantillon.
2. Tracer sur le même graphique de deux couleurs différentes les nuages de points du poids et de l'âge pour les hommes et les femmes de l'échantillon.
3. Calculer les deux droites de régressions du poids y en fonction de l'âge x, pour chaque groupe.

1. Sources : `http://www.biostatisticien.eu/springeR/jeuxDonnees4.html` et `https://www.atmo-auvergnerhonealpes.fr/dataviz/mesures-aux-stations`

Suppléments : Intervalles de prédiction autour de la régression linéaire.

Dans la suite on va aussi utiliser une librairie qui donne plus d'informations sur la régression linéaire.

```
import statsmodels.api as sm
```

Exercice TP2.4 On reprend la régression du poids en fonction de la taille.

1. On donne le code suivant qui calcule toute l'information sur la régression linéaire

   ```
   X=sm.add_constant(taille)#besoin de donner vecteur constant pour avoir b
   lmdf=sm.OLS(poids,X);resdf=lmdf.fit()
   print(resdf.summary())
   ```

   Trouver la droite de régression $y = ax + b$.
2. Tracer la droite de régression en utilisant `resdf.fittedvalues`
3. Soit `y= a* taille + b`. Calculer le rapport de la variance de `y` et de la variance de `poids`. Où retrouve-t-on cette proportion de la variance du poids capturée par la régression linéaire dans le résultat de `resdf.summary()` ? Comment l'interprétez vous ? Commenter de même les données de l'exerice 3.
4. Dans le cas où le nuage de point est large autour de la droite de régression, il est intéressant de trouver un intervalle de droites parallèles à celle-ci telle que l'on sache avec 95% de chance que les points doivent se trouver entre ses droites. La fonction `get_prediction().summary_frame( alpha=0.05)` permet cela. En utilisant le code suivant, trouvez ce que cette fonction renvoie dans `prediction`.

   ```
   prediction=resdf.get_prediction().summary_frame(alpha=0.05)#1-alpha=0.95 niveau
       de confiance pour la prédiction.
   fig, ax = plt.subplots(figsize=(8, 6))
   ax.plot(taille, poids, "o", label="data")
   ax.plot(taille, prediction["mean"], label="OLS",color="orange")
   ax.plot(taille, prediction["obs_ci_lower"], color="red")
   ax.plot(taille, prediction["obs_ci_upper"], color="red")
   ```

Exercice TP2.5 Le jeu de données `Air` contient des informations issues de relevés de la pollution à la station de Lyon-centre entre 2017 et 2018. Il contient les 5 polluants principaux mesurés presque tous les jours. Elles sont mesurées en microgrammes par mètre cube d'air. Les valeurs indiquées "-" dans le fichier, ou NA après importation par `pandas`, sont les valeurs non disponibles, correspondant à des jours de non-observation. On pourra les supprimer en utilisant par exemple la fonction `dropna`.

1. Créer un `DataFrame` des 652 jours où le Dioxyde d'Azote et les Particules PM10 sont observées.
2. Soit x le logarithme népérien (obtenu en Python par la fonction `np.log`) de la mesure en Dioxyde d'Azote N et y le logarithme népérien de la mesure P en Particules PM10.

   Trouvez (avec la commande `sm.OLS`) la droite de régression de y en fonction de x. Discutez la significativité statistique du résultat (en commentant les résultats de la commande `fit().summary()`). Tracer le nuage de points (x, y) et la droite de régression en couleur, puis deux droites donnant les bornes de l'intervalle de prédiction au niveau 95% (d'une autre couleur). Commenter.
3. En déduire en passant à l'exponentielle, les nuages de points des mesures en Dioxyde d'Azote et en Particules PM10 avec la courbe de régression ($N = CP^{\alpha}$, image de celles de x, y par `np.exp`) et les bornes des intervalles de prédiction au niveau 95% pour ces mesures. Quelle est l'équation de la courbe de régression ainsi obtenue ?