TP 6

Statistiques descriptives bivariées

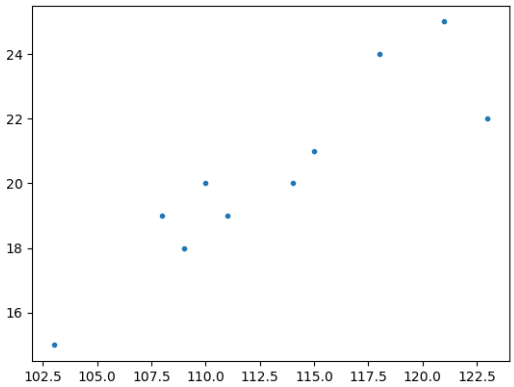
Exercice 1 (★)

1. Pour la série statistique double proposée en exemple, quel est le caractère explicatif et le caractère expliqué ?
2. Représenter le nuage de points associé à cette série statistique double. Quelle fonction de régression vous semble appropriée ?
3. Proposer une droite qui passe « très près » de tous ces points.

1. On a ici le choix, on peut expliquer le poids en fonction de la taille, ou l'inverse. Prenons comme caractère explicatif la taille X, et comme caractère expliqué le poids Y.
2. On utilise le code suivant pour représenter le nuage de points :

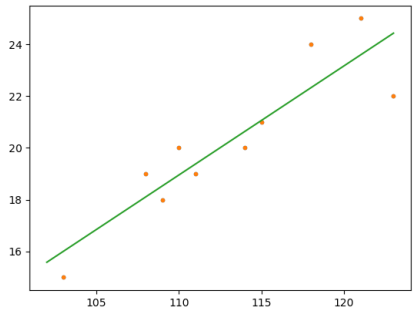
```
X = [121 123 108 118 111 109 114 103 110 115]
Y = [25 22 19 24 19 18 20 15 20 21]
plot2d(X,Y,-1)
```

On obtient :



Les points, bien que non alignés, semblent malgré tout être disposé dans la même direction. On peut donc penser à une fonction affine pour la fonction de régression $f : x \mapsto ax + b$.

3. On peut proposer approximativement la droite suivante qui semble passer « la plus près » de tous ces points :



Exercice 2 (★★)

Reprenons la série statistique de l'exemple de départ.

1. Appliquer la méthode qui précède afin d'obtenir les valeurs de a et b.
2. Représenter la droite des moindres carrés sur le même graphique que le nuage de points. Le résultat est-il conforme à vos attentes ?
3. Quel est le signe du coefficient directeur de cette droite ? Comment l'interprétez-vous ?

1. On doit tout d'abord construire la matrice $A = \begin{pmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}$. On peut le faire à l'aide des commandes suivantes :

```
n = np.shape(x)[0]
A = np.ones((n,2))
A[:,0] = np.transpose(x)
```

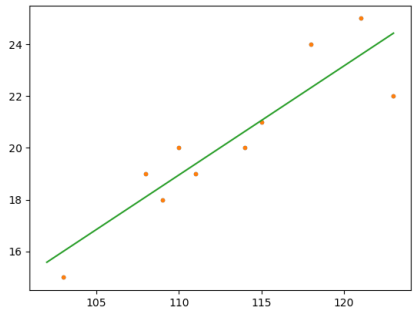
On calcule ensuite la matrice $U = ({}^tA \times A)^{-1} \times {}^tAY$ permettant d'obtenir a et b pour construire la droite des moindres carrés :

```
B = al.inv(np.dot(np.transpose(A),A))
C = np.dot(np.transpose(A),np.transpose(y))
U = np.dot(B,C)
a = U[0]
b = U[1]
```

Reste alors à la tracer avec le nuage de points :

```
u = np.linspace(102,123,2)
plt.plot(u, a*u+b)
plt.show()
```

On obtient :



ce qui correspond (environ) à la droite à laquelle on pensait.

2. Le coefficient directeur est de signe positif (la droite est croissante), ce qu'on peut interpréter par le fait que les caractères quantitatifs X et Y ont tendance à varier dans le même sens. Cela parait conforme à l'intuition, le poids et la taille aillant tendance à varier effectivement dans le même sens.

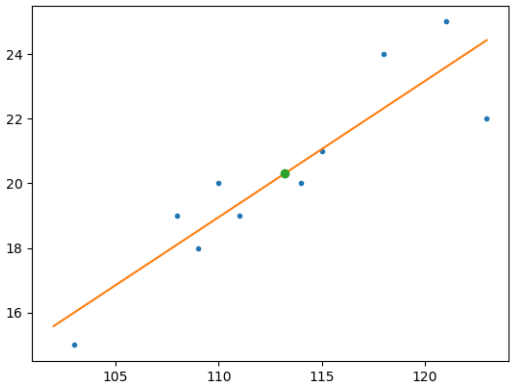
Exercice 3 (★)

Représenter le point moyen dans notre exemple (on pourra utiliser la commande `plt.plot(·, ·,"o")` pour le différentier des autres points). Que constate-t-on ?

On représente le point moyen à l'aide des instructions suivantes :

```
xm = np.mean(x)
ym = np.mean(y)
plt.plot(xm,ym,"o")
```

On obtient le graphe suivant :



On remarque que le point moyen se trouve sur la droite des moindres carrés. On verra que c'est en fait toujours le cas. Ce qui peut permettre de tracer la droite des moindres carrés plus facilement.

Exercice 4 (★ - Formules de Huygens)

Montrer que :

$$\sigma_x^2 = \left(\frac{1}{n}\sum_{i=1}^{n} x_i^2\right) - \overline{x}^2 \quad \text{et} \quad \text{Cov}(x,y) = \overline{xy} - \overline{x}\cdot\overline{y}.$$

> Il suffit de développer. À vous de jouer !

Exercice 5 (★★★)
Montrer ce résultat en effectuant le calcul $U = ({}^tA \times A)^{-1} \times {}^tAY$.

> On a déjà fait une partie du calcul :
>
> $$ {}^tAA = \begin{pmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & n \end{pmatrix} \quad \text{et} \quad \det({}^tA \times A) = n\sum x_i^2 - (\sum x_i)^2 = n^2\sigma_x^2$$
>
> par la formule de Huygens. On obtient :
>
> $$ ({}^tAA)^{-1} = \frac{1}{n^2\sigma_x^2}\begin{pmatrix} n & -\sum x_i \\ -\sum x_i & \sum x_i^2 \end{pmatrix} = \frac{1}{n\sigma_x^2}\begin{pmatrix} 1 & -\overline{x} \\ -\overline{x} & \overline{xx} \end{pmatrix}.$$
>
> D'autre part, on a :
>
> $$ {}^tAY = \begin{pmatrix} x_1 & \dots & x_n \\ 1 & \dots & 1 \end{pmatrix}\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} \sum x_iy_i \\ \sum y_i \end{pmatrix} = n\begin{pmatrix} \overline{xy} \\ \overline{y} \end{pmatrix}.$$
>
> D'où finalement :
>
> $$ U = ({}^tAA)^{-1}\,{}^tAY = \frac{1}{\sigma_x^2}\begin{pmatrix} \mathrm{Cov}(x,y) \\ \overline{xx}\cdot\overline{y} - \overline{x}\cdot\overline{xy} \end{pmatrix}.$$
>
> On obtient bien $a = \dfrac{\mathrm{Cov}(x,y)}{\sigma_x^2}$ et :
>
> $$ b = \frac{\overline{xx}\cdot\overline{y} - \overline{x}\cdot\overline{xy}}{\sigma_x^2} = \frac{\overline{xx}\cdot\overline{y} - \overline{x}^2\cdot\overline{y} + \overline{x}^2\cdot\overline{y} - \overline{x}\cdot\overline{xy}}{\sigma_x^2} = \overline{y} - \overline{x}\frac{\overline{xy} - \overline{x}\cdot\overline{y}}{\sigma_x^2} = \overline{y} - \overline{x}\frac{\mathrm{Cov}(x,y)}{\sigma_x^2}$$

Exercice 6 (★★)
À l'aide de l'inégalité de Cauchy-Schwarz dans $\mathbb{R}^n$, démontrer la propriété précédente.

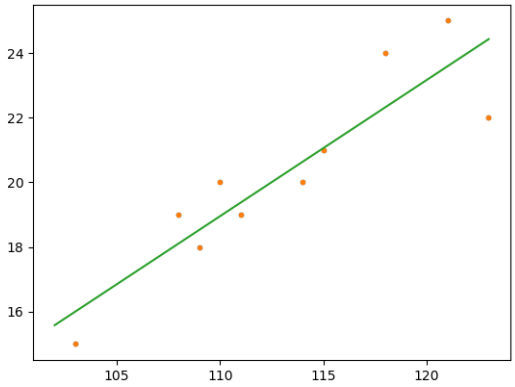
> Ca a déjà été fait au chapitre 7 dans une situation analogue. À vous de jouer !

Exercice 7 (★)
1. Calculer le coefficient de corrélation linéaire pour notre série statistique double. Cela correspond-il à ce que vous vous attendiez ?
2. Estimer graphiquement le poids d'un enfant de 6 ans qui mesure 1m20.

> 1. On calcule le coefficient de corrélation linéaire de notre série statistique double grâce à la commande :
>
> ```
> print(np.corrcoef(x,y)[0,1])
> ```

On obtient `0.9001375`. La corrélation linéaire entre les séries Y et Y est donc forte. On s'y attendait puisqu'une dépendance linéaire forte était prévisible étant donné l'allure du nuage de points.

2. Étant donné cette corrélation linéaire forte, notre modèle de régression linéaire se justifie. Ce qui nous permet de faire des analyses prédictives à partir du graphe :



On peut ainsi estimer qu'un enfant de 6 ans mesurant 1m20 pèse 23kg.

Exercice 8 (★ - Sensibilités aux valeurs extrêmes)

Le petit Badr arrive en retard en classe ce matin-là. Il prétend mesurer 105 cm et peser 27 kg.

1. Calculer le coefficient de corrélation linéaire pour la série statistique double associée aux 11 enfants. Comparer cette valeur avec celle obtenue dans l'exercice précédent.
2. Représenter le nouveau nuage de points, en distinguant les points correspondants aux 10 enfants avec celui représentant Badr.
3. Tracer la droite des moindres carrés correspondant à la nouvelle série statistique. Que constatez vous ?

1. On calcule le nouveau coefficient de corrélation linéaire :

```
x1 = np.array([121, 123, 108, 118, 111, 109, 114, 103, 110, 115, 105])
y1 = np.array([25, 22, 19, 24, 19, 18, 20, 15, 20, 21, 27])

print(np.corrcoef(x1,y1)[0,1])
```

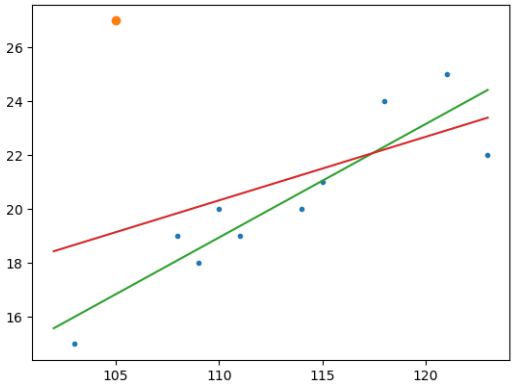
On obtient cette fois un coefficient de corrélation linéaire de `0.4412470`. Avec ce dernier point, le coefficient de corrélation linéaire a été divisé par 2 ! On peut douter de ce qu'affirme Badr. On notera aussi qu'un seul point très éloigné de la droite des moindres carré a une grosse influence sur le coefficient de corrélation linéaire.

2. et 3. On utilise pour cela les instructions suivantes :

```
# nuage de point
plt.plot(x,y,".")
plt.plot([105],[27],"o")

```

```
# regression pour 10 élèves
a,b = np.polyfit(x,y,1)
u = np.linspace(102,123,2)
plt.plot(u, a*u+b)

# regression pour 11 élèves
a1,b1 = np.polyfit(x1,y1,1)
u = np.linspace(102,123,2)
plt.plot(u, a1*u+b1)

plt.show()
```

On obtient le graphe suivant, avec en vert la régression linéaire pour les 10 élèves, et en rouge celle pour les 11 élèves :



On remarque que cette valeur erronée a beaucoup fait dévier la droite des moindres carrés, et rend la régression linéaire inadaptée et inutilisable pour faire des prévisions éventuelles.

Exercice 9 (★★ - Indépendance et corrélation linéaire)

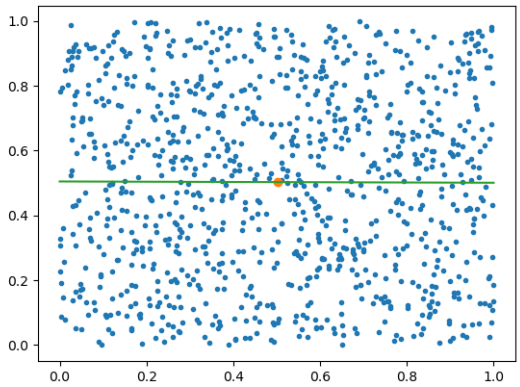
1. (a) Créer deux vecteurs `x` et `y` contenant chacun 1000 nombres tirés au hasard suivant une loi uniforme sur $[0,1]$.
   (b) Représenter le nuage de points ainsi que le point moyen et la droite des moindres carrés (à l'aide de la commande `np.polyfit`). Qu'en pensez vous ?
   (c) Calculer le coefficient de corrélation linéaire. Comment expliquer le résultat obtenu ?
2. On pose à présent `x = 2*rd.random(100)-1` et `y = x**2`.
   (a) Représenter le nuage de points associés à ces séries statistiques ainsi que la droite des moindres carrés (à l'aide de la commande `np.polyfit`). Qu'en pensez vous ?
   (b) Calculer le coefficient de corrélation linéaire. Les variables `x` et `y` sont-elles indépendantes ?

1. On utilise le code suivant :

```
# nuage de points
x = rd.random(1000)
y = rd.random(1000)
plt.plot(x,y,".")

```

```
# point moyen
xm = np.mean(x)
ym = np.mean(y)
plt.plot(xm,ym,"o")

# regression linéaire
a,b = np.polyfit(x,y,1)
u = np.array([0,1])
plt.plot(u,a*u+b)

plt.show()

# coeff de correl lin
print(np.corrcoef(x,y)[0,1])
```

On obtient le graphe suivant :



La droite de régression linéaire est quasi horizontale. Elle ne semble pas du tout adaptée pour approximer notre nuage de points, il ne semble pas y avoir de corrélation linéaire entre les deux séries statistiques.

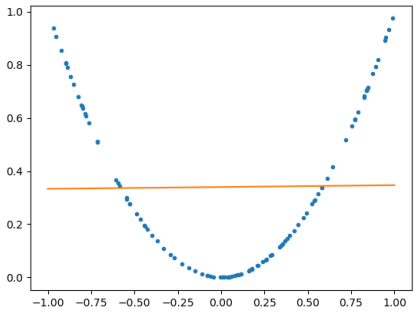
Le coefficient de corrélation linéaire obtenu est `0.0181605`. Il confirme l'indépendance linéaire qu'on avait constaté auparavant. Il explique également pourquoi la droite de régression linéaire est horizontale. Rappelons en effet que le coefficient directeur de cette droite fait intervenir la covariance, qui dans notre cas est voisin de 0.

Tous ces résultats ne sont pas étonnants : les séries statistiques `x` et `y` ont été choisies de manière indépendante l'une de l'autre, la fonction `rand` générant des réalisations indépendantes de la loi $\mathscr{U}([0,1])$. Elles sont en particulier linéairement indépendantes, ce qui implique que leur coefficient de corrélation linéaire est proche de 0.

2. On adapte le code précédent :

```
# nuage de points
x = 2*rd.random(100)-1
y = x**2
plt.plot(x,y,".")

# regression linéaire
a,b = np.polyfit(x,y,1)
u = np.array([-1,1])
plt.plot(u,a*u+b)

plt.show()

# coeff de correl lin
print(np.corrcoef(x,y)[0,1])
```

On obtient le graphe suivant :



La droite de régression linéaire est là aussi quasi horizontale (ce qui indique un coefficient de corrélation linéaire proche de 0) et ne semble également pas du tout adaptée pour approximer notre nuage de points. Le coefficient de corrélation linéaire obtenu le confirme : il est de `0.0131898`.

On peut conclure que les séries statistiques `x` et `y` sont **linéairement** indépendantes. Mais attention à ne pas conclure à l'indépendance de ces séries : elles ne le sont clairement pas puisque `y = x.^2`.

Exercice 10 (★)

Considérons les séries statistiques `x` et `y` suivantes :

```
x = np.arange(1,51)
y = np.log(x)+rd.normal(0,1/2,50)
```

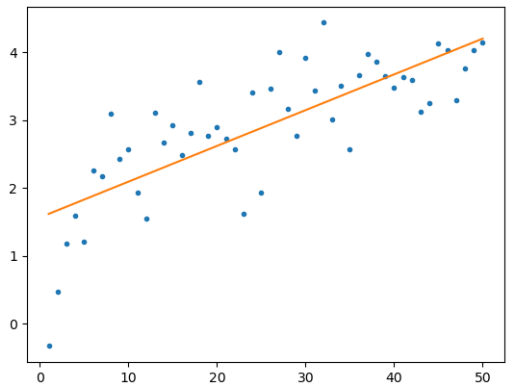
1. Représenter le nuage de points associé.
2. (a) Calculer le coefficient de corrélation linéaire. Vous semble-t-il bon ?
   (b) Déterminer l'équation de la droite de régression de `y` par rapport à `x`.
   (c) Superposer la droite de régression linéaire au nuage de points.
3. Vérifier que le nuage de points se superpose bien avec courbe de la fonction $f : x \mapsto \ln(x)$.
4. Étudier la corrélation linéaire de `y` par rapport à `log(x)`.

1. et 2. On utilise le code suivant :

```
# nuage de points
x = np.arange(1,51)
y = np.log(x)+rd.normal(0,1/2,50)
plt.plot(x,y,".")

# coeff de correl lin
print(np.corrcoef(x,y)[0,1])

# regression linéaire
a,b = np.polyfit(x,y,1)
u = np.array([1,50])
plt.plot(u,a*u+b)

plt.show()
```

On obtient `0.771325` comme coefficient de corrélation linéaire, en deçà de la barre des 0.9. La corrélation linéaire entre les deux séries statistiques n'est donc pas bonne.

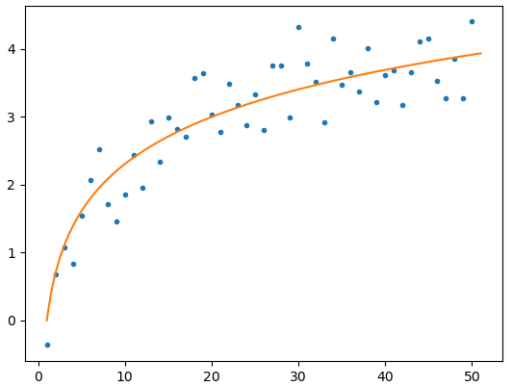
On obtient la représentation suivante du nuage de points et de la droite de régression linéaire associée :



3. Avec le code :

```python
# nuage de points
x = np.arange(1,51)
y = np.log(x)+rd.normal(0,1/2,50)
plt.plot(x,y,".")

# courbe du log
u = np.linspace(1,51,100)
plt.plot(u,np.log(u))

plt.show()

# coeff de correl lin
print(np.corrcoef(np.log(x),y)[0,1])
```

on obtient :



Le nuage de point se superpose effectivement bien avec la courbe représentative du logarithme.

4. En exécutant la commande `print(np.corrcoef(np.log(x),y)[0,1])`, on obtient `0.9179` comme coefficient de corrélation linéaire. C'est supérieur à 0.9, ce qui confirme une forte corrélation linéaire entre `log(x)` et `y`.

Exercice 11 (★ - Corrélation et causalité)

Une bonne corrélation entre deux séries de données ne signifie pas pour autant qu'il existe un lien de cause à effet entre les deux. À titre d'exemple, considérons la série statistique suivante :

Année	1996	1997	1998	1999	2000
Morts	15.85	15.7	15.39	15.32	14.85
Importations de citrons	230	280	360	410	525

Ce tableau donne le nombre de morts (pour un million d'habitants) sur les autoroutes américaines, ainsi que le nombre de tonnes de citrons mexicains importés aux États-Unis de 1996 à 2000.

Calculer le coefficient de corrélation linéaire pour cette série double. En déduisez vous une information pertinente ?

On obtient avec le code :

```
m = np.array([15.85, 15.7, 15.39,
15.32, 14.85])
i = np.array([230, 280, 360, 410,
525])
print(np.corrcoef(m,i)[0,1])
```

un coefficient de corrélation linéaire de `-0.9951611`. Ces deux séries statistiques sont donc fortement corrélées linéairement. Pourtant il semble bien difficile d'envisager un lien rationnel entre le nombre de morts sur les routes et les importations de citrons mexicains. Le travail du mathématicien s'arrête donc au calcul de la corrélation, l'exploitation de ces chiffres nécessite d'autres compétences !
