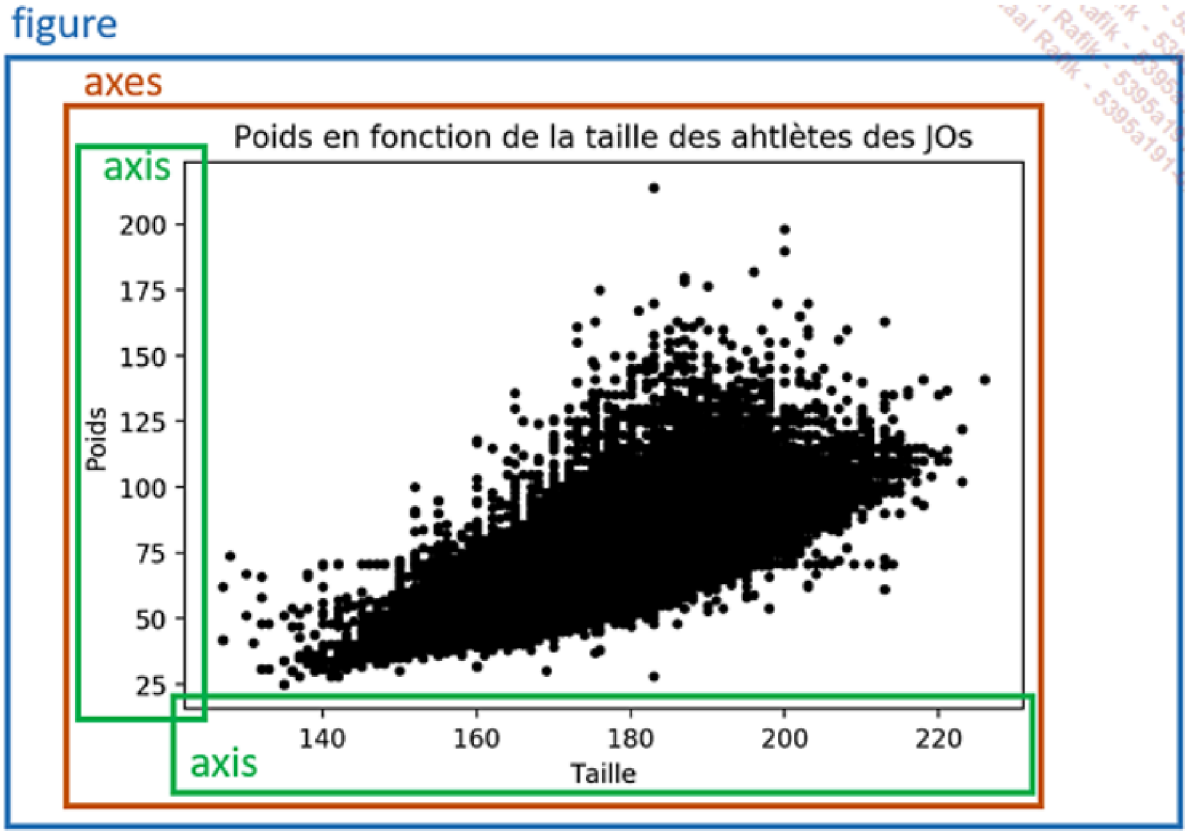
Matplotlib & Seaborn

Customiser ses graphiques

Organisation des figures avec Matplotlib

Voici comment est organisée une figure selon Matplotlib :



Organisation des figures avec Matplotlib

- L'**objet figure** peut être représenté comme la feuille blanche sur laquelle on va tracer des graphiques.
- Cette figure peut contenir un ou plusieurs **objets axes** (qui signifie axes, au pluriel).Ceux-ci représente un graphique, c'est-à-dire qu'il contient les deux axes x et y.
- Les axes x et y eux sont des **objets axis** (qui signifie axe, au singulier).

Organisation des figures avec Matplotlib

- Chaque objet axes peut avoir un titre principal et un titre lié aux axis.

- Ainsi, un objet axes est en fait une partie d'une figure et il peut y avoir plusieurs axes, plusieurs graphiques, par figure.

Organisation des figures avec Matplotlib

- Les objets axis permettent de fixer les limites du graphique et génèrent ce qu'on appelle les ticks (marques en français) qui permettent de marquer les axes x et y.
  - ⇒Sur l'exemple, les ticks pour l'axe x représentant la taille sont 140, 160, 180, 200 et 220.
  - ⇒Ces ticks peuvent être définis soit par le dataframe, soit par nous-mêmes, manuellement, lors de l'écriture du code permettant de générer la figure.

Premier graphique : Nuage de points

Au sein de la librairie Matplotlib, c'est en réalité le module Pyplot que nous allons utiliser majoritairement, qui permet de créer les graphiques et de les customiser.

La syntaxe générale pour tracer un graphique est la suivante :

```
import matplotlib.pyplot as plt
plt.figure()
subplot(111)
plt.plot(x,y)
plt.show()
```

Premier graphique : Nuage de points

- Ici, nous importons le module **Pyplot** de la librairie **Matplotlib** et nous le renommons avec l'alias plt, ce qui est un classique.

- Puis on utilise la fonction **figure()**, qui permettra de créer une nouvelle figure, encore vierge à ce moment-là. Cette commande n'est pas nécessaire, en utilisant la commande plt.plot().

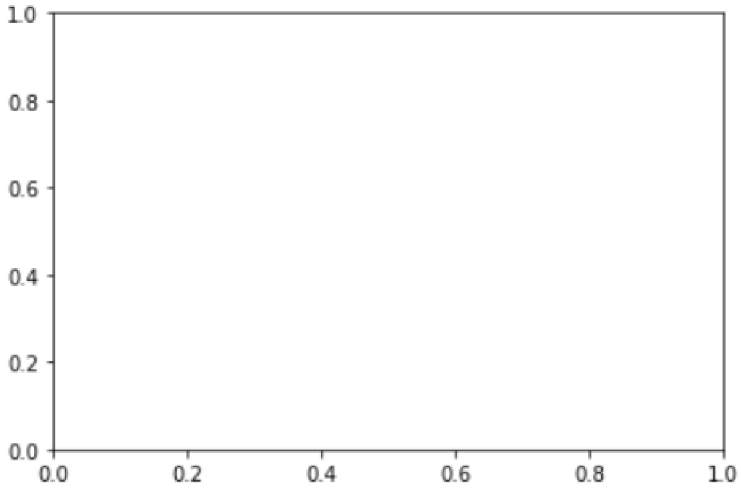
- Puis vient la commande **subplot()**. Dans l'exemple, le code subplot(111) permet de créer une grille de 1*1 puis de placer le graphique à la position 1. Par défaut, Pyplot crée un subplot(111).

- La fonction subplot crée l'objet axes, unique ici, puisqu'on a défini qu'on ne voulait qu'un seul graphique sur la figure (grille 1*1).

Premier graphique : Nuage de points

- Ensuite, la fonction **plot()** crée les objets axis et trace les points, en prenant les valeurs qu'on lui donne en x (abscisses) et en y (ordonnées).

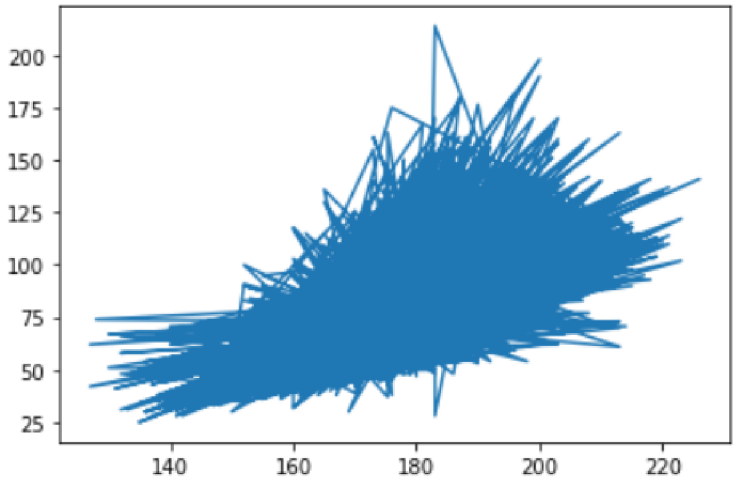
- Enfin, la fonction **show()** permet d'afficher le graphique courant, ici le graphique généré avec la fonction plot().



Premier graphique : Nuage de points

```
import matplotlib.pyplot as plt
import pandas as pd
donnees=pd.read_csv("..\\Jeux_de_donnees\\120-years-of-olympic-history-athletes-and-results1.csv",
                    index_col=[1])

plt.figure()
plt.subplot(111)
plt.plot(donnees["Height"],
         donnees["Weight"])
plt.show()
```



Premier graphique : Nuage de points

- Par défaut, la fonction plot() de Pyplot va tracer des lignes entre tous les points, ce qui n'est pas nécessaire ici.

- Il existe une fonction de Pyplot, nommée **scatter()**, qui permet de tracer des nuages de points sans les relier.

- Toutefois, avec la fonction plot(), il est possible d'indiquer de ne pas relier les points entre eux, grâce à l'option **linestyle**.

- Il est également possible de changer la forme des points grâce à l'option **marker** et la couleur des points grâce à l'option **color**.

Premier graphique : Nuage de points

- Pour les différents markers voir : https://matplotlib.org/3.1.3/api/markers_api.html#module-matplotlib.markers (et l'option markersize pour la taille des points).

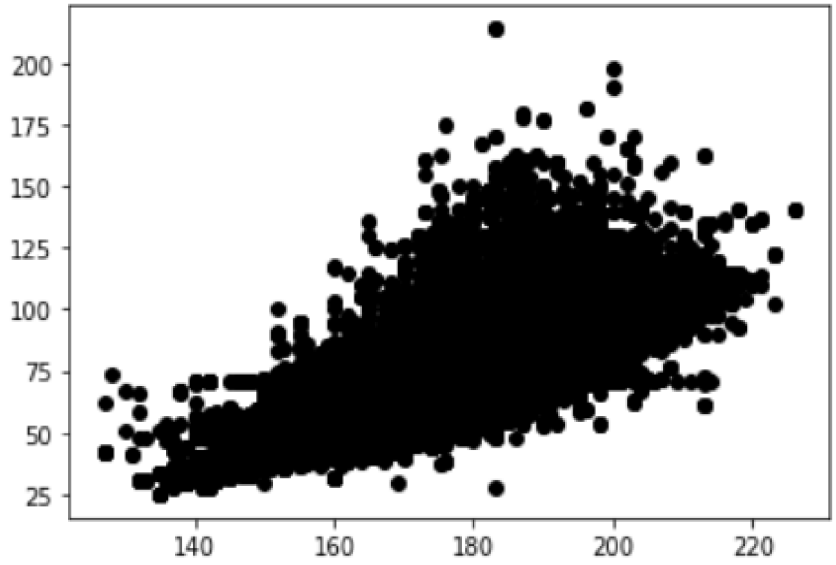
- Voici la liste des couleurs basiques dans Matplotlib : b = bleu, g = vert, r = rouge, c = cyan, m = magenta, y = jaune, k = : noir, w = : blanc.

- Enfin, pour qu'il n'y ait pas de ligne tracée entre tous les points, on utilise l'option linestyle en lui mettant une valeur nulle ' '.

- Il est aussi possible de modifier le type de ligne entre les points, par exemple avec linestyle="--", qui permet de tracer des lignes en pointillés.

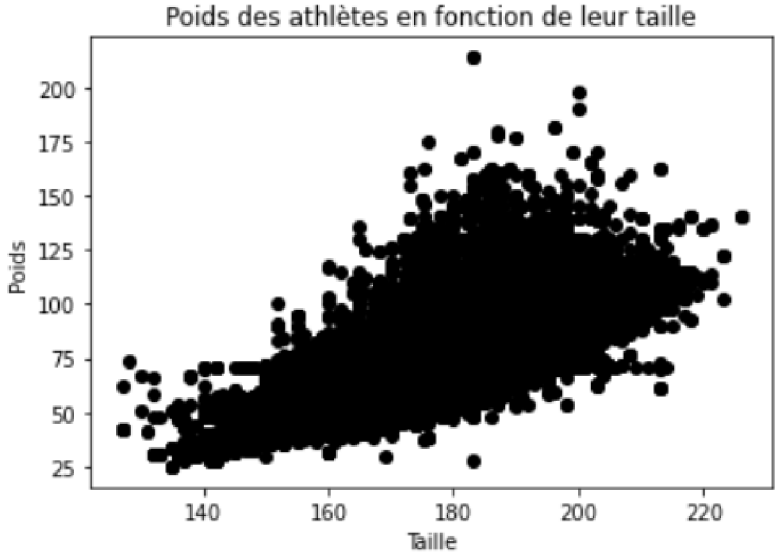
Premier graphique : Nuage de points

```
plt.figure()
plt.subplot(111)
plt.plot(donnees["Height"],
         donnees["Weight"],
         marker='o',
         color="k",
         linestyle='')
plt.show()
```



Ajouter un titre principal et des labels aux axes du nuage de points

```
import matplotlib.pyplot as plt
plt.figure()
plt.subplot(111)
plt.plot(donnees["Height"],donnees["Weight"],
         marker='o', color="black", linestyle='')
plt.title("Poids des athlètes en fonction de leur taille")
plt.ylabel("Poids")
plt.xlabel("Taille")
plt.show()
```



Ajouter un titre principal et des labels aux axes du nuage de points

- Ici, on utilise la fonction title pour ajouter un titre principal.
- La fonction ylabel pour ajouter un titre à l'axe y (axe des ordonnées).
- La fonction xlabel pour ajouter un titre à l'axe x (axe des abscisses). title, ylabel et xlabel étant des fonctions.
- Il est possible de les paramétrer finement, comme par exemple changer la taille du texte, la couleur, etc (option color, size).
- Pour les couleurs voir : https://matplotlib.org/3.1.3/gallery/color/named_colors.html#sphx-glr-gallery-color-named-colors-py

Enregistrer son graphique

Lorsque votre graphique est terminé, il est possible de l'enregistrer dans un fichier pour ensuite l'utiliser dans un rapport ou le partager avec d'autres personnes.

Pour enregistrer un graphique dans un fichier, il faut utiliser la fonction savefig() de Pyplot, dont la syntaxe est la suivante :

```
import matplotlib.pyplot as plt
plt.figure()
plt.subplot(111)
plt.plot(donnees["Height"],donnees["Weight"],
         marker='o', color="black", linestyle='')
plt.title("Poids en fonction de la taille des ahtlètes des JOs")
plt.ylabel("Poids")
plt.xlabel("Taille")
plt.savefig("..\\Jeux_de_donnees\\athlètes.png", dpi=300, format="png")
plt.close()
```

Enregistrer son graphique

- **format** : vous pouvez mettre une extension connue à votre fichier (jpg, jpeg, png, pdf, svg, eps…).

- **dpi** : permet de spécifier la résolution de l'image. Plus la résolution est élevée, plus l'image générée sera de qualité.

- **transparent** : pour avoir un fond transparent, il est possible de le spécifier avec l'option transparent=True.

La fonction plt.close() permet de fermer le graphique et de le faire disparaître de la mémoire de votre ordinateur. Cela fait partie des bonnes pratiques après avoir enregistré un graphique.

Taille de la fenêtre graphique et la résolution de son graphique

Il n'est pas possible de spécifier la taille du graphique lors de l'enregistrement de celui-ci dans un fichier. Il faut le faire lors de la création de la figure, en spécifiant des options dans la fonction figure().

```
import matplotlib.pyplot as plt
plt.figure(figsize=[5,5], dpi=150, facecolor='grey')
plt.subplot(111)
plt.plot(donnees["Height"],donnees["Weight"],
         marker='o', color="black", linestyle='', markersize=1.8)
plt.title("Poids en fonction de la taille des ahtlètes des JOs")
plt.ylabel("Poids")
plt.xlabel("Taille")
plt.show()
```

Taille de la fenêtre graphique et la résolution de son graphique

- La première option, **figsize**, permet de définir la taille de l'image (en inches, pouces en français). La valeur à gauche correspond à la largeur et la valeur à droite correspond à la hauteur de la figure.

- Ensuite, on peut définir l'option **dpi**, qui par défaut, est à 100.

- Enfin, il est possible de changer la couleur de fond du graphique, avec l'option **facecolor**, par défaut en blanc (w = white).

Tracer plusieurs courbes sur un même graphique

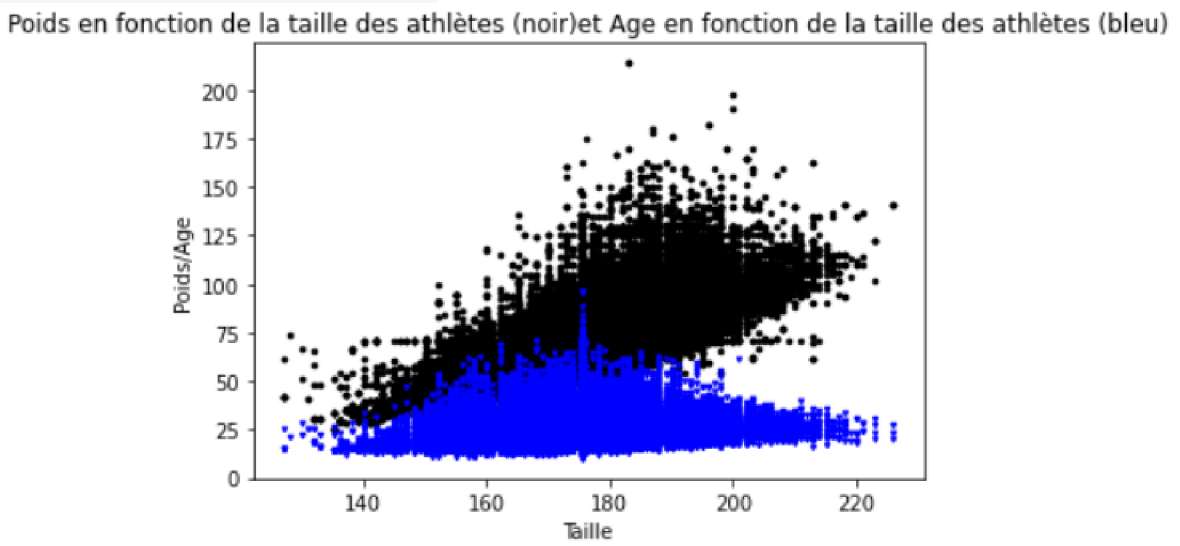
```
import numpy as np

plt.figure()
plt.subplot(111)

plt.plot(donnees["Height"],donnees["Weight"],
         marker='o', color="black", linestyle='', markersize=2.5)
plt.plot(donnees["Height"],donnees["Age"],
         marker='v', color="blue", linestyle='', markersize=2.5)

plt.title("Poids en fonction de la taille des athlètes (noir)\
et Age en fonction de la taille des athlètes (bleu)")

plt.ylabel("Poids/Age")
plt.xlabel("Taille")
plt.show()
```



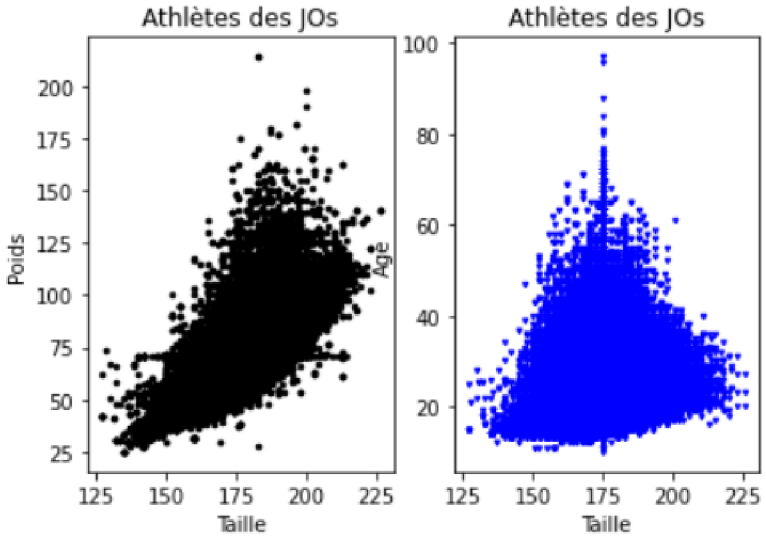
```
plt.figure()

plt.subplot(121)
plt.plot(donnees["Height"],donnees["Weight"],
         marker='o', color="black", linestyle='', markersize=2.5,)
plt.title("Athlètes des JOs")
plt.ylabel("Poids")
plt.xlabel("Taille")

plt.subplot(122)
plt.plot(donnees["Height"],donnees["Age"],
         marker='v', color="blue", linestyle='', markersize=2.5)

plt.title("Athlètes des JOs")
plt.ylabel("Age")
plt.xlabel("Taille")

plt.show()
```



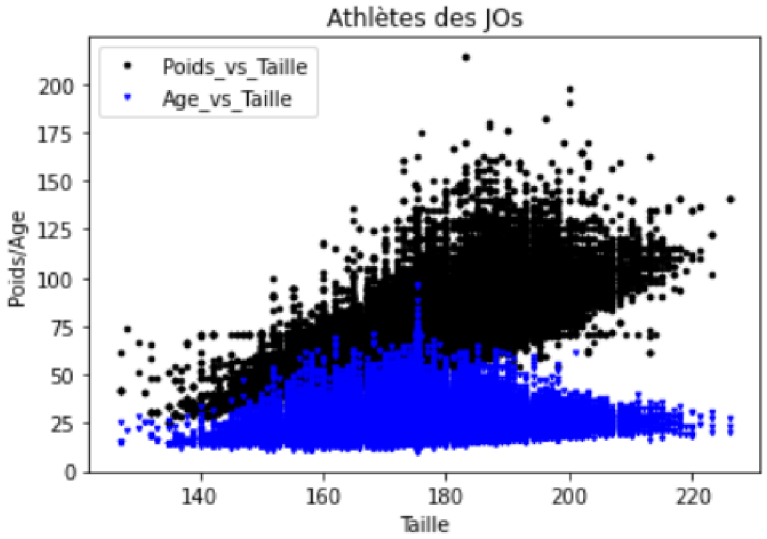
Ajouter une légende

```
plt.figure()
plt.subplot(111)

plt.plot(donnees["Height"],donnees["Weight"],
         marker='o', color="black", linestyle='', markersize=2.5, label='Poids_vs_Taille')
plt.plot(donnees["Height"],donnees["Age"],
         marker='v', color="blue", linestyle='', markersize=2.5, label='Age_vs_Taille')

plt.title("Athlètes des JOs")
plt.ylabel("Poids/Age")
plt.xlabel("Taille")

plt.legend(loc='upper left')
plt.show()
```



Annoter son graphique avec du texte

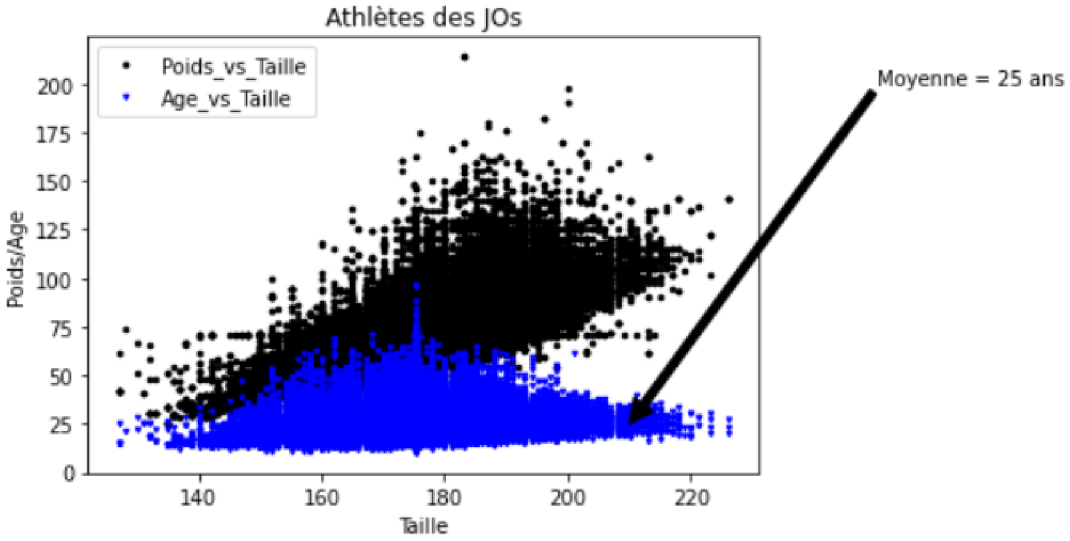
```
plt.figure()
plt.subplot(111)

plt.plot(donnees["Height"],donnees["Weight"],
         marker='o', color="black", linestyle='', markersize=2.5, label='Poids_vs_Taille')
plt.plot(donnees["Height"],donnees["Age"],
         marker='v', color="blue", linestyle='', markersize=2.5, label='Age_vs_Taille')

plt.annotate("Moyenne = 25 ans", xy=(210,25), xytext=(250,200),arrowprops=dict(facecolor='black'))

plt.title("Athlètes des JOs")
plt.ylabel("Poids/Age")
plt.xlabel("Taille")

plt.legend(loc='upper left')
plt.show()
```



Seaborn

Esthétique des figures

Paramétrer les styles Seaborn

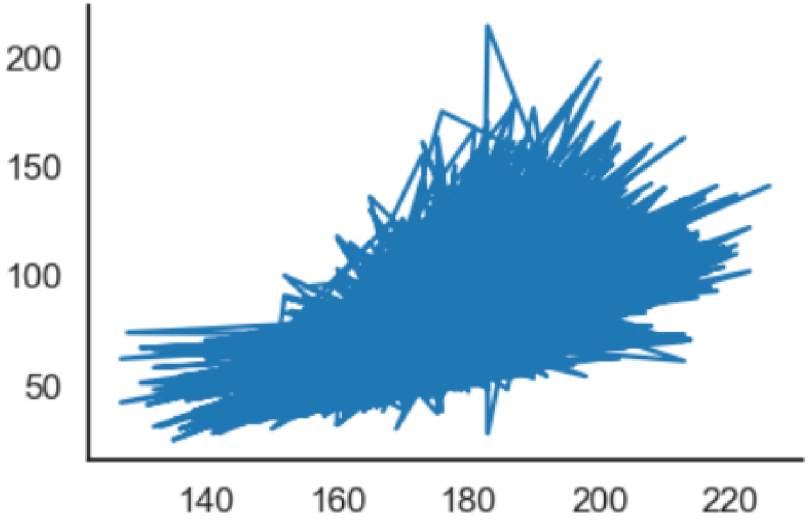
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_style("white")
sns.set_context("talk")

plt.figure()
plt.subplot(111)
plt.plot(donnees["Height"],donnees["Weight"])

sns.despine()

plt.show()
```



Paramétrer les styles Seaborn

- Ici, on utilise la fonction **set_style()** de Seaborn, qui permet de définir le thème de la figure à utiliser : darkgrid, whitegrid, dark, white, ticks.

- Seaborn fournit une fonction appelée **despine()**, qui permet de supprimer les axes du haut et de droite. Il est possible de supprimer l'ensemble des axes avec le code sns.despine(left=True, bottom=True)

- Il est possible de dire à Seaborn dans quel contexte les images seront utilisées avec **set_context()**. En fonction de ce contexte, elle adaptera la taille des courbes et des axes :
  - **paper** représente le contexte le plus petit, par exemple pour créer une figure pour une publication scientifique
  - **poster** représente le contexte le plus grand, par exemple pour créer une figure à mettre sur un poster scientifique pour présenter ses résultats.
  - **notebook** est utilisée pour une figure à intégrer dans un notebook.
  - **talk** permet de dire à Seaborn que la figure sera placée dans un fichier PowerPoint pour une présentation orale de ses résultats.

Palettes de couleurs

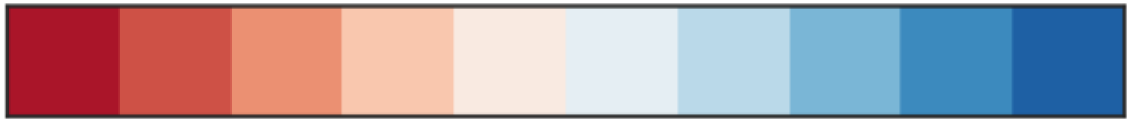
```
sns.palplot(sns.color_palette("dark",15))
```

```
sns.palplot(sns.color_palette("Paired"))
```

```
sns.palplot(sns.color_palette("OrRd",10))
sns.palplot(sns.color_palette("Blues",10))
```

Palettes de couleurs

```
sns.palplot(sns.color_palette("RdBu", 10))
```



```
palette= sns.color_palette(["#9b59b6","#3498db","#e74c3c","#34495e","#2ecc71"])
sns.set_palette(palette)
sns.palplot(palette)
```

- Pour les palettes de couleurs voir : https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/34087/versions/2/screenshot.jpg

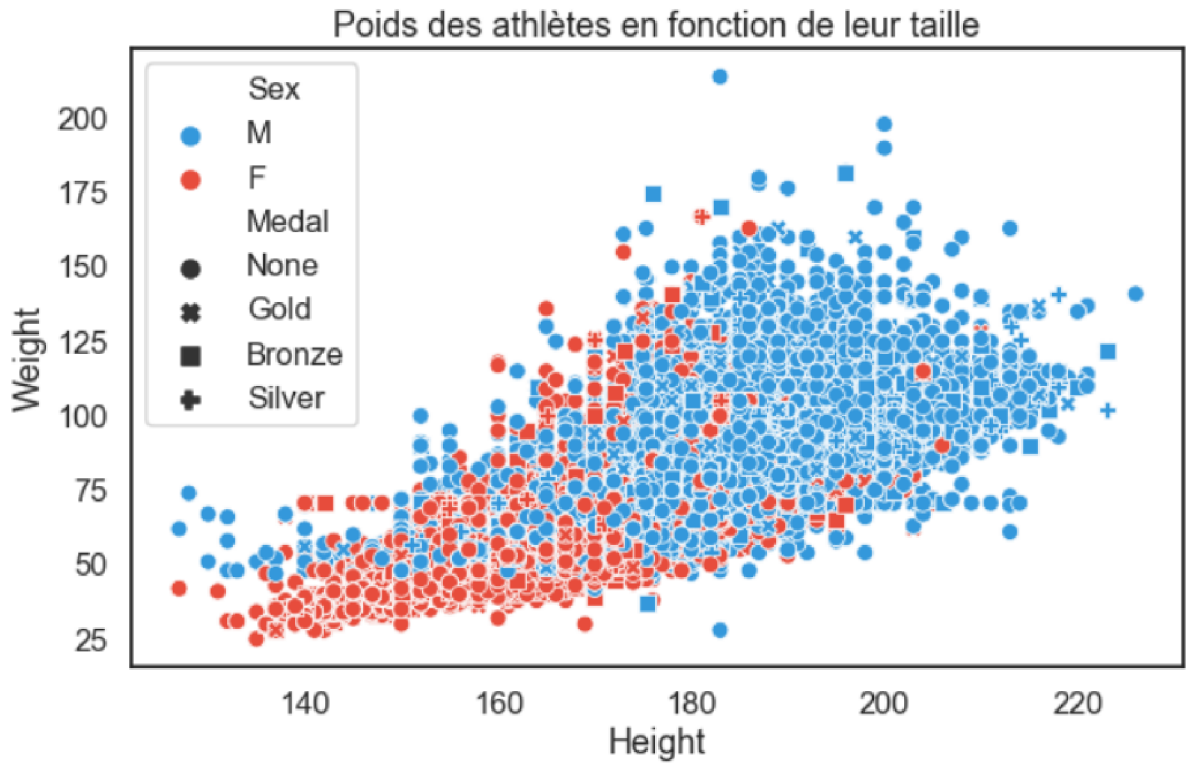
Nuage de points (scatterplot)

```
palette= sns.color_palette(["#3498db","#e74c3c"])

sns.set_palette(palette)
sns.set_style("white")
plt.figure(figsize=[10,6])
plt.subplot(111)

sns.scatterplot(data=donnees, x='Height', y='Weight',hue='Sex',style='Medal')

plt.title("Poids des athlètes en fonction de leur taille")
plt.legend(loc='upper left')
plt.show()
```



Nuage de points (scatterplot)

- L'option **data** de cette fonction attend un dataframe. En donnant un dataframe à Seaborn, il est alors au courant de l'ensemble des variables de ce dataframe et du noms des colonnes. Ainsi, pour tracer x et y, il suffit de donner le nom de la colonne qu'on souhaite tracer.

- **hue** attend une variable en entrée (un nom de colonne) et donnera une couleur pour chaque modalité d'une variable qualitative ou pour chaque valeur d'une variable quantitative.
- **style** attend aussi une variable, et donnera un style de point différent à chaque modalité/valeur de la variable.
- **size** donnera une taille de point différente selon les modalités/valeurs de la variable.

Autres graphiques

- Pointplot
- stripplot
- Boxplots
- countplot