femto-st SCIENCES & TECHNOLOGIES

Analyse et visualisation avancée des données

N. Sirri

UFC UNIVERSITÉ DE FRANCHE-COMTÉ cnrs ensmm utbm université de technologie Belfort-Montbéliard

Objectif de ce cours

- Utiliser Python pour analyser des données.
- Manipuler et transformer les données avec Numpy et Pandas.
- Intégrer, nettoyer et traiter des données avec Pandas.
- Créer et personnaliser des graphiques avec Matplotlib, Seaborn.

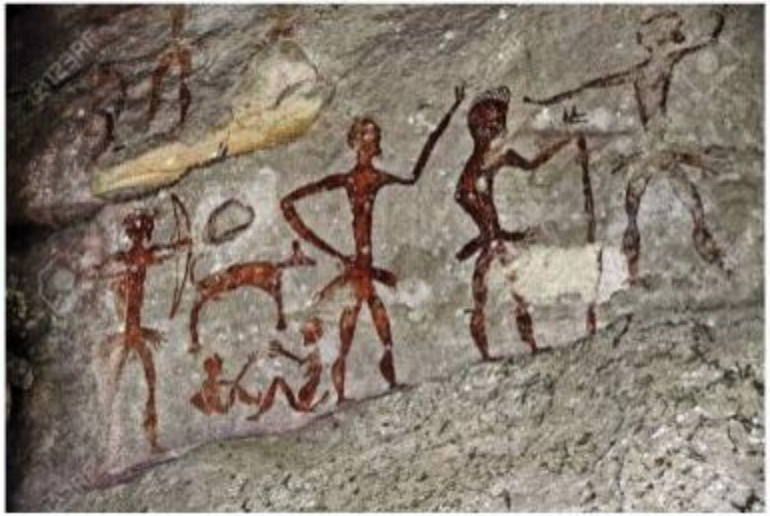
Introduction à la Manipulation et à la Visualisation des données

Objectifs de visualisation

- Structurer et consigner l'information
- Analyser les données afin d'étayer un raisonnement
- Valider des hypothèses à partir des résultats obtenus
- Communiquer efficacement les idées et les conclusions

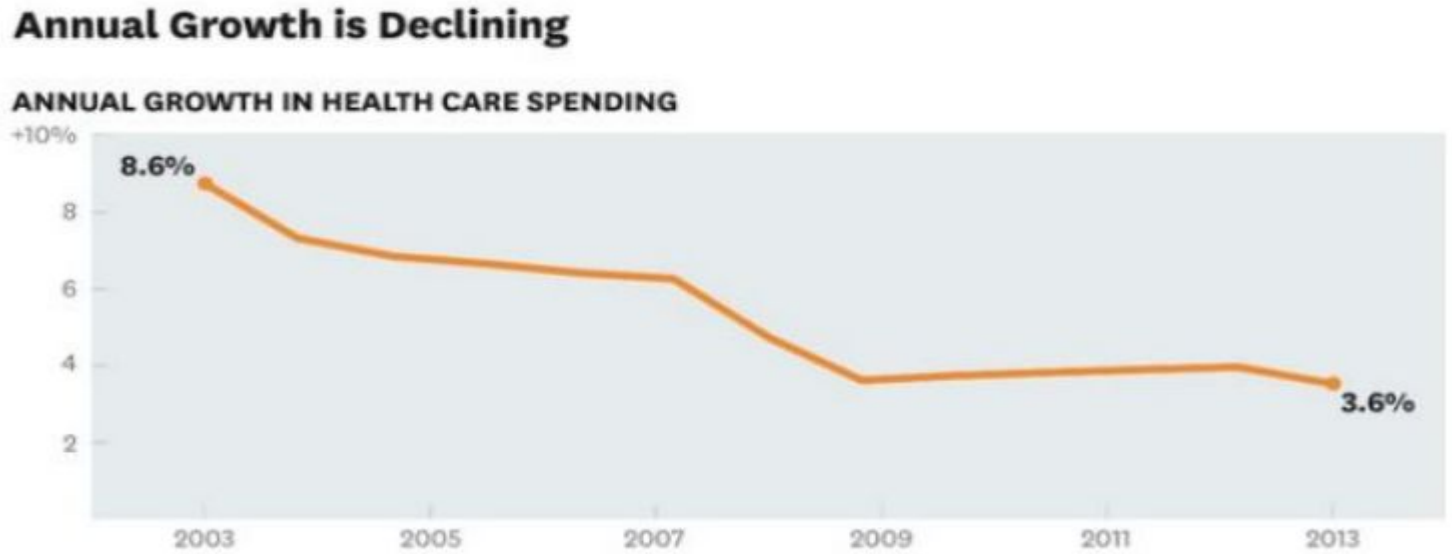
Pourquoi visualiser ?

Structurer et enregistrer l'information



Pourquoi visualiser ?

Analyser et Communiquer l'information



Pourquoi visualiser ?

Analyser les données

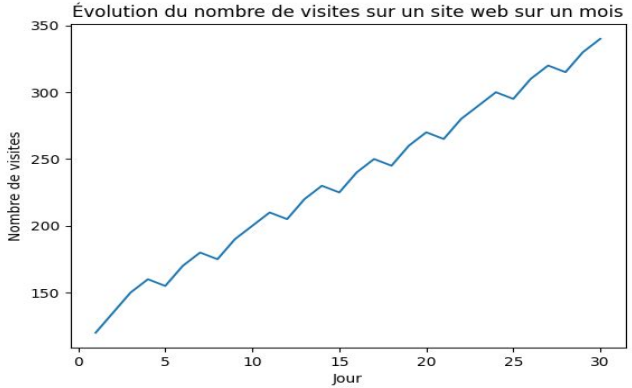
2020 US Elections (NYTimes)

Types de graphiques

- Graphiques en lignes (Line plots)
- Nuages de points (Scatter plots)
- Diagrammes en barres (Bar plots)
- Histogrammes (Histograms)
- Diagrammes en boîtes à moustaches (Box plots)

Graphiques en lignes

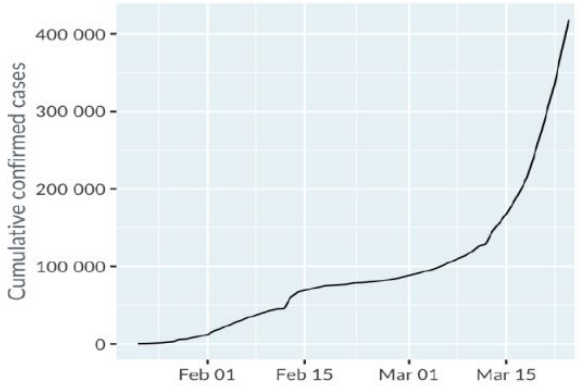
- Utilisés pour représenter des données numériques
- Permettent de visualiser des tendances
- Comparent deux ou plusieurs variables dans le temps
- Peuvent être utilisés pour effectuer des prédictions



Quand utiliser un graphique en lignes ?

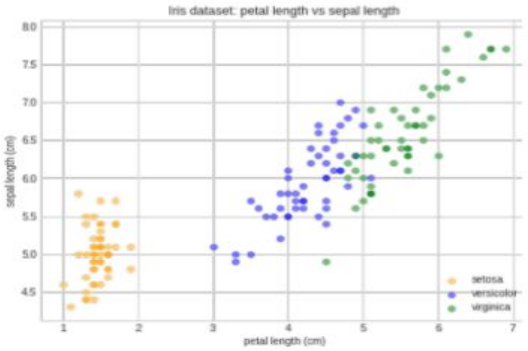
- Lorsque vous disposez de deux variables continues
  -> Données continues : généralement numériques
  Exemples : tailles, températures, revenus

- Lorsque les observations consécutives sont liées entre elles ; le plus souvent, l'axe des abscisses représente des dates ou des instants de temps



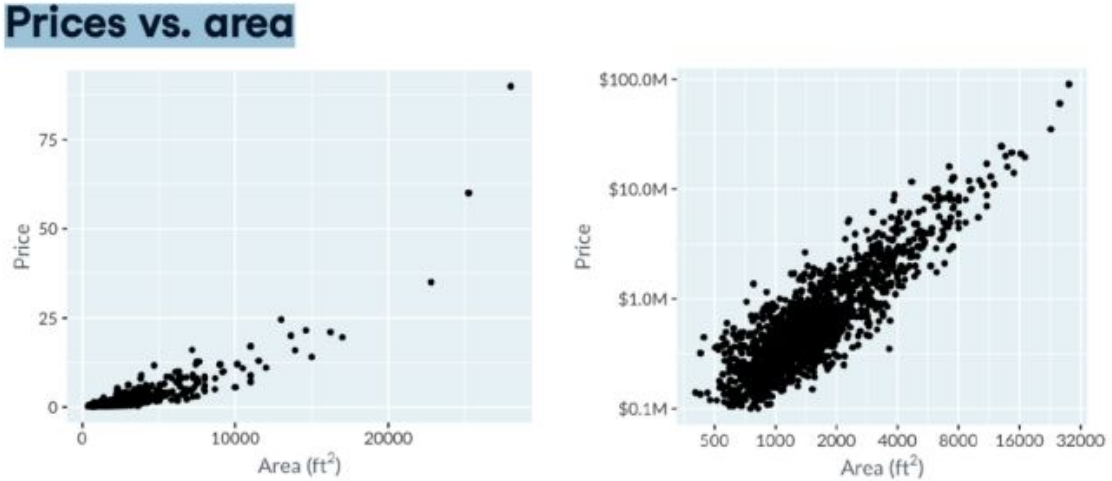
Nuages de points

- Étudier les relations entre des valeurs quantitatives
- Utilisés pour visualiser la relation entre deux variables numériques
- Utilisés pour mettre en évidence la corrélation au sein d'un jeu de données volumineux

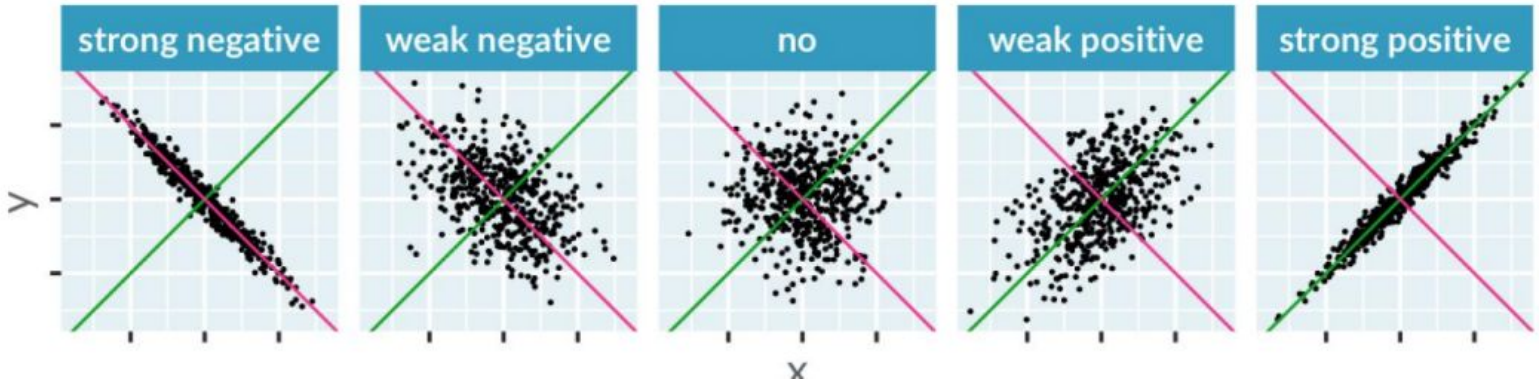


Quand utiliser un nuage de points ?

- Lorsque vous disposez de deux variables continues
- Lorsque vous souhaitez analyser la relation entre ces deux variables



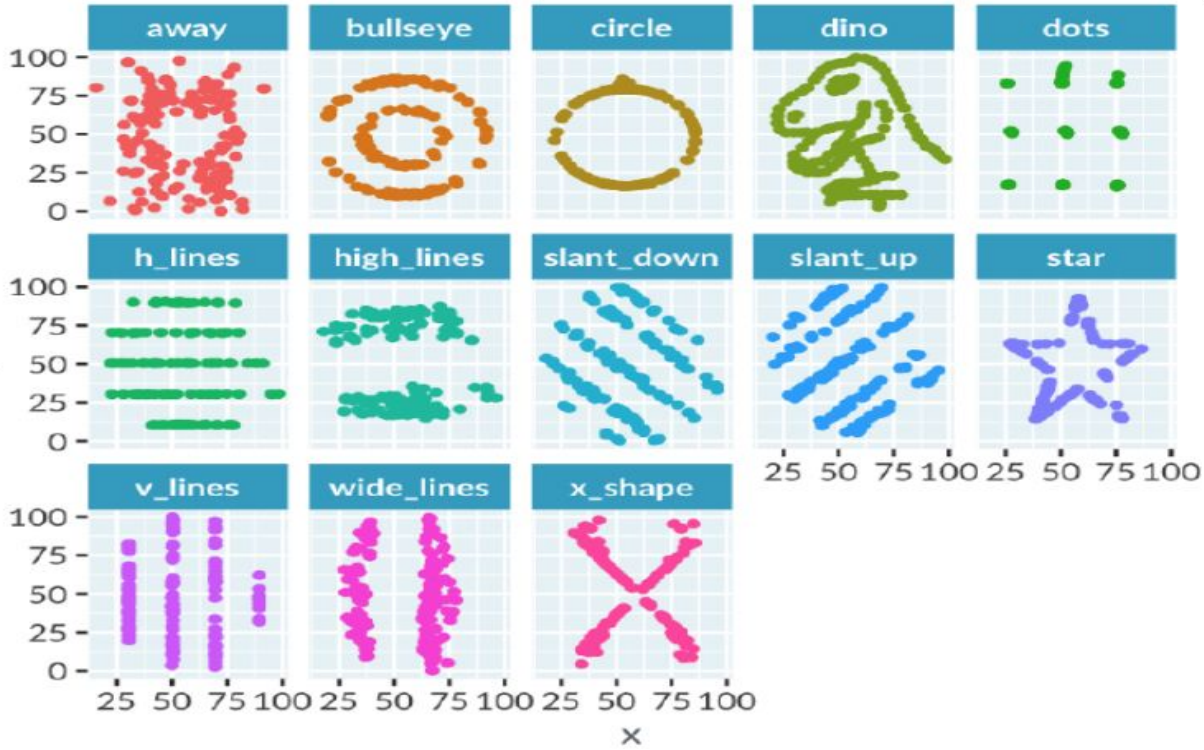
Correlation !



$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

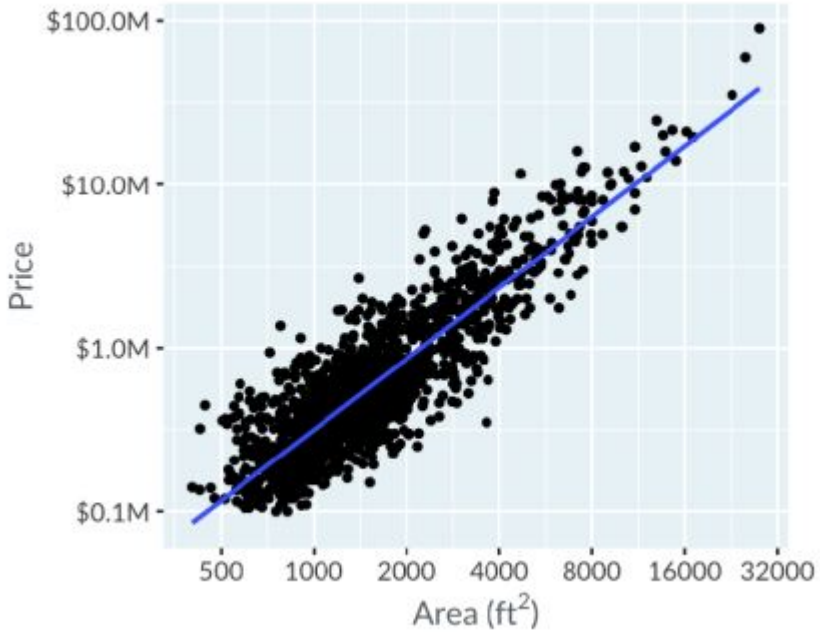
- x_i, y_i : valeurs observées de X et Y
- $\bar{x}, \bar{y}$: moyennes de X et Y
- n : nombre d'observations
- $r \in [-1, 1]$

Parfois, la corrélation n'est pas utile



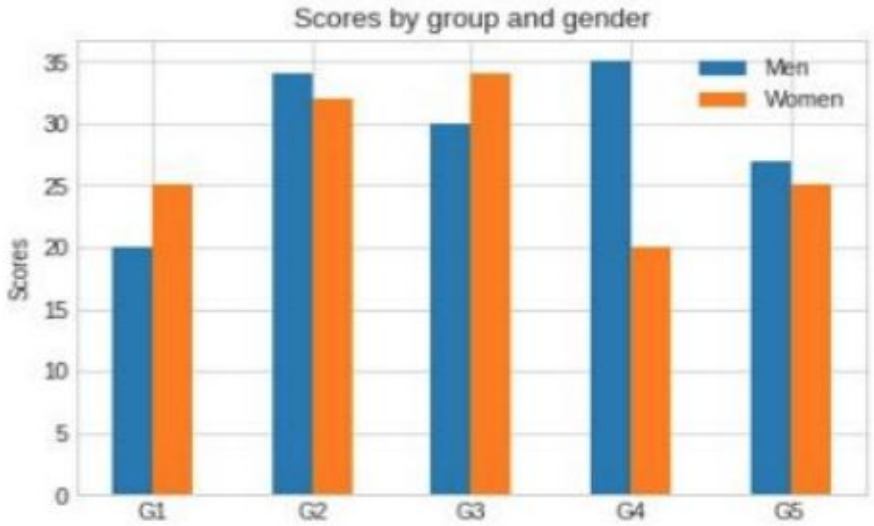
femto-st SCIENCES & TECHNOLOGIES

Ajout de lignes de tendance



Diagrammes en barres

- Comparer les données entre différentes catégories

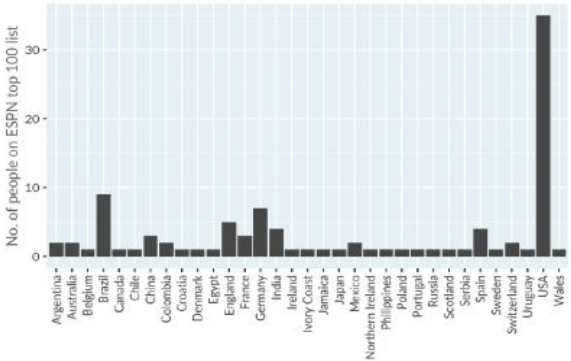


Quand utiliser un diagramme en barres ?

- Cas les plus courants :

- Lorsque vous disposez d'une variable catégorielle
  -> Données catégorielles : généralement textuelles
  Exemples : couleurs des yeux, pays, secteurs d'activité

- Lorsque vous souhaitez représenter des effectifs ou des pourcentages pour chaque catégorie



Histogrammes

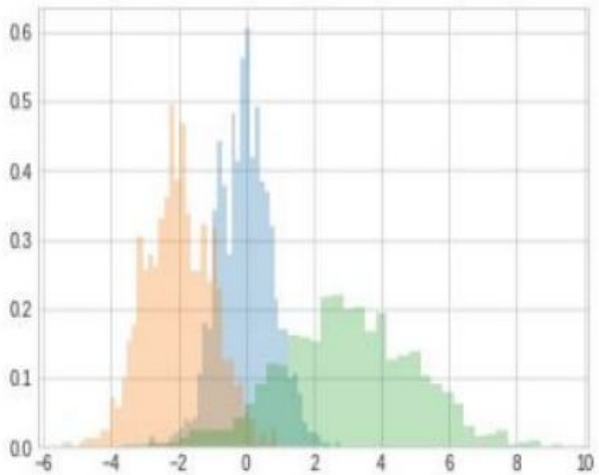
- Comprendre la distribution des données: Représenter la distribution de fréquence (forme des données)

- Synthétiser graphiquement de grands ensembles de données

- Comparer plusieurs distributions

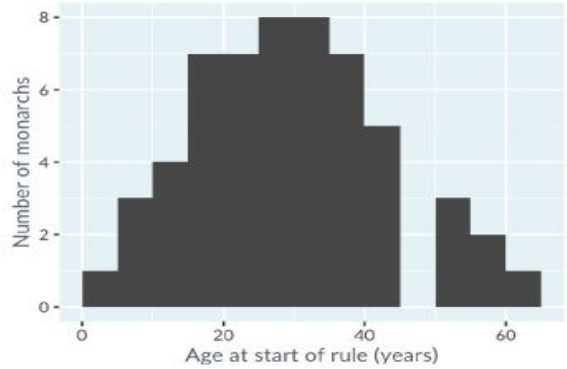
- Exemples :

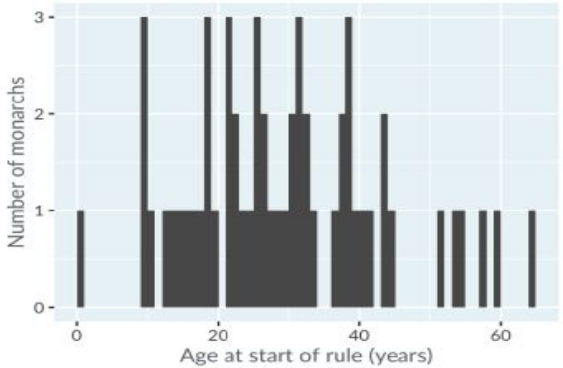
  Nombre de clients selon la taille de l'entreprise
  Résultats des étudiants à un examen
  Fréquence d'un défaut de produit



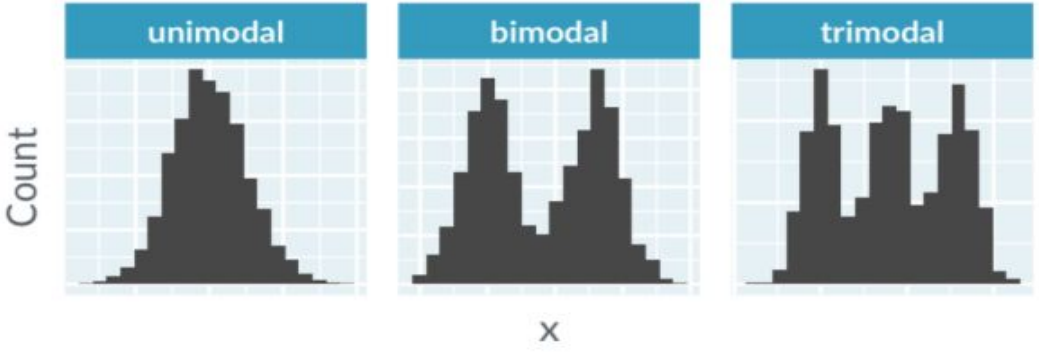
Quand utiliser un histogramme ?

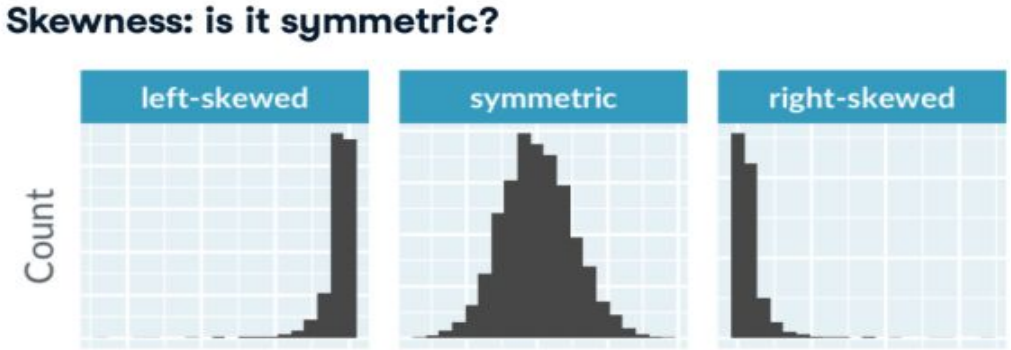
- Lorsque vous disposez de variables continues
- Lorsque vous souhaitez analyser la forme de la distribution des données





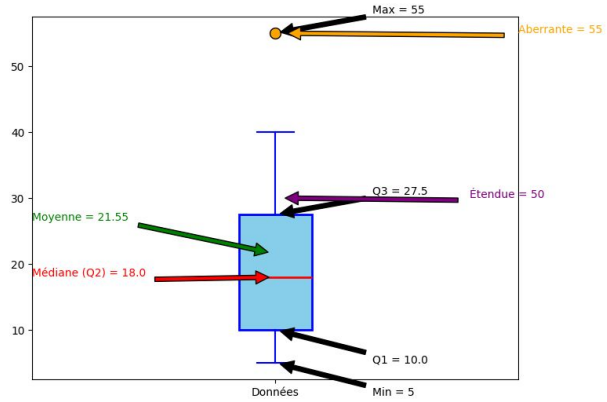
Modalité : combien de pics comporte la distribution ?





Diagrammes en en boîtes à moustaches

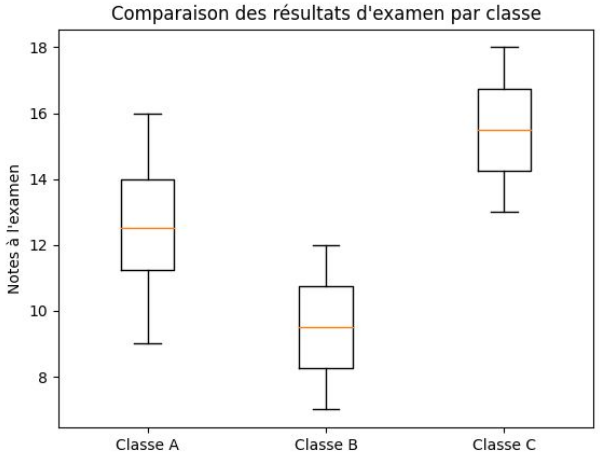
- Représente la distribution d'une variable quantitative
- Met en évidence la médiane, les quartiles et l'étendue
- Permet d'identifier les valeurs aberrantes (outliers)
- Utile pour comparer plusieurs distributions entre groupes



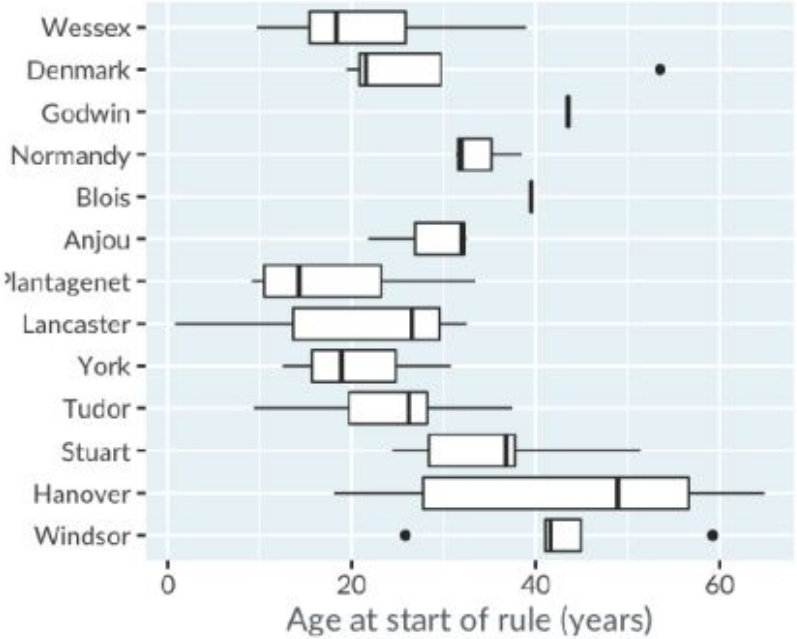
Quand utiliser un diagramme en boîtes à moustaches ?

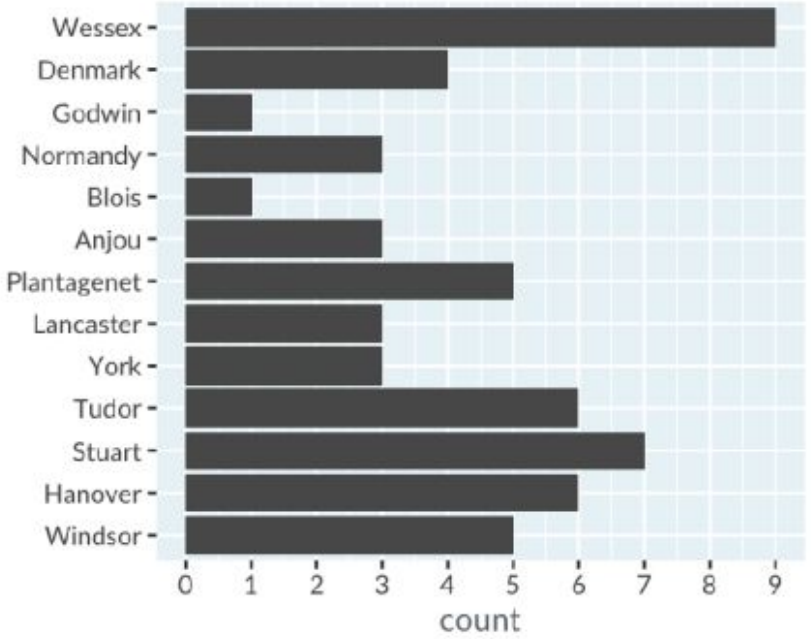
- Lorsque vous disposez d'une variable continue, répartie selon une variable catégorielle

- Lorsque vous souhaitez comparer les distributions de la variable continue pour chaque catégorie

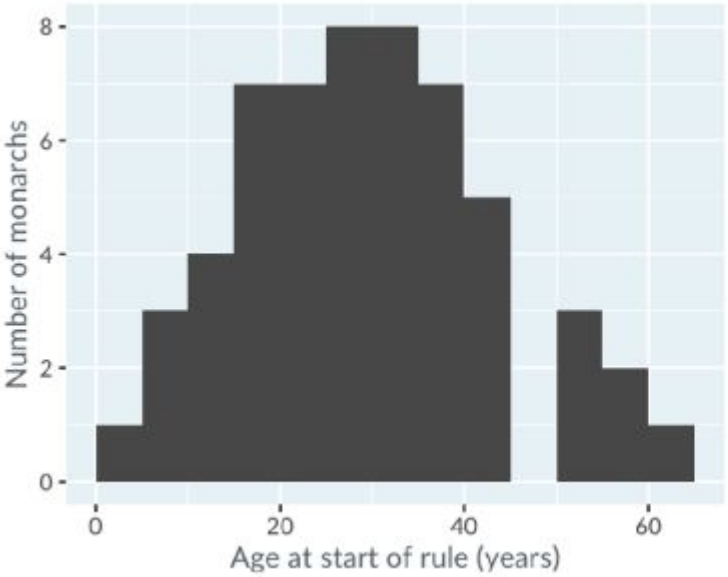


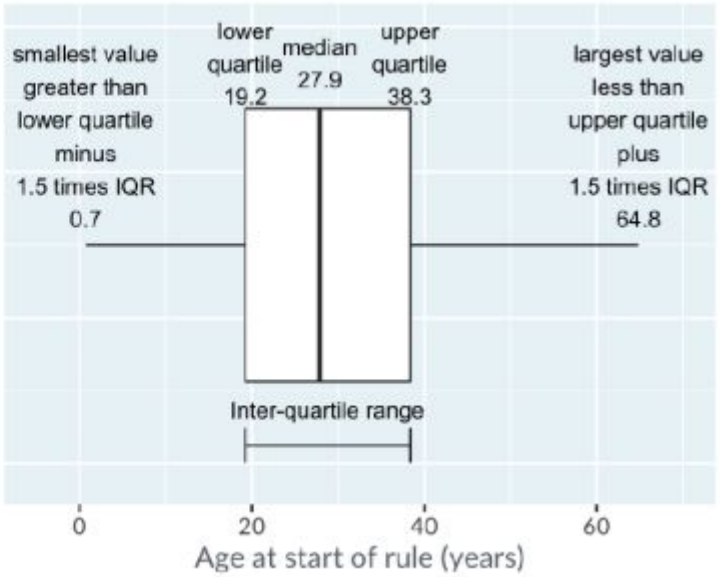
Diagrammes à barres vs. diagrammes en boîte





Histogramme vs. diagrammes en boîte





Dimensions

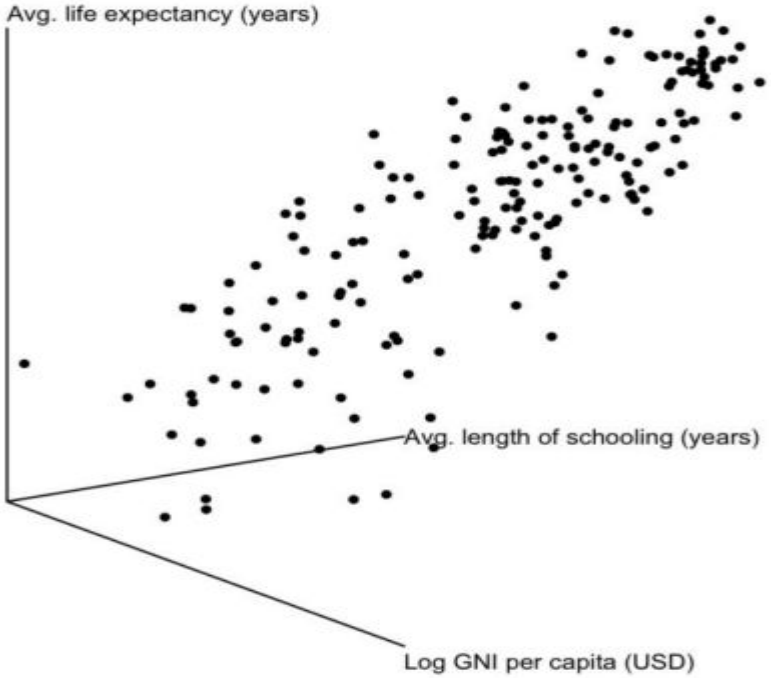
- Graphiques 2D:
  -> Représentent deux variables : x et y, permettent d'analyser les relations simples et les tendances

- Graphiques 3D:
  -> Représentent trois variables : x, y et z, permettent d'explorer des relations complexes entre plusieurs variables

- Choix de la dimension:
  -> 2D : plus lisible et simple pour les présentations

  -> 3D : utile pour des analyses plus complexes, mais peut être moins clair visuellement

Dimensions


Avg. life expectancy (years)
Avg. length of schooling (years)
Log GNI per capita (USD)

Autres Dimensions

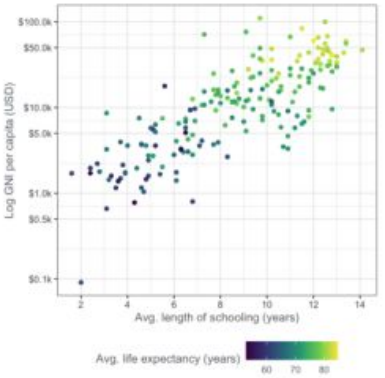
- Les points possèdent également ces dimensions :

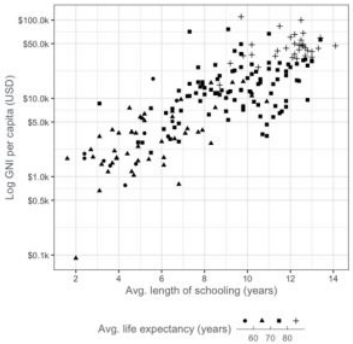
  -> Couleur

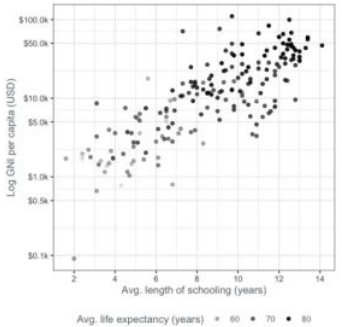
  -> Taille

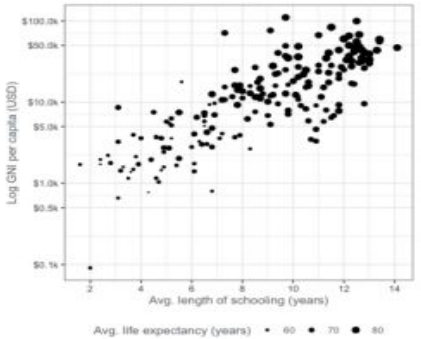
  -> Transparence

  -> Forme



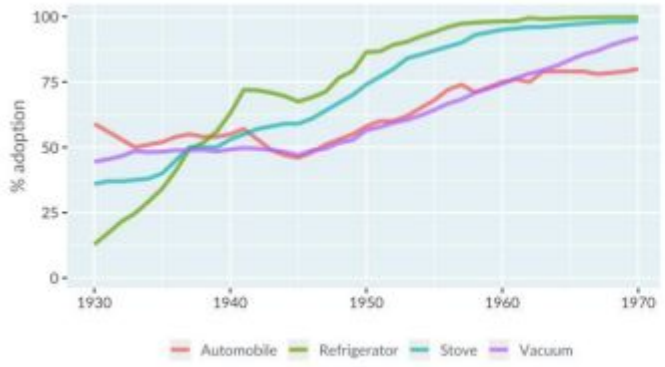


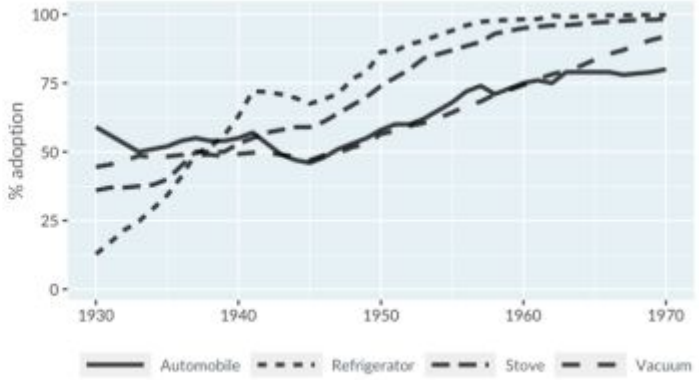




femto-st SCIENCES & TECHNOLOGIES

Autres Dimensions

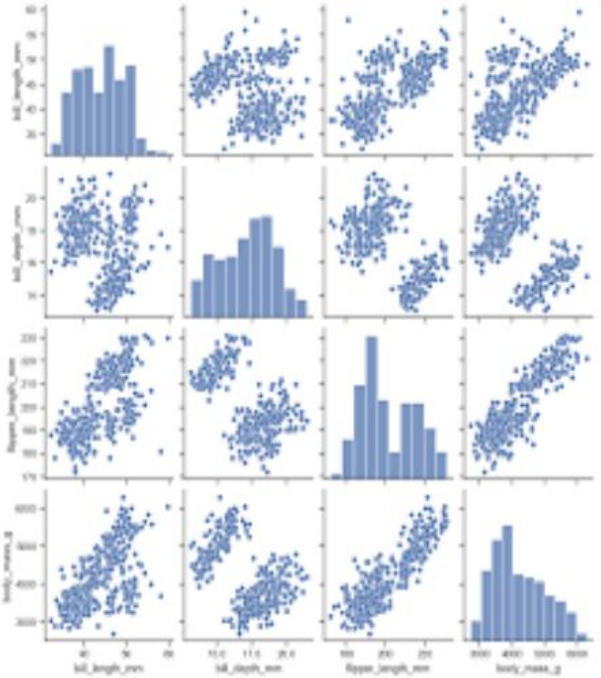




femto-st SCIENCES & TECHNOLOGIES

Diagrammes de paires (Pair plot)

- Vous disposez d'un ensemble de variables (continues, catégorielles ou mixtes)
- Vous souhaitez visualiser la distribution de chaque variable
- Vous souhaitez analyser la relation entre chaque paire de variables



femto-st SCIENCES & TECHNOLOGIES

Carte de corrélation (Correlation heatmap)

- Vous disposez de nombreuses variables continues
- Vous souhaitez avoir une vue d'ensemble simple des relations entre chaque paire de variables

femto-st SCIENCES & TECHNOLOGIES

Résumé

- Graphiques en lignes : montrent les tendances dans le temps
- Nuages de points : comparent deux variables numériques
- Diagrammes en barres : montrent les effectifs par catégorie
- Histogrammes : représentent la distribution des données
- Diagrammes boîtes à moustaches : Représente la distribution d'une variable quantitative
- Dimensions : representation des donnees avancée

Résumé

- Diagrammes de paires (Pair plot) : comparent plusieurs variables simultanément

- Carte de corrélation (Correlation heatmap) : visualise les relations entre variables

Fondements théoriques du traitement et de l'exploration de données

Données versus informations

- Données (Data): Toute observation recueillie concernant une caractéristique ou un événement est appelée donnée

- Information: les données brutes ont peu de signification lorsqu'elles sont considérées seules, donc ces données sont traitées et analysées, puis présentées de manière systématique, Elles sont ensuite converties en information

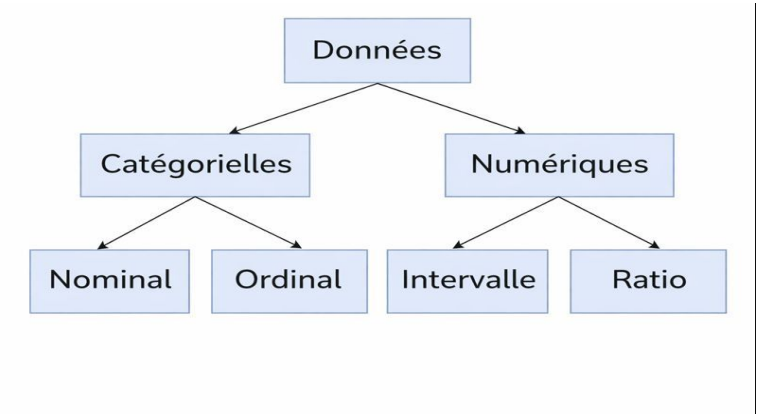
- Les données non converties en information ont peu de valeur pour l' évaluation et la planification, et ne peuvent pas être utilisées efficacement pour la prise de décision

Classification des données

- Les données peuvent être divisées en deux types :

  -> Données quantitatives : numériques

  -> Données qualitatives : descriptives, catégorielles ou basées sur des effectifs (fréquences)



Données quantitatives vs Données qualitatives

- Les données quantitatives se divisent en deux types :

  -> Discrètes : Les variables discrètes ne peuvent prendre que certaines valeurs bien définies.

  -> Continues : Les variables continues peuvent prendre n'importe quelle valeur, généralement comprise entre des limites données.

- Les données qualitatives, également appelées données descriptives, catégorielles ou de comptage de fréquences, se caractérisent par les éléments suivants :

  -> Les données sont regroupées en catégories selon leur nature ou leurs caractéristiques, avec une discontinuité entre les valeurs.

  -> Elles sont initialement exprimées sous une forme non numérique.

Gestion des données

- Se connecter: datasource (excel,csv,json,xml,text,oracle,mysql,cloud,binaire)

- analyser: traiter,analyser et visualiser ces données

- communiquer: dashboard ou rapport

Qualité des données

Dans le monde réel, les données sont souvent imparfaites (dirty) :

- Incomplètes : certaines valeurs d'attributs sont manquantes, certains attributs d'intérêt ne sont pas renseignés, ou les données ne sont disponibles que sous forme agrégée.

- Bruitées (noisy) : elles contiennent des erreurs ou des valeurs aberrantes, par exemple : Salaire = « -10 ».

- Incohérentes : elles présentent des discordances dans les codes, les formats ou les libellés. Exemples : Âge = « 42 » alors que Date de naissance = « 03/07/1997 »

Pourquoi le prétraitement des données ?

Le prétraitement des données est un ensemble de techniques visant à transformer des données brutes en un jeu de données propre et exploitable.

- Il comprend une série d'étapes permettant d'améliorer la qualité des données.

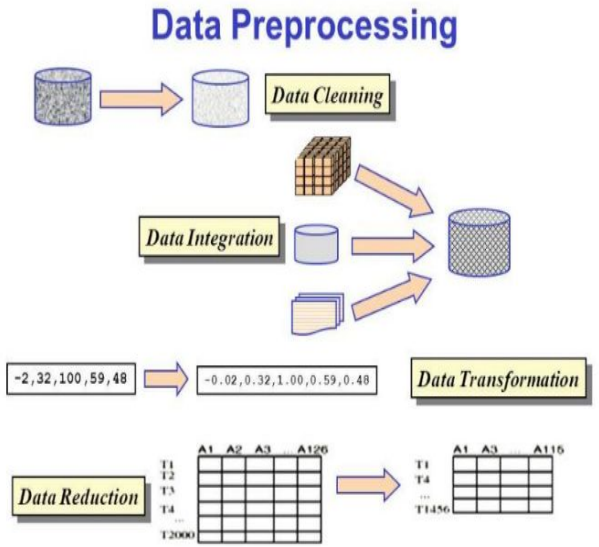
- Les données sont souvent collectées à partir de sources multiples, sous une forme brute qui n'est pas directement adaptée à l'analyse.

- Le prétraitement des données est effectué avant l'exécution d'analyses itératives ou de méthodes d'apprentissage.

Pourquoi le prétraitement des données ?

Étapes du prétraitement des données :

- Nettoyage des données (Data Cleaning)
- Intégration des données (Data Integration)
- Transformation des données (Data Transformation)
- Réduction des données (Data Reduction)



Nettoyage des données (Data Cleaning)

La qualité des données constitue un enjeu majeur et se pose à tous les niveaux des systèmes d'information. Ces problèmes peuvent être traités grâce au nettoyage des données (Data Cleaning) :

- il s'agit d'un processus permettant d'identifier les données inexactes, incomplètes ou non plausibles

- puis d'améliorer la qualité des données en corrigeant les erreurs détectées

- ce processus permet ainsi de réduire les erreurs et d'augmenter la fiabilité des données.

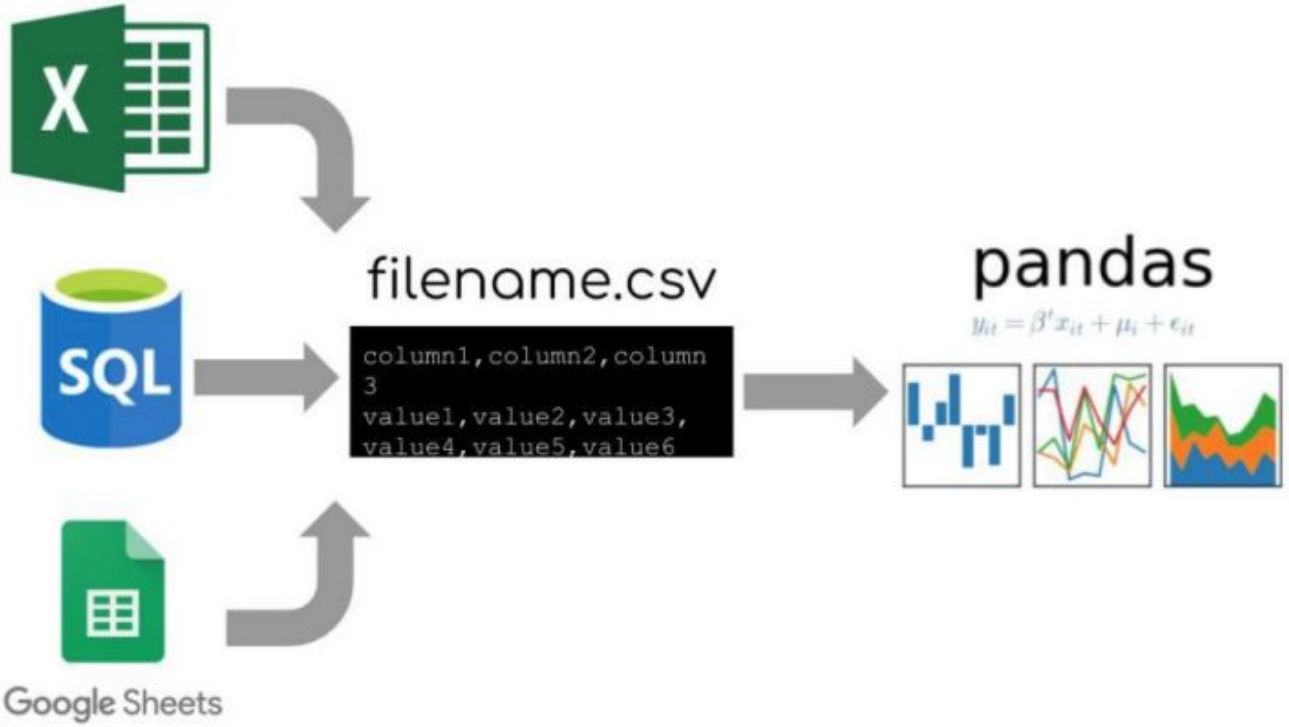
- Bien que le nettoyage des données soit une tâche chronophage et fastidieuse, il demeure indispensable.

Nettoyage des données (Data Cleaning)

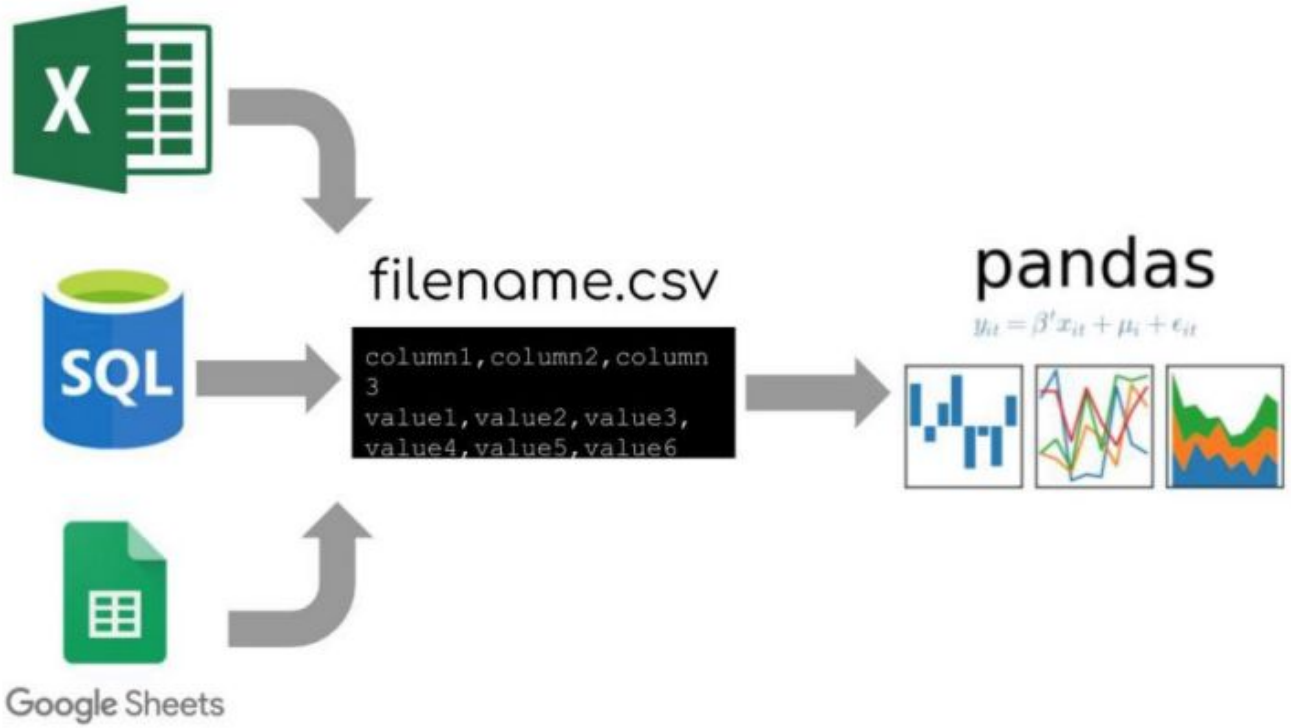
Critères de qualité des données :

- Exactitude (accuracy)
- Intégrité (integrity)
- Complétude (completeness)
- Validité (validity)
- Cohérence (consistency)
- Unicité (uniqueness)

Intégration des données (Data Integration)



Transformation des données (Data Transformation)



Réduction des données (Data Reduction)

	home	score	away	year
396	NaN	NaN	NaN	2010
397	NaN	NaN	NaN	2010
398	NaN	NaN	NaN	2010
399	NaN	NaN	NaN	2010
400	NaN	NaN	NaN	2010
...	...	...	...	...
455	NaN	NaN	NaN	2010
456	NaN	NaN	NaN	2010
457	NaN	NaN	NaN	2010
458	NaN	NaN	NaN	2010
459	NaN	NaN	NaN	2010

64 rows × 4 columns

Langage et bibliothèques utiles

NumPy

matplotlib

seaborn

pandas

bokeh

Python

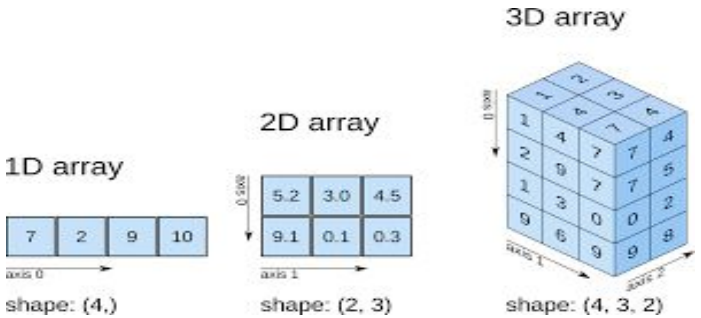
- Langage de programmation simple et lisible
- Très utilisé en data science, IA et automatisation
- Large écosystème de bibliothèques
- Communauté active et open-source

Python

```python
# Création de variables
x = 10  # Entier
y = "Bonjour"  # Chaîne de caractères
z = 3.14  # Nombre à virgule flottante

# Affectation multiple
a, b, c = 1, 2, 3

# Changement de type
x = "Valeur"

# Affichage des valeurs
print(x, y, z)
print(a, b, c)
```

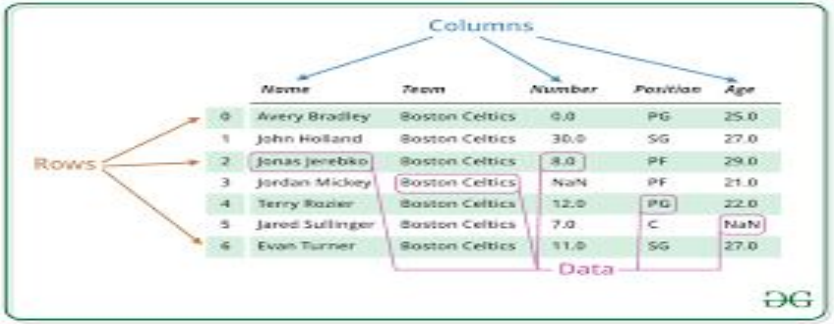
Numpy

- Bibliothèque pour le calcul numérique
- Manipulation efficace de tableaux multidimensionnels
- Fournit des fonctions mathématiques rapides
- Base de nombreuses autres bibliothèques scientifiques



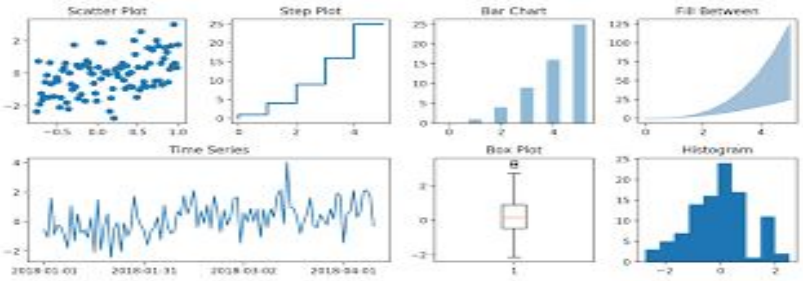
Pandas

- Bibliothèque essentielle pour l'analyse de données
- Gestion des données via Séries et DataFrame
- Nettoyage, transformation et agrégation des données
- Compatible avec NumPy et les outils de visualisation



Matplotlib

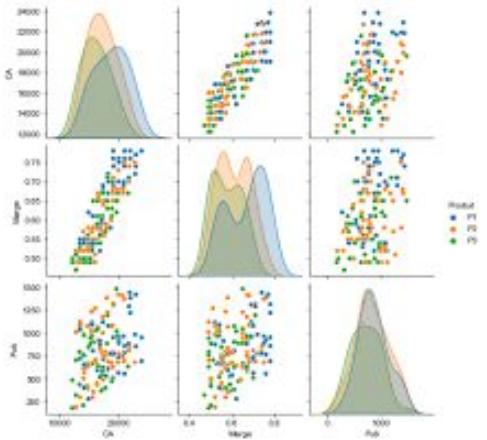
- Bibliothèque de visualisation de données
- Création de graphiques simples et personnalisables
- Supporte différents types de graphiques (courbes, barres, histogrammes)
- Fonctionne avec NumPy et Pandas



femto-st SCIENCES & TECHNOLOGIES

Seaborn

- Basée sur Matplotlib
- Visualisations statistiques avancées et esthétiques
- Interface de haut niveau, facile à utiliser
- Idéale pour des graphiques colorés et informatifs



Travaux pratiques

Manipulation des données avec Numpy

Introduction à NumPy

- Bibliothèque Python pour le calcul numérique
- Optimisée pour les tableaux multidimensionnels
- Très performante (implémentée en C)
- Base de nombreuses bibliothèques scientifiques

```
import numpy as np
```

Structure principale : ndarray

- Tableau N-dimensionnel
- Tous les éléments ont le même type
- Plus rapide et plus efficace que les listes Python

```
arr = np.array([1, 2, 3, 4, 5])
print(arr)
```

```
[1 2 3 4 5]
```

Création de tableaux NumPy

- Création manuelle

- Tableaux remplis automatiquement

```
np.zeros((3, 3))      # matrice de zéros
np.ones((2, 4))       # matrice de uns
np.arange(0, 10, 2)   # valeurs espacées
np.linspace(0, 1, 5)  # valeurs uniformes
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

```
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]]
```

```
[0 2 4 6 8]
```

```
[0.   0.25 0.5  0.75 1.  ]
```

Accès et sélection des données (Slicing)

- Indexation simple
- Slicing (découpage)
- Accès par lignes et colonnes

```
arr = np.array([10, 20, 30, 40, 50])

arr[0]        # premier élément
arr[1:4]      # slicing

mat = np.array([[1, 2], [3, 4]])
mat[0, 1]     # ligne 0, colonne 1
```

```
10
[20 30 40]
2
```

Opérations mathématiques

- Opérations vectorisées
- Calculs rapides sans boucles

```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

a + b
a * b
a ** 2
```

```
[5 7 9]
[ 4 10 18]
[1 4 9]
```

Fonctions statistiques

- Analyse descriptive
- Calculs globaux ou par axe

```
data = np.array([10, 20, 30, 40])

np.mean(data)
np.sum(data)
np.min(data)
np.max(data)
np.std(data)
```

```
25.0
100
10
40
11.180339887498949
```

Manipulation de la forme (reshape)

- Changement de dimensions
- Utile pour matrices et images

```
arr = np.arange(6)
print(arr.reshape((2, 3)))

mat = np.array([[1, 2], [3, 4]])
print(mat.T)
```

```
[[0 1 2]
 [3 4 5]]

[[1 3]
 [2 4]]
```

Filtrage conditionnel

- Sélection basée sur des conditions
- Très utilisé en analyse de données

```
data = np.array([5, 10, 15, 20])

print(data[data > 10])
```

```
[15 20]
```

Génération aléatoire

- Données de test
- Résultats reproductibles

```
np.random.seed(1)
print(np.random.randint(1, 10, 5))
```

```
[6 9 6 1 1]
```

Manipulation des données avec Pandas

Introduction à Pandas

- Bibliothèque Python dédiée à la manipulation et l'analyse de données
- Basée sur NumPy
- Structure principale : DataFrame
- Utilisée en data science, data analysis et machine learning

```
import pandas as pd
```

Création de Séries

- Tableau unidimensionnel
- Index automatique

```
ages = pd.Series([22, 25, 30])
print(ages)
```

```
0    22
1    25
2    30
dtype: int64
```

Création de DataFrames

- Structure tabulaire
- Colonnes hétérogènes

```
data = {
    "Nom": ["Alice", "Bob", "Charlie"],
    "Age": [22, 25, 30],
    "Salaire": [3000, 3500, 4000]
}

df = pd.DataFrame(data)
print(df)
```

	Nom	Age	Salaire
0	Alice	22	3000
1	Bob	25	3500
2	Charlie	30	4000

Chargement des données

- Lecture depuis un fichier CSV

```
df = pd.read_csv("data.csv")
print(df.head())
```

	Nom	Age	Salaire
0	Alice	22	3000
1	Bob	25	3500
2	Charlie	30	4000

Informations sur les données

- Taille, types, valeurs non nulles

```
df.info()
```

```
df.dtypes
```

```
df.values
```

```
df.index
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 3 columns):
Nom        3 non-null object
Age        3 non-null int64
Salaire    3 non-null int64
```

Informations et opérations sur les colonnes

- Accès à une colonne
- Opérations simples
- Affichage des noms de colonnes

```
print(df["Salaire"])
print(df["Salaire"].mean())
```

```
0    3000
1    3500
2    4000
Name: Salaire, dtype: int64

3500.0
```

```
df.columns
```

```
Index(['Nom', 'Age', 'Ville'], dtype='object')
```

Informations et opérations sur les colonnes

- Sélection de plusieurs colonnes
- Comptage des valeurs uniques
- Suppression d'une colonne

```
df[["Nom", "Ville"]]
```

	Nom	Ville
0	Alice	Paris
1	Bob	Lyon

```
df["Nom"].value_counts()
```

```
Alice      2
Bob        1
Charlie    1
Name: Nom, dtype: int64
```

```
df.drop("Age", axis=1)
```

	Nom	Ville
0	Alice	Paris
1	Bob	Lyon
2	Alice	Paris
3	Charlie	Marseille

Sélection des données (indexing & slicing)

- Sélection par index
- Sélection conditionnelle

```
print(df.iloc[0])
print(df.loc[df["Age"] > 23])
```

```
Nom         Alice
Age            22
Salaire      3000
Name: 0, dtype: object
```

```
       Nom  Age  Salaire
1      Bob   25     3500
2  Charlie   30     4000
```

Sélection des données (indexing & slicing)

- Sélection basé sur les labels
- Filtrage selon une liste de valeurs

```
df.loc[0:2, ["Nom", "Ville"]]
```

	Nom	Ville
0	Alice	Paris
1	Bob	Lyon
2	Charlie	Marseille

```
df.loc[df["Age"] > 25]
```

	Nom	Age	Ville
2	Charlie	30	Marseille
3	Alice	28	Paris

```
df[df["Ville"].isin(["Paris", "Lyon"])]
```

	Nom	Age	Ville
0	Alice	22	Paris
1	Bob	25	Lyon
3	Alice	28	Paris

Nettoyage des données

- Valeurs manquantes
- Suppression de doublons

```
df.isnull().sum()

df.drop_duplicates()
```

```
Nom       0
Age       0
Salaire   0
dtype: int64
```

Nettoyage des données

- Remplacement des valeurs manquantes par la moyenne

- Suppression des valeurs manquantes (modification directe)

```
df["target"].fillna(df["target"].mean())
```

```
0    10.0
1    20.0
2    20.0
3    30.0
4    20.0
Name: target, dtype: float64
```

```
df["target"].dropna(inplace=True)
```

```
   target
0    10.0
1    20.0
3    30.0
```

Transformation des données

- Création de nouvelles colonnes

```
df["Salaire_annuel"] = df["Salaire"] * 12
print(df)
```

```
      Nom  Age  Salaire  Salaire_annuel
0   Alice   22     3000           36000
1     Bob   25     3500           42000
2  Charlie   30     4000           48000
```

Analyse descriptive

- Statistiques de base

```
df.describe()
```

	Age	Salaire	Salaire_annuel
count	3.000000	3.000000	3.000000
mean	25.666667	3500.000000	42000.000000
min	22.000000	3000.000000	36000.000000
max	30.000000	4000.000000	48000.000000

Groupement et agrégation

- Analyse par catégorie

```
df.groupby("Age")["Salaire"].mean()
```

```
Age
22    3000.0
25    3500.0
30    4000.0
Name: Salaire, dtype: float64
```

Tri

- Tri croissant / décroissant

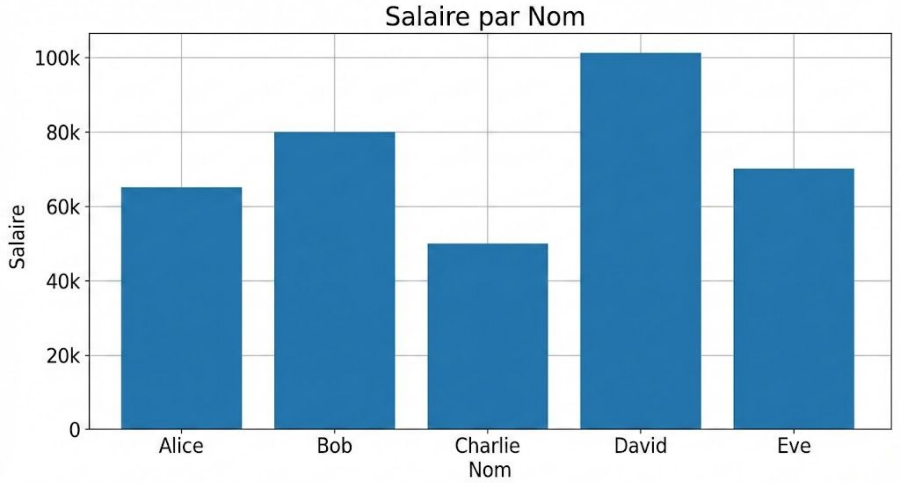
```
df.sort_values(by="Salaire", ascending=False)
```

	Nom	Age	Salaire
2	Charlie	30	4000
1	Bob	25	3500
0	Alice	22	3000

Visualisation rapide avec Pandas

- Graphiques intégrés
- Basés sur Matplotlib

```
df.plot(x="Nom", y="Salaire", kind="bar")
```



Visualisation des données avec Matplotlib

Introduction à Matplotlib

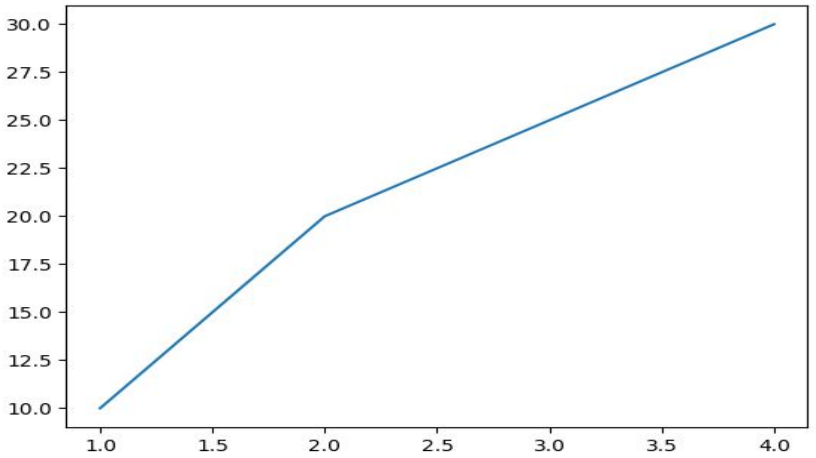
- Bibliothèque Python de visualisation de données
- Permet de créer des graphiques simples et personnalisables
- Base de nombreuses bibliothèques (Seaborn, Pandas)
- Module principal : matplotlib.pyplot

```
import matplotlib.pyplot as plt
```

Graphique simple (Line Plot)

- Représentation de données continues
- Très utilisé pour l'évolution dans le temps

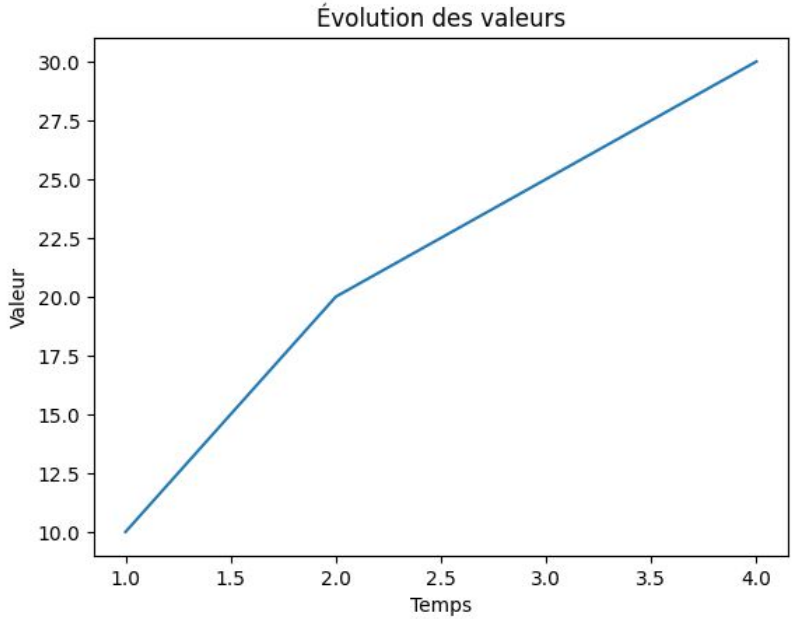
```
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]

plt.plot(x, y)
plt.show()
```



Ajout de titre et labels

- Améliore la lisibilité du graphique

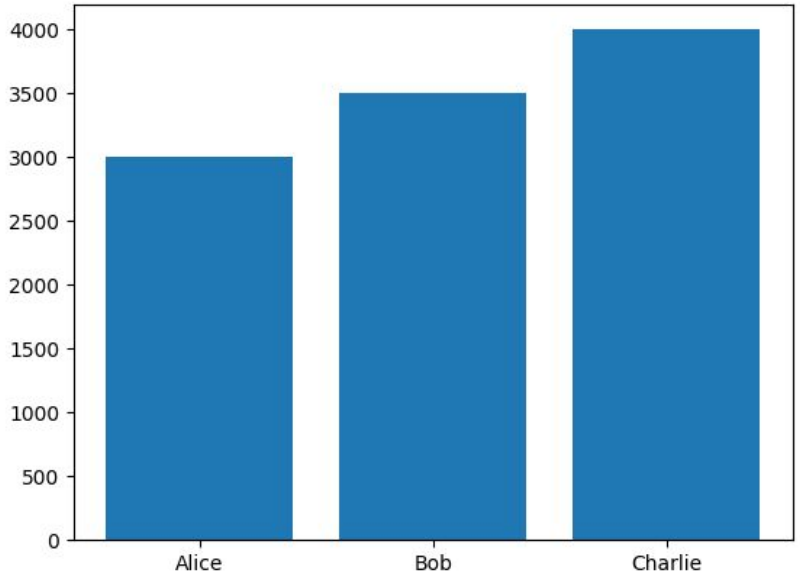
```
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]

plt.plot(x, y)
plt.title("Évolution des valeurs")
plt.xlabel("Temps")
plt.ylabel("Valeur")
plt.show()
```



Graphique en barres (Bar Plot)

- Comparaison entre catégories
- Très utilisé avec Pandas

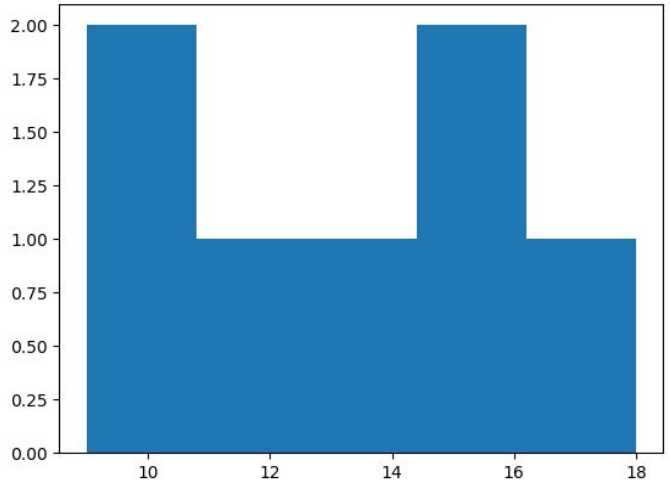
```
noms = ["Alice", "Bob", "Charlie"]
salaires = [3000, 3500, 4000]

plt.bar(noms, salaires)
plt.show()
```



Histogramme

- Visualisation de la distribution des données
- Détection des tendances et anomalies

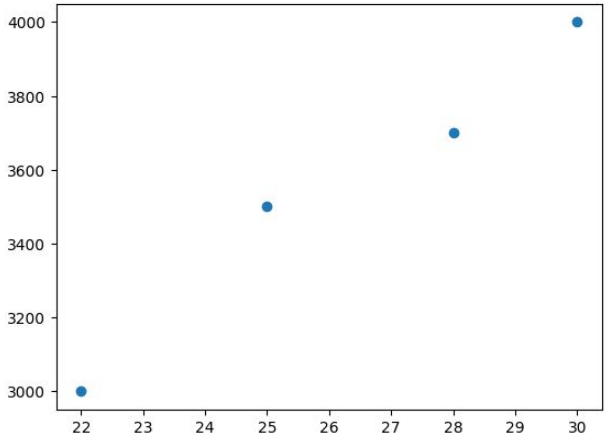
```
notes = [12, 15, 9, 18, 14, 10, 16]

plt.hist(notes, bins=5)
plt.show()
```



Nuage de points (Scatter Plot)

- Analyse de la relation entre deux variables
- Détection de corrélation

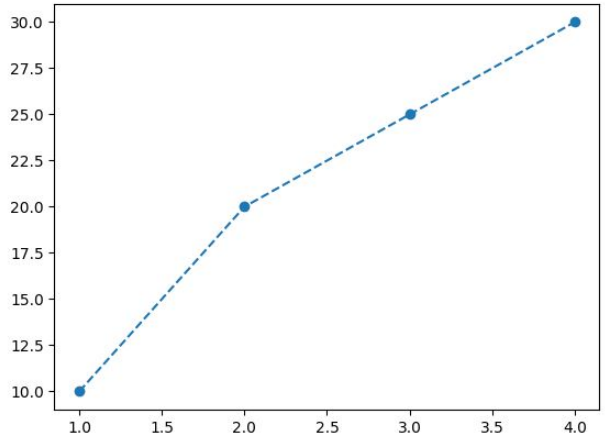
```
ages = [22, 25, 30, 28]
salaires = [3000, 3500, 4000, 3700]

plt.scatter(ages, salaires)
plt.show()
```



Personnalisation (couleurs, styles)

- Amélioration visuelle
- Lecture plus intuitive

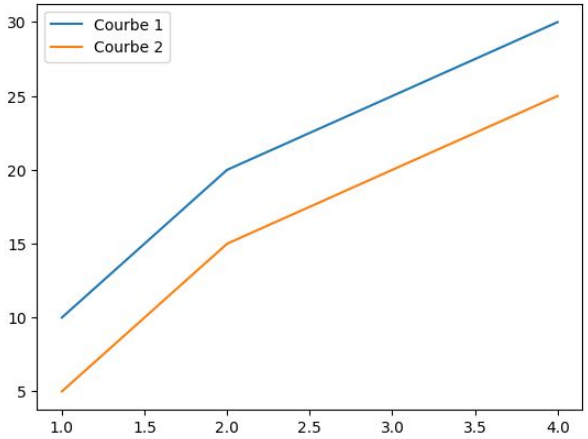
```
ages = [22, 25, 30, 28]
salaires = [3000, 3500, 4000, 3700]

plt.plot(x, y, marker='o', linestyle='--')
plt.show()
```



Plusieurs courbes sur un même graphique

- Amélioration visuelle
- Lecture plus intuitive

```
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]
y2 = [5, 15, 20, 25]

plt.plot(x, y, label="Courbe 1")
plt.plot(x, y2, label="Courbe 2")
plt.legend()
plt.show()
```

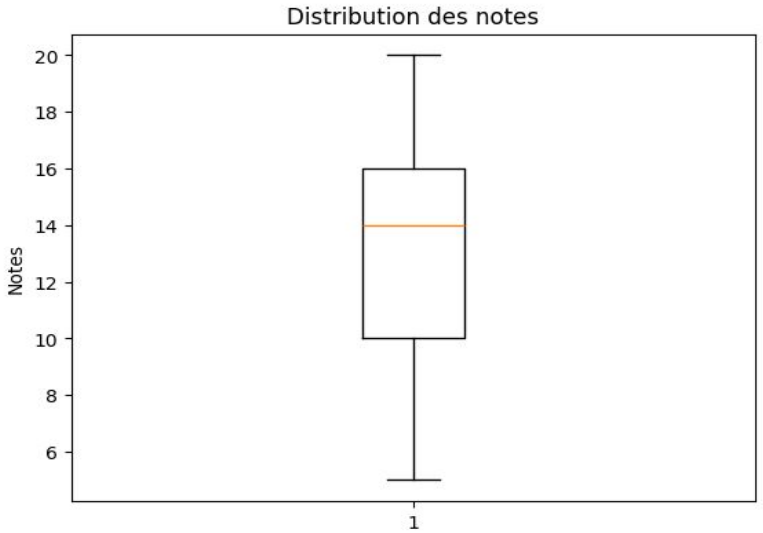


Boxplot (Boîte à moustaches)

- Représente la distribution des données

  Montre : médiane, quartiles, valeurs aberrantes (outliers)

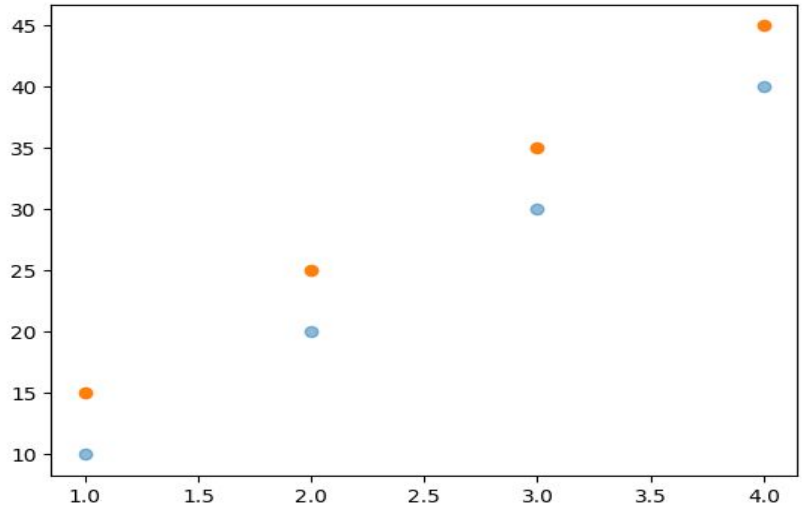
```
notes = [12, 15, 9, 18, 14, 10, 16, 5, 20]

plt.boxplot(notes)
plt.title("Distribution des notes")
plt.ylabel("Notes")
plt.show()
```



Transparence (alpha)

- Permet de gérer le chevauchement des données
- Valeur entre 0 (transparent) et 1 (opaque)

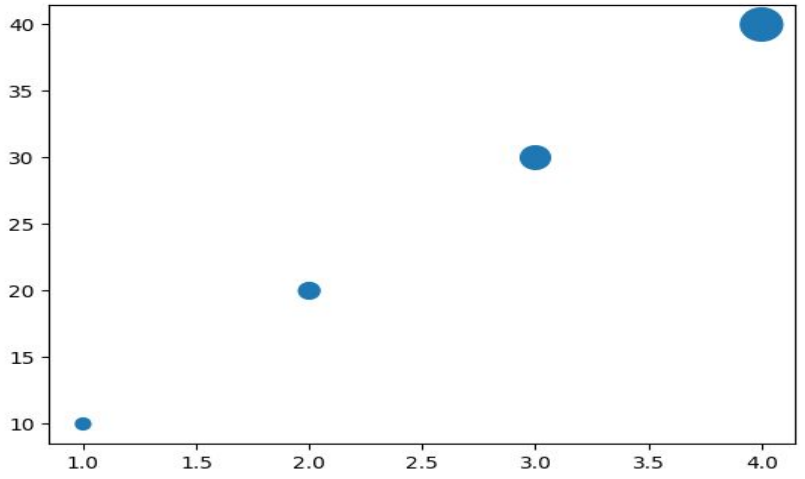
```
x = [1, 2, 3, 4]
y1 = [10, 20, 30, 40]
y2 = [15, 25, 35, 45]

plt.scatter(x, y1, alpha=0.5)
plt.scatter(x, y2, alpha=0.5)
plt.show()
```



Taille des marqueurs (marker size)

- Permet de gérer le chevauchement des données
- Valeur entre 0 (transparent) et 1 (opaque)

```
x = [1, 2, 3, 4]
y1 = [10, 20, 30, 40]
sizes = [50, 100, 200, 400]

plt.scatter(x, y1, s=sizes)
plt.show()
```



Sauvegarde d’un graphique

- Export pour rapports ou présentations

```
plt.plot(x, y)
plt.savefig("graphique.png")
plt.show()
```

Visualisation des données avec Seaborn

Introduction à Seaborn

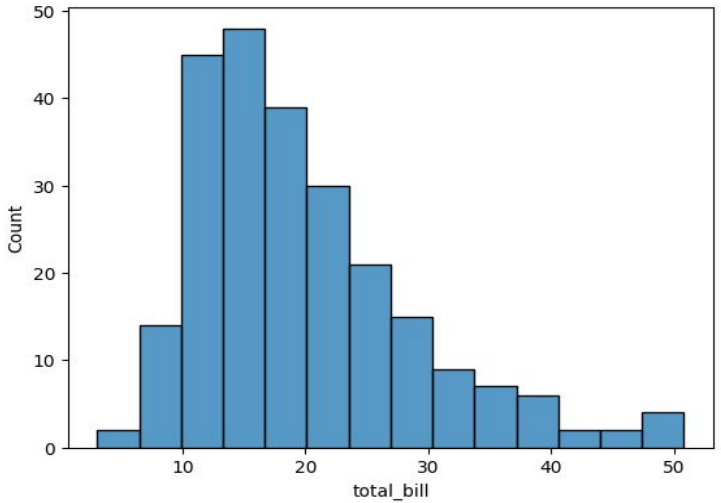
- Bibliothèque Python de visualisation statistique
- Basée sur Matplotlib
- Interface haut niveau
- Graphiques esthétiques par défaut

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

Graphique de distribution (Histogramme)

- Visualisation de la distribution d'une variable

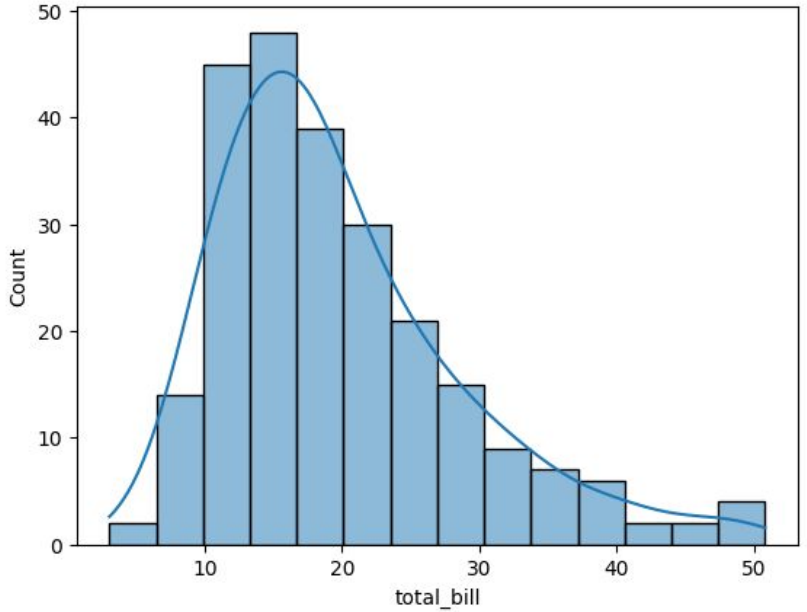
```
sns.histplot(df["total_bill"])
plt.show()
```



Histogramme avec densité (KDE)

- Ajout d'une courbe de densité

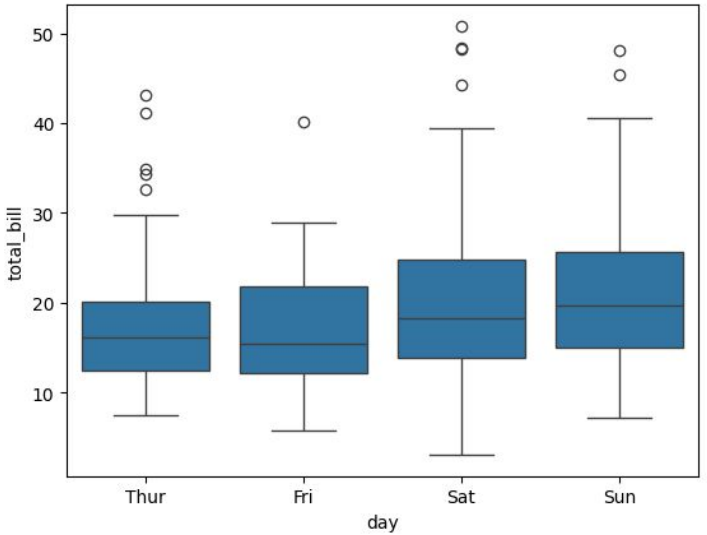
```
sns.histplot(df["total_bill"], kde=True)
plt.show()
```



Boxplot (boîte à moustaches)

- Analyse de la distribution
- Détection des valeurs aberrantes

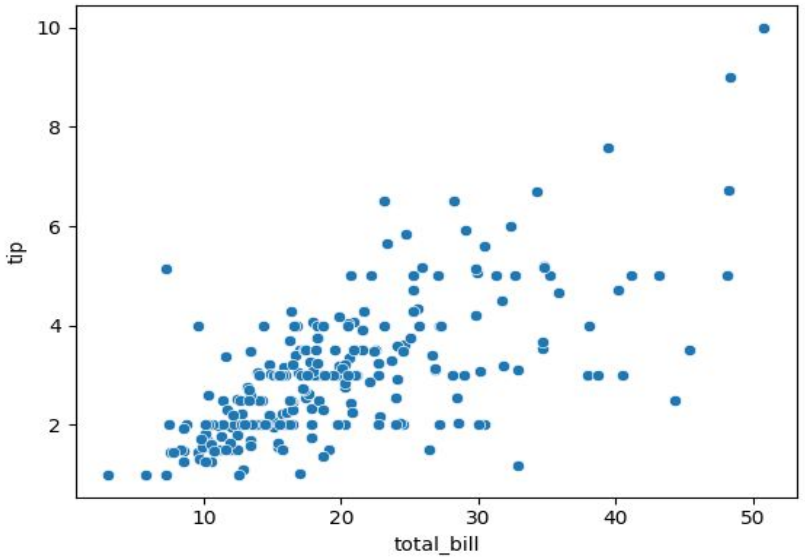
```
sns.boxplot(x="day", y="total_bill", data=df)
plt.show()
```



Nuage de points (Scatter plot)

- Analyse de la relation entre deux variables

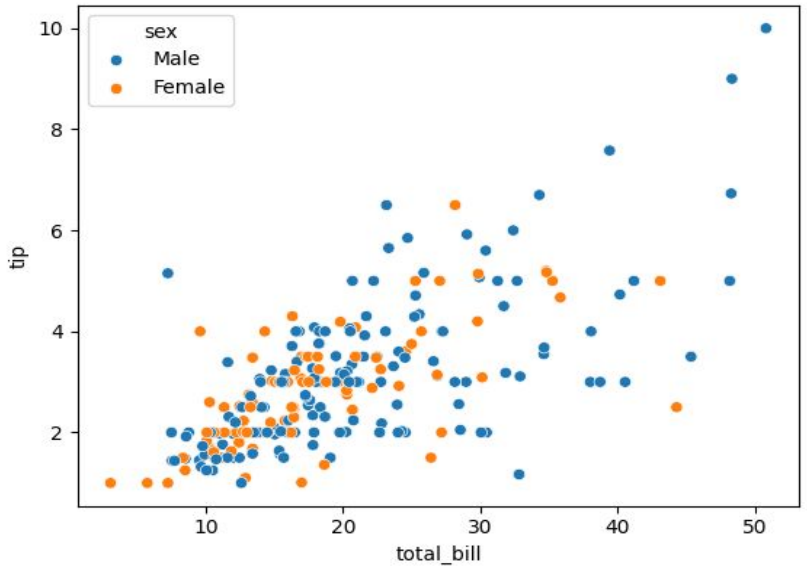
```
sns.scatterplot(x="total_bill", y="tip", data=df)
plt.show()
```



Scatter plot avec catégorie (hue)

- Ajout d'une variable catégorielle

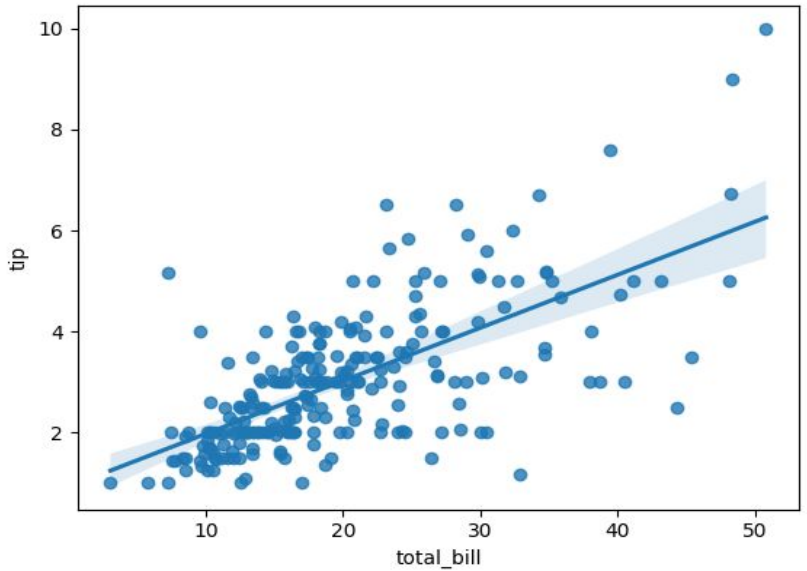
```
sns.scatterplot(
    x="total_bill",
    y="tip",
    hue="sex",
    data=df
)
plt.show()
```



Régression linéaire

- Visualisation + modèle de tendance

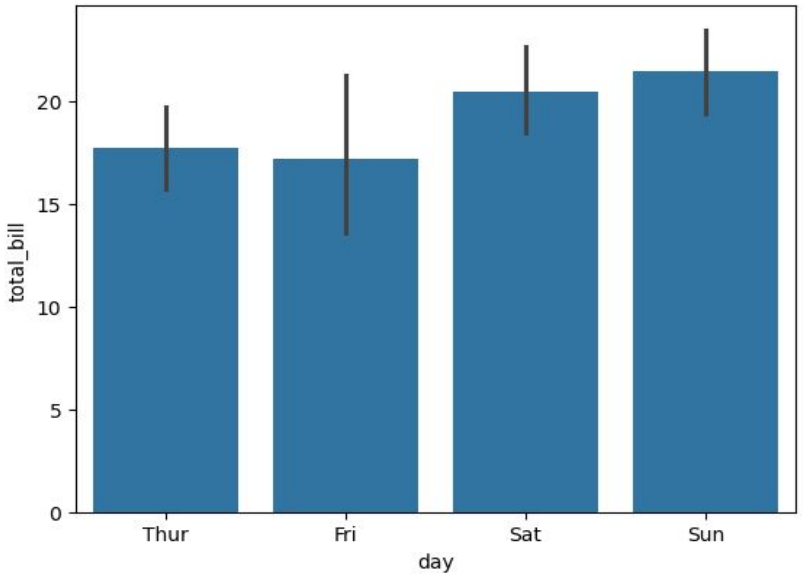
```
sns.regplot(x="total_bill", y="tip", data=df)
plt.show()
```



Barplot (statistiques)

- Moyenne par catégorie (par défaut)

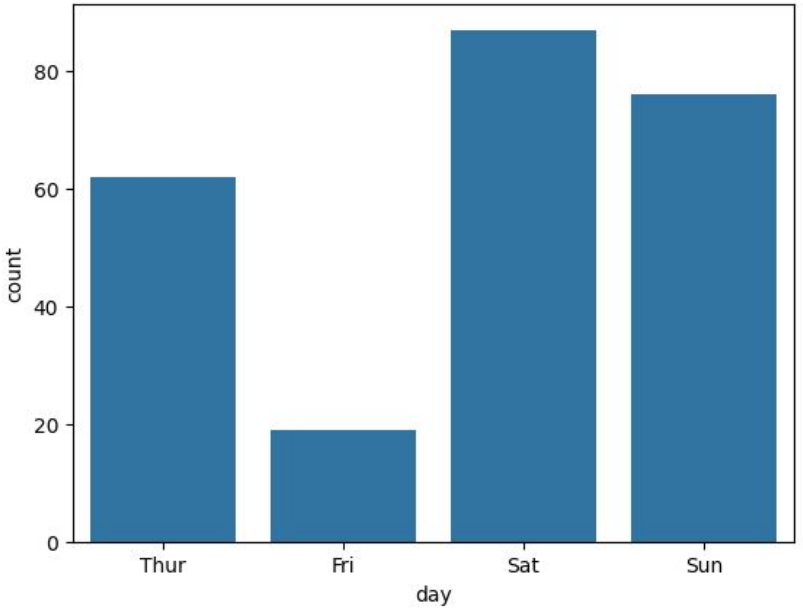
```
sns.barplot(x="day", y="total_bill", data=df)
plt.show()
```



Countplot (comptage)

- Fréquence des catégories

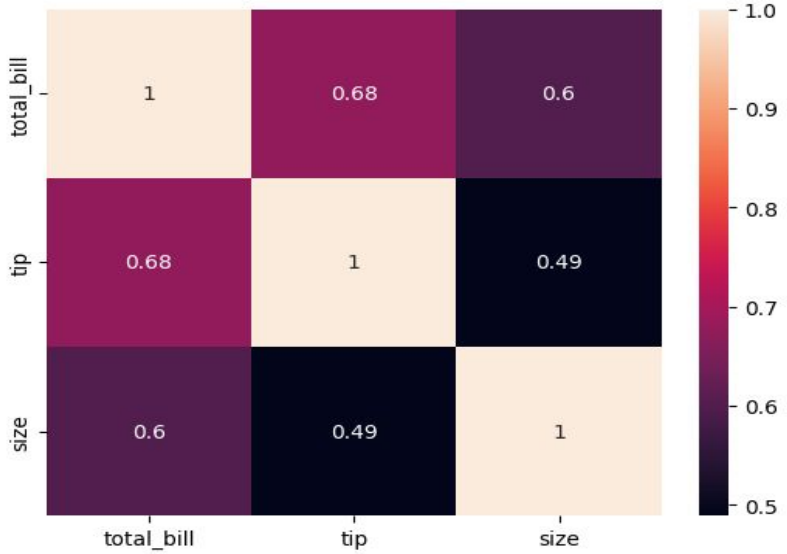
```
sns.countplot(x="day", data=df)
plt.show()
```



Heatmap (matrice de corrélation)

- Visualisation des corrélations numériques

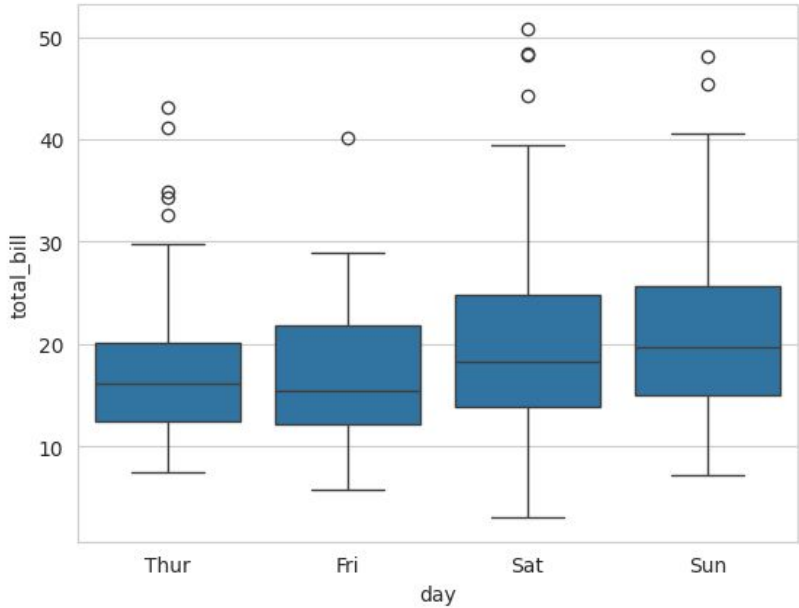
```
corr = df.corr(numeric_only=True)
sns.heatmap(corr, annot=True)
plt.show()
```



Styles et thèmes Seaborn

- Styles intégrés

```
sns.set_style("whitegrid")
sns.boxplot(x="day", y="total_bill", data=df)
plt.show()
```



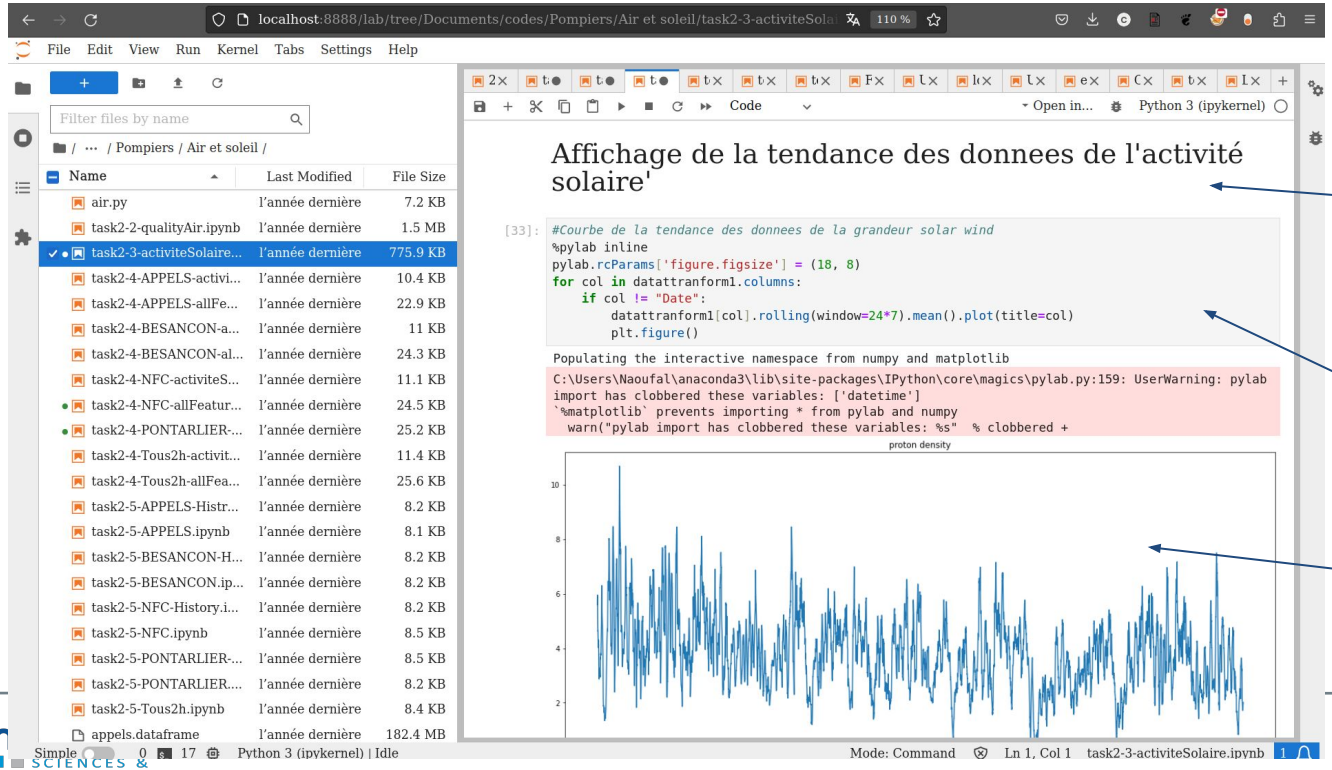
Travaux pratiques

Travaux dirigés

Préparer son environnement de travail

Jupyter notebooks

`Jupyter notebook, jupyter-lab, google colaboratory, jupyter-book…`



markdown cells

python cells

output cells

\+ pdf/latex/html…