

Analyse et Manipulation de Données avec Python

- 1. Introduction à l'Analyse et à la Manipulation de Données**
- 2. Rappel mathématique**
- 3. Préparation et Chargement des Données**
- 4. Manipulation et Exploration des Données avec Pandas**
- 5. Visualisation de Données avec Pandas et Matplotlib**
- 6. Analyse Avancée et Cas Pratiques**
- 7. Conclusion et Perspectives**

- Fondements de l'Analyse de Données
- Les Outils
- La bibliothèque Pandas

L'analyse des données est un **processus** qui consiste à collecter, à traiter et à **interpréter** des **données** pour en extraire des informations **utiles**.

Les entreprises font du **“Data Driven” =>**
Approche qui consiste à prendre des décisions stratégiques sur la base d'une analyse et d'une interprétation des données.

- **Amélioration de l'expérience client**
 - ✓ Comprendre le client
 - ✓ Trouver de nouveaux clients
 - ✓ Améliorer la satisfaction client
- **Amélioration de l'efficacité dans l'entreprise**
 - ✓ Automatiser des tâches.
 - ✓ Optimiser des processus
- **Détection d'anomalies ou de fraude**
 - ✓ Utiliser en cybersécurité
- **Assister la prise de décision en entreprise**
 - ✓ Permettre la prédiction
 - ✓ Tendance du marché
 - ✓ Nouvelle opportunité

Exemple de **manipulation de données** dans un carnet :

- **Ranger les données** dans des structures, les manipuler, les filtrer
- **Nettoyer un jeu de données** afin de le rendre exploitable et ordonné.
- **Visualiser graphiquement les données**
- **Utiliser l'interactivité et Manipulation dynamique des données** à l'intérieur des carnets.

Exemple de **transformation des données en information** dans un carnet :

- **analyser des données via la statistique**, l'algèbre linéaire, l'optimisation, l'analyse de Fourier ...
- **apprendre à partir des données** (machine learning, IA).

<https://www.astera.com/fr/type/blog/data-manipulation-tools/>
<https://jhub.cnam.fr/doc/pages/utilisation/manipulation/>

- **Python** : <https://www.python.org/downloads/>
- **Anaconda** : <https://www.anaconda.com/download>
- **Jupyter & IPython** :
 - <https://jupyter.org/install>
 - <https://ipython.org/install.html>
- **Panda** : https://pandas.pydata.org/docs/getting_started/install.html
- **NumPy** : <https://numpy.org/install/>
- **Matplotlib** : <https://matplotlib.org/stable/users/installing/index.html>
- **SCIPY** : <https://scipy.org/install/>
- **scikit-learn** : <https://scikit-learn.org/stable/install.html>

Le projet IPython est une version améliorée du shell Python standard, proposant de nombreuses fonctionnalités supplémentaires créée en 2001 par Fernando Perez

IPython est conçu pour optimiser la productivité dans le domaine du développement de logiciels.

<https://ipython.org/>

Le Jupyter Notebook (Ex IPython Notebooks) est une application web **open-source** permettant de créer et de partager des documents contenant du code, des équations, des visualisations, et du texte narratif.

Il s'agit d'un environnement de calcul interactif basé sur le web permettant de créer des documents notebooks.

« Jupyter » est un acronyme désignant **Julia, Python et R**

<https://jupyter.org/>

<https://datascientest.com/jupyter-notebook-tout-savoir>

Pandas est une bibliothèque logicielle open-source du langage de programmation Python, entièrement dédiée à la Data Science et qui est conçue pour permettre :

- **la manipulation et l'analyse de données en langage Python,**
- **de charger, aligner, de manipuler ou de fusionner des données.**
- Performant

<https://pandas.pydata.org>

Pandas fournit des outils de manipulation de données de haut niveau, notamment des outils pour :

- Lire et écrire des données de différents formats (CSV, Excel, JSON, ...)
- Effectuer des requêtes SQL sur des données.
- Manipuler des tables de données (ajouter, supprimer, modifier des lignes et des colonnes ...)
- effectuer des calculs sur les données (agrégation, filtrage, ...)



1 - Introduction à l'Analyse et à la Manipulation de Données

-> la bibliothèque Pandas

The screenshot shows a Jupyter Notebook interface with the following structure:

- File Edit View Run Kernel Tabs Settings Help** (top menu bar)
- OPEN TABS** (left sidebar):
 - Panda_Exemple001.py (marked with a red X)
 - Panda_Exemple001.py (marked with a green checkmark)
- KERNELS** (left sidebar): Python 3 (ipykernel)
- LANGUAGE SERVERS** (left sidebar): Shut Down All
- TERMINALS** (left sidebar): Shut Down All
- Panda_Exemple001.py** (top cell):

```
1 import pandas
2 if __name__ == '__main__':
3     mydataset = {
4         'Modèle de voiture': ['Toyota Prius", "Honda Civic", "Ford Escape"],
5         'Capacité': [4, 5, 5]
6     }
7
8     myvar = pandas.DataFrame(mydataset)
9
10    print(myvar)
11
```
- Panda_Exemple001.py** (cell 1):

```
[1]: import pandas
if __name__ == '__main__':
    mydataset = {
        'Modèle de voiture': ['Toyota Prius", "Honda Civic", "Ford Escape"],
        'Capacité': [4, 5, 5]
    }

    myvar = pandas.DataFrame(mydataset)

    print(myvar)
```
- Output:**

	Modèle de voiture	Capacité
0	Toyota Prius	4
1	Honda Civic	5
2	Ford Escape	5

Les séries comme les DataFrame en Pandas peuvent être utilisées pour :

- **Stocker et organiser des données** , elles sont particulièrement adaptées pour les données indexées, telles que les données temporelles ou les données géographiques.
- **Effectuer des analyses statistiques**, telles que le calcul de la moyenne, de la médiane et de la variance.
- **Visualiser des données**, pour créer des graphiques et des diagrammes pour visualiser les données.

1 - Introduction à l'Analyse et à la Manipulation de Données

-> la bibliothèque Pandas : DataFrame & Serie

Caractéristique	Série	DataFrame
Dimensionnalité	Unidimensionnelle	Bidimensionnelle
Nombre d'axes	1	2 ou plus
Type de données	Tableau de valeurs indexées	Tableau de données indexé
Représentation des données	Une colonne de valeurs indexées	Deux axes de données indexées
Exemples d'utilisation	Stocker des données de température, suivre les performances d'un investissement, analyser les données de trafic Web	Stocker des données sur les ventes, effectuer des analyses statistiques sur les données démographiques, visualiser les données sur les changements climatiques

Un objet Séries est un objet unidimensionnel de type tableau contenant :

- une séquence de valeurs
- un tableau associé d'étiquettes de données, appelé index.

```
class pandas.Series(data=None, index=None, dtype=None, name=None, copy=None,  
fastpath=False) [source]
```

1 - Introduction à l'Analyse et à la Manipulation de Données

-> la bibliothèque Pandas : Séries - data & index

```
[1]: import pandas
if __name__ == '__main__':
    import pandas as pd
    a = [14, 27, "232"]
    myvar = pd.Series(data=a)
    print(myvar)
    print(myvar[1])
```



```
if __name__ == '__main__':
    import pandas as pd
    a = [14, 27, "232"]
    myvar = pd.Series(a)
    print(myvar)
    print(myvar[1])
```

```
0      14
1      27
2     232
dtype: object
27
```

```
if __name__ == '__main__':
    import pandas as pd
    a = [14, 27, "232"]
    myvar = pd.Series(a, index = ["a", "b", "g"])
    print(myvar)
    print(myvar["b"])
```

```
a      14
b      27
g     232
dtype: object
27
```

1 - Introduction à l'Analyse et à la Manipulation de Données

-> la bibliothèque Pandas : Séries - data & index

```
[3]: if __name__ == '__main__':
    import pandas as pd
    a = {"a": 14, "b": 27, "g": "232"}
    myvar = pd.Series(a)
    print(myvar)
    print(myvar["b"])
```

```
a    14
b    27
g    232
dtype: object
27
```

```
[2]: if __name__ == '__main__':
    import pandas as pd
    a = {"a": 14, "b": 27, "g": "232"}
    myvar = pd.Series(a, index=["a", "d", "g"])
    print(myvar)
    print(myvar["g"])
    print(myvar["d"])
```

```
a    14
d    NaN
g    232
dtype: object
232
nan
```

1 - Introduction à l'Analyse et à la Manipulation de Données

-> la bibliothèque Pandas : Séries - name

```
[4]: if __name__ == '__main__':
    import pandas as pd
    a = [14, 27, "232"]
    myvar = pd.Series(data=a, name="myserie")
    print(myvar)
```

```
0      14
1      27
2     232
Name: myserie, dtype: object
```

1 - Introduction à l'Analyse et à la Manipulation de Données

-> la bibliothèque Pandas : Séries - numpy

```
[23]: if __name__ == '__main__':
    import pandas as pd
    import pandas as np
    a = np.array([14, 27, 232])
    myvar = pd.Series(data=a, copy=False)
    print(myvar)
    myvar[0]=10
    print(myvar)
    print(a)
```

```
# import numpy as np
import numpy as np
```

```
0    14
1    27
2   232
dtype: Int64
0    10
1    27
2   232
dtype: Int64
<IntegerArray>
[10, 27, 232]
Length: 3, dtype: Int64
```

1 - Introduction à l'Analyse et à la Manipulation de Données

-> la bibliothèque Pandas : Séries - copy

```
[18]: if __name__ == '__main__':
    import pandas as pd
    a = [14, 27, "232"]
    myvar = pd.Series(data=a, copy=True)
    print(myvar)
    myvar[0]=10
    print(myvar)
    print(a)
```

```
0    14
1    27
2   232
dtype: object
0    10
1    27
2   232
dtype: object
[14, 27, '232']
```

```
[16]: if __name__ == '__main__':
    import pandas as pd
    a = [14, 27, "232"]
    myvar = pd.Series(data=a, copy=False) #par default
    print(myvar)
    myvar[0]=10
    print(myvar)
    print(a)
```

```
0    14
1    27
2   232
dtype: object
0    10
1    27
2   232
dtype: object
[14, 27, '232']
```

1 - Introduction à l'Analyse et à la Manipulation de Données

-> la bibliothèque Pandas : Séries - dtype

```
[31]: if __name__ == '__main__':
    import pandas as pd
    import pandas as np
    a = np.array([14, 27, 232])
    myvar = pd.Series(a,dtype=type(2.5))
    print(myvar)
    myvar[0]=10
    print(myvar)
    print(a)

0    14.0
1    27.0
2   232.0
dtype: float64
0    10.0
1    27.0
2   232.0
dtype: float64
<IntegerArray>
[14, 27, 232]
Length: 3, dtype: Int64
```

Attributes

Axes

<code>Series.index</code>	The index (axis labels) of the Series.
<code>Series.array</code>	The ExtensionArray of the data backing this Series or Index.
<code>Series.values</code>	Return Series as ndarray or ndarray-like depending on the dtype.
<code>Series.dtype</code>	Return the dtype object of the underlying data.
<code>Series.shape</code>	Return a tuple of the shape of the underlying data.
<code>Series nbytes</code>	Return the number of bytes in the underlying data.
<code>Series.ndim</code>	Number of dimensions of the underlying data, by definition 1.
<code>Series.size</code>	Return the number of elements in the underlying data.
<code>Series.T</code>	Return the transpose, which is by definition self.
<code>Series.memory_usage</code> ([<code>index</code> , <code>deep</code>])	Return the memory usage of the Series.
<code>Series.hasnans</code>	Return True if there are any NaNs.
<code>Series.empty</code>	Indicator whether Series/DataFrame is empty.
<code>Series.dtypes</code>	Return the dtype object of the underlying data.
<code>Series.name</code>	Return the name of the Series

 On this page

Constructor
Attributes
 Conversion
 Indexing, iteration
 Binary operator functions
 Function application, GroupBy & window
 Computations / descriptive stats
 Reindexing / selection / label manipulation
 Missing data handling
 Reshaping, sorting
 Combining / comparing / joining / merging
 Time Series-related
 Accessors
 Plotting
 Serialization / IO / conversion

 Show Source

Input/output
 General functions
Series
`pandas.Series`
`pandas.Series.index`
`pandas.Series.array`
`pandas.Series.values`
`pandas.Series.dtype`
`pandas.Series.shape`
`pandas.Series.nbytes`
`pandas.Series.ndim`
`pandas.Series.size`
`pandas.Series.T`
`pandas.Series.memory_usage`
`pandas.Series.hasnans`
`pandas.Series.empty`
`pandas.Series.dtypes`
`pandas.Series.name`
`pandas.Series.flags`
`pandas.Series.set_flags`
`pandas.Series.astype`

<https://pandas.pydata.org/pandas-docs/stable/api.html>

<http://www.python-simple.com/python-pandas/creation-series.php>



<https://www.kaggle.com/code/arnopub/pandas-pr-sentation-de-l-objet-series>

kaggle

<https://www.geeksforgeeks.org/python-pandas-series/?ref=lbp>



Le type DataFrame (tables de données) sont des tableaux à deux (ou plus) dimensions, avec des étiquettes (labels) sur les lignes et les colonnes.

Les données ne sont pas forcément homogènes et certaines données peuvent être manquantes.

```
class pandas.DataFrame(data=None, index=None, columns=None, dtype=None,  
copy=None)
```

[\[source\]](#)

<https://pandas.pydata.org/docs/reference/frame.html>

1 - Introduction à l'Analyse et à la Manipulation de Données

-> la bibliothèque Pandas : DataFrame

```
[1]: import pandas as pd

# Définition des données
marques = ["Renault", "Peugeot", "Citroen", "Volkswagen", "Audi"]
modeles = ["Clio", "208", "C3", "Golf", "A3"]
annees = [2023, 2022, 2021, 2023, 2022]
kilometrages = [10000, 20000, 30000, 50000, 10000]
prix = [15000, 20000, 25000, 30000, 35000]

# Création du dataframe
df = pd.DataFrame({
    "marque": marques,
    "modele": modeles,
    "annee": annees,
    "kilometrage": kilometrages,
    "prix": prix
})

# Affichage du dataframe
print(df)
```

	marque	modele	annee	kilometrage	prix
0	Renault	Clio	2023	10000	15000
1	Peugeot	208	2022	20000	20000
2	Citroen	C3	2021	30000	25000
3	Volkswagen	Golf	2023	50000	30000
4		Audi	2022	10000	35000

1 - Introduction à l'Analyse et à la Manipulation de Données

-> la bibliothèque Pandas : Read CSV

```
Pays,Type,Nombre d'espèces
France,Comestibles,100
France,Toxiques,20
France,Inconnus,10
Italie,Comestibles,150
Italie,Toxiques,30
Italie,Inconnus,20
Espagne,Comestibles,200
Espagne,Toxiques,40
Espagne,Inconnus,30
Allemagne,Comestibles,250
Allemagne,Toxiques,50
Allemagne,Inconnus,40
Royaume-Uni,Comestibles,300
Royaume-Uni,Toxiques,60
Royaume-Uni,Inconnus,50
```

DataChampignons.csv

```
import pandas as pd

# Lecture du fichier de données
df = pd.read_csv("DataChampignons.csv")

print("Affichage de la taille de la série")
print(df.shape)
print("Affichage des colonnes de la série")
print(df.columns)
print("Affichage de la série")
print(df)
print("Création de la série")
serie = df["Type"]
print("Affichage de la série")
print(serie)
```

1 - Introduction à l'Analyse et à la Manipulation de Données

-> la bibliothèque Pandas : Read CSV

```
Affichage de la taille de la série
(15, 3)
Affichage des colonne de la série
Index(['Pays', 'Type', 'Nombre d'espèces'], dtype='object')
Affichage de la série
      Pays        Type  Nombre d'espèces
0    France  Comestibles            100
1    France    Toxiques             20
2    France    Inconnus             10
3    Italie  Comestibles            150
4    Italie    Toxiques             30
5    Italie    Inconnus             20
6   Espagne  Comestibles            200
7   Espagne    Toxiques             40
8   Espagne    Inconnus             30
9  Allemagne  Comestibles            250
10  Allemagne    Toxiques             50
11  Allemagne    Inconnus             40
12  Royaume-Uni  Comestibles            300
13  Royaume-Uni    Toxiques             60
14  Royaume-Uni    Inconnus             50
Création de la série
Affichage de la série
0    Comestibles
1    Toxiques
2    Inconnus
3    Comestibles
4    Toxiques
5    Inconnus
6    Comestibles
7    Toxiques
8    Inconnus
9    Comestibles
10   Toxiques
11   Inconnus
12   Comestibles
13   Toxiques
14   Inconnus
Name: Type, dtype: object
```

1 - Introduction à l'Analyse et à la Manipulation de Données -> la bibliothèque Pandas : Read CSV

usecol : pour sélectionner certaines colonnes du fichier csv

skiprows : pour sauter des lignes

skipfooter : pour des lignes en démarrant en bas

```
df = pd.read_csv("DataChampignons.csv")
print(df.shape)
print(df.columns)

df = pd.read_csv("DataChampignons.csv",usecols=[1,2])
print(df.shape)
print(df.columns)

df = pd.read_csv("DataChampignons.csv",skiprows=2)
print(df.shape)

df = pd.read_csv("DataChampignons.csv",skipfooter=3, engine='python')
print(df.shape)
```

```
Name: Type, dtype: object
(15, 3)
Index(['Pays', 'Type', 'Nombre d'espèces'], dtype='object')
(15, 2)
Index(['Type', 'Nombre d'espèces'], dtype='object')
(13, 3)
(12, 3)
```

1 - Introduction à l'Analyse et à la Manipulation de Données

-> la bibliothèque Pandas : Write CSV

```
df = pd.read_csv("DataChampignons.csv")

df.to_csv('DataChampignons2.csv', index=False)

df.to_csv('DataChampignons3.csv', index=True)
```

Pays	Type	Nombre d'espèces
France	Comestibles	100
France	Toxiques	20
France	Inconnus	10
Italie	Comestibles	150
Italie	Toxiques	30
Italie	Inconnus	20
Espagne	Comestibles	200
Espagne	Toxiques	40
Espagne	Inconnus	30
Allemagne	Comestibles	250
Allemagne	Toxiques	50
Allemagne	Inconnus	40
Royaume-Uni	Comestibles	300
Royaume-Uni	Toxiques	60
Royaume-Uni	Inconnus	50

DataChampignons2.csv

	Pays	Type	Nombre d'espèces
0	France	Comestibles	100
1	France	Toxiques	20
2	France	Inconnus	10
3	Italie	Comestibles	150
4	Italie	Toxiques	30
5	Italie	Inconnus	20
6	Espagne	Comestibles	200
7	Espagne	Toxiques	40
8	Espagne	Inconnus	30
9	Allemagne	Comestibles	250
10	Allemagne	Toxiques	50
11	Allemagne	Inconnus	40
12	Royaume-Uni	Comestibles	300
13	Royaume-Uni	Toxiques	60
14	Royaume-Uni	Inconnus	50

DataChampignons3.csv

1 - Introduction à l'Analyse et à la Manipulation de Données

-> la bibliothèque Pandas : Write Json

```
import json
dicJson=pd.Series({
    "Pays": {"0": "France", "1": "France", "2": "France", "3": "Italie", "4": "Italie", "5": "Italie"}, \
    "Type": {"0": "Comestibles", "1": "Toxiques", "2": "Inconnus", "3": "Comestibles", "4": "Toxiques", "5": "Inconnus"}, \
    "Nombre d'espèces": {"0": 100, "1": 20, "2": 10, "3": 150, "4": 30, "5": 20}})
```

```
print("sauvegarde json orient = index ( valeur par defaut ) ")
# <=>dicJson.to_json('DataChampignons.json',orient="index")
```

```
dicJson.to_json('DataChampignons.json')
```

```
result=json.load(open('DataChampignons.json'))
```

```
print(json.dumps(result, indent=4))
```

```
    "sauvegarde json orient = index ( valeur par defaut ) "
    "Pays": {
        "0": "France",
        "1": "France",
        "2": "France",
        "3": "Italie",
        "4": "Italie",
        "5": "Italie"
    },
    "Type": {
        "0": "Comestibles",
        "1": "Toxiques",
        "2": "Inconnus",
        "3": "Comestibles",
        "4": "Toxiques",
        "5": "Inconnus"
    },
    "Nombre d'espèces": {
        "0": 100,
        "1": 20,
        "2": 10,
        "3": 150,
        "4": 30,
        "5": 20
    }
}
```

<https://pythonbasics.org/pandas-json/>

1 - Introduction à l'Analyse et à la Manipulation de Données

-> la bibliothèque Pandas : Write Json

```
print("sauvegarde json orient = split")
dicJson.to_json('DataChampignons.json',orient="split")
result=json.load(open('DataChampignons.json'))
print(json.dumps(result, indent=4))
```

```
sauvegarde json orient = split
{
    "name": null,
    "index": [
        "Pays",
        "Type",
        "Nombre d'espèces"
    ],
    "data": [
        {
            "0": "France",
            "1": "France",
            "2": "France",
            "3": "Italie",
            "4": "Italie",
            "5": "Italie"
        },
        {
            "0": "Comestibles",
            "1": "Toxiques",
            "2": "Inconnus",
            "3": "Comestibles",
            "4": "Toxiques",
            "5": "Inconnus"
        },
        {
            "0": 100,
            "1": 20,
            "2": 10,
            "3": 150,
            "4": 30,
            "5": 20
        }
    ]
}
```

1 - Introduction à l'Analyse et à la Manipulation de Données

-> la bibliothèque Pandas : Write Json

```
print("sauvegarde json orient = records")
dicJson.to_json('DataChampignons.json',orient="records")
result=json.load(open('DataChampignons.json'))
print(json.dumps(result, indent=4))
```

```
sauvegarde json orient = records
[
  {
    "0": "France",
    "1": "France",
    "2": "France",
    "3": "Italie",
    "4": "Italie",
    "5": "Italie"
  },
  {
    "0": "Comestibles",
    "1": "Toxiques",
    "2": "Inconnus",
    "3": "Comestibles",
    "4": "Toxiques",
    "5": "Inconnus"
  },
  {
    "0": 100,
    "1": 20,
    "2": 10,
    "3": 150,
    "4": 30,
    "5": 20
  }
]
```



1 - Introduction à l'Analyse et à la -> la bibliothèque Pandas : Write J

```
print("sauvegarde json orient = table")
dicJson.to_json('DataChampignons.json',orient="table")
result=json.load(open('DataChampignons.json'))
print(json.dumps(result, indent=4))
```

```
sauvegarde json orient = table
{
  "schema": {
    "fields": [
      {
        "name": "index",
        "type": "string"
      },
      {
        "name": "values",
        "type": "string"
      }
    ],
    "primaryKey": [
      "index"
    ],
    "pandas_version": "1.4.0"
  },
  "data": [
    {
      "index": "Pays",
      "values": {
        "0": "France",
        "1": "France",
        "2": "France",
        "3": "Italie",
        "4": "Italie",
        "5": "Italie"
      }
    },
    {
      "index": "Type",
      "values": {
        "0": "Comestibles",
        "1": "Toxiques",
        "2": "Inconnus",
        "3": "Comestibles",
        "4": "Toxiques",
        "5": "Inconnus"
      }
    },
    {
      "index": "Nombre d'espèces",
      "values": {
        "0": 100,
        "1": 20,
        "2": 10,
        "3": 150,
        "4": 30,
        "5": 20
      }
    }
  ]
}
```

1 - Introduction à l'Analyse et à la Manipulation de Données

-> la bibliothèque Pandas : Read Json

```
sauvegarde json orient = index ( valeur par defaut )
```

```
{
  "Pays": {
    "0": "France",
    "1": "France",
    "2": "France",
    "3": "Italie",
    "4": "Italie",
    "5": "Italie"
  },
  "Type": {
    "0": "Comestibles",
    "1": "Toxiques",
    "2": "Inconnus",
    "3": "Comestibles",
    "4": "Toxiques",
    "5": "Inconnus"
  },
  "Nombre d'especes": {
    "0": 100,
    "1": 20,
    "2": 10,
    "3": 150,
    "4": 30,
    "5": 20
  }
}
```

	Pays	Type	Nombre d'especes
0	France	Comestibles	100
1	France	Toxiques	20
2	France	Inconnus	10
3	Italie	Comestibles	150
4	Italie	Toxiques	30
5	Italie	Inconnus	20

```

print("sauvegarde json orient = index ( valeur par defaut ) ")
dicJson.to_json('DataChampignons.json')
result=json.load(open('DataChampignons.json'))
print(json.dumps(result, indent=4))

df = pd.read_json('DataChampignons.json')

print(df.to_string())

```

https://jhub.cnam.fr/doc/notebooks/Manipulations_de_donnees_Python_Pandas.html

le **cnam**

<https://moncoachdata.com/blog/manipulation-de-donnees-avec-pandas/>

<https://egallic.fr/Enseignement/Python/pandas.html>

1 - Introduction à l'Analyse et à la Manipulation de Données

-> MEMENTO

Mémento Python

https://www.foucherconnect.fr/miniliens/mie/2019/9782216156009/15493_P4_maths_doc2.pdf

https://joly415.perso.math.cnrs.fr/memento_python.pdf

https://perso.limsi.fr/pointal/_media/python:cours:mementopython3.pdf

Mémento Python 3 pour le calcul scientifique

https://www.cpge-brizeux.fr/wordpress/wp-content/uploads/CI9_m03-Memento-Python-ENSAM-comp.pdf

Mémo – Pandas pour le Traitement de Données

<https://moncoachdata.com/blog/pandas-pour-le-traitement-de-donnees/>

Livres

<https://riptutorial.com/Download/pandas-fr.pdf>

2 - Rappel mathématique

Les mathématiques les plus importantes pour la science des données sont les suivantes :

- **Probabilités et statistiques** : les probabilités et les statistiques sont utilisées pour décrire la variabilité et l'incertitude des données,
 - Utiliser pour comprendre et analyser les données, et pour construire des modèles prédictifs.
- **Algèbre linéaire** : l'algèbre linéaire est une branche des mathématiques qui étudie les vecteurs, les matrices et les espaces vectoriels.
 - Utiliser dans de nombreux domaines de la science des données, notamment l'apprentissage automatique, la visualisation de données et la compression de données.
- **Analyse numérique** : l'analyse numérique est une branche des mathématiques qui étudie les méthodes numériques pour résoudre des problèmes mathématiques
 - Utiliser dans de nombreux domaines de la science des données, notamment l'apprentissage automatique, l'optimisation et la modélisation.

suite

2 - Rappel mathématique

- **Calcul différentiel et intégral** : le calcul est une branche des mathématiques qui étudie les changements des quantités
 - Utiliser dans de nombreux domaines de la science des données, notamment l'apprentissage automatique, la modélisation et l'optimisation.
- **Logique** : la logique est une branche des mathématiques qui étudie les raisonnements corrects
 - Utiliser dans de nombreux domaines de la science des données, notamment l'apprentissage automatique, la modélisation et l'analyse de données.
- **Graphes et théorie des graphes** : Certains problèmes de science des données peuvent être modélisés comme des graphes, et la théorie des graphes
 - Utiliser pour représenter des relations complexes de manière efficace, des corrélations entre les données, détection de fraudes

Probabilités : La probabilité est l'étude mathématique des événements aléatoires.

- Permet de décrire la variabilité et l'incertitude des données.
- Utilisé dans de nombreux domaines de la science des données, notamment l'apprentissage automatique, la visualisation de données et la prise de décision.

Statistiques : Les statistiques sont l'étude des données.

- Permet de collecter, organiser, analyser et interpréter des données.
- Utilisé dans de nombreux domaines de la science des données, notamment l'analyse descriptive, l'analyse inférentielle et la modélisation

Lien entre probabilités et statistiques

- Les probabilités et les statistiques sont étroitement liées.
- La théorie des probabilités fournit le cadre théorique pour les statistiques.
- **Les statistiques étudient des événements déroulés dans le passé à partir d'un grand nombre de données et les probabilités s'intéressent quant à elles à l'avenir**

- **Théorème de Bayes** est utilisé dans de nombreux domaines de la science des données, notamment la classification, la détection d'anomalies et la prise de décision.
- **Loi forte des grands nombres** est utilisée pour estimer la moyenne d'une population à partir d'un échantillon.
- **Loi des erreurs types** est utilisée pour calculer l'intervalle de confiance d'une estimation.

2 - Rappel mathématique -> Probabilités : Théorème de Bayes

Théorème de Bayes permet de calculer la probabilité d'un événement A, sachant qu'un événement B s'est déjà produit.

Théorème : Soit (A_n) un système complet d'événements, tous de probabilité non nulle. Alors, pour tout événement B, on a :

$$P(B) = \sum_{n \geq 1} P_{A_n}(B)P(A_n).$$

Si de plus $P(B) > 0$, on a pour tout entier k l'égalité :

$$P_B(A_k) = \frac{P_{A_k}(B)P(A_k)}{P(B)} = \frac{P_{A_k}(B)P(A_k)}{\sum_{n \geq 1} P_{A_n}(B)P(A_n)}.$$

Cette formule est souvent utilisée lorsque le système complet est constitué de A et \bar{A} , un événement et son contraire. Dans ce cas, la formule se simplifie en :

$$P_B(A) = \frac{P_A(B)P(A)}{P(B)} = \frac{P_A(B)P(A)}{P_A(B)P(A) + P_{\bar{A}}(B)P(\bar{A})}.$$

<https://www.bibmath.net/dico/index.php?action=affiche&quoi=../b/bayes.html>

https://fr.wikipedia.org/wiki/Th%C3%A9or%C3%A8me_de_Bayes

<https://www.linkedin.com/pulse/le-th%C3%A9or%C3%A8me-de-bayes-cas-pratique-virginie-mathivet/?originalSubdomain=fr>

Vous êtes directeur de cabinet du ministre de la santé. Une maladie est présente dans la population, dans la proportion d'une personne malade sur 10000. Un responsable d'un grand laboratoire pharmaceutique vient vous vanter son nouveau test de dépistage : si une personne est malade, le test est positif à 99%. Si une personne n'est pas malade, le test est positif à 0,1%.

Ces chiffres ont l'air excellent, vous ne pouvez qu'en convenir. Toutefois, avant d'autoriser la commercialisation de ce test, vous faites appel au statisticien du ministère : ce qui vous intéresse, ce n'est pas vraiment les résultats présentés par le laboratoire, c'est la probabilité qu'une personne soit malade si le test est positif. La formule de Bayes permet de calculer cette probabilité.

On note M l'événement : "La personne est malade", et T l'événement : "Le test est positif". Le but est de calculer $P_T(M)$. Les données que vous avez en main sont $P(M) = 0,0001$ (et donc $P(\bar{M}) = 0,9999$), $P_M(T) = 0,99$ et $P_{\bar{M}}(T) = 0,001$. La formule de Bayes donne :

$$\begin{aligned} P_T(M) &= \frac{P_M(T)P(M)}{P_M(T)P(M) + P_{\bar{M}}(T)P(\bar{M})} \\ &= \frac{10^{-4} \times 0,99}{10^{-4} \times 0,99 + 0,9999 \times 10^{-3}} \simeq 0,09. \end{aligned}$$

C'est catastrophique! Il n'y a que 9% de chances qu'une personne positive au test soit effectivement malade! C'est tout le problème des tests de dépistage pour des maladies rares : ils doivent être excessivement performants, sous peine de donner beaucoup trop de "faux-positifs".

2 - Rappel mathématique

-> Probabilités : Loi forte des grands nombres

La loi des grands nombres stipule que la moyenne d'un échantillon de données tend vers la moyenne de la population lorsque la taille de l'échantillon augmente, donne une convergence presque sûre de se réaliser

Considérons une suite $(X_n)_{n \in \mathbb{N}}$ de **variables aléatoires indépendantes** qui suivent la même **loi de probabilité**, intégrables, *i. e.* $E(|X_0|) < +\infty$. En reprenant les notations ci-dessus, la loi forte des grands nombres précise que $(\bar{X}_n)_{n \in \mathbb{N}}$ converge vers $E(X)$ « *presque sûrement* ».

C'est-à-dire que :

Théorème — $\mathbb{P} \left(\lim_{n \rightarrow +\infty} \bar{X}_n = E(X) \right) = 1.$

Autrement dit, selon la loi forte des grands nombres, la moyenne empirique est un **estimateur fortement convergent** de l'espérance.

https://fr.wikipedia.org/wiki/Loi_des_grands_nombres

<https://www.techno-science.net/definition/5994.html>

2 - Rappel mathématique

-> Probabilités : Loi forte des grands nombres

Écarts entre pile et face lors d'une simulation de lancers sur Python

Nombre de lancers	100	1000	10000	100000
Nombre de « pile »	51	477	5074	50026
Nombre de « face »	49	523	4926	49974
Écart absolu	2	46	148	52
Écart relatif	2%	4,6%	1,48%	0,05%

Simulation sur Python de 100000 lancers d'un dé équilibré, de moyenne 3,50131

Valeur	1	2	3	4	5	6
Occurrences	16620	16815	16558	16687	16461	16859

2 - Rappel mathématique -> Probabilités : Loi des erreurs types

Loi des erreurs types stipule que la variance d'une estimation de la moyenne d'une population est proportionnelle à $1/\sqrt{n}$, où n est la taille de l'échantillon.

Population [\[modifier \]](#) [\[modifier le code \]](#)

L'erreur type de la moyenne vaut :

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$$

avec

σ est l'écart type de la population ;

n est la taille de l'échantillon (nombre de tirages).

Estimation [\[modifier \]](#) [\[modifier le code \]](#)

Lorsque l'écart type est inconnu, l'erreur type de la moyenne est souvent déterminé à partir de l'estimateur avec biais de l'écart type s , sous réserve que les tirages soient indépendants :

$$\sigma_{\bar{x}} \approx \frac{s}{\sqrt{n}}$$

https://fr.wikipedia.org/wiki/Erreur_type

https://www.deleze.name/marcel/sec2/applmaths/csud/calcul_erreur/1_a_2-calcul_erreur.pdf

Supposons que nous avons un échantillon de 100 individus et que la moyenne de l'échantillon est de 10. L'écart type de l'échantillon est de 2.

L'erreur type de la moyenne de l'échantillon est de :

$$\sigma^e = \sigma / \sqrt{n} = 2 / \sqrt{100} = 0,2$$

Cela signifie que la valeur réelle de la moyenne de la population est susceptible de se trouver dans l'intervalle [9,8 ; 10,2] avec une probabilité de 95 %.

2 - Rappel mathématique -> Statistiques descriptives

- **Mesures de dispersion** sont utilisées pour décrire la variabilité d'un ensemble de données.
 - **Écart type** : mesure de la dispersion des données autour de la moyenne.
 - **Variance** : moyenne des carrés des distances des données à la moyenne
 - **Ecart Interquartile** : différence entre le troisième quartile et le premier quartile.
- **Mesures de tendance centrale** sont utilisées pour décrire la tendance centrale d'un ensemble de données.
 - **Moyenne** : somme des données divisée par le nombre de données. Elle est la mesure de tendance centrale la plus couramment utilisée.
 - **Médiane** : valeur au milieu d'un ensemble de données, lorsque les données sont triées par ordre croissant ou décroissant. Elle est utilisée pour décrire la tendance centrale d'un ensemble de données qui n'est pas symétrique.
 - **Mode** valeur la plus fréquente dans un ensemble de données. Elle est utilisée pour décrire la tendance centrale d'un ensemble de données qui n'est pas symétrique.
 - **Etendue** différence entre la plus petite valeur et la plus grande
- **Mesure de la relation entre deux variables**
 - **Covariance** : mesure de la relation entre deux variables. Elle est utilisée pour identifier les relations entre des variables.
 - **Corrélation** : mesure de la force de la relation entre deux variables. Elle est utilisée pour identifier les relations entre des variables.

2 - Rappel mathématique

→ Statistiques descriptives : Mesures de tendance centrale

l'Etudiant

QU'EST-CE QUE LE MODE ?

- Le mode est la valeur qui revient le plus souvent dans une série

Le mode est le prix qui revient le plus souvent pour les baskets rouges, c'est-à-dire 49,90€


 l'Etudiant **TROUVER LA MÉDIANE D'UNE SÉRIE IMPAIRE**

- Si la série est impaire, la médiane sera la valeur du milieu
- La valeur 8 partage en deux parts égales la série



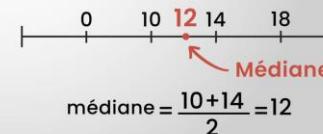
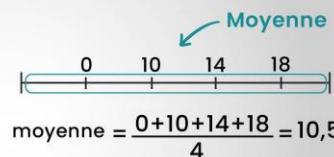
l'Etudiant

QUELLE DIFFÉRENCE ENTRE MÉDIANE ET MOYENNE ?

- La moyenne est la somme des valeurs divisé par leur nombre

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

- La médiane divise une série en deux parts égales



<https://www.letudiant.fr/college/methodologie-college/article/le-calcul-de-mediane-soyez-meilleur-que-la-moyenne.html>

- **Variance** : moyenne des carrés des distances des données à la moyenne. Elle est utilisée pour calculer l'écart type

$$V = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$$

- **Estimateur non biaisé de la variance** $S^2 = \frac{\sum (x_i - \bar{X})^2}{n-1}$ ←
- **Écart type** : mesure de la dispersion des données autour de la moyenne. Il est calculé en calculant la racine carrée de la variance.

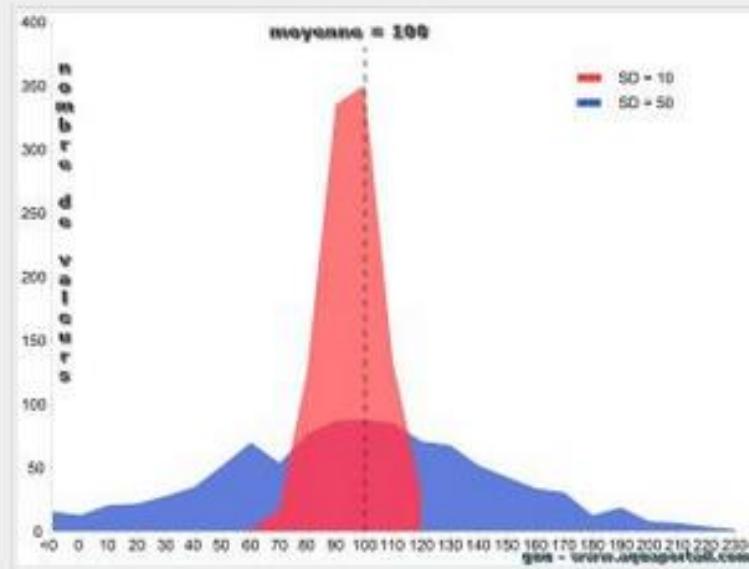
$$\sigma = \sqrt{V} = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2} = \sqrt{\frac{1}{n} \left(\sum_{i=1}^n x_i^2 \right) - \bar{x}^2}$$

[https://fr.wikipedia.org/wiki/Variance_\(math%C3%A9matiques\)](https://fr.wikipedia.org/wiki/Variance_(math%C3%A9matiques))

https://fr.wikipedia.org/wiki/%C3%89cart_type

https://www.umr-lastig.fr/paul-chapron/resources/cours_site/dispersion.html

Différences de variance entre deux populations :



Exemple de deux échantillons provenant de deux populations ayant les mêmes variances moyennes mais différentes, c'est-à-dire ayant les mêmes écarts-types moyens et différents. La population en échantillon rouge a une moyenne de 100 et SD 10; l'échantillon bleu a une moyenne de 100 et un écart-type 50. Chaque échantillon a 1000 valeurs tirées au hasard à partir d'une distribution gaussienne avec les paramètres spécifiés.

2 - Rappel mathématique

-> Statistiques descriptives : Mesures de dispersion

- **Ecart Interquartile** : différence entre le troisième quartile et le premier quartile. Il est utilisé pour décrire la dispersion d'un ensemble de données

Tableau de données [\[modifier \]](#) [\[modifier le code \]](#)

Valeurs	%	Quartile
1	102	
2	104	
3	105	Q ₁
4	107	
5	108	
6	109	Q ₂ (médiane)
7	110	
8	112	
9	115	Q ₃
10	116	
11	118	

L'écart interquartile de cette distribution de données (noté EI), est $EI = Q_3 - Q_1 = 115 - 105 = 10$.

- **Étendue** : différence entre la plus petite valeur et la plus grande

https://www.umr-lastig.fr/paul-chapron/resources/cours_site/dispersion.html

https://fr.wikipedia.org/wiki/%C3%89cart_interquartile

<https://www150.statcan.gc.ca/n1/edu/power-pouvoir/ch12/5214890-fra.htm>

<https://www.maxicours.com/se/cours/statistiques-etendue-mediane-quartiles/>

- **Covariance** : mesure de la relation entre deux variables. Elle est utilisée pour identifier les relations entre des variables.

La formule générale de covariance :

**formule de covariance
de la population**

$$Cov(x, y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{N}$$

covariance de l'échantillon

$$Cov(x, y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{N-1}$$

www.aquaportal.com

La formule de covariance est l'une des formules statistiques utilisées pour déterminer la relation entre deux variables ou nous pouvons dire que la covariance montre la relation statistique entre deux variances entre les deux variables.

<https://www.aquaportal.com/dictionnaire/definition/13155/covariance>

2 - Rappel mathématique

→ Statistiques descriptives : Mesure de la relation entre deux variables

- **Corrélation** : mesure de la force de la relation entre deux variables. Elle est utilisée pour identifier les relations entre des variables.

La formule pour calculer le coefficient de corrélation :



Correlation
Coefficient =
$$\frac{\sum [(X - X_m) * (Y - Y_m)]}{\sqrt{[\sum (X - X_m)^2 * \sum (Y - Y_m)^2]}}$$

www.aquaportal.com

En statistique, les résultats spécifiques sont directement liés à d'autres situations ou variables, et le coefficient de corrélation mesure l'association directe de deux variables ou situations. Ces variables présentent un coefficient de corrélation positif lorsqu'elles évoluent simultanément dans la même direction. S'ils se déplacent dans des directions opposées, un coefficient de corrélation négatif est présent.

<https://www.aquaportal.com/dictionnaire/definition/13153/coefficient-de-correlation>

L'objectif des statistiques inférentielles est de déterminer si un résultat observé sur un échantillon est significatif ou non

Test d'hypothèse est une méthode statistique qui permet de tester l'hypothèse que la valeur d'un paramètre est égale à une valeur donnée.

Pour tester une hypothèse, il faut d'abord formuler deux hypothèses :

- **L'hypothèse nulle (H₀)** est l'hypothèse que le résultat observé n'est pas significatif.
- **L'hypothèse alternative (H₁)** est l'hypothèse que le résultat observé est significatif.

Test statistique

- Un test statistique est utilisé pour déterminer si l'hypothèse nulle doit être rejetée ou non.(généralement fixé à 5 % ou 1 %.)

https://www.imo.universite-paris-saclay.fr/~pierre.pansu/web_ifips/Tests.pdf

Estimation est le processus d'obtention d'une approximation d'une valeur inconnue.

Il existe deux types d'estimation :

- **Estimation ponctuelle** : Elle consiste à déterminer une valeur unique pour le paramètre de population.
- **Estimation par intervalle** : Elle consiste à déterminer un intervalle de valeurs probables pour le paramètre de population.

Il existe de nombreuses méthodes d'estimation, chacune adaptée à un type de situation spécifique.

Les méthodes d'estimation ponctuelle les plus courantes sont :

- La moyenne échantillonnage
- La médiane échantillonnage
- La mode échantillonnage

Les méthodes d'estimation par intervalle les plus courantes sont :

- L'intervalle de confiance
- L'intervalle de prédiction

Intervalle de confiance est un intervalle de valeurs qui a une probabilité donnée de contenir la valeur réelle d'un paramètre.

Calcul de l'intervalle de confiance (IC)

$$IC = \bar{x} \pm t \frac{\sigma}{\sqrt{n}}$$

- \bar{x} Moyenne sur l'échantillon
- t Coefficient fonction du niveau de confiance
- σ Écart type sur la population (connu ou estimé)
- n Taille de l'échantillon

<http://villemin.gerard.free.fr/aMaths/Statisti/IntConf.htm>

<https://www.demarcheiso17025.com/fiche007.html>

Statistique descriptive

- http://math.univ-lyon1.fr/~chekroun/Files/chekroun_statistiques.pdf
- <https://homepages.laas.fr/kader/StatDes.pdf>
- <https://www.imo.universite-paris-saclay.fr/~damien.thomine/fr/archives/Enseignement2021/IUT/C3-Stat-Pres-donnees.pdf>
- https://epub.ub.uni-muenchen.de/8377/1/oa_8377.pdf
- <https://slideplayer.fr/slide/13792329/>

L'algèbre linéaire est un domaine des mathématiques qui étudie les vecteurs, les matrices et les transformations linéaires.

Il est un outil fondamental pour l'analyse des données, car il peut être utilisé pour :

- Représenter les données de manière compacte et efficace.
- Manipuler et analyser les données en utilisant des opérations mathématiques simples.
- Construire et évaluer des modèles statistiques.
- Développer des algorithmes d'apprentissage automatique.

Un vecteur est un objet mathématique qui a une grandeur et une direction. Un vecteur peut être représenté par une colonne de nombres.

Une matrice est une table de nombres.

Les matrices peuvent être utilisées pour représenter des ensembles de vecteurs, des systèmes d'équations linéaires et des transformations linéaires.

https://www.unilim.fr/pages_perso/jean.debord/math/matrices/matrices.htm

<https://alir-jsturcotte.profweb.ca/sec-transfodef.html>

Une transformation linéaire est une fonction qui mappe des vecteurs vers d'autres vecteurs, en préservant les opérations d'addition et de produit scalaire.

Les transformations linéaires peuvent être représentées par des matrices.

Définition 2.1.3. Une fonction vectorielle de \mathbb{R}^n vers \mathbb{R}^m est une fonction T qui prend un vecteur $\vec{x} \in \mathbb{R}^n$ et lui associe un vecteur $\vec{y} \in \mathbb{R}^m$. On écrit alors $T(\vec{x}) = \vec{y}$.

<https://alir-jsturcotte.profweb.ca/sec-transfodef.html>

L'algèbre linéaire peut être utilisée pour résoudre une large gamme de problèmes d'analyse des données, notamment :

- Classification des données
- Régression linéaire et non linéaire
- Clustering des données
- Réduction de la dimensionnalité
- Détection d'anomalies
- Prévision

2 - Rappel mathématique -> Analyse numérique

L'analyse numérique est un domaine des mathématiques qui s'intéresse à la résolution d'équations et d'inéquations, à l'intégration et à la dérivation de fonctions, et à la solution d'autres problèmes mathématiques à l'aide d'algorithmes numériques

Les avantages de l'analyse numérique pour l'analyse des données sont notamment :

- **Efficacité** : Les algorithmes numériques peuvent être utilisés pour résoudre des problèmes complexes de manière plus efficace que les méthodes analytiques.
- **Précision** : Les algorithmes numériques peuvent être utilisés pour obtenir des solutions précises, même pour des problèmes avec des données bruitées ou incomplètes.
- **Flexibilité** : Les algorithmes numériques peuvent être adaptés à un large éventail de problèmes d'analyse des données.

2 - Rappel mathématique -> Analyse numérique

L'analyse numérique est utilisée dans une grande variété d'applications d'analyse des données, notamment :

- **Classification** : L'analyse numérique peut être utilisée pour construire des modèles de classification qui peuvent prédire l'appartenance d'une observation à une classe donnée.
- **Régression** : L'analyse numérique peut être utilisée pour construire des modèles de régression qui peuvent prédire une valeur d'une variable en fonction d'autres variables.
- **Clustering** : L'analyse numérique peut être utilisée pour regrouper des observations similaires en fonction de leurs caractéristiques.
- **Réduction de la dimensionnalité** : L'analyse numérique peut être utilisée pour réduire la dimensionnalité des données, tout en préservant les informations les plus importantes.
- **Détection d'anomalies** : L'analyse numérique peut être utilisée pour détecter les observations qui sont anormales par rapport au reste des données.

2 - Rappel mathématique -> Analyse numérique

Les différentes branches de l'analyse numérique peuvent être classées en fonction de la nature du problème à résoudre. Exemples de branches de l'analyse numérique :

- **L'analyse linéaire numérique** s'intéresse à la résolution d'équations et d'inéquations linéaires, ainsi qu'à l'intégration et à la dérivation de fonctions linéaires.
- **L'analyse non linéaire numérique** s'intéresse à la résolution d'équations et d'inéquations non linéaires, ainsi qu'à l'intégration et à la dérivation de fonctions non linéaires.
- **L'analyse numérique multidimensionnelle** s'intéresse à la résolution de problèmes mathématiques dans plusieurs dimensions
- **L'analyse numérique des équations différentielles** s'intéresse à la résolution d'équations différentielles.
- **L'analyse numérique des systèmes dynamiques** s'intéresse à l'étude du comportement des systèmes dynamiques.

2 - Rappel mathématique -> Analyse numérique

http://math.univ-lyon1.fr/~ebretin/fichier/analyse_numerique.pdf

https://www.ceremade.dauphine.fr/~legendre/enseignement/methnum/cohurs_ananum_dauphine.pdf

<https://www-fourier.ujf-grenoble.fr/~parisse/mat249/polyAnaNum.pdf>

Le calcul différentiel et intégral est un domaine des mathématiques qui s'intéresse à l'étude des fonctions et de leurs dérivées.

Il est un outil essentiel pour l'analyse des données, car il peut être utilisé pour :

- **Décrire les relations entre les variables**
- **Prédire les valeurs futures**
- **Optimiser les processus**

La dérivée d'une fonction est une mesure de sa variation à un point donné.

Elle peut être utilisée pour décrire la pente d'une courbe, la vitesse d'un objet ou la variation d'une quantité au fil du temps.

L'intégration est le processus inverse de la dérivation. Elle permet de trouver la valeur d'une fonction sur un intervalle donné.

Le calcul différentiel et intégral est un domaine des mathématiques qui s'intéresse à l'étude des fonctions et de leurs dérivées.

Il est un outil essentiel pour l'analyse des données, car il peut être utilisé pour :

- **Décrire les relations entre les variables**
- **Prédire les valeurs futures**
- **Optimiser les processus**

2 - Rappel mathématique -> Calcul différentiel et intégral

Le calcul différentiel et intégral peut être utilisé pour résoudre une grande variété de problèmes d'analyse des données, notamment :

- **Classification** : Les dérivées peuvent être utilisées pour construire des modèles de classification qui peuvent prédire l'appartenance d'une observation à une classe donnée.
- **Régression** : Les dérivées peuvent être utilisées pour construire des modèles de régression qui peuvent prédire une valeur d'une variable en fonction d'autres variables.
- **Clustering** : Les dérivées peuvent être utilisées pour regrouper des observations similaires en fonction de leurs caractéristiques.
- **Réduction de la dimensionnalité** : Les dérivées peuvent être utilisées pour réduire la dimensionnalité des données, tout en préservant les informations les plus importantes.
- **Détection d'anomalies** : Les dérivées peuvent être utilisées pour détecter les observations qui sont anormales par rapport au reste des données.

2 - Rappel mathématique -> Calcul différentiel et intégral

- https://fr.wikibooks.org/wiki/Calcul_différentiel_et_intégral_pour_débutant
- <https://www.math.univ-toulouse.fr/~jroyer/TD/2015-16-L2PS/L2PS-poly.pdf>

La logique est un outil puissant qui peut être utilisé pour résoudre des problèmes d'analyse des données. Elle peut être utilisée pour :

- **Identifier les relations entre les données**
- **Déduire des conclusions à partir des données**
- **Valider les résultats de l'analyse**

Il existe deux grands types de logique : la logique déductive et la logique inductive.

- **La logique déductive** part de prémisses connues pour en déduire une conclusion. Par exemple, si nous savons que tous les chats sont des mammifères et que tous les mammifères ont des poils, nous pouvons déduire que tous les chats ont des poils.
- **La logique inductive** part d'observations pour en déduire des généralisations. Par exemple, si nous observons que tous les chats que nous avons vus ont des poils, nous pouvons induire que tous les chats ont des poils.

La logique peut être utilisée pour résoudre une grande variété de problèmes d'analyse des données, notamment :

- **La classification** : la logique peut être utilisée pour construire des modèles de classification qui peuvent prédire l'appartenance d'une observation à une classe donnée.
- **La régression** : la logique peut être utilisée pour construire des modèles de régression qui peuvent prédire une valeur d'une variable en fonction d'autres variables.
- **Le clustering** : la logique peut être utilisée pour regrouper des observations similaires en fonction de leurs caractéristiques.
- **La réduction de la dimensionnalité** : la logique peut être utilisée pour réduire la dimensionnalité des données tout en préservant les informations les plus importantes.
- **La détection d'anomalies** : la logique peut être utilisée pour détecter les observations qui sont anormales par rapport au reste des données.

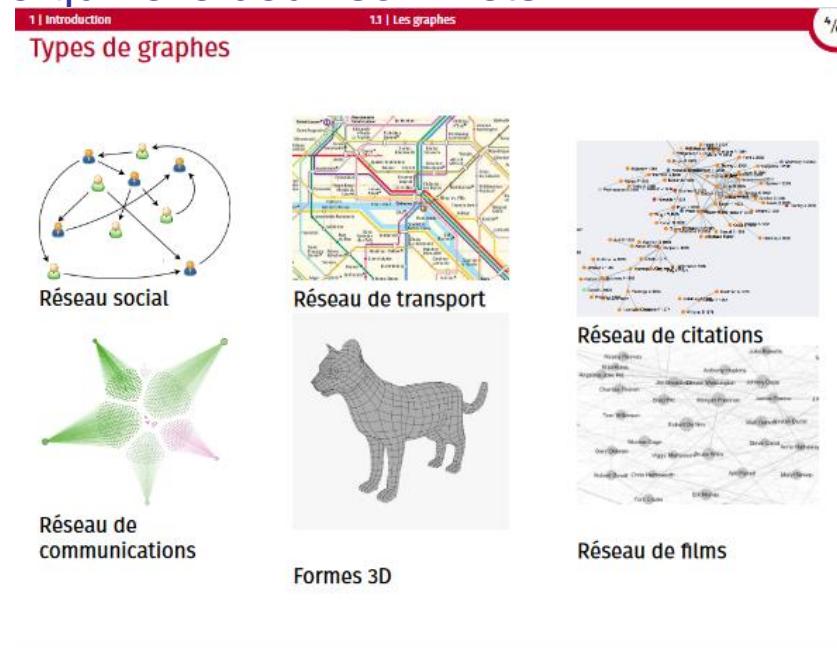
2 - Rappel mathématique -> Logique

- <https://www.apprendre-en-ligne.net/info/logique/logique.pdf>
- https://fac.umc.edu.dz/ista/pdf/cours/chapitre%203_alg%C3%A8bre_de-boole.pdf

2 - Rappel mathématique -> Graphes et théorie des graphes

La théorie des graphes est un domaine des mathématiques qui étudie les structures composées de points reliés par des lignes. Ces structures peuvent être utilisées pour modéliser une grande variété de phénomènes, tels que les réseaux sociaux, les systèmes biologiques et les transports.

Un graphe est un ensemble de sommets et d'arêtes. Un sommet est un point, et une arête est une ligne qui relie deux sommets.



<https://cedric.cnam.fr/vertigo/cours/RCP217/docs/RCP217-GraphML1.pdf>

La théorie des graphes peut être utilisée pour résoudre un large éventail de problèmes d'analyse de données, tels que :

- **La détection de communautés** : les graphes peuvent être utilisés pour identifier des groupes de données qui sont fortement interconnectés.
- **La recommandation** : les graphes peuvent être utilisés pour recommander des produits ou des services aux utilisateurs en fonction de leurs relations avec d'autres utilisateurs.
- **La détection d'anomalies** : les graphes peuvent être utilisés pour identifier des données qui ne correspondent pas au modèle attendu.

Exemples d'applications supplémentaires

- **L'analyse de réseaux sociaux** : les graphes peuvent être utilisés pour analyser les relations entre les utilisateurs de réseaux sociaux, tels que Facebook ou Twitter.
- **La visualisation de données** : les graphes peuvent être utilisés pour visualiser des données de manière informative et interactive.
- **La recherche d'informations** : les graphes peuvent être utilisés pour rechercher des informations dans des bases de données volumineuses.

2 - Rappel mathématique -> Graphes et théorie des graphes

- <https://www.imo.universite-paris-saclay.fr/~ruette/mathsdiscrtes/polygraph-Sigward.pdf>
- <https://www.apprendre-en-ligne.net/graphes/graphes.pdf>
- <https://perso.liris.cnrs.fr/christine.solnon/polyGraphes.pdf>
- <https://www.apprendre-en-ligne.net/graphes/>
- https://fr.wikipedia.org/wiki/Th%C3%A9orie_des_graphes
- <https://datascientest.com/top-10-des-pre-requis-mathematiques-importants-en-data-science>
- <https://moncoachdata.com/blog/mathematiques-du-machine-learning/>

NumPy <https://numpy.org/> est le package fondamental pour les calculs scientifiques avec Python.. qui ajoute

- np.array, un tableau multidimensionnel, fournissant des opérations arithmétiques rapides orientées tableaux ainsi que des capacités de diffusion souples.
- Fonctions mathématiques
- Outils de lecture/écriture de données de tableaux sur disque, et outils de travail avec des fichiers mappés en mémoire.
- Algèbre linéaire, génération de nombres aléatoires et transformation de Fourier.
- API C pour connecter NumPy avec des librairies écrites en C, C++ ou Fortran.

Installation : pip install numpy

Utilisation : import numpy as np

2 - Rappel mathématique -> la bibliothèque NumPy : array

np.array, ou tableau NumPy, est une structure de données multidimensionnelle optimisée pour les calculs numériques => similaire à une liste Python, mais il est stocké en mémoire de manière contiguë, ce qui le rend beaucoup plus rapide pour les opérations telles que l'accès aux éléments, le calcul vectoriel et les opérations matricielles.

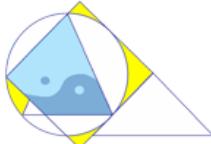
Les tableaux NumPy peuvent être créés à partir de listes, de tuples ou d'autres tableaux NumPy en utilisant la fonction `np.array()`. Ils peuvent également être créés à partir de zéro en spécifiant la forme du tableau et le type de données des éléments.

Une fois qu'un tableau NumPy a été créé, il peut être accédé à ses éléments en utilisant des indices. Les indices des tableaux NumPy sont basés sur zéro, et chaque dimension du tableau a son propre index.

Par exemple, pour accéder à l'élément à la ligne 2 et à la colonne 3 d'un tableau NumPy bidimensionnel, vous utiliseriez l'index `[2, 3]`.

Les tableaux NumPy peuvent également être utilisés pour effectuer des opérations mathématiques et matricielles. NumPy fournit un large éventail de fonctions pour ces opérations, notamment :

- `np.sum()`: Calcule la somme de tous les éléments d'un tableau.
- `np.mean()`: Calcule la moyenne de tous les éléments d'un tableau.
- `np.dot()`: Calcule le produit matriciel de deux tableaux.
- `np.linalg.inv()`: Calcule l'inverse d'une matrice.
- `np.fft.fft()`: Calcule la transformée de Fourier rapide d'un tableau.



2 - Rappel mathématique -> la bibliothèque NumPy : array

```
main.py x
1 import numpy as np
2
3 # Crée un tableau NumPy bidimensionnel contenant les nombres de 1 à 9.
4 arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
5
6 # Accéder à l'élément à la ligne 2 et à la colonne 3 du tableau.
7 element = arr[1,2]
8
9 # Calcule la somme de tous les éléments du tableau.
10 sum = np.sum(arr)
11
12 # Calcule le produit matriciel du tableau avec lui-même.
13 dot_product = np.dot(arr, arr)
14
15 # Imprime les résultats.
16 print(arr)
17 print(element)
18 print(sum)
19 print(dot_product)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
6
45
[[ 30  36  42]
 [ 66  81  96]
 [102 126 150]]
```

2 - Rappel mathématique -> la bibliothèque NumPy : array

```
import numpy as np

# Créer un tableau NumPy bidimensionnel contenant les valeurs de température pour chaque jour d'un mois.
temperatures = np.array([[10, 12, 14], [15, 17, 19], [20, 22, 24]])

# Calculer la moyenne de la température pour chaque jour.
averages = np.mean(temperatures, axis=0)

print(temperatures)
# Imprimer les résultats.
print("Résultat = ", averages)
```

```
[[10 12 14]
 [15 17 19]
 [20 22 24]]
Résultat =  [15. 17. 19.]
```

2 - Rappel mathématique -> la bibliothèque NumPy : La transformée de Fourier rapide

La transformée de Fourier est une transformation mathématique qui convertit un signal temporel en un signal fréquentiel. Cela permet d'analyser le signal et d'identifier les fréquences qu'il contient.

https://fr.wikipedia.org/wiki/Transformation_de_Fourier_rapide

La FFT est un algorithme itératif, ce qui signifie qu'il résout le problème original en une série d'étapes.

Voici quelques exemples d'utilisation de la FFT :

- **Analyse de séries chronologiques:** La FFT peut être utilisée pour identifier les tendances et les modèles dans les données de séries chronologiques. Cela peut être utilisé pour prédire les valeurs futures des données ou pour identifier les anomalies dans les données.
- **Traitements du signal:** La FFT peut être utilisée pour analyser les signaux et identifier les fréquences qu'ils contiennent. Cela peut être utilisé pour améliorer la qualité du signal, supprimer le bruit, ou identifier les composants d'un signal.
- **Image et vidéo:** La FFT peut être utilisée pour analyser les images et les vidéos et identifier les fréquences qu'elles contiennent. Cela peut être utilisé pour améliorer la qualité de l'image, supprimer le bruit, ou identifier les objets dans une image.
- **Communications sans fil:** La FFT peut être utilisée pour analyser les signaux sans fil et identifier les fréquences qu'ils contiennent. Cela peut être utilisé pour améliorer la qualité de la transmission, supprimer le bruit, ou identifier les interférences.

2 - Rappel mathématique -> la bibliothèque NumPy : La transformée de Fourier rapide

```

def Echantillon(plt, t, x_e, Duree, N, Te, te, f):
    plt.grid()
    plt.plot(t, f(t), '--', label="Signal réel")
    plt.scatter(te, x_e, color="red", label="Signal échantillonné")
    plt.xlabel(r"$t$ (s)")
    plt.ylabel(r"$x(t)$")
    plt.title(f"Échantillonnage d'un signal D={Duree:.2f}, P={Te}, N={N}")
    plt.legend()

def TF(plt, freq, X, etiquette):
    plt.title("Transformée de Fourier")
    plt.grid()
    plt.plot(freq, X, label=etiquette)
    plt.xlabel(r"Fréquence (Hz)")
    plt.ylabel(r"Amplitude $X(f)$")
    plt.legend()

def ITF(plt, te, iX):
    plt.grid()
    plt.title("Inverse transformée de Fourier")
    plt.plot(te, iX.real, label="Partie réel")
    plt.plot(te, iX.imag, label="Partie imaginaire")
    plt.xlabel(r"$t$ (s)")
    plt.ylabel(r"Amplitude $X(f)$")
    plt.legend()
  
```

```

from numpy.fft import ifft, fft, fftfreq
import numpy as np
import matplotlib.pyplot as plt

def x(t):
    # Calcul du signal
    return np.sin(2*np.pi*t)
  
```

2 - Rappel mathématique -> la bibliothèque NumPy : La transformée de Fourier rapide

```

def TFF(f, Duree, N, Te, te):
    x_e=f(te)
    plt.figure(figsize=[9, 9])
    t = np.linspace(0, Duree, 2000)
    plt.subplot(411)
    Echantillon=plt.t,x_e,Duree,N,Te,te,f)
    # Calcul FFT
    X = fft(x_e) # Transformée de fourier
    print(x_e)
    freq = fftfreq(x_e.size, d=Te) # Fréquences de la transformée de Fourier
    plt.subplot(412)
    TF=plt.freq,X.real,"partie réel")
    plt.subplot(413)
    TF=plt.freq,X.imag,"partie imaginaire")
    iX = ifft(X) # Inverse transformée de fourier
    plt.subplot(414)
    ITF=plt.te,iX)
    plt.tight_layout()
    plt.show()
  
```

```

def DataTFF(Duree, Te, f):
    # Échantillonnage du signal
    # Duree du signal en secondes
    # Te -> Période d'échantillonnage en seconde
    N = int(Duree/Te) + 1 # Nombre de points du signal échantillonné
    te = np.linspace(0, Duree, N) # Temps des échantillons
    TFF(f, Duree, N, Te, te)
  
```

DataTFF(2, 0.8, x)
 DataTFF(2, 0.2, x)
 DataTFF(2, 0.05, x)
 DataTFF(2, 0.0125, x)
 DataTFF(2, 0.00625, x)

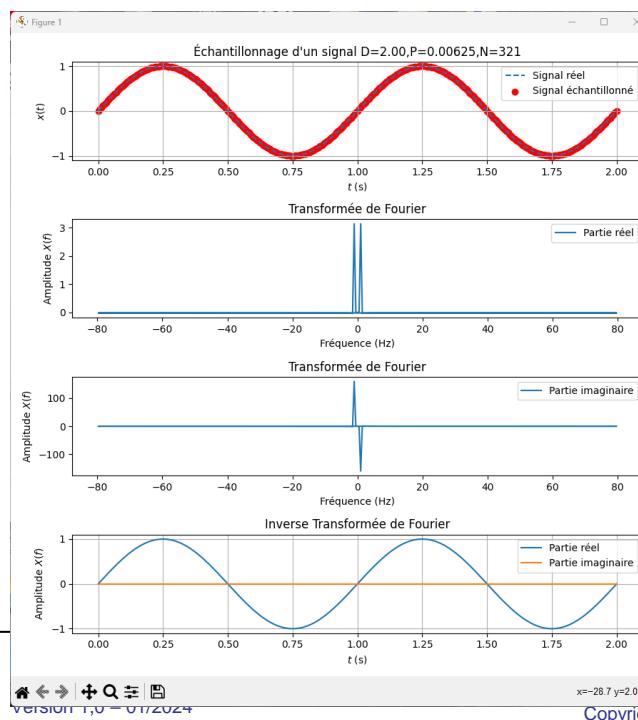
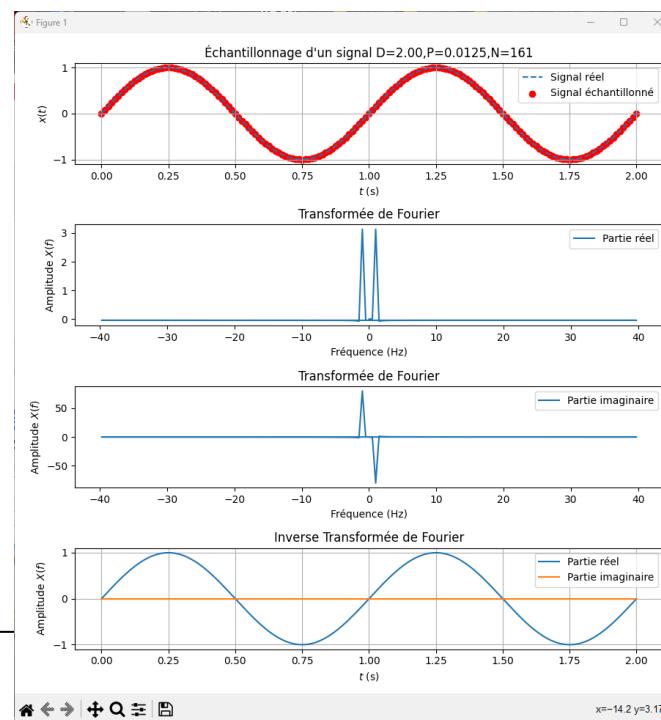
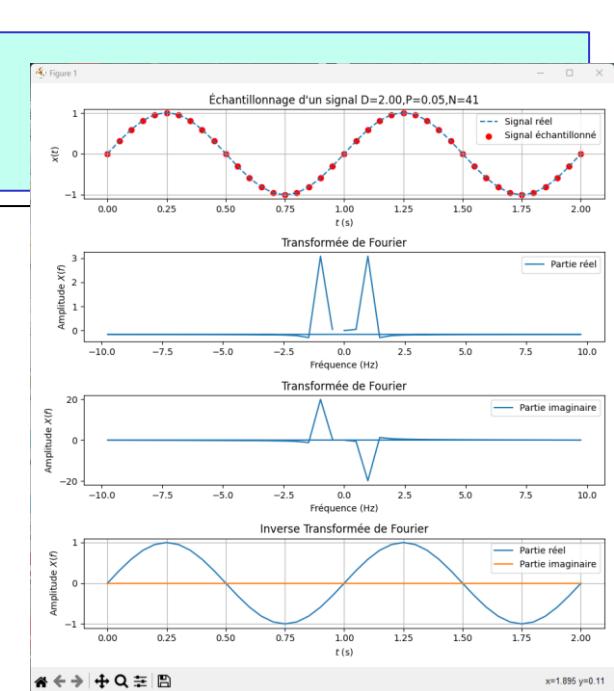
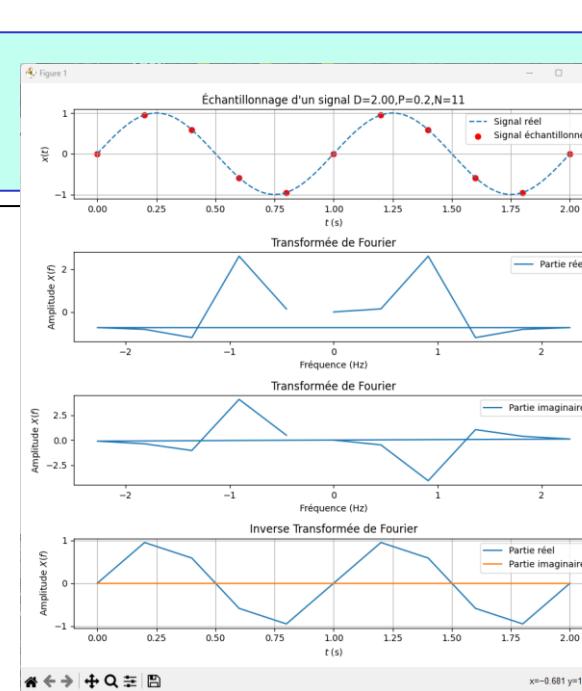
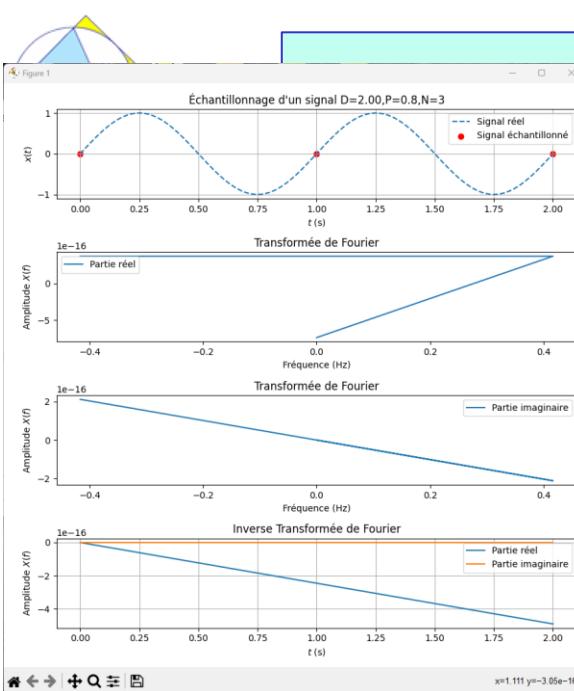
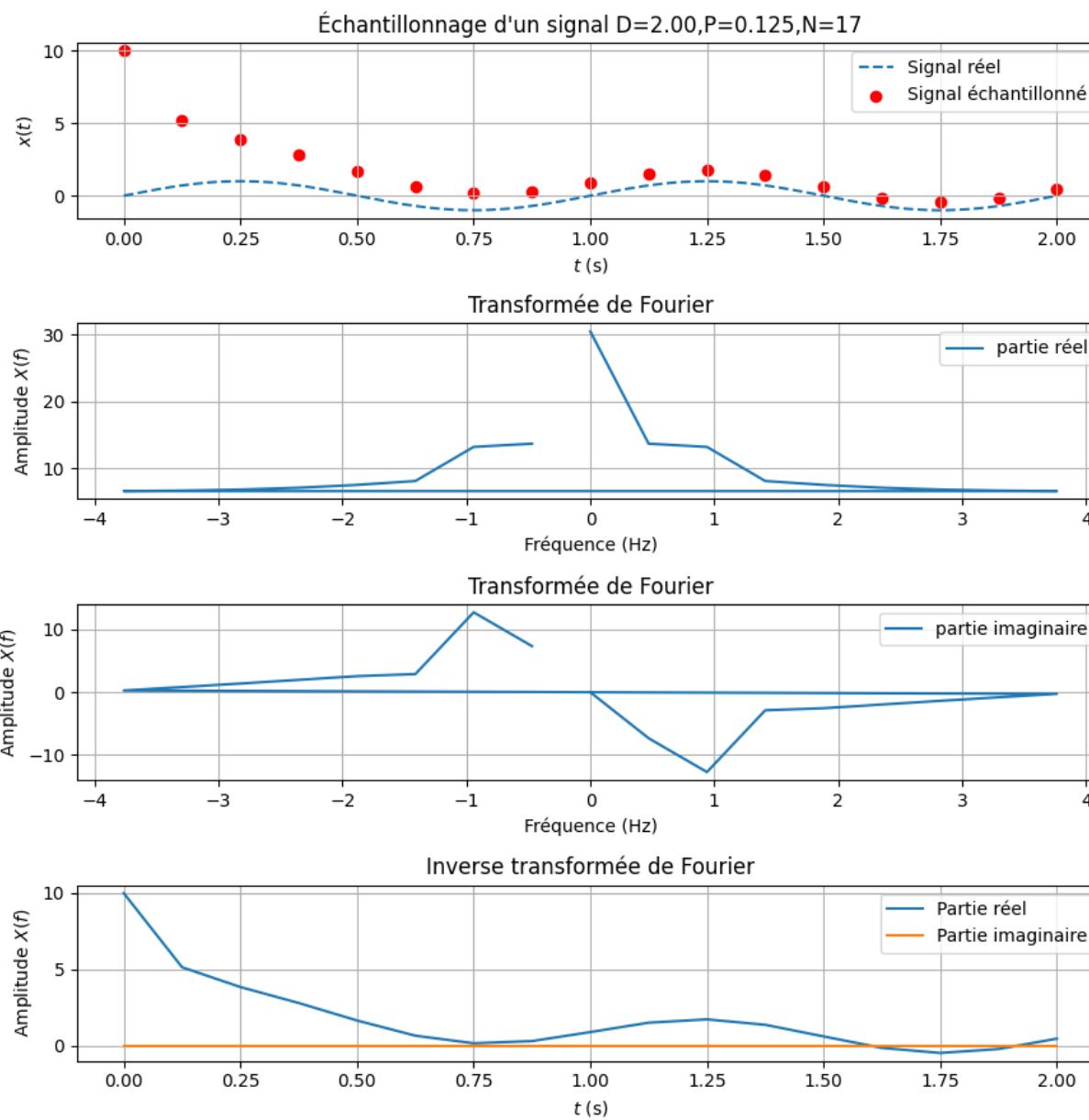




Figure 1

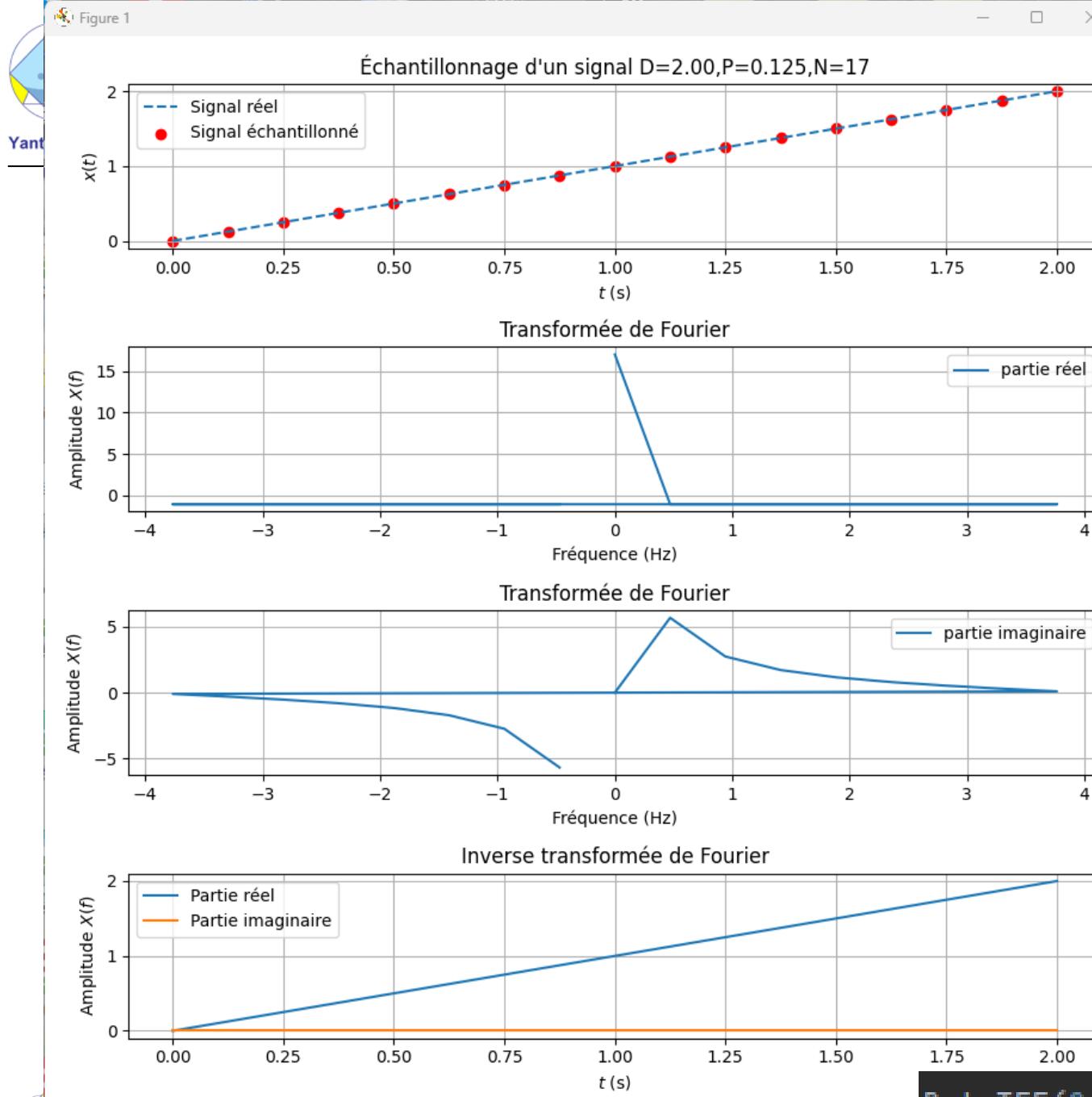
Yant



DataTFF(2, 0.125, lambda t: np.sin(2*np.pi*t) + 1/(t+0.1))

Version 1.0 – 01/2024

Copyright : Yantra Technologies 2004-2024



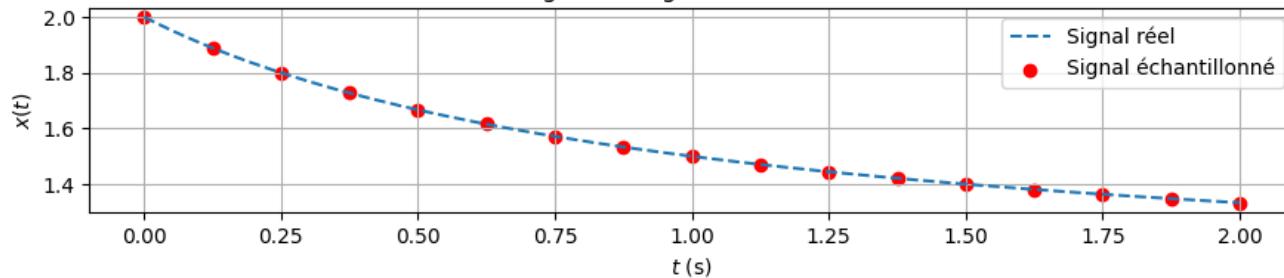
DataTFF(2, 0.125, lambda t : t)



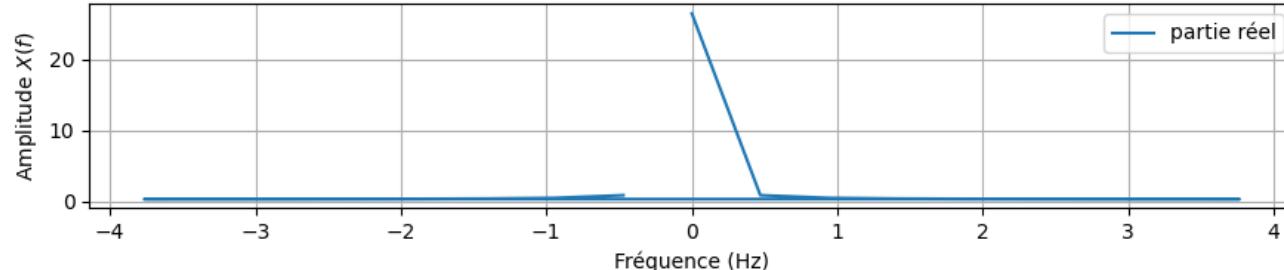
Yantra Te

Figure 1

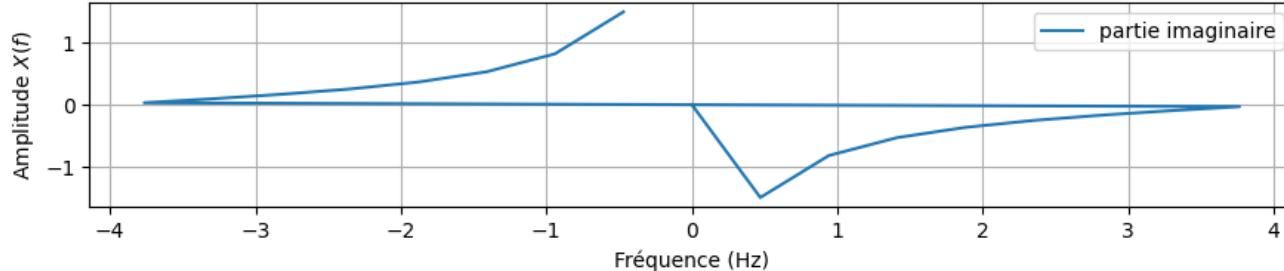
Échantillonnage d'un signal $D=2.00, P=0.125, N=17$



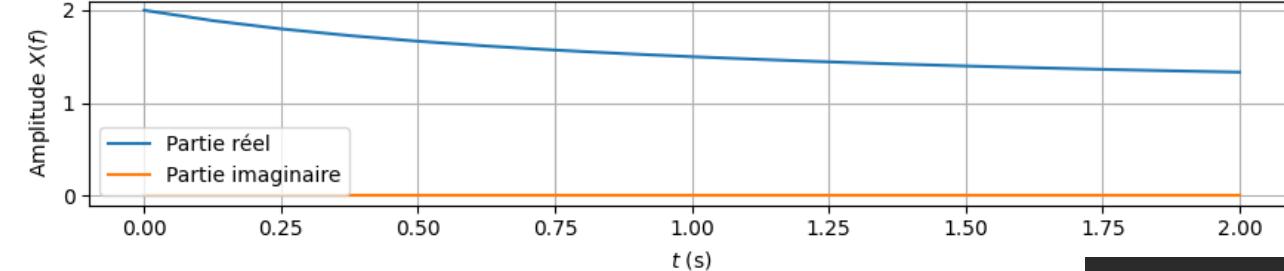
Transformée de Fourier



Transformée de Fourier



Inverse transformée de Fourier

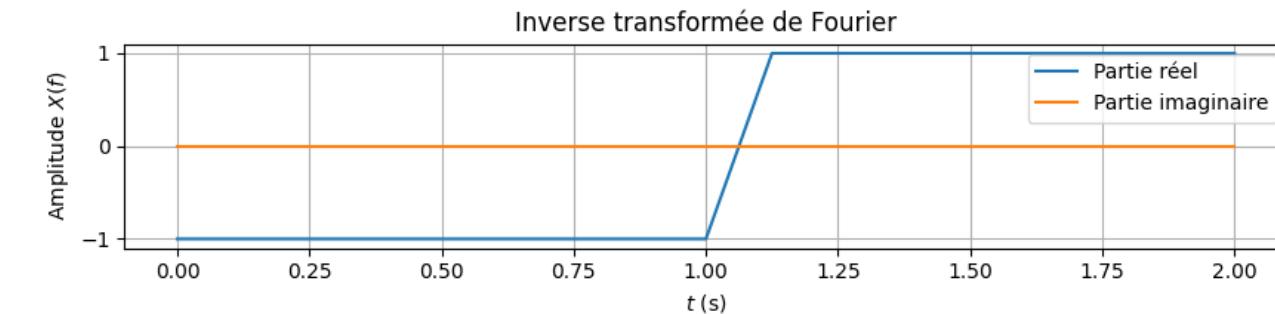
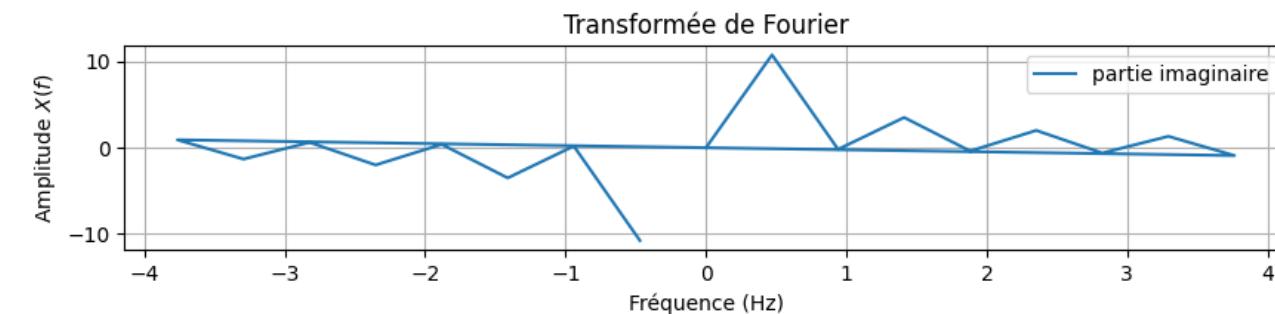
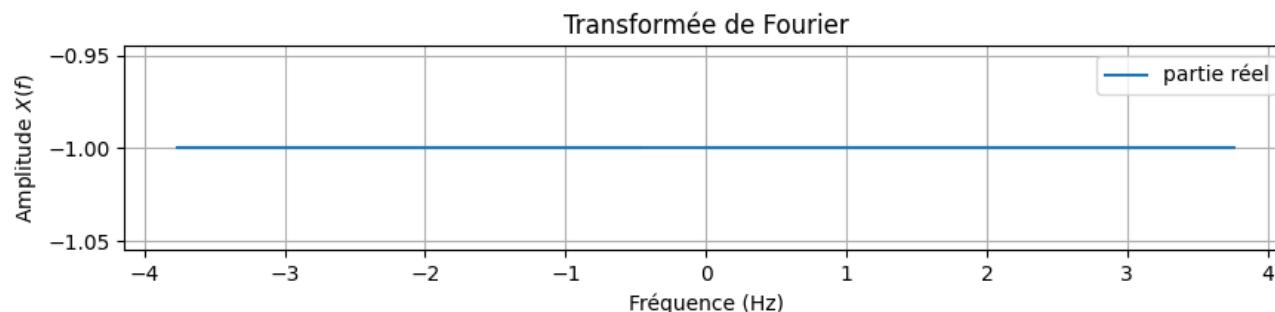
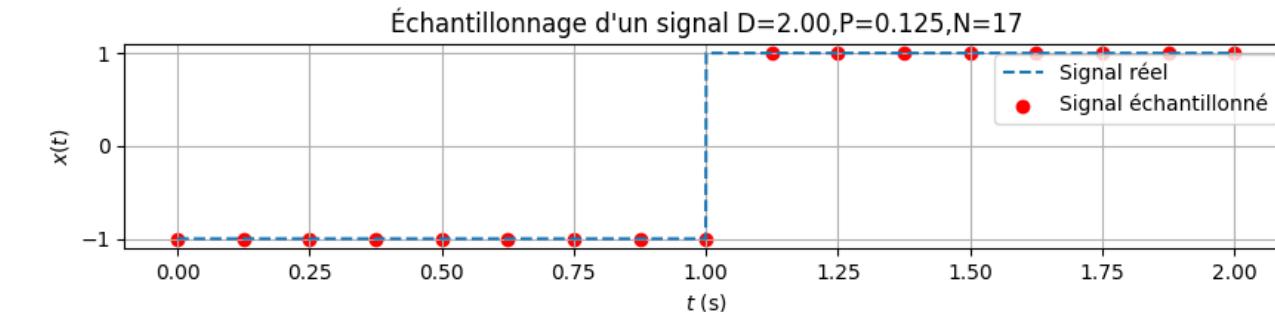


DataTFF(2, 0.125, lambda t : 1+1/(1+t))

Copyright Yantra Technologie 2007-2014



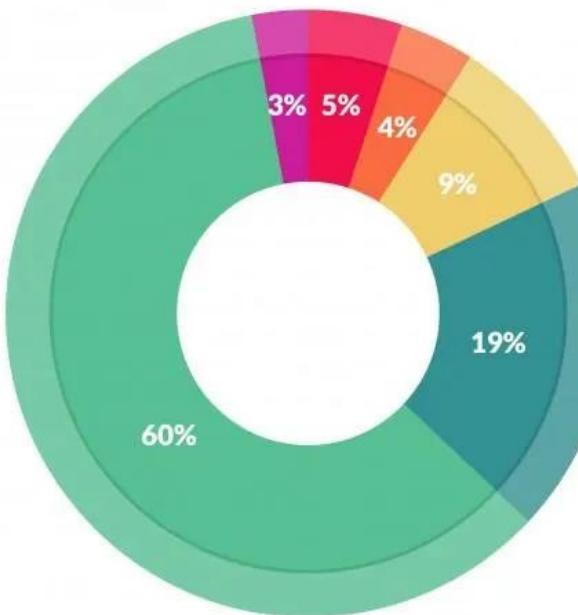
Figure 1



```
def f1(t):
    res=[]
    for i in t:
        if i%3 > 1:
            res+=[1.]
        else:
            res+=[-1]
    return np.array(res)
DataTFF(2, 0.125, f1)
```

- **Présentation**
- **Nettoyage Initial des Données**

La préparation des données consiste à rassembler, combiner, structurer et organiser les données afin de pouvoir les analyser dans le cadre de programmes d'informatique décisionnelle (BI, Business Intelligence) et d'analytique métier (BA, Business Analytics).



What data scientists spend the most time doing

- *Building training sets: 3%*
- *Cleaning and organizing data: 60%*
- *Collecting data sets; 19%*
- *Mining data for patterns: 9%*
- *Refining algorithms: 4%*
- *Other: 5%*

<https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/?sh=d5ce0f6f637d>

Le nettoyage des données, également connu sous le nom d'assainissement ou d'épuration des données, est le processus de correction, de mise à jour, de suppression ou de transformation des données pour les rendre plus précises, complètes et cohérentes.

Il vise à améliorer la qualité des données, ce qui est essentiel pour leur bonne utilisation au cours des opérations, lors des prises de décision et de la planification.

Pourquoi préparer les données :

- **Corriger les erreurs rapidement** – Une préparation des données efficace facilite la détection des erreurs avant le traitement des données.
- Lorsque les données ont quitté leur source, ces erreurs deviennent plus difficiles à détecter, comprendre et corriger.
- **Obtenir des données de grande qualité** – Le nettoyage et reformatage des « datasets » ou « jeu de données » garantissent que toutes les données utilisées dans les analyses seront de grande qualité.
- **Prendre des décisions plus avisées** – Lorsque les données sont de meilleure qualité et qu'elles peuvent être traitées et analysées plus rapidement et plus efficacement, les décisions internes sont également plus rapides, plus efficaces et de meilleure qualité.

Il existe différents types de nettoyage des données, notamment :

- **La correction des erreurs**, qui consiste à corriger les erreurs de saisie, de codage ou de transcription.
- **La suppression des données en double**, qui consiste à supprimer les données redondantes ou inutiles.
- **La complétude des données**, qui consiste à ajouter des données manquantes ou à compléter des données incomplètes.
- **La cohérence des données**, qui consiste à garantir que les données sont cohérentes entre elles.

Le nettoyage des données est un processus important qui peut avoir un impact positif sur de nombreux aspects d'une entreprise, notamment :

- **L'amélioration de la précision des analyses**
- **La réduction des coûts**
- **L'amélioration de la prise de décision**
- **La conformité aux réglementations**

- **Étape 1 – Identifier les données essentielles**
- **Étape 2 – Collecter les données**
- **Étape 3 – Éliminer les doublons**
- **Étape 4 – Résoudre les valeurs manquantes (complétude des données)**
- **Étape 5 – Effectuer la détection des valeurs aberrantes / anomalies (cohérence des données, correction des erreurs)**

Les données essentielles sont les données qui sont nécessaires pour répondre aux objectifs de l'analyse.

Il existe plusieurs façons d'identifier les données essentielles.

- Une approche consiste à se concentrer sur les données qui sont pertinentes pour les questions que l'on cherche à répondre.
- Une autre approche consiste à se concentrer sur les données qui sont utilisées dans les modèles ou les algorithmes d'analyse de données.

Conseil :

- **Comprendre les objectifs de l'analyse.** Quelles sont les questions que l'on cherche à répondre ? Quelles sont les informations dont on a besoin pour répondre à ces questions ?
- **Examiner les données.** Quelles sont les données disponibles ? Quelles sont les données pertinentes pour les objectifs de l'analyse ?
- **Consulter les experts.** Les experts peuvent fournir des informations précieuses sur les données essentielles.

La collecte de données consiste à obtenir les données nécessaires pour répondre aux objectifs de l'analyse.

Les données peuvent être collectées à partir de sources :

- internes, telles que des bases de données, des systèmes de suivi ou des logs.
- externes, telles que des enquêtes, des sondages ou des réseaux sociaux.

Conseil :

- **Déterminez les objectifs de l'analyse.** Quelles sont les informations dont vous avez besoin ?
- **Identifiez les sources de données.** Où pouvez-vous trouver les données dont vous avez besoin ?
- **Choisissez la méthode de collecte de données appropriée.** Quelle méthode est la plus efficace pour collecter les données dont vous avez besoin ?
- **Collectez les données de manière systématique.** Assurez-vous que les données sont collectées de manière cohérente et précise.
- **Vérifiez la qualité des données.** Assurez-vous que les données sont complètes, exactes et cohérentes.

L'élimination des doublons est une étape importante du nettoyage initial des données.

Les doublons sont des enregistrements qui sont identiques ou très similaires à d'autres enregistrements. Ils peuvent être causés par une saisie erronée, une erreur de système ou une autre cause.

Il existe plusieurs façons d'éliminer les doublons.

- Utiliser une fonction de suppression des doublons dans un logiciel de feuille de calcul ou de base de données.
- Utiliser une fonction de recherche de doublons pour identifier les enregistrements qui sont identiques ou très similaires.

Dans Pandas, il existe deux méthodes principales pour supprimer les doublons :

- **drop_duplicates()** supprime les enregistrements qui sont identiques sur toutes les colonnes.
- **duplicated()** renvoie un tableau indiquant les enregistrements qui sont des doublons.

La méthode `drop_duplicates()` est la méthode la plus simple pour supprimer les doublons. Elle prend en entrée un tableau et un ensemble de colonnes sur lesquelles baser la comparaison.

```
import pandas as pd

# Créez un tableau avec des doublons
df = pd.DataFrame({
    "nom": ["Jean", "Marc", "Marie", "Jean", "Marc"],
    "âge": [20, 25, 30, 20, 25]
})

# Supprimez les doublons sur la colonne "nom"
df = df.drop_duplicates(subset="nom")

# Affiche le tableau
print(df)
```

	nom	âge
0	Jean	20
1	Marc	25
2	Marie	30

3 - Préparation et Chargement des Données

-> Nettoyage Initial des Données : Etape 3 - Éliminer les doublons

La méthode `duplicated()` renvoie un tableau indiquant les enregistrements qui sont des doublons. Ce tableau peut ensuite être utilisé pour supprimer les doublons à l'aide de la méthode `drop()`.

```
import pandas as pd

# Créez un tableau avec des doublons
df = pd.DataFrame({
    "nom": ["Jean", "Marc", "Marie", "Jean", "Marc"],
    "âge": [20, 25, 30, 20, 25]
})

# Retourne un tableau indiquant les enregistrements qui sont des doublons
df_duplicated = df.duplicated(subset="nom")

# Affiche le tableau
print(df_duplicated)

# Supprime les doublons
df = df.drop(df_duplicated[df_duplicated == True].index)

# Affiche le tableau
print(df)
```

	nom	âge
0	Jean	20
1	Marc	25
2	Marie	30

3 - Préparation et Chargement des Données

-> Nettoyage Initial des Données : Etape 3 - Éliminer les doublons

```
#https://www.thepinguin.com/2023/03/28/python-suppression-de-doublons/

import pandas as pd

# Création d'un DataFrame avec des doublons
data = {'Nom': ['Jean', 'Pierre', 'Marie', 'Luc', 'Jean'],
        'Age': [25, 35, 28, 32, 25],
        'Ville': ['Paris', 'Lyon', 'Marseille', 'Lille', 'Paris']}
df = pd.DataFrame(data)

# Affichage du DataFrame original
print("DataFrame original :")
print(df)

# Identification des doublons
duplicates = df[df.duplicated()]
print("Doublons identifiés :")
print(duplicates)

# Suppression des doublons en utilisant drop_duplicates()
df_drop = df.drop_duplicates()
print("DataFrame sans doublons :")
print(df_drop)

# Suppression des doublons en utilisant set() et frozenset()
df_set = pd.DataFrame([frozenset(row) for row in df.values], columns=df.columns)
df_unique = pd.DataFrame(list(set(tuple(row) for row in df_set.values))), columns=df.columns
print("DataFrame sans doublons :")
print(df_unique)
```

<https://www.thepinguin.com/2023/03/28/python-suppression-de-doublons/>

Les valeurs manquantes sont un problème courant dans les ensembles de données. Elles peuvent être causées par une saisie erronée, une erreur de système ou une autre cause.

Les valeurs manquantes peuvent biaiser les résultats de l'analyse de données. Par exemple, si une valeur manquante est comptée comme une valeur nulle, cela peut entraîner une diminution artificielle de la moyenne ou de la variance d'une variable.

Il existe plusieurs façons de résoudre les valeurs manquantes. La méthode la plus appropriée dépend du type de données, de la proportion de valeurs manquantes et des objectifs de l'analyse.

Voici quelques techniques pour résoudre les valeurs manquantes :

- Supprimer les enregistrements contenant des valeurs manquantes. Cette méthode est simple à mettre en œuvre, mais elle peut entraîner une perte de données.
- Combler les valeurs manquantes avec des valeurs estimées. Cette méthode est plus complexe à mettre en œuvre, mais elle permet de conserver les données.
- Laisser les valeurs manquantes inchangées. Cette méthode est la plus simple à mettre en œuvre, mais elle peut entraîner un biais dans les résultats de l'analyse.

https://gorenja.com/article/display/python_valeurs_manquantes

Supprimer les enregistrements contenant des valeurs manquantes

la méthode `dropna()` pour supprimer les enregistrements contenant des valeurs manquantes

```
import pandas as pd
import numpy as np

# Créez un tableau avec des valeurs manquantes
df = pd.DataFrame({
    "nom": ["Jean", "Marc", pd.NA],
    "âge": [20, 25, np.nan]
})

# Supprimez les enregistrements contenant des valeurs manquantes
df = df.dropna()

# Affiche le tableau
print(df)
```

	nom	âge
0	Jean	20.0
1	Marc	25.0

Combler les valeurs manquantes avec des valeurs estimées

La méthode `fillna()` pour combler les valeurs manquantes avec la moyenne des valeurs non manquantes

```
import numpy as np
import pandas as pd

# Créez un tableau avec des valeurs manquantes
df = pd.DataFrame({
    "nom": ["Jean", "Marc", np.nan],
    "âge": [20, 25, np.nan]
})

# Comblez les valeurs manquantes avec la moyenne des valeurs non manquantes
df["âge"].fillna(df["âge"].mean(), inplace=True)
print(df)

df["nom"].fillna("Inconnue", inplace=True)

# Affiche le tableau
print(df)
```

	nom	âge
0	Jean	20.0
1	Marc	25.0
2	NaN	22.5

	nom	âge
0	Jean	20.0
1	Marc	25.0
2	Inconnue	22.5

La détection des valeurs aberrantes ou anomalies est une étape importante du nettoyage des données.

Les valeurs aberrantes sont des données qui s'écartent de manière significative de la distribution générale des données. Elles peuvent être causées par des erreurs de saisie, des erreurs de mesure ou des événements rares.

Il existe plusieurs méthodes pour détecter les valeurs aberrantes. Les méthodes les plus courantes sont les suivantes :

- **Méthodes statistiques** : Ces méthodes utilisent des statistiques descriptives telles que la moyenne, l'écart-type et les quartiles pour identifier les données qui s'écartent de la distribution générale.
- **Méthodes visuelles** : Ces méthodes utilisent des graphiques et des visualisations pour identifier les données qui s'écartent du reste des données.
- **Méthodes d'apprentissage automatique** : Ces méthodes utilisent des algorithmes d'apprentissage automatique pour identifier les données qui sont susceptibles d'être des valeurs aberrantes.

Une fois que les valeurs aberrantes ont été détectées, elles doivent être traitées. Il existe plusieurs méthodes pour traiter les valeurs aberrantes :

- **Elimination** : Les valeurs aberrantes peuvent être simplement éliminées des données.
- **Modification** : Les valeurs aberrantes peuvent être modifiées pour qu'elles correspondent à la distribution générale des données.
- **Laisser comme tel** : Il est parfois préférable de laisser les valeurs aberrantes telles qu'elles sont, si elles sont susceptibles d'être des données réelles.

Effectuer la détection des valeurs aberrantes / anomalies

- **Z-score** : [https://datascience.eu/fr/mathematiques-
et-statistiques/quest-ce-quun-z-score/](https://datascience.eu/fr/mathematiques-et-statistiques/quest-ce-quun-z-score/)
- **DBSCAN**
[https://www.kdnuggets.com/2022/08/implementing-
dbscan-python.html](https://www.kdnuggets.com/2022/08/implementing-dbscan-python.html)
- **Forêt d'isolation** :
[https://france.devoteam.com/paroles-
dexperts/algorithme-n3-comprendre-lisolation-
forest-en-5-min/](https://france.devoteam.com/paroles-dexperts/algorithme-n3-comprendre-lisolation-forest-en-5-min/)

3 - Préparation et Chargement des Données -> Nettoyage Initial des Données : Etape 5 - Z-score

Le z-score, également appelé score standard ou valeur z, est une mesure statistique qui indique le nombre de standard déviations par lequel une valeur est au-dessus ou en dessous de la moyenne.

Le z-score est calculé de la manière suivante :

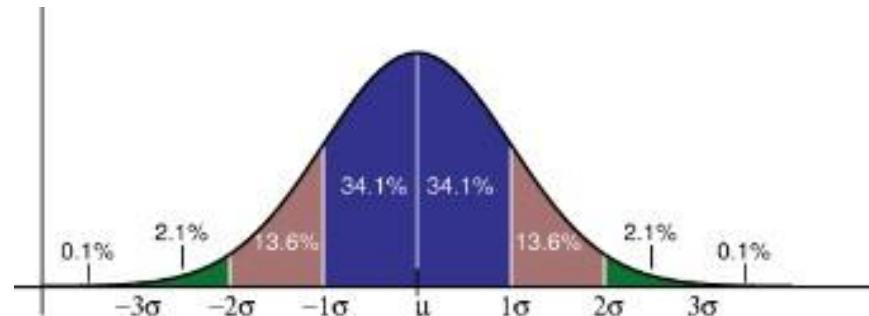
$$z = (\text{valeur} - \text{moyenne}) / \text{écart-type}$$

- Un z-score de 0 indique que la valeur est égale à la moyenne.
- Un z-score positif indique que la valeur est au-dessus de la moyenne,
- Un z-score négatif indique que la valeur est en dessous de la moyenne.

Plus le z-score est élevé, plus la valeur s'écarte de la moyenne.

Cela suppose une distribution gaussienne (une courbe normale, en forme de cloche).

En général, on considère qu'une valeur est aberrante si son z-score est supérieur ou égal à 3 ou inférieur ou égal à -3. Cela signifie que la valeur est située à au moins 3 écarts-types de la moyenne.



https://docs.oracle.com/cloud/help/fr/pbcs_common/PFUSU/insights_metrics_Z-Score.htm#PFUSU-GUID-640CEBD1-33A2-4B3C-BD81-EB283F82D879

 <https://datascience.eu/fr/mathematiques-et-statistiques/quest-ce-quun-z-score/>

3 - Préparation et Chargement des Données -> Nettoyage Initial des Données : Etape 5 - Z-score

```

import pandas as pd
import numpy as np
import scipy.stats as stats

data = np.array([6, 7, 7, 12, 13, 13, 15, 16, 19, 22])
zscores = stats.zscore(data)
print(data)
print(zscores)
print("\n")

data = np.array([[5, 6, 7, 7, 8], [8, 8, 8, 9, 9], [2, 2, 4, 4, 5]])
zscores = stats.zscore(data, axis=1)
print(data)
print(zscores)
print("\n")

data = pd.DataFrame(np.random.randint(0, 10, size=(5, 3)), columns=['A', 'B', 'C'])
zscores = data.apply(stats.zscore)
print(data)
print(zscores)
print("\n")
  
```

	A	B	C
0	1	7	8
1	4	5	5
2	8	6	6
3	6	3	5
4	4	2	9

	A	B	C
0	-1.543487	1.293993	0.861640
1	-0.257248	0.215666	-0.984732
2	1.457738	0.754829	-0.369274
3	0.600245	-0.862662	-0.984732
4	-0.257248	-1.401826	1.477098

3 - Préparation et Chargement des Données -> Nettoyage Initial des Données : Etape 5 - Z-score

```
import pandas as pd
import numpy as np
import scipy.stats as stats

data = np.array([6, 7, 7, 12, 13, 13, 15, 16, 19, 22])
zscores = stats.zscore(data)
print(data)
print(zscores)
print("\n")
```

```
[ 6  7  7 12 13 13 15 16 19 22]
[-1.39443338 -1.19522861 -1.19522861 -0.19920477  0.
 0.39840954  0.5976143   1.19522861  1.79284291]
```

```
import pandas as pd
import numpy as np
import scipy.stats as stats

data = np.array([[5, 6, 7, 7, 8], [8, 8, 8, 9, 9], [2, 2, 4, 4, 5]])
zscores = stats.zscore(data, axis=1)
print(data)
print(zscores)
print("\n")
```

```
[[5 6 7 7 8]
 [8 8 8 9 9]
 [2 2 4 4 5]]
[[-1.56892908 -0.58834841  0.39223227  0.39223227  1.37281295]
 [-0.81649658 -0.81649658 -0.81649658  1.22474487  1.22474487]
 [-1.16666667 -1.16666667  0.5       0.5       1.33333333]]
```

3 - Préparation et Chargement des Données -> Nettoyage Initial des Données : Etape 5 - Z-score

```
import pandas as pd
import numpy as np
import scipy.stats as stats

data = pd.DataFrame(np.random.randint(0, 10, size=(5, 3)), columns=['A', 'B', 'C'])
zscores = data.apply(stats.zscore)
print(data)
print(zscores)
print("\n")
```

	A	B	C
0	5	6	9
1	3	2	9
2	3	4	0
3	6	0	3
4	7	1	4

	A	B	C
0	0.125	1.578410	1.135924
1	-1.125	-0.278543	1.135924
2	-1.125	0.649934	-1.419905
3	0.750	-1.207020	-0.567962
4	1.375	-0.742781	-0.283981

```
import pandas as pd
import numpy as np
import scipy.stats as stats

data = pd.DataFrame({
    "a": [1, 2, 3, 4, 5],
    "b": [6, 7, 8, 9, 10]
})

z_scores = data.apply(stats.zscore)

print(data)
print(z_scores)
```

	a	b
0	1	6
1	2	7
2	3	8
3	4	9
4	5	10

	a	b
0	-1.414214	-1.414214
1	-0.707107	-0.707107
2	0.000000	0.000000
3	0.707107	0.707107
4	1.414214	1.414214

Le **DBSCAN** est un **algorithme non supervisé** très connu en matière de **Clustering**. Il a été proposé 1996 par Martin Ester, Hans-Peter Kriegel, Jörg Sander et Xiawei Xu.

DBSCAN étant un **algorithme non supervisé**, ce qui signifie qu'il n'a pas besoin de connaître le nombre de clusters à l'avance.

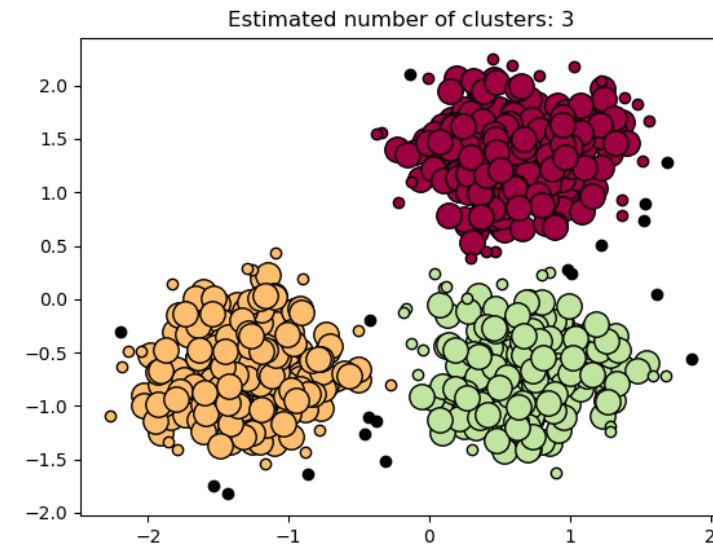
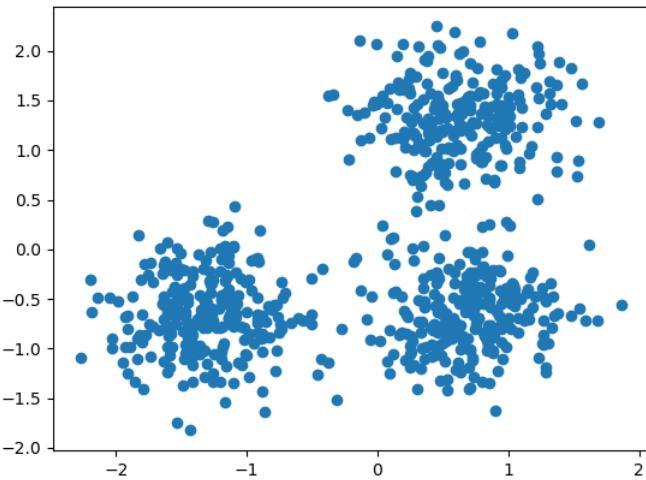
3 - Préparation et Chargement des Données -> Nettoyage Initial des Données : Etape 5 - DBSCAN

DBSCAN utilise deux paramètres principaux : MinPts et Epsilon.

- MinPts est le nombre minimum de points de données qui doivent se trouver à Epsilon d'un point de données pour que ce point de données soit considéré comme un point central.
- Epsilon est la distance maximale entre deux points de données pour qu'ils soient considérés comme des voisins.

DBSCAN fonctionne comme suit :

1. Il commence par un point de données arbitraire et trouve tous ses voisins à Epsilon.
2. Si le nombre de voisins est supérieur ou égal à MinPts, alors le point de données est un point central et un nouveau cluster est créé.
3. Tous les voisins du point central sont ajoutés au cluster.
4. L'algorithme est répété pour chaque point de données dans l'ensemble de données.



3 - Préparation et Chargement des Données -> Nettoyage Initial des Données : Etape 5 - DBSCAN

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN

# Créer un ensemble de données
X = np.array([[1, 1], [2, 2], [3, 2], [12, 12], [5, 5], [6, 6], [5, 6], [-1, 7], [-1, 8], [-1, 9], [12, 13], [-12, -12]])

# Créer un modèle DBSCAN
dbSCAN = DBSCAN(eps=2, min_samples=3)

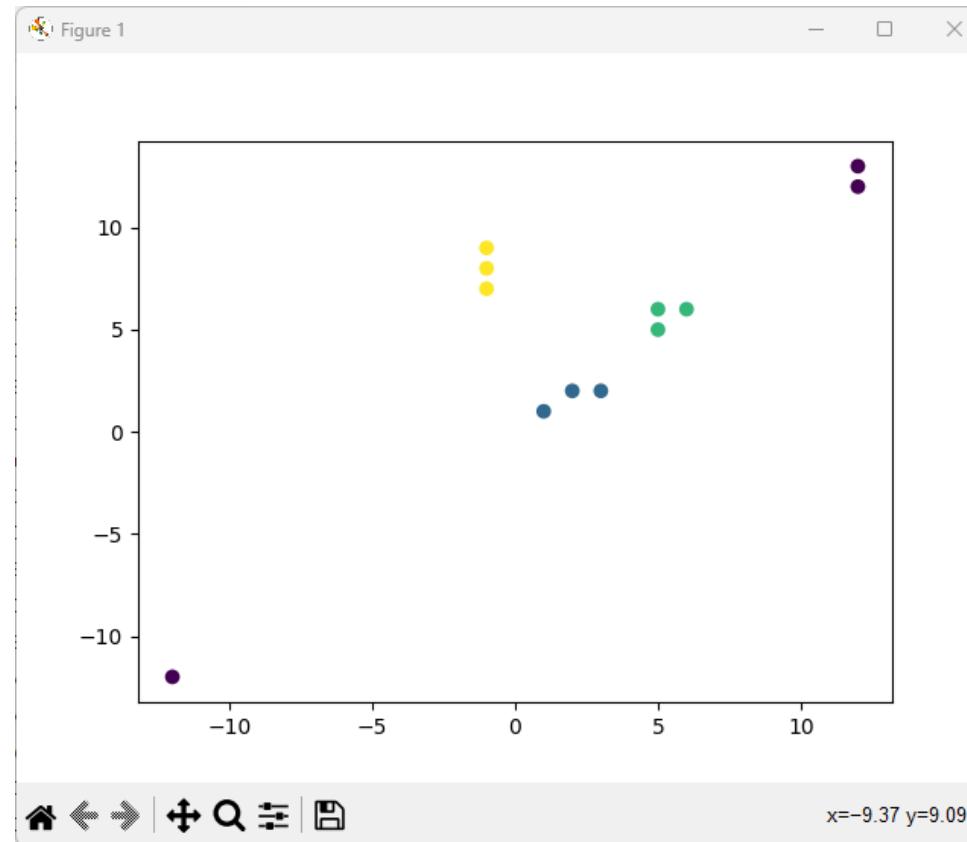
# Appliquer le modèle à l'ensemble de données
labels = dbSCAN.fit_predict(X)

# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)

print("Estimation des clusters: %d" % n_clusters_)
print("Estimation du nombre de point correspondant à du bruit: %d" % n_noise_)

# Afficher les clusters
plt.scatter(X[:, 0], X[:, 1], c=labels)
plt.show()
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='rainbow')
plt.show()
```

3 - Préparation et Chargement des Données -> Nettoyage Initial des Données : Etape 5 - DBSCAN



```
Estimation des clusters: 3
Estimation du nombre de point correspondant à du bruit: 3
```

<https://datascientest.com/machine-learning-clustering-dbscan>

<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html>

<https://www.kdnuggets.com/2022/08/implementing-dbscan-python.htm>

3 - Préparation et Chargement des Données

-> Nettoyage Initial des Données : Etape 5 - Forêt d'isolement

- <https://france.devoteam.com/paroles-dexperts/algorithme-n3-comprendre-isolation-forest-en-5-min/>

L'isolation forest (moins connu sous son nom français « forêt d'isolement ») calcule un score d'anomalie pour chaque observation du dataset.

Ce score donne une mesure de la normalité de chaque observation en fonction de l'ensemble des données.

Pour calculer ce score, l'algorithme isole la donnée en question de manière récursive : il choisit une variable au hasard et fixe un seuil de coupure au hasard, puis il évalue si cela permet d'isoler une observation en particulier.

Ce type d'algorithme est à privilégier dans le cas de problèmes mathématiques de grandes dimensions (grand nombre d'observations et de variables).

La forêt d'isolement est un algorithme de détection d'anomalies **qui fonctionne en mesurant la distance de chaque point de données au reste de l'ensemble de données.**

Les points de données qui sont plus isolés que les autres sont considérés comme des anomalies.

La forêt d'isolement fonctionne comme suit :

- L'ensemble de données est divisé en plusieurs arbres de décision.
- Chaque arbre de décision est formé sur un sous-ensemble aléatoire de l'ensemble de données.
- Pour chaque point de données, la forêt d'isolement calcule la distance entre le point de données et chaque arbre de décision.
- La distance maximale de chaque point de données est calculée.
- Les points de données dont la distance maximale est supérieure à un seuil donné sont considérés comme des anomalies.

La forêt d'isolement présente les avantages suivants :

- Elle est robuste au bruit.
- Elle peut trouver des anomalies dans des ensembles de données de grande taille.
- Elle est facile à utiliser.

Un inconvénient la forêt d'isolement peut être sensible aux paramètres de l'algorithme.

La forêt d'isolement peut être utilisée pour une variété de tâches, notamment :

- La détection de fraude
- La détection d'erreurs
- La surveillance des systèmes

Voici quelques exemples de points de données que la forêt d'isolement peut identifier comme des anomalies :

- Une transaction bancaire qui est beaucoup plus importante que les autres transactions du client.
- Une erreur dans un programme informatique qui n'apparaît que dans une petite partie du code.
- Une panne d'équipement qui ne se produit que dans un petit nombre d'unités.

```

import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import IsolationForest

# Créer un ensemble de données de températures
data = np.array([
    [20, 30], [25, 35], [30, 40], [35, 45], [40, 50],
    [45, 55], [50, 60], [55, 65], [60, 70],
    [65, 80], [75, 30], [80, 85], [85, 40],
    [90, 45], [95, 35], [100, 55], [105, 60]
])

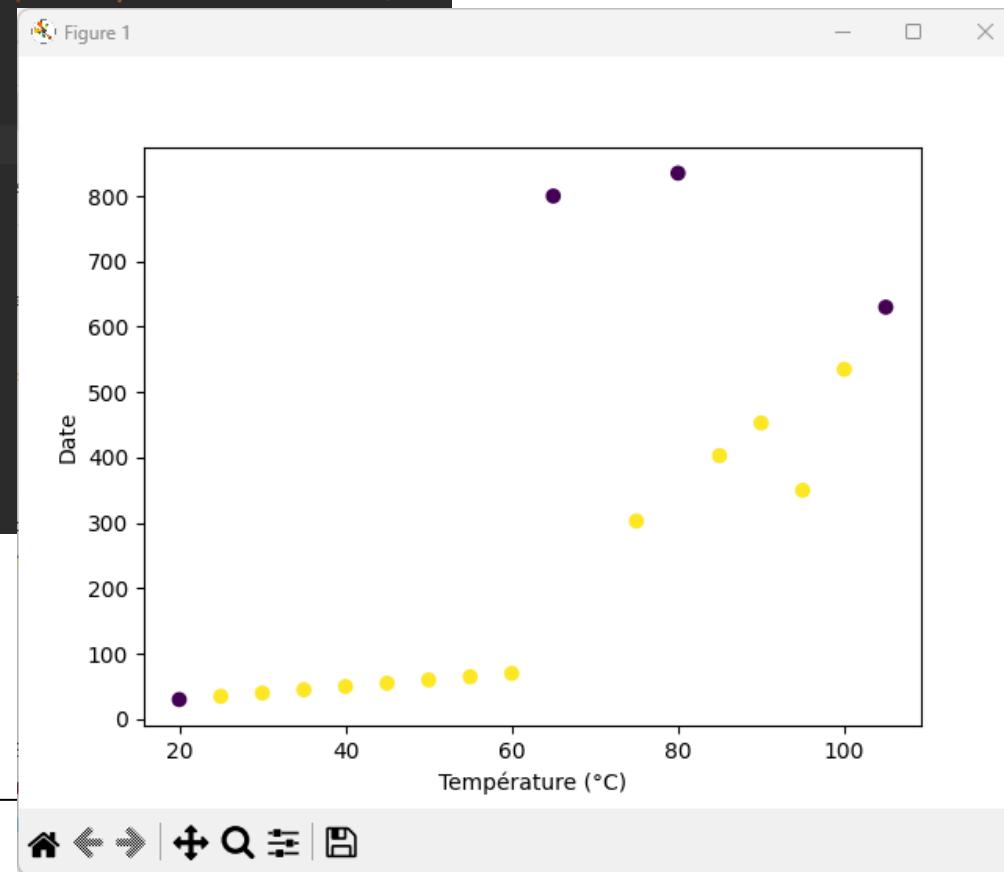
# Créer un modèle de forêt d'isolement
model = IsolationForest(n_estimators=100, max_samples=17, contamination=0.2)

# Entrainer le modèle sur l'ensemble de données
model.fit(data)

# Prédire les anomalies
labels = model.predict(data)
print(labels)

# Afficher les résultats
plt.scatter(data[:, 0], data[:, 1], c=labels)
plt.xlabel("Température (°C)")
plt.ylabel("Date")
plt.show()

```



```

# Créer un ensemble de données de transactions bancaires
data = np.array([
    [100, 200], [200, 300], [300, 400], [400, 500],
    [500, 600], [700, 800], [600, 700], [700, 500],
    [800, 900], [900, 1000], [1000, 2000], [1250, 500]
])

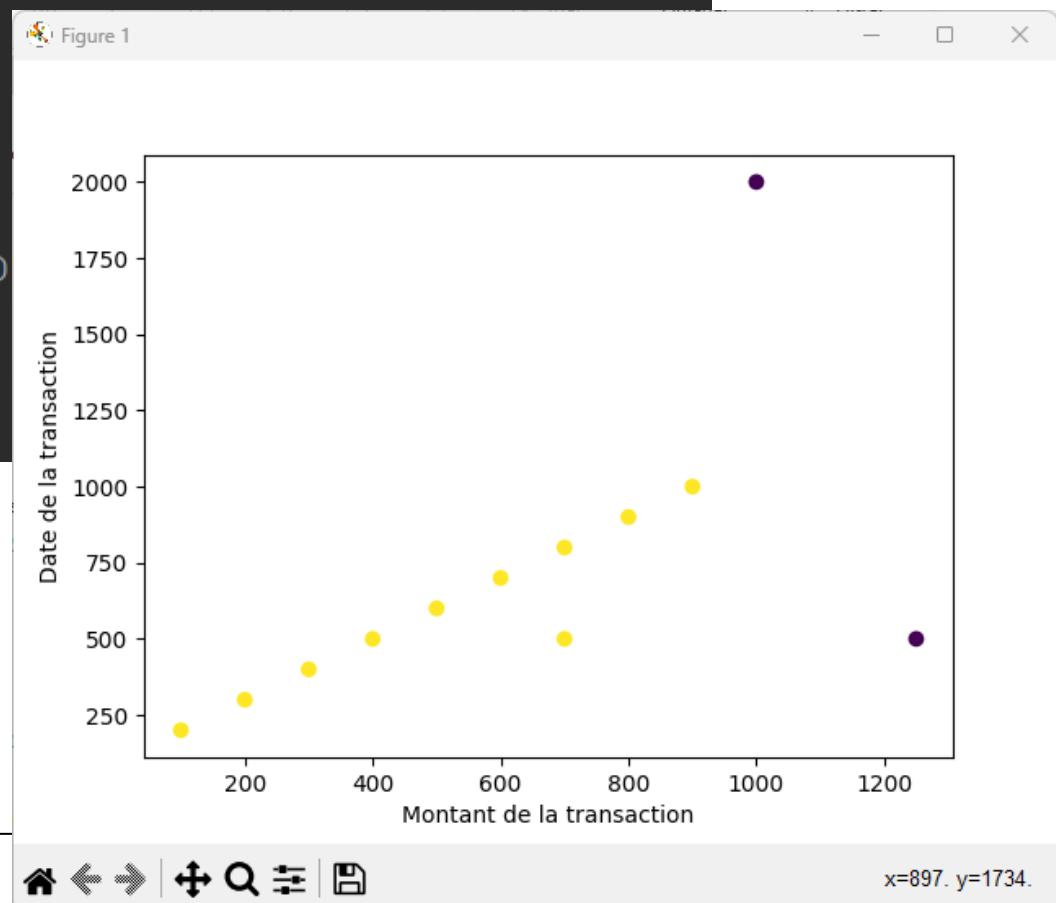
# Créer un modèle de forêt d'isolement
model = IsolationForest(n_estimators=50, max_samples=data.size//2, contamination=0.16)

# Entrainer le modèle sur l'ensemble de données
model.fit(data)

# Prédire les anomalies
labels = model.predict(data)

# Afficher les résultats
plt.scatter(data[:, 0], data[:, 1], c=labels)
plt.xlabel("Montant de la transaction")
plt.ylabel("Date de la transaction")
plt.show()

```



- **Exploration des Données avec Pandas**
- **Filtrage et Sélection des Données**
- **Manipulation des Données avec Pandas**

L'exploration des données est le processus d'analyse des données pour en tirer des informations. Il s'agit d'une étape importante de toute analyse de données, car elle permet de comprendre les données et de déterminer les questions qui peuvent être posées.

Pandas est une bibliothèque puissante pour l'analyse et le traitement des données. Elle offre une variété de fonctionnalités qui peuvent être utilisées pour l'exploration des données, notamment :

- La visualisation des données : Pandas fournit une variété de fonctions pour visualiser les données, notamment des graphiques et des tableaux.
- Les statistiques descriptives : Pandas fournit une variété de fonctions pour calculer des statistiques descriptives sur les données, notamment la moyenne, la médiane et la variance.
- Les tests statistiques : Pandas fournit une variété de fonctions pour effectuer des tests statistiques sur les données, notamment des tests de t-student et des tests de khi-deux.

L'exploration des données est le processus d'analyse des données pour en tirer des informations. Il s'agit d'une étape importante de toute analyse de données, car elle permet de comprendre les données et de déterminer les questions qui peuvent être posées.

Pandas est une bibliothèque puissante pour l'analyse et le traitement des données. Elle offre une variété de fonctionnalités qui peuvent être utilisées pour l'exploration des données, notamment :

- La visualisation des données : Pandas fournit une variété de fonctions pour visualiser les données, notamment des graphiques et des tableaux.
- Les statistiques descriptives : Pandas fournit une variété de fonctions pour calculer des statistiques descriptives sur les données, notamment la moyenne, la médiane et la variance.
- Les tests statistiques : Pandas fournit une variété de fonctions pour effectuer des tests statistiques sur les données, notamment des tests de t-student et des tests de khi-deux.

Rappels sur Pandas

- DataFrame : Structure de données tabulaire
- Index : Permet d'accéder aux données par ligne ou par colonne
- Méthodes pour la manipulation des données :
 - head() : Affiche les premières lignes d'un DataFrame
 - tail() : Affiche les dernières lignes d'un DataFrame
 - describe() : Affiche des statistiques descriptives d'un DataFrame
 - filter() : Filtre les lignes d'un DataFrame selon une condition
 - drop() : Supprime les lignes ou les colonnes d'un DataFrame
 - rename() : Renomme les colonnes d'un DataFrame
 - fillna() : Remplace les valeurs manquantes d'un DataFrame
 - merge() : Fusionne deux DataFrames

Comprendre les données :

- Quelles sont les colonnes ?
- Quels sont les types de données ?
- Y a-t-il des valeurs manquantes ?

Identifier les tendances :

- Statistiques descriptives
- Visualisation des données

Les étapes de l'exploration des données avec Pandas sont les suivantes :

- 1. Importation des données** : La première étape consiste à importer les données dans un format compatible avec Pandas.
- 2. Visualisation des données** : La deuxième étape consiste à visualiser les données pour obtenir une compréhension de base de leurs tendances et de leurs distributions.
- 3. Calcul des statistiques descriptives** : La troisième étape consiste à calculer des statistiques descriptives sur les données pour obtenir une compréhension plus approfondie de leurs valeurs.
- 4. Exécution de tests statistiques** : La quatrième étape consiste à exécuter des tests statistiques sur les données pour tester des hypothèses ou identifier des relations entre les variables.

```
import pandas as pd

#Importer les données
df = pd.read_csv("ExempleData.csv")
```

Age	Taille	Poids	Genre
20	175	70	Homme
25	180	65	Femme
30	185	80	Homme
35	170	60	Femme
40	190	75	Homme
21	176	70	Homme
26	181	65	Femme
31	186	80	Homme
36	171	60	Femme
41	191	75	Homme

ExempleData.csv

4 - Manipulation et Exploration des Données avec Pandas

-> Exploration des Données avec Pandas :

Les Etapes : 2 - Visualisation des données

```

import pandas as pd

# Importer les données
df = pd.read_csv("ExempleData.csv")

print(df)

print("\n-> info")
df.info()
print("\n-> head")
# Display the first 5 rows of the DataFrame
print(df.head())
print("\n-> tail")
# Display the last 5 rows of the DataFrame
print(df.tail())
  
```

	Age	Taille	Poids	Genre
0	20	175	70	Homme
1	25	180	65	Femme
2	30	185	80	Homme
3	35	170	60	Femme
4	40	190	75	Homme
5	21	176	70	Homme
6	26	181	65	Femme
7	31	186	80	Homme
8	36	171	60	Femme
9	41	191	75	Homme

```

-> info
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype  
---  -- 
 0   Age      10 non-null    int64  
 1   Taille   10 non-null    int64  
 2   Poids    10 non-null    int64  
 3   Genre    10 non-null    object  
dtypes: int64(3), object(1)
memory usage: 452.0+ bytes
  
```

```

-> head
   Age   Taille   Poids   Genre
0   20    175     70   Homme
1   25    180     65   Femme
2   30    185     80   Homme
3   35    170     60   Femme
4   40    190     75   Homme
  
```

```

-> tail
   Age   Taille   Poids   Genre
5   21    176     70   Homme
6   26    181     65   Femme
7   31    186     80   Homme
8   36    171     60   Femme
9   41    191     75   Homme
  
```

4 - Manipulation et Exploration des Données avec Pandas

→ Exploration des Données avec Pandas :

Les Etapes : 2 - Visualisation des données

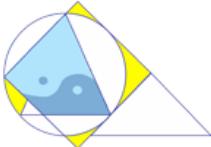
```
import matplotlib.pyplot as plt

df.boxplot(column="Taille")
plt.title("Box Plot de la Taille")
plt.ylabel("Taille (cm)")
plt.show()

df.plot.scatter(x="Age", y="Taille")
plt.title("Scatter Plot de Age / Taille")
plt.xlabel("Age (an)")
plt.ylabel("Taille (cm)")
plt.show()

df.groupby("Genre").Taille.mean().plot.bar()
plt.title("Bar Chart de Taille by Genre")
plt.xlabel("Genre")
plt.ylabel("Taille")
plt.show()

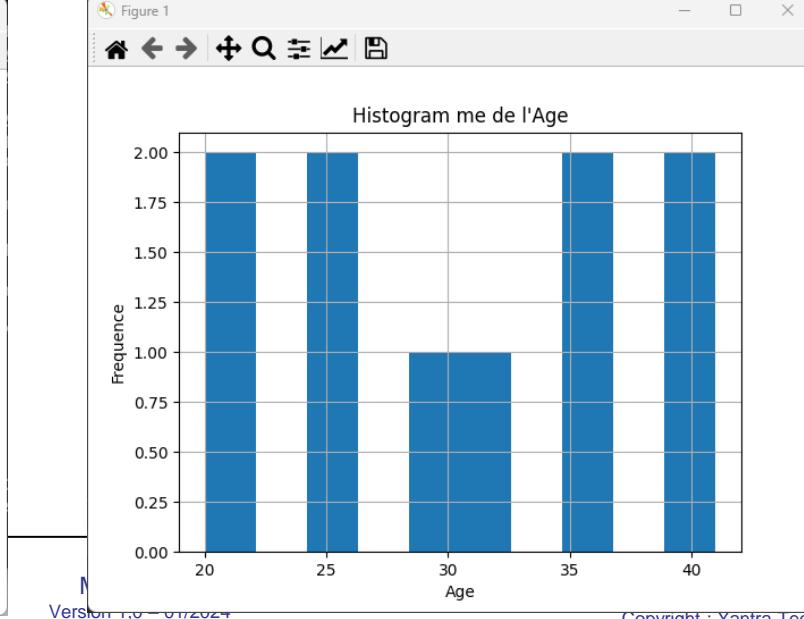
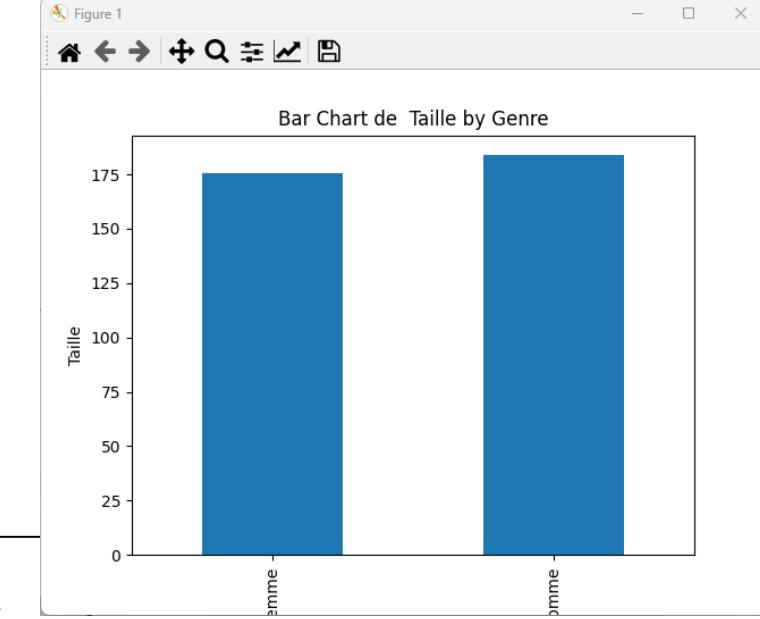
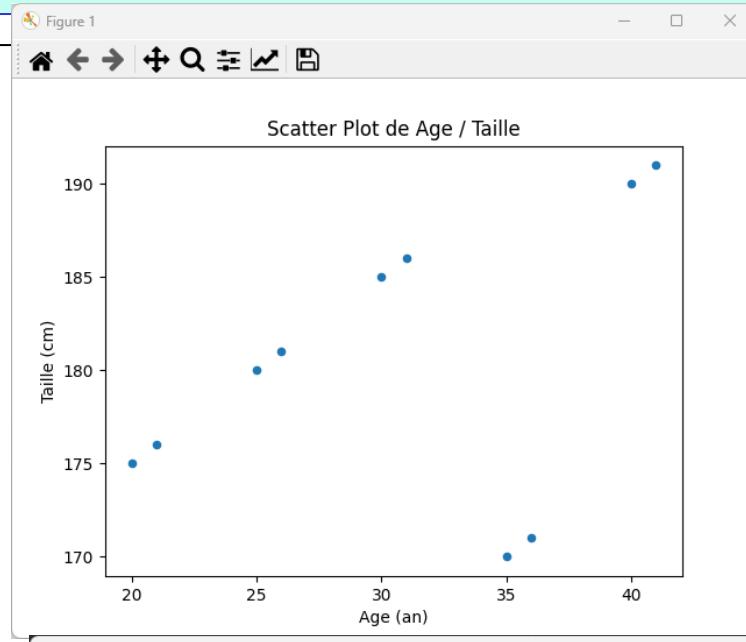
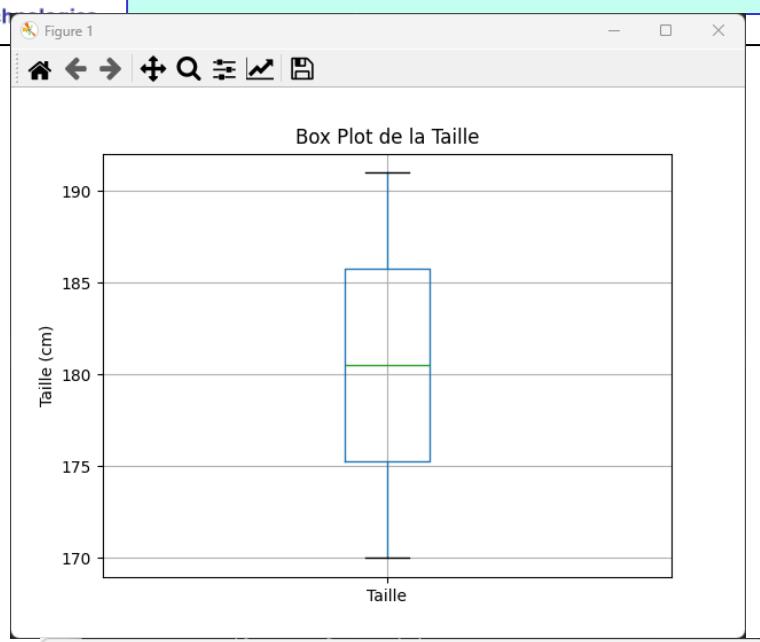
df.Age.hist()
plt.title("Histogramme de l'Age")
plt.xlabel("Age")
plt.ylabel("Frequence")
plt.show()
```



4 - Manipulation et Exploration des Données avec Pandas

→ Exploration des Données avec Pandas :

Les Etapes : 2 - Visualisation des données



- <https://sites.google.com/view/aide-python/graphiques/explorer-et-visualiser-les-donn%C3%A9es-dune-dataframe>

```

df["Genre"] = np.where(df["Genre"] == "Homme", 1, 0)
print(df.describe())
taille_mean = df["Taille"].mean()
print("Calcule de la moyenne",taille_mean)
taille_std = df["Taille"].std()
print("Calcule de l'écart type de la taille",taille_std)
  
```

- **count** : Comptez le nombre d'observations non NA/nulles.
- **mean**: Moyenne des valeurs.
- **std** : Écart type des observations.
- **min** : Minimum des valeurs dans l'objet.
- **25 %** : la valeur au-dessus de laquelle se trouve 25 % des données.
- **50%** : la valeur au milieu des données.
- **75%** : la valeur au-dessus de laquelle se trouve 75 % des données
- **max** : Maximum des valeurs dans l'objet.

```

-> describe
      Age      Taille      Poids      Genre
count  10.000000  10.000000  10.00000  10.000000
mean   30.500000 180.500000  70.00000  0.600000
std    7.472171  7.472171  7.45356  0.516398
min   20.000000 170.000000  60.00000  0.000000
25%  25.250000 175.250000  65.00000  0.000000
50%  30.500000 180.500000  70.00000  1.000000
75%  35.750000 185.750000  75.00000  1.000000
max   41.000000 191.000000  80.00000  1.000000
Calcule de la moyenne 180.5
Calcule de l'écart type de la taille 7.472170590486631
  
```

- L'âge moyen est de 30,5 ans.
- La taille moyenne est de 180,5 centimètres.
- Le poids moyen est de 70 kilogrammes.
- Le sexe le plus courant est l'homme (60 %).
- L'âge moyen est de 30,5 ans. La moitié des personnes dans l'échantillon ont moins de 30,5 ans et l'autre moitié ont plus de 30,5 ans.
- La taille moyenne est de 180,5 cm. La moitié des personnes dans l'échantillon mesurent moins de 180,5 cm et l'autre moitié mesurent plus de 180,5 cm.
- Le poids moyen est de 70 kg. La moitié des personnes dans l'échantillon pèsent moins de 70 kg et l'autre moitié pèsent plus de 70 kg.
- La distribution de la taille et du poids est approximativement normale car la plupart des valeurs se situent autour de la moyenne et que les valeurs extrêmes sont rares..

```
-> describe
      Age      Taille      Poids      Genre
count 10.000000 10.000000 10.000000 10.000000
mean 30.500000 180.500000 70.000000 0.600000
std 7.472171 7.472171 7.45356 0.516398
min 20.000000 170.000000 60.000000 0.000000
25% 25.250000 175.250000 65.000000 0.000000
50% 30.500000 180.500000 70.000000 1.000000
75% 35.750000 185.750000 75.000000 1.000000
max 41.000000 191.000000 80.000000 1.000000
Calcule de la moyenne 180.5
Calcule de l'écart type de la taille 7.472170590486631
```

Correlation Poids/Taille 0.7980074688861063

```
correlation = df['Taille'].corr(df['Poids'])

print("Correlation Poids/Taille ",correlation)
```

4 - Manipulation et Exploration des Données avec Pandas

-> Exploration des Données

Les Etapes

il existe :

- une corrélation positive entre l'âge et la taille, ce qui signifie que les gens ont tendance à grandir en taille en vieillissant.
- une corrélation positive entre l'âge et le poids, ce qui signifie que les gens ont tendance à prendre du poids en vieillissant
- une corrélation positive entre le sexe et la taille, ce qui signifie que les hommes ont tendance à être plus grands que les femmes.
- une corrélation positive entre le sexe et le poids, ce qui signifie que les hommes ont tendance à être plus lourds que les femmes.

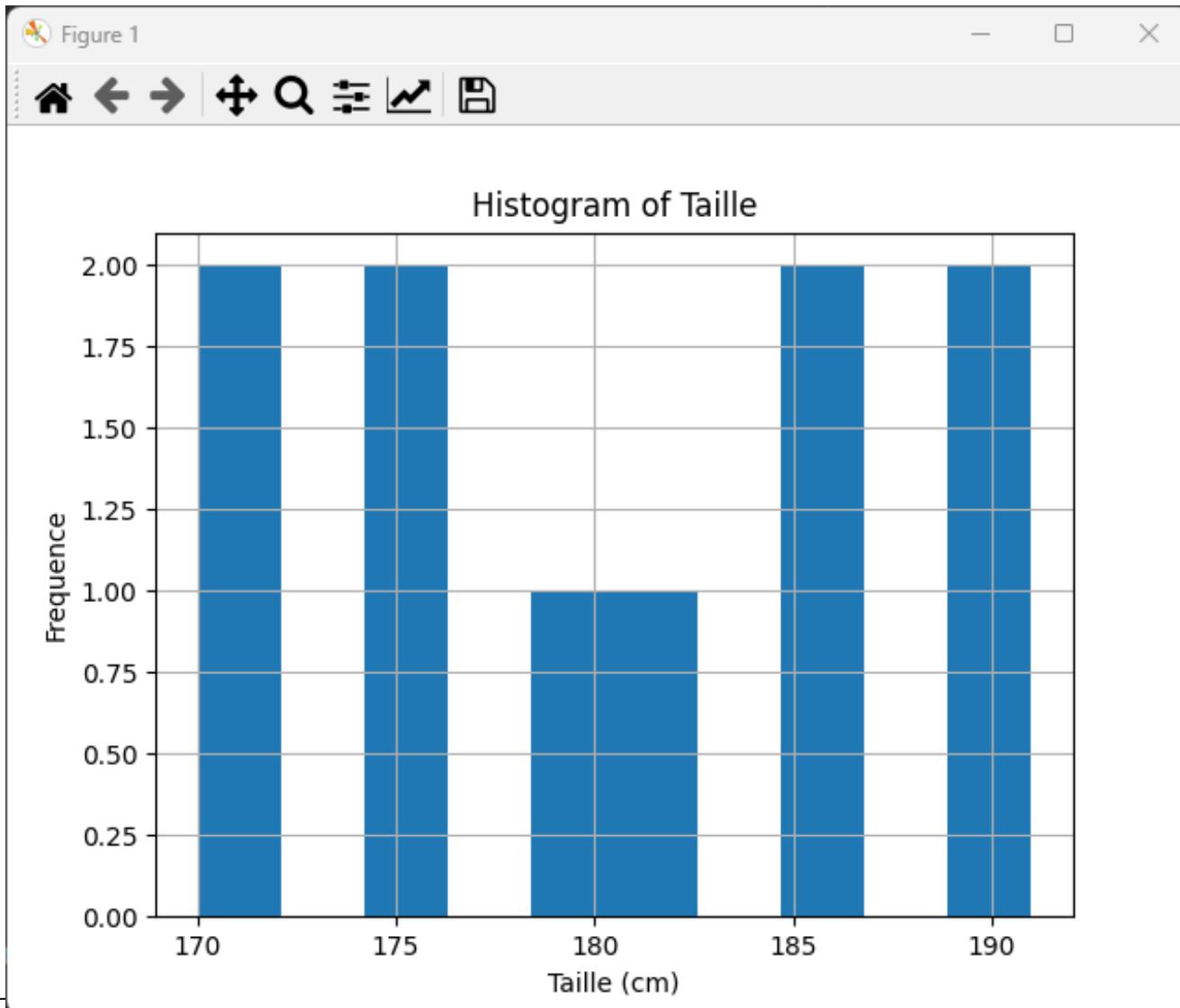
Corrélation entre l'âge et la taille : 0.40298507462686567
 Corrélation entre l'âge et le poids : 0.09975093361076329
 Corrélation entre le sexe et la taille : 0.5759122837209069
 Corrélation entre le sexe et le poids : 0.8660254037844387
 Correlation Poids/Taille 0.7980074688861063
 Genre
 Femme 175.500000
 Homme 183.833333
 Name: Taille, dtype: float64
 Genre
 Femme 62.5
 Homme 75.0
 Name: Poids, dtype: float64

```
df_gender_height = df.groupby("Genre")["Taille"].mean()
df_gender_weight = df.groupby("Genre")["Poids"].mean()

correlation_age_height = df["Age"].corr(df["Taille"])
correlation_age_weight = df["Age"].corr(df["Poids"])
df["Genre"] = np.where(df["Genre"] == "Homme", 1, 0)
correlation_gender_height = df["Genre"].corr(df["Taille"])
correlation_gender_weight = df["Genre"].corr(df["Poids"])

print("Corrélation entre l'âge et la taille :", correlation_age_height)
print("Corrélation entre l'âge et le poids :", correlation_age_weight)
print("Corrélation entre le sexe et la taille :", correlation_gender_height)
print("Corrélation entre le sexe et le poids :", correlation_gender_weight)

correlation = df['Taille'].corr(df['Poids'])
print("Correlation Poids/Taille ", correlation)
print(df_gender_height)
print(df_gender_weight)
```



Pour effectuer un test t indépendant (ttest_ind) avec Pandas, vous pouvez utiliser la fonction `scipy.stats.ttest_ind()`.

Cette fonction prend deux arguments : les deux échantillons à comparer.

4 - Manipulation et Exploration des Données avec Pandas

-> Exploration des Données avec Pandas : Les Etapes : 4 - Exécution de tests statistiques

```
men_height_mean = df[df["Genre"] == "Homme"]["Taille"].mean()
women_height_mean = df[df["Genre"] == "Femme"]["Taille"].mean()
df["Genre"] = np.where(df["Genre"] == "Homme", 1, 0)
ttest_results = stats.ttest_ind(df["Taille"], df["Genre"], alternative="two-sided", equal_var=False)
```

```
TtestResult(statistic=75.95383953264275, pvalue=4.734279191933172e-14, df=9.085968188206175)
```

- La valeur pvalue est extrêmement faible (4,741570082474393e-14), ce qui est bien inférieur au niveau de signification typique de 0,05.
 - Cela suggère que la différence de moyenne observée est très probablement due à autre chose que le hasard.
- La valeur élevée de la statistique de test (75,93095778612616) renforce encore la conclusion que les deux moyennes de population sont différentes.
 - (La statistique de test mesure l'ampleur de la différence de moyenne, et une valeur plus élevée indique une différence plus importante)
- En résumé :
 - le résultat du test T fourni indique qu'il existe une différence statistiquement significative entre les deux moyennes de population. La nature exacte de cette différence peut être explorée plus en détail en examinant les moyennes des échantillons et la distribution des données.
- Conclusion : les données fournissent suffisamment de preuves pour conclure que la moyenne de la hauteur des hommes est significativement plus élevée que la moyenne de la hauteur des femmes.

- **Présentation**
- **Visualisation avec Matplotlib**
- **Visualisation avec Pandas**

La visualisation des données est le processus de représentation des données sous forme graphique ou visuelle. Elle est un outil important pour l'analyse des données car elle permet de comprendre les données de manière plus intuitive.

La visualisation des données peut être utilisée pour :

- **Identifier les tendances et les modèles dans les données**
- **Comprendre les relations entre les variables**
- **Communiquer les résultats de l'analyse des données**

Il existe de nombreux types de visualisations de données, chacun ayant ses propres avantages et inconvénients. Les types de visualisations de données les plus courants incluent :

- **Graphiques de barres** : Les graphiques de barres sont utilisés pour représenter des données catégorielles.
- **Histogrammes** : Les histogrammes sont utilisés pour représenter la distribution d'une variable quantitative.
- **Diagrammes à secteurs** : Les diagrammes à secteurs sont utilisés pour représenter des données sous forme de parts d'un tout.
- **Diagrammes en ligne** : Les diagrammes en ligne sont utilisés pour représenter l'évolution des données au fil du temps.
- **Cartes** : Les cartes sont utilisées pour représenter des données géographiques.

Outils de visualisation des données

Il existe de nombreux outils disponibles pour la visualisation des données

Exemple :

<https://moncoachdata.com/blog/visualisation-de-donnees-avec-python/>

- **Matplotlib** : Une bibliothèque Python pour la visualisation des données.
<https://www.xarala.co/blog/comment-visualiser-des-donnees-avec-python-et-matplotlib/>
- **Seaborn** : Une bibliothèque Python pour la visualisation des données basée sur Matplotlib. <https://datascientest.com/seaborn-tout-savoir>
- **Plotly** : Un outil de visualisation des données en ligne et hors ligne.
- **Power BI** : Un outil de visualisation des données de Microsoft..

- <https://sites.google.com/view/aide-python/graphiques/les-graphiques-courbes-et-nuages-de-points-scatter-plot>
- <https://www.xarala.co/blog/comment-visualiser-des-donnees-avec-python-et-matplotlib/>
- https://gvallverdu.gitbooks.io/python_sciences/content/pandas_capp.html
- <https://python.doctor/page-creer-graphiques-scientifiques-python-apprendre>
- <https://moncoachdata.com/blog/seaborn-pour-la-data-science/>
- <https://infoforall.fr/python/python-act160.html>
- <https://plotly.com/python/getting-started/>
- <https://learn.microsoft.com/fr-fr/power-bi/visuals/power-bi-line-chart?tabs=powerbi-desktop>

- Agrégation et Groupement
- Analyse de Séries Temporelles

L'agrégation et le regroupement sont deux concepts importants en analyse de données.

L'agrégation est le processus de combinaison de données pour créer un résumé ou une mesure globale. Le regroupement est le processus de division des données en sous-ensembles basés sur une ou plusieurs variables.

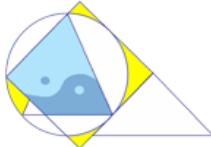
En Python, l'agrégation et le regroupement peuvent être effectués à l'aide de la bibliothèque Pandas.

Pandas fournit une variété de fonctions pour effectuer des opérations d'agrégation et de regroupement.

Agrégation

Les fonctions d'agrégation de Pandas sont utilisées pour calculer des résumés ou des mesures globales des données. Les fonctions d'agrégation les plus courantes incluent :

- `sum()` : Calcule la somme des valeurs d'une colonne.
- `mean()` : Calcule la moyenne des valeurs d'une colonne.
- `median()` : Calcule la médiane des valeurs d'une colonne.
- `mode()` : Calcule la valeur la plus courante d'une colonne.
- `std()` : Calcule l'écart type des valeurs d'une colonne.



6 - Analyse Avancée et Cas Pratiques-> Agrégation et Groupement

```
Yantra Technologies
import pandas as pd

df = pd.DataFrame({
    "Date": pd.to_datetime(["2023-01-01", "2023-01-02", "2023-01-03", "2023-01-04", "2023-01-05", "2023-01-06", "2023-01-07", "2023-01-08", "2023-01-09", "2023-01-10", "2023-02-01", "2023-02-02", "2023-02-03", "2023-02-04", "2023-02-05"]),
    "NbVente": [100, 105, 110, 115, 120, 80, 85, 80, 95, 70]
})
# Somme
somme = df["NbVente"].sum()
# Moyenne
moyenne = df["NbVente"].mean()
# Médiane
médiane = df["NbVente"].median()
# Mode
mode = df["NbVente"].mode()
# Écart type
écart_type = df["NbVente"].std()

# Afficher les statistiques descriptives
print(f"Nombre de valeurs : {len(df)}")
print(f"Somme : {somme}")
print(f"Moyenne : {moyenne}")
print(f"Écart type : {écart_type}")
print(f"Valeur minimale : {df['NbVente'].min()}")
print(f"Quartile 25 : {df['NbVente'].quantile(0.25)}")
print(f"Médiane : {médiane}")
print(f"Quartile 75 : {df['NbVente'].quantile(0.75)}")
print(f"Valeur maximale : {df['NbVente'].max()}")
```

Nombre de valeurs : 10
Somme : 960
Moyenne : 96.0
Écart type : 16.79947089113887
Valeur minimale : 70
Quartile 25 : 81.25
Médiane : 97.5
Quartile 75 : 108.75
Valeur maximale : 120
Nombre de valeurs : 8

6 - Analyse Avancée et Cas Pratiques-> Agrégation et Groupement

```

df = pd.DataFrame({
    "Date": pd.to_datetime(["2023-01-01", "2023-01-02", "2023-01-03", "2023-01-04", "2023-01-05",
                           "2023-02-01", "2023-02-02", "2023-02-03", "2023-02-04", "2023-02-05"]),
    "NbVente": [100, 105, 110, 115, 120, 80, 85, 80, 95, 70]
})
  
```

```

df_stats = df[["NbVente"]].describe()
print(df_stats)
  
```

```

# Afficher les statistiques descriptives
print(f"Nombre de valeurs : {df_stats.count().iloc[0]}")
print(f"Moyenne : {df_stats.mean().iloc[0]}")
print(f"Écart type : {df_stats.std().iloc[0]}")
print(f"Valeur minimale : {df_stats.min().iloc[0]}")
print(f"Quartile 25 : {df_stats.quantile(0.25).iloc[0]}")
print(f"Médiane : {df_stats.median().iloc[0]}")
print(f"Quartile 75 : {df_stats.quantile(0.75).iloc[0]}")
print(f"Valeur maximale : {df_stats.max().iloc[0]}")
  
```

	NbVente
count	10.000000
mean	96.000000
std	16.799471
min	70.000000
25%	81.250000
50%	97.500000
75%	108.750000
max	120.000000
Nombre de valeurs :	8
Moyenne :	75.03743386139236
Écart type :	41.03598660792319
Valeur minimale :	10.0
Quartile 25 :	56.69986772278472
Médiane :	88.625
Quartile 75 :	100.3125
Valeur maximale :	120.0

Groupement

Les fonctions de regroupement de Pandas sont utilisées pour diviser les données en sous-ensembles basés sur une ou plusieurs variables. Les fonctions de regroupement les plus courantes incluent :

- `groupby()` : Permet de regrouper les données en fonction d'une ou plusieurs variables.
- `agg()` : Permet d'appliquer une fonction d'agrégation à chaque groupe.

6 - Analyse Avancée et Cas Pratiques-> Agrégation et Groupement

```

import pandas as pd

df = pd.DataFrame({
  "nom": ["John", "Lydia", "Laury", "Arnaud", "Enzo"],
  "genre": ["Homme", "Femme", "Femme", "Homme", "Homme"],
  "taille": [182, 166, 170, 178, 172],
  "education": ["Master", "Master", "Licence", "Licence", "Master"],
  "salaire": [65000, 72000, 74000, 68000, 80000]
})

# Sélectionner uniquement les colonnes numériques
numeric_columns = df.select_dtypes(include=['int64', 'float64']).columns
# Appliquer la fonction mean() sur les colonnes numériques
print("\n- moyenne 1 \n", df.groupby("genre")[numeric_columns].mean())

# Sélectionner directement les colonnes pertinentes
selected_columns = ["genre", "salaire"]
# Appliquer la fonction mean() sur les colonnes sélectionnées
print("\n- moyenne 2 \n", df[selected_columns].groupby("genre").mean())

print("\n- moyenne 3\n", df.groupby("genre").agg(
  salaire_moyen=pd.NamedAgg("salaire", "mean")
))

print("\n- moyenne 4\n", df.groupby("genre").agg(salaire_moyen=("salaire", "mean")))
  
```

```

- moyenne 1
      taille  salaire
genre
Femme  168.000000  73000.0
Homme  177.333333  71000.0

- moyenne 2
      salaire
genre
Femme  73000.0
Homme  71000.0

- moyenne 3
      salaire_moyen
genre
Femme  73000.0
Homme  71000.0

- moyenne 4
      salaire_moyen
genre
Femme  73000.0
Homme  71000.0
  
```

```
print("\n- mediane&moyenne 5\n",df.groupby("genre").agg(  
    salaire_median=("salaire","median"),  
    taille_moyenne=("taille","mean"))  
))
```

```
print("\n- mediane&moyenne 6\n",df.groupby([ "genre","education"], as_index=False).agg(  
    salaire_median=("salaire","median"),  
    taille_moyenne=("taille","mean"))  
))
```

```
print("\n- mediane&moyenne 7\n",df.groupby([ "genre","education"], as_index=False).agg(  
    salaire_median=("salaire","median"),  
    taille_moyenne=("taille","mean"))  
.sort_values(by="salaire_median", ascending=False))
```

- mediane&moyenne 5				
		salaire_median	taille_moyenne	genre
	Femme	73000.0	168.000000	
	Homme	68000.0	177.333333	
- mediane&moyenne 6				
0	genre	education	salaire_median	taille_moyenne
0	Femme	Licence	74000.0	170.0
1	Femme	Master	72000.0	166.0
2	Homme	Licence	68000.0	178.0
3	Homme	Master	72500.0	177.0
- mediane&moyenne 7				
0	genre	education	salaire_median	taille_moyenne
0	Femme	Licence	74000.0	170.0
3	Homme	Master	72500.0	177.0
1	Femme	Master	72000.0	166.0
2	Homme	Licence	68000.0	178.0

L'analyse de séries temporelles est le processus d'identification et d'étude des tendances, des modèles et des anomalies dans les données qui varient au fil du temps. Elle est utilisée dans une variété d'applications, notamment la finance, la météorologie et la santé.

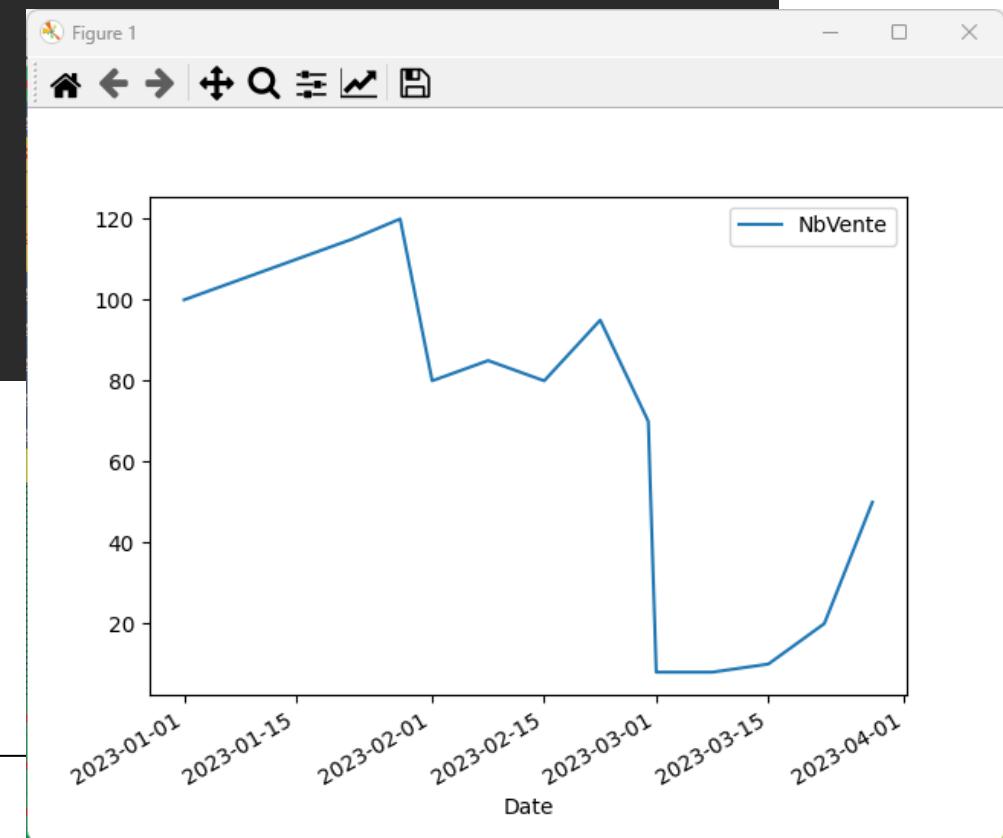
En Python, l'analyse de séries temporelles peut être effectuée à l'aide de la bibliothèque Pandas. Pandas fournit une variété de fonctions et de méthodes pour l'analyse de séries temporelles, notamment :

- La fonction `to_datetime()` : Permet de convertir une colonne de dates ou d'heures en un objet `DatetimeIndex`.
- La méthode `asfreq()` : Permet de convertir une série temporelle en une nouvelle fréquence.
- La fonction `rolling()` : Permet de calculer des statistiques sur des fenêtres glissantes de données.
- La fonction `diff()` : Permet de calculer les différences entre les valeurs successives d'une série temporelle.
- La fonction `detrend()` : Permet de supprimer la tendance d'une série temporelle.
- La fonction `seasonal_decompose()` : Permet de décomposer une série temporelle en ses composantes saisonnière, tendance et résiduelle.
- Les modèles ARIMA : Des modèles statistiques qui peuvent être utilisés pour prédire les valeurs futures d'une série temporelle.



6 - Analyse Avancée et Cas Pratiques -> Analyse de Séries Temporelles

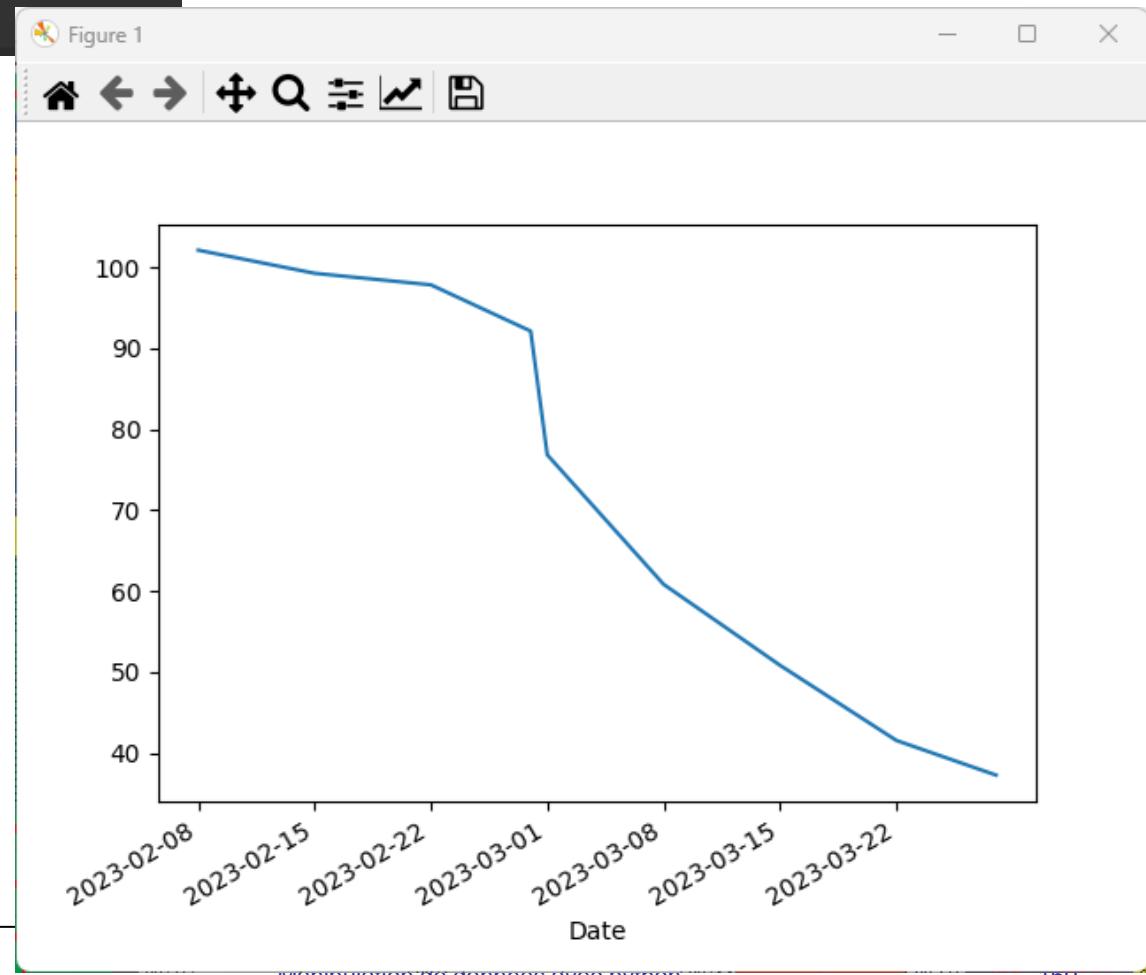
```
import pandas as pd
from matplotlib import pyplot
from scipy import signal
from statsmodels.tsa.seasonal import seasonal_decompose
df = pd.DataFrame({
    "Date": pd.to_datetime(["2023-01-01", "2023-01-08", "2023-01-15", "2023-01-22", "2023-01-28",
                           "2023-02-01", "2023-02-08", "2023-02-15", "2023-02-22", "2023-02-28",
                           "2023-03-01", "2023-03-08", "2023-03-15", "2023-03-22", "2023-03-28"]),
    "NbVente": [100, 105, 110, 115, 120,
               80, 85, 80, 95, 70,
               8, 8, 10, 20, 50]
})
# Convertir la colonne Date en une série temporelle
df_ts = df.set_index("Date")
df_ts.plot()
pyplot.show()
```



6 - Analyse Avancée et Cas Pratiques -> Analyse de Séries Temporelles

```
res= df_ts["NbVente"].rolling(7).mean()  
print("moyenne sur 7 jours : ", res)  
res.plot()  
pyplot.show()
```

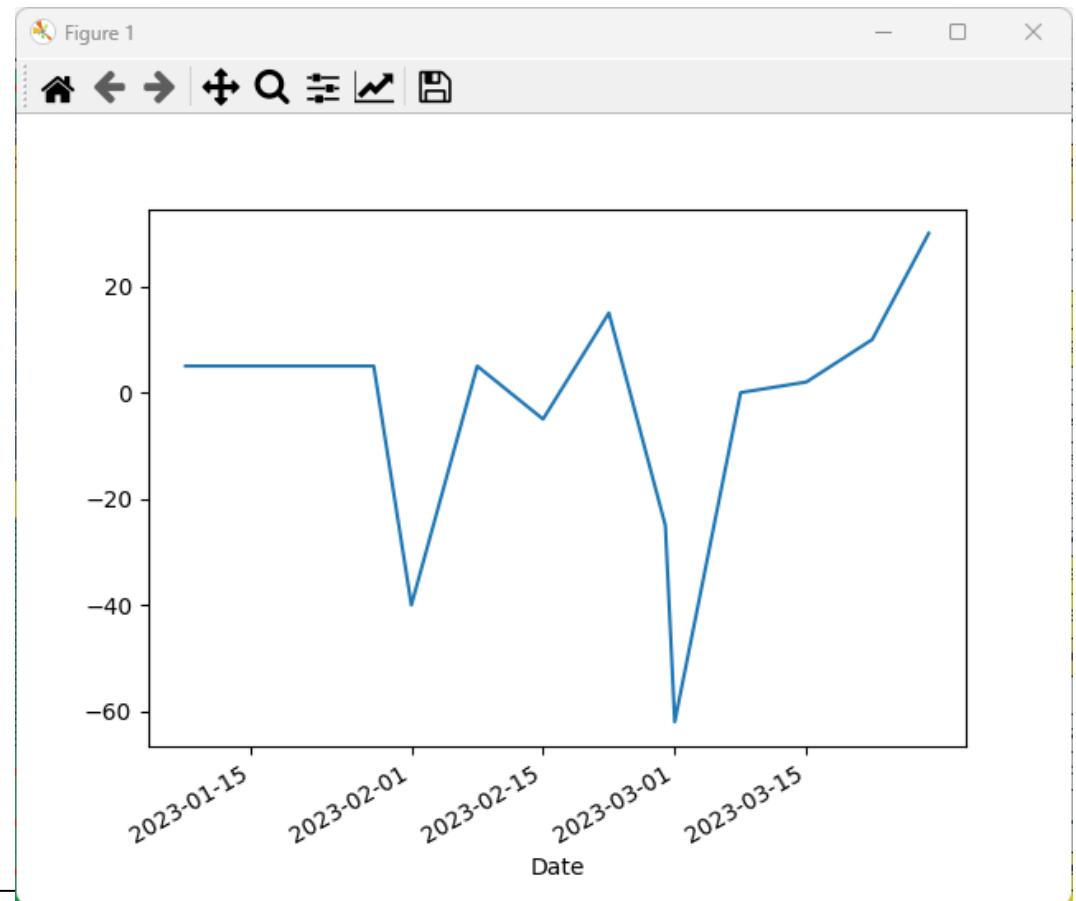
```
moyenne sur 7 jours :  Date  
2023-01-01      NaN  
2023-01-08      NaN  
2023-01-15      NaN  
2023-01-22      NaN  
2023-01-28      NaN  
2023-02-01      NaN  
2023-02-08  102.142857  
2023-02-15  99.285714  
2023-02-22  97.857143  
2023-02-28  92.142857  
2023-03-01  76.857143  
2023-03-08  60.857143  
2023-03-15  50.857143  
2023-03-22  41.571429  
2023-03-28  37.285714  
Name: NbVente, dtype: float64
```



6 - Analyse Avancée et Cas Pratiques -> Analyse de Séries Temporelles

```
res = df_ts["NbVente"].diff()  
print("différence entre les valeurs successives : ", res)  
res.plot()  
pyplot.show()
```

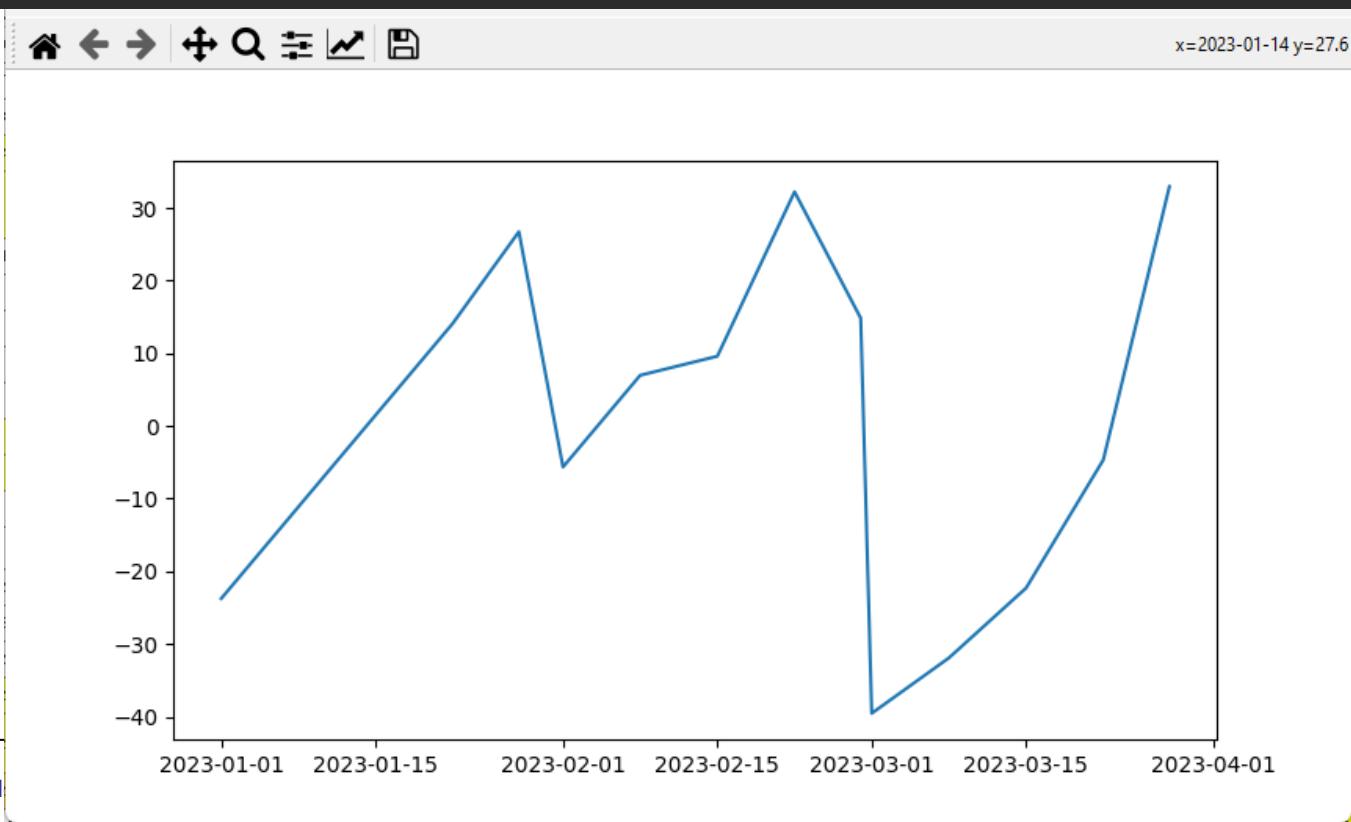
```
différence entre les valeurs successives :  Date  
2023-01-01      NaN  
2023-01-08      5.0  
2023-01-15      5.0  
2023-01-22      5.0  
2023-01-28      5.0  
2023-02-01     -40.0  
2023-02-08      5.0  
2023-02-15     -5.0  
2023-02-22     15.0  
2023-02-28    -25.0  
2023-03-01    -62.0  
2023-03-08      0.0  
2023-03-15      2.0  
2023-03-22     10.0  
2023-03-28     30.0  
Name: NbVente, dtype: float64
```



6 - Analyse Avancée et Cas Pratiques -> Analyse de Séries Temporelles

```
res= signal.detrend(df_ts['NbVente'], type='linear')
print("tendance à l'aide d'une régression linéaire : ", res)
pyplot.plot(*args: df['Date'], res, label="detrended")
pyplot.show()
```

```
tendance à l'aide d'une régression linéaire :  [-23.75      -11.12857143   1.49285714  14.11428571  26.73571429
 -5.64285714   6.97857143   9.6       32.22142857  14.84285714
 -39.53571429 -31.91428571 -22.29285714  -4.67142857   32.95      ]
```



- **La composante saisonnière** : variations régulières de la série temporelle au cours d'une période donnée, comme les variations mensuelles, trimestrielles ou annuelles.
- **La composante tendancielle** : la tendance à long terme de la série temporelle.
- **La composante résiduelle** : variations de la série temporelle qui ne sont pas expliquées par la composante saisonnière ou la composante tendancielle.

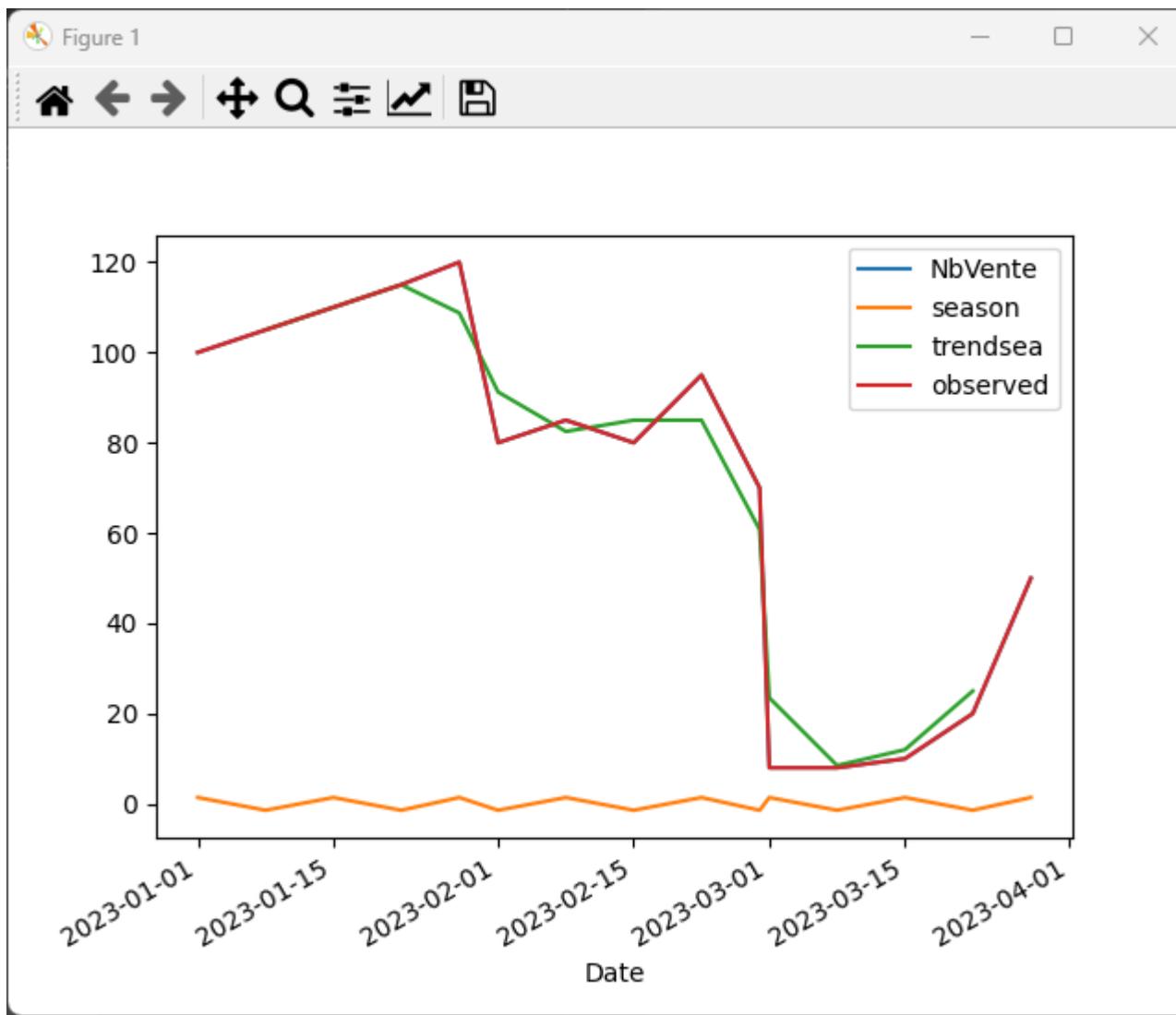
<https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality/>

http://www.xavierdupre.fr/app/ensae_teaching_cs/helpsphinx/notebooks/td2a_timeseries_correction.html

<https://blog.statoscop.fr/timeseries-4.html>

```
df_ts_decomp = seasonal_decompose(df_ts["NbVente"], model='additive', period=2)

df_ts["season"] = df_ts_decomp.seasonal
df_ts["trendsea"] = df_ts_decomp.trend
df_ts["observed"] = df_ts_decomp.observed
df_ts.plot(y=["NbVente", "season", "trendsea", "observed"])
pyplot.show()
```



Modèles ARIMA

Les modèles ARIMA sont des modèles statistiques qui peuvent être utilisés pour prédire les valeurs futures d'une série temporelle.

Ils sont basés sur les concepts d'autocorrélation et d'intercorrélation.

<https://vincmazet.github.io/signal1/xcorr/intercorrelation.html>

<https://fr.wikipedia.org/wiki/Autocorr%C3%A9lation>

https://arthurgarnier.fr/telecom/A1/S1/SICA/Diapo_Cours/CM-SIC1App_Ch3_TransfFourier.pdf

Conseils pour l'analyse de séries temporelles avec Pandas

- Assurez-vous que les données sont stationnaires avant d'appliquer des modèles ARIMA.
- Testez différents modèles ARIMA pour trouver le modèle qui convient le mieux aux données.
- Utilisez des techniques de validation croisée pour évaluer les performances des modèles ARIMA.

<https://moncoachdata.com/blog/modele-arima-avec-python/>

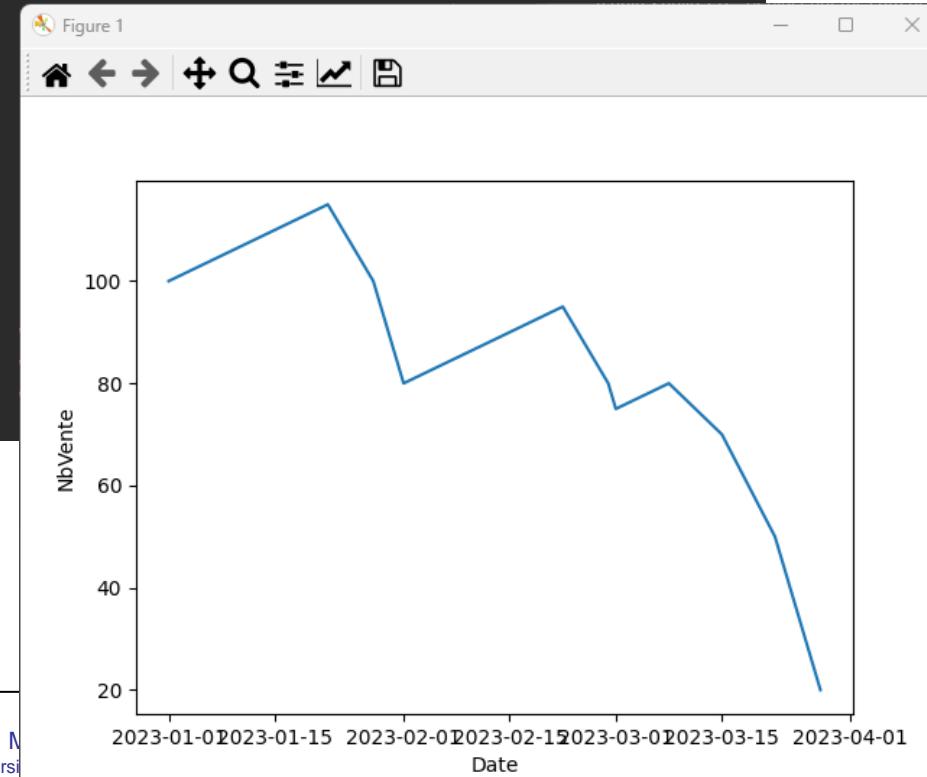
```

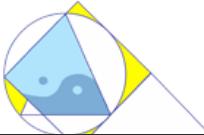
import pandas as pd
from scipy import signal
from statsmodels.tsa.seasonal import seasonal_decompose
from matplotlib import pyplot
from statsmodels.tsa.arima.model import ARIMA
from pandas.plotting import register_matplotlib_converters
from statsmodels.tsa.stattools import adfuller
register_matplotlib_converters()

df = pd.DataFrame({
    "Date": pd.to_datetime(["2023-01-01", "2023-01-08", "2023-01-15", "2023-01-22", "2023-01-28",
                           "2023-02-01", "2023-02-08", "2023-02-15", "2023-02-22", "2023-02-28",
                           "2023-03-01", "2023-03-08", "2023-03-15", "2023-03-22", "2023-03-28"]),
    "NbVente": [100, 105, 110, 115, 100,
                80, 85, 90, 95, 80,
                75, 80, 70, 50, 20]
})

print(df)
pyplot.xlabel('Date')
pyplot.ylabel('NbVente')
pyplot.plot(*args: df['Date'], df['NbVente'])
pyplot.show()

```





6 - Analyse Avancée et Cas Pratiques -> Analyse de Séries Temporelles

```
# https://moncoachdata.com/blog/modele-arima-avec-python/
_, p, _, _, _ = adfuller(df['NbVente'])
print("La p-value (doit etre < 0.05) et est de: ", round(p, 3))

# Estimer les paramètres du modèle ARIMA
model1 = ARIMA(df['NbVente'], order=(1, 2, 3)).fit()
print(model1.summary())
model2 = ARIMA(df['NbVente'], order=(1, 0, 1)).fit()
print(model2.summary())
```

La p-value (doit etre < 0.05) et est de: 0.996

```
SARIMAX Results
=====
Dep. Variable: NbVente   No. Observations: 15
Model: ARIMA(1, 2, 3)   Log Likelihood: -6855952.536
Date: Mon, 20 Nov 2023   AIC: 13711915.073
Time: 14:11:30   BIC: 13711917.897
Sample: 0 - 15   HQIC: 13711914.492
Covariance Type: opg
=====
              coef    std err      z      P>|z|      [0.025      0.975]
-----
ar.L1      2.64e-09  2.45e-07   0.011      0.991    -4.77e-07  4.82e-07
ma.L1      2.639e-09  2.45e-07   0.011      0.991    -4.77e-07  4.82e-07
ma.L2     -8.502e-08  4.06e-07  -0.209      0.834    -8.81e-07  7.11e-07
ma.L3     -1.928e-08  3.32e-07  -0.058      0.954    -6.7e-07   6.32e-07
sigma2      0.0002  8.82e-11  1.72e+06      0.000      0.000      0.000
=====
Ljung-Box (L1) (Q):      0.04      Jarque-Bera (JB):      0.57
Prob(Q):      0.84      Prob(JB):      0.75
Heteroskedasticity (H):      1.24      Skew:      0.51
Prob(H) (two-sided):      0.84      Kurtosis:      2.81
=====
```

6 - Analyse Avancée et Cas Pratiques -> Analyse de Séries Temporelles

```

# https://moncoachdata.com/blog/modele-arima-avec-python/
_, p, _, _, _ = adfuller(df['NbVente'])
print("La p-value (doit être < 0.05) et est de: ", round(p, 3))

# Estimer les paramètres du modèle ARIMA
model1 = ARIMA(df['NbVente'], order=(1, 2, 3)).fit()
print(model1.summary())
model2 = ARIMA(df['NbVente'], order=(1, 0, 1)).fit()
print(model2.summary())
  
```

SARIMAX Results

```

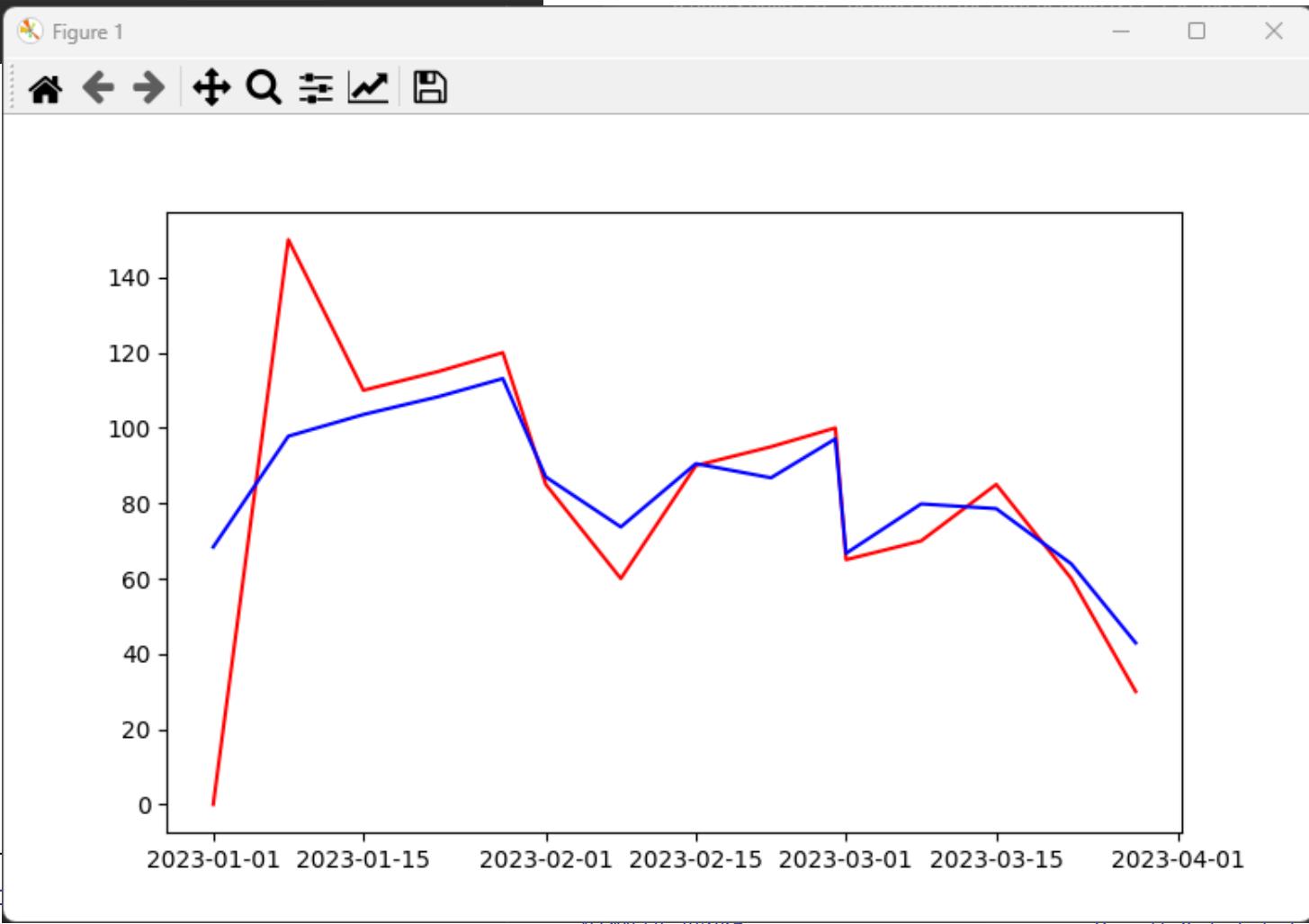
=====
Dep. Variable:          NbVente    No. Observations:             15
Model:                  ARIMA(1, 0, 1)    Log Likelihood       -58.551
Date:                  Mon, 20 Nov 2023   AIC                  125.102
Time:                  14:11:30          BIC                  127.934
Sample:                  0 - 15          HQIC                 125.072
Covariance Type:             opg
=====

            coef    std err      z      P>|z|      [0.025      0.975]
-----
const      68.3716    33.922     2.016      0.044      1.886    134.857
ar.L1      0.8663     0.215     4.033      0.000      0.445     1.287
ma.L1      0.6877     0.306     2.247      0.025      0.088     1.287
sigma2    118.0942    70.276     1.680      0.093    -19.644    255.833
=====

Ljung-Box (L1) (Q):                  0.20      Jarque-Bera (JB):       1.42
Prob(Q):                           0.66      Prob(JB):            0.49
Heteroskedasticity (H):              2.29      Skew:                 -0.51
Prob(H) (two-sided):                0.38      Kurtosis:            1.88
=====
```

6 - Analyse Avancée et Cas Pratiques -> Analyse de Séries Temporelles

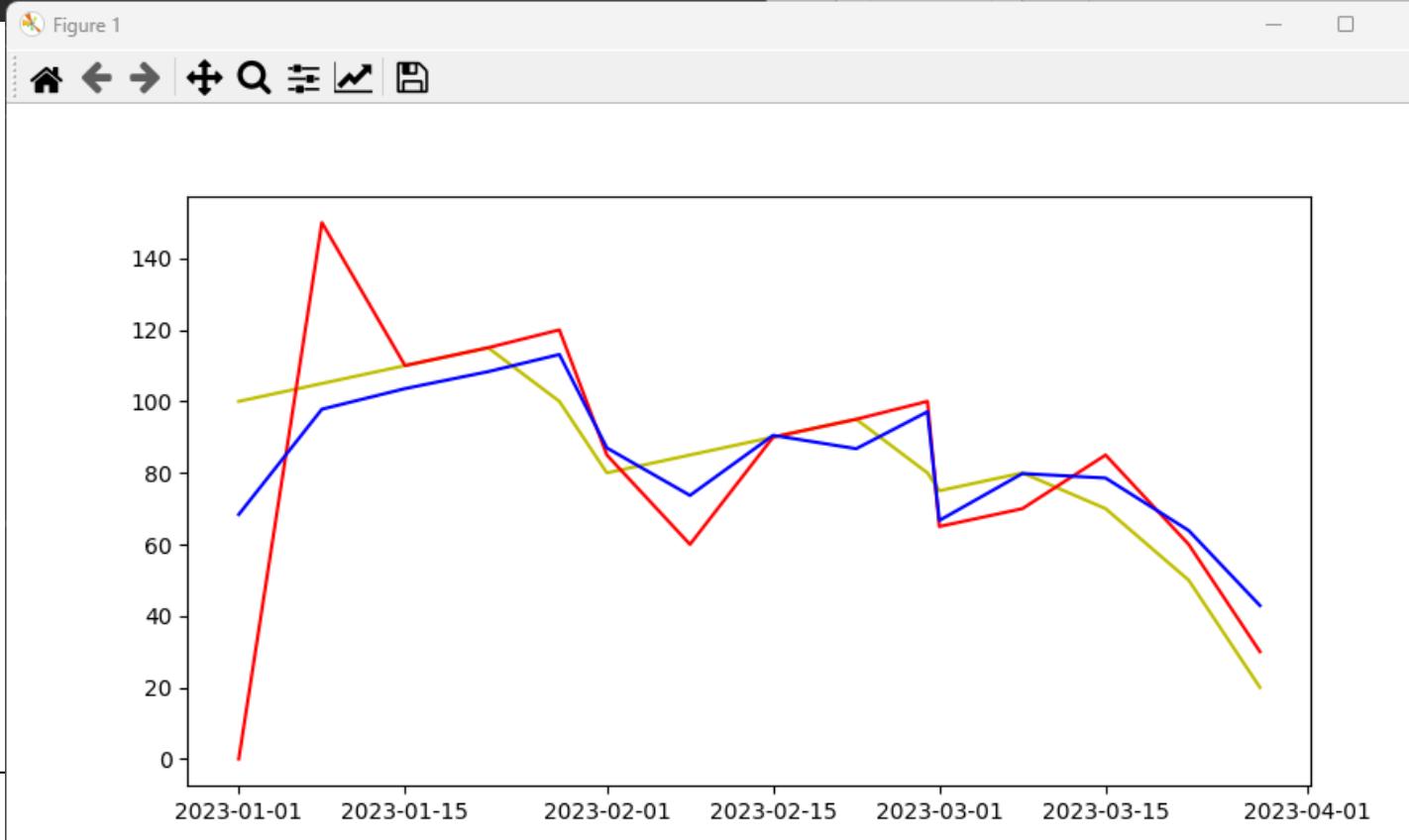
```
residuals1 = pd.DataFrame(model1.fittedvalues)
residuals2 = pd.DataFrame(model2.fittedvalues)
pyplot.plot(*args: df['Date'], residuals1, color = "r")
pyplot.plot(*args: df['Date'], residuals2, color = "b")
pyplot.show()
```



6 - Analyse Avancée et Cas Pratiques -> Analyse de Séries Temporelles

```
pred1 = model1.predict()
pred2 = model2.predict()

pyplot.plot(*args: df['Date'], df['NbVente'], color="y")
pyplot.plot(*args: df['Date'], pred1, color = "r")
pyplot.plot(*args: df['Date'], pred2, color = "b")
pyplot.show()
```



6 - Analyse Avancée et Cas Pratiques

- <https://www.stat4decision.com/fr/10-sites-de-reference-open-data/>
- <https://www.stat4decision.com/fr/10-sites-pour-trouver-des-donnees-pour-modeles/>

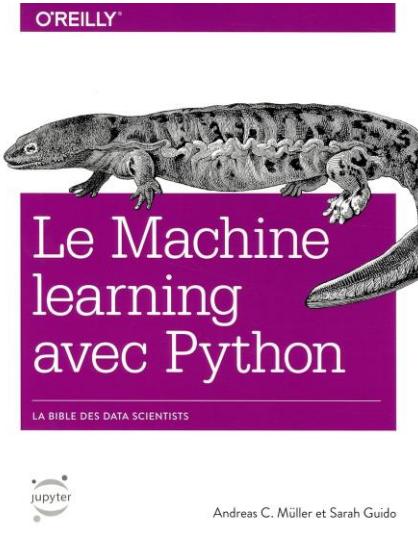
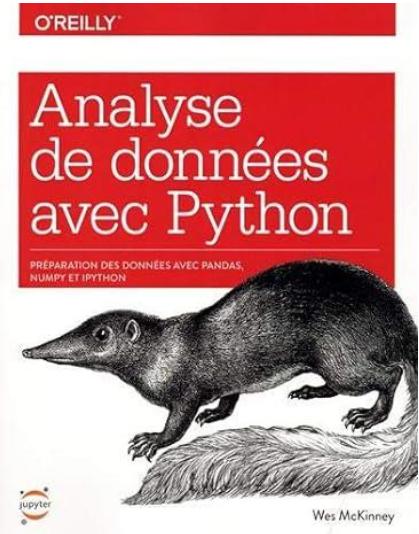
7 - Conclusion et Perspectives

La manipulation des données est un domaine en constante évolution, avec de nouveaux développements qui apparaissent régulièrement.

Perspectives pour la manipulation des données :

- L'apprentissage automatique et l'intelligence artificielle seront utilisés pour automatiser les tâches de manipulation des données.
 - Exemple, les algorithmes d'apprentissage automatique peuvent être utilisés pour identifier les valeurs aberrantes dans les données, ou pour nettoyer les données de manière automatique.
- La manipulation des données en temps réel
 - Les entreprises et les organisations devront être en mesure de traiter les données en temps réel pour prendre des décisions rapides et éclairées
- La manipulation des données sur les grands ensembles de données
 - Les entreprises et les organisations devront être en mesure de traiter des ensembles de données massifs pour tirer des insights significatifs.

8 - Bibliographie



Frédéric Bro
Chantal Remy

Python & Pandas et les 36 problèmes de data science

Problèmes et exercices corrigés pas à pas

- Data visualisation, cartes statistiques
- Fouille et analyse de données
- Modélisation, simulation, lanceur d'cerf
- Prédiction et premiers pas vers lIA

