Partie 3
Les statistiques

Chapitre 3-1
Statistiques

1. Objectif du chapitre

Les statistiques regroupent un ensemble de méthodes dédiées à l'échantillonnage de données ainsi qu'à leur analyse afin de tirer des conclusions et de comprendre les phénomènes sous-jacents à ces données. Ces méthodes statistiques font partie intégrante de la Data Science.

Il est quasiment impossible d'aborder l'ensemble des méthodes statistiques en un seul ouvrage vu leur diversité. Il existe plusieurs livres qui traitent des statistiques. L'objectif de ce chapitre est double : le premier est la présentation des outils statistiques élémentaires que tout Data Scientist devrait connaître, et le deuxième objectif est d'attirer l'attention du lecteur sur l'intérêt des statistiques et leur relation avec la Data Science. Ainsi, nous allons porter une attention particulière à la partie inférentielle des statistiques.

À la fin de ce chapitre, le lecteur aura abordé :

– les statistiques descriptives,
– les lois de probabilité,
– la loi normale et la loi normale centrée réduite,
– le principe de l'échantillonnage,
– le théorème central limite,
– l'estimation ponctuelle,

– l'estimation par intervalle de confiance,
– les tests d'hypothèses,
– le paradoxe de Simpson,
– les séries temporelles.

2. Les statistiques descriptives

Les statistiques descriptives permettent de résumer un ensemble de données de manière concise. Avec les statistiques descriptives, et pour un échantillon de valeurs $E = \{x_1, x_2, \dots x_n\}$, nous pouvons calculer certains paramètres afin de cerner la nature de la distribution associée aux valeurs x_i.

Ainsi, nous distinguons deux types de paramètres que nous pouvons calculer sur une série statistique de type quantitative : les paramètres de position et les paramètres de dispersion présentés dans les sous-sections suivantes.

2.1 Paramètres de position

Les paramètres de position permettent d'avoir une idée précise sur la nature du domaine de définition d'un ensemble de données. Ces paramètres de position, également appelés indicateurs de position, sont des nombres réels utilisés comme référence pour un ensemble $E = \{x_1, x_2, \dots x_n\}$.

2.1.1 La moyenne

La moyenne $\overline{X}$ associée à une série de valeurs $S = (x_1, x_2, \dots, x_n)$ se calcule comme suit $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} x_i$.

La moyenne ainsi calculée correspond à la moyenne arithmétique. Il existe d'autres types de moyennes telles que la moyenne harmonique, la moyenne quadratique ou encore la moyenne géométrique. Généralement, en statistiques, la moyenne utilisée est la moyenne arithmétique.

Remarque

Si la série S est un échantillon issu d'une population plus grande, alors il ne faut pas confondre la moyenne $\overline{X}$ calculée sur cet échantillon avec la moyenne de la population ! Dans la suite de ce chapitre, nous allons revenir sur la relation entre la moyenne d'un échantillon et la moyenne de la population.

2.1.2 Le mode

Le mode d'une série de valeurs est tout simplement la valeur qui apparaît le plus fréquemment.

Par exemple, dans la série de valeurs $S=(1, 2, 5, 2, 5, 5, 6, 8, 5, 9, 5)$, on dira que le mode est la valeur 5, car c'est bien cette valeur qui apparaît avec le plus d'occurrences. La valeur 5 apparaît cinq fois, la valeur 2 deux fois, puis les autres valeurs apparaissent une fois chacune.

La série S de notre exemple est dite unimodale, car elle dispose d'un seul mode. La série $S^1=(1, 2, 5, 2, 5, 5, 2, 2, 5, 2, 5)$ est dite bimodale, car elle dispose de deux modes, à savoir le mode 5 et le mode 2.

2.1.3 La médiane

La médiane associée à une série de valeurs rangée dans l'ordre croissant $S = (x_1, x_2, \ldots, x_n)$ avec $x_i < x_{i+1}\ \forall\, i$ est une valeur qui partage toutes les valeurs de S en deux groupes de valeurs de taille égale. La valeur de la médiane peut ou pas faire partie de S.

Si les valeurs x_i de S sont des valeurs discrètes, alors la médiane se calcule comme suit :

– Si la taille n de la série S est impaire, alors nous pouvons écrire $n=2p+1$. Comme les valeurs de S sont ordonnées, la médiane est la $(p+1)^{\text{ième}}$ valeur. Dans ce cas, la médiane fait partie de la série S.

  Par exemple, pour la série $S=(1, 2, 5, 8, 15, 55, 68, 82, 125, 199, 500)$, la médiane est la valeur 55, alors que la moyenne est égale à 96,36.

– Si la taille n de la série S est paire, alors nous pouvons écrire $n=2p$. Comme les valeurs de S sont ordonnées, la médiane est la moyenne entre la $(p+1)^{\text{ième}}$ valeur et la $p^{\text{ième}}$ valeur. Dans ce cas, la médiane ne fait pas partie de la série S.

  Par exemple, pour la série $S=(1,2,5,8,15,55,68,82,125,199)$, la médiane est la valeur $\frac{15+55}{2} = 35$ et la moyenne est égale à 56.

Remarquez que dans des deux derniers exemples, les différences entre les moyennes et les médianes étaient importantes !

Si les valeurs x_i de S sont des valeurs continues, alors la médiane correspond à la valeur centrale que nous pouvons calculer par interpolation linéaire du centre des effectifs cumulés sur les x_i.

Par exemple, soit le tableau récapitulatif des notes obtenues par un groupe de 100 étudiants :

Notes	Effectifs
[0;5[	15
[5;7[	30
[7;11[	20
[11;16[	25
[16;20[	10
Total	100

À partir de ce tableau, nous allons procéder au calcul des effectifs cumulés comme suit :

Notes	Effectifs	Effectifs cumulés
[0;5[	15	15
[5;7[	30	45
[7;11[	20	65

Notes	Effectifs	Effectifs cumulés
[11;16[	25	90
[16;20[	10	100

D'après la colonne des effectifs cumulés, nous avons :

– 15 étudiants qui ont une note en dessous de 5.

– 45 étudiants qui ont une note en dessous de 7.

– 65 étudiants qui ont une note en dessous de 11.

– 90 étudiants qui ont une note en dessous de 16.

– 100 étudiants qui ont une note en dessous de 20.

Le nombre total des étudiants est égal à 100, donc la moitié est égale à 50 étudiants. L'intervalle sur l'axe des effectifs cumulés où se situe la médiane est l'intervalle [45;65], puisque 50 appartient à cet intervalle.

Nous pouvons calculer la valeur de la médiane par interpolation linéaire comme sur la figure 6-1 suivante :

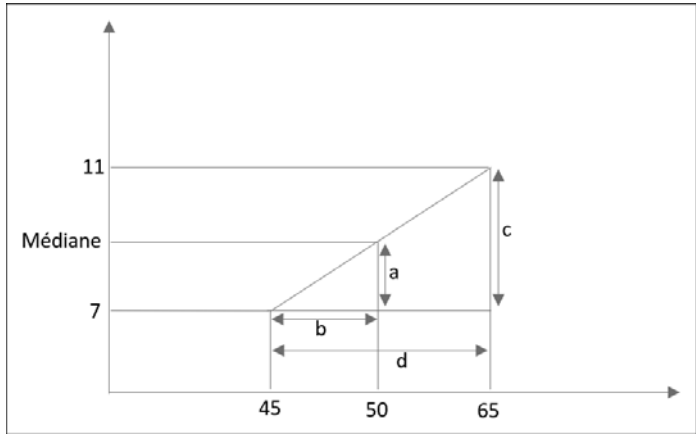

Figure 6-1 : Calcul de la médiane par interpolation linéaire du centre des effectifs cumulés

Grâce au théorème de Thalès, nous savons que $\frac{a}{b} = \frac{c}{d}$ (voir la figure précédente).

a = médiane - 7.

b = 50 - 45 = 5.

c = 11 - 7 = 4.

d = 65 - 45=20.

En procédant au remplacement de a, b, c et d par leurs valeurs respectives, on obtient $\frac{médiane - 7}{5} = \frac{4}{20}$.

Donc la médiane est égale à 8.

2.1.4 Les quartiles

Pour une série de valeurs rangées dans l'ordre croissant $S = (x_1, x_2, \ldots, x_n)$ avec $x_i < x_{i+1} \ \forall \ i$, les quartiles sont définis par trois valeurs qui partagent les valeurs de la série S en quatre groupes de valeurs de même taille. Ces trois valeurs sont définies comme suit :

– Le deuxième quartile correspond à la valeur de la médiane définie ci-dessus.

– Le premier quartile partage les valeurs de S situées entre la première valeur x_1 et la médiane en deux groupes de valeurs de même taille. En d'autres termes, le premier quartile est la médiane de la série S_1 constituée des valeurs entre x_1 et la médiane.

– De même que pour le premier quartile, le troisième quartile partage les valeurs de S situées entre la médiane et la dernière valeur xn en deux groupes de valeurs de même taille. En d'autres termes, le troisième quartile est la médiane de la série S_2 constituée des valeurs entre la médiane et x_n.

2.2 Paramètres de dispersion

Les paramètres de dispersion permettent de comprendre la variabilité associée à une série de données quantitatives.

2.2.1 La variance

La variance associée à un échantillon de données est une mesure qui nous renseigne sur la moyenne des carrés des écarts à la moyenne. On peut également dire que la variance est une valeur qui nous donne une idée sur la dispersion d'un ensemble de valeurs autour de leur moyenne.

Pour une série statistique $X = (x_1, x_2, \dots, x_n)$, la variance *Var(X)* se calcule comme suit : $Var(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{X})^2$.

Avec $\overline{X}$ la moyenne de la série X et qui, pour rappel, se calcule comme suit $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} x_i$

À partir de la formule de *Var(X)*, on voit bien que la variance est une distance moyenne au carré qui sépare chaque élément x_i de sa moyenne $\overline{X}$.

2.2.2 Calcul de la variance avec la formule de Koenig

Le calcul de la variance *Var(X)* avec la formule vue précédemment peut être compliqué à réaliser sans l'aide d'un logiciel. La formule de Koenig est une simplification de la formule précédente et qui permet de calculer la variance comme suit : $Var(X) = \frac{1}{n}\sum_{i=1}^{n} {x_i}^2 - \overline{X}^2$.

Avec cette formule, la variance est la différence entre la moyenne des carrés des x_i et le carré de la moyenne x_i.

Avec la formule de Koenig, le calcul de la variance devient plus facile et une simple calculatrice serait largement suffisante.

2.2.3 L'écart-type

Nous avons vu que la variance est la distance moyenne au carré d'un ensemble de valeurs par rapport à leur moyenne. L'écart-type σ est tout simplement la distance moyenne, ou la distance type, qui sépare les valeurs de leur moyenne et il se calcule à partir de la variance comme suit : $\sigma = \sqrt{Var(X)}$.

2.2.4 L'écart interquartile

L'écart interquartile correspond à la distance entre le troisième et le premier quartile.

Par exemple, pour la série $S=(1, 2, 5, 8, 15, 55, 68, 82,125, 199, 500)$, les quartiles sont les suivants :

– Le premier quartile est égal à 5.

– Le deuxième quartile est égal à 55, ce qui est également la médiane.

– Le troisième quartile est égal à 125.

Pour cette série S, l'espace interquartile est donc égal à $125\text{-}5 = 120$.

La valeur de l'écart interquartile, aussi appelé l'espace interquartile ou bien l'étendue interquartile, nous renseigne sur l'ampleur de la dispersion des valeurs d'une série.

En combinant les mesures de la moyenne, du mode, de la médiane, de la variance, de l'écart-type et de l'écart interquartile, nous obtenons un résumé de l'ensemble des valeurs étudiées. Toutes ces mesures peuvent être calculées sur un ensemble de valeurs associées à une seule variable quantitative. Dans le chapitre Analyse en composantes principales, nous aborderons l'analyse en composantes principales, qui est une méthode permettant de résumer un ensemble de données définies avec plusieurs variables.

3. Les lois de probabilité

Une loi de probabilité permet de cerner le comportement d'une variable aléatoire. Dans le domaine des probabilités, une variable aléatoire dépend du hasard. Justement, c'est le comportement de ce hasard que l'on tente de décrire avec une loi de probabilité. Avec une loi de probabilité, nous pouvons calculer la probabilité qu'une variable aléatoire soit fixée à une valeur donnée.

Par exemple, si nous considérons une variable X associée au résultat obtenu après le lancer d'un dé à six chiffres, alors cette variable X sera appelée une variable aléatoire, puisque la survenue de l'un des six chiffres est un événement aléatoire.

Si nous supposons que notre dé est parfait, c'est-à-dire que chacun des six chiffres est équiprobable avec une probabilité de $\frac{1}{6}$, alors la loi de la variable X est tout simplement $F(X) = \frac{1}{6}$.

Le choix d'une loi de probabilité est en fonction de la nature de la variable aléatoire étudiée et en fonction du phénomène associé à cette variable aléatoire. En effet, une variable aléatoire X peut être discrète ou continue et elle peut être définie dans un intervalle fini, semi-fini ou infini. Le phénomène associé à une variable aléatoire peut concerner des durées, des quantités comme le poids, des longueurs comme la taille ou toute autre mesure qui peut être observée et relative à un événement quelconque.

En résumé, une loi de probabilité décrit la manière dont sont distribuées les valeurs possibles d'une variable aléatoire X.

Il existe un nombre important de lois de probabilité. Loin d'être exhaustifs, parmi les plus usuelles nous pouvons citer :

- Loi de Bernoulli : décrit la distribution d'une variable aléatoire associée à un événement à deux issues possibles.
- Loi binomiale : décrit la distribution d'une variable aléatoire associée à un événement à deux issues possibles répété plusieurs fois avec remise, c'est-à-dire que le même événement peut survenir plusieurs fois.
- Loi de Poisson : décrit la distribution d'une variable aléatoire associée au nombre d'événements se produisant dans un intervalle de temps donné. Par exemple, le nombre de personnes supplémentaires dans une file d'attente au bout d'un temps donné.
- Loi uniforme : décrit la distribution d'une variable aléatoire associée à un ensemble d'événements équiprobables.
- Loi exponentielle : décrit la distribution d'une variable aléatoire associée au délai de la survenue d'un événement. Par exemple, le délai de l'arrivée d'une personne supplémentaire dans une file d'attente.

Chapitre 3
Préparer les données avec Pandas et Numpy

1. Pandas, la bibliothèque Python incontournable pour manipuler les données

Nous sommes désormais prêts à explorer et analyser nos données. Pour ce faire, nous nous appuierons sur l'un des modules Python les plus essentiels : Pandas.

Pandas est la bibliothèque de Python permettant de manipuler et analyser les données. Elle a été créée en 2008 par Wes McKinney, un statisticien et développeur Python. Cette bibliothèque s'est progressivement imposée comme un élément incontournable pour gérer des données et a contribué à faire de Python une référence en la matière.

1.1 Installation

Pour commencer, assurons-nous qu'elle est bien installée. Si tel n'est pas le cas, validons la ligne suivante dans une invite de commande :

```
pip install pandas
```

Une fois installée, il suffit de l'importer :

```
import pandas as pd
```

L'alias `pd` est couramment utilisé pour simplifier les commandes liées à Pandas. Il permet ensuite de s'assurer que les commandes pandas que nous lançons s'y réfèrent bien.

1.2 Structure et type de données

Avant de l'utiliser, prenons le temps d'expliquer les différentes structures que peut prendre pandas. Celle que tout le monde a en tête est le DataFrame à deux dimensions se présentant comme un tableur Excel, mais il existe en réalité un type par dimension :

Nom de la structure	Nombre de dimensions	Principe
Séries	1	Données unidimensionnelles indexées
DataFrames	2	Feuille de calcul avec lignes et colonnes
Panels	3	Collection de DataFrames
DataFrames multi-index	4	DataFrames avec multi-index pour les lignes ou les colonnes

Remarque

Il existe aussi une structure `panel4D` de quatre dimensions mais elle est dépréciée et il est déconseillé de l'utiliser.

Dans l'immense majorité des cas, seuls les séries et les DataFrames seront rencontrés, mais il est utile de connaître l'existence des autres structures.

L'indexation est au cœur de ces structures, de manière beaucoup plus présente que dans les tableurs. Elle permet d'accéder facilement aux lignes, colonnes ou cases. Pandas propose d'ailleurs plusieurs façons en offrant la possibilité de s'y référer soit par leur indice soit par leur nom.

Petite précision utile : l'indexation commence à 0 et non à 1.

Le tableau suivant indique les quatre façons de faire :

Action	iloc	loc	at	query
Accès à une case	`df.iloc[2,1]`	`df.loc[2,'Col2']`	`df.at[2,'Col2']`	Inapproprié
Accès à une ligne	`df.iloc[2, :]`	`df.loc[2, :]`	Inapproprié	`df.query ("index==2")`
Accès à une colonne	`df.iloc[ :,1]`	`df.loc[ :, 'Col2']`	Inapproprié	Inapproprié
Filtrage avec condition	`masque = df['A'] > 2 df.iloc[masque .values, 1]`	`masque = df['A'] > 2 df.loc[masque .values, "B"]`	Inapproprié	`df.query ("A>2")`

Remarque

La grande différence entre `iloc` et `loc`, qui sont les fonctions les plus utilisées, réside dans le fait que tout est numérique avec `iloc` alors que `loc` oblige à préciser le nom de la variable.

1.3 Possibilités offertes

Passé le côté structurel, regardons ensemble ce qui a fait le succès de Pandas : sa capacité à répondre à tous les besoins inhérents aux manipulations de données.

Dès le départ, Pandas offre l'assurance de pouvoir lire quasiment tous les types de fichiers. Outre les formes les plus classiques comme les fichiers texte, CSV, Excel ou JSON, nous est aussi offerte la possibilité de lire du SQL, des fichiers propriétaires comme SAS ou SPSS et d'autres formats que nous rencontrerons plus tard comme HDF5, Parquet ou Pickle.

Une fois les données acquises, nous pouvons accéder simplement à différentes informations comme les dimensions, le type des variables, le nombre d'observations non nulles par colonne ou des statistiques descriptives de base.

Pandas nous offre ensuite toute la panoplie de fonctions pour préparer les données : supprimer des observations ou des colonnes, imputer, remplacer, fusionner d'autres fichiers ou créer de nouvelles variables.

À cela s'ajoutent des capacités de visualisations graphiques, très pratiques pour appréhender les données. Cependant, pour des besoins plus avancés, nous pourrions préférer les fonctionnalités offertes par des modules dédiés tels que Matplotlib ou Seaborn.

Cette interopérabilité avec les autres modules est d'ailleurs un des autres piliers qui fait la force de Pandas. En plus des bibliothèques graphiques mentionnées précédemment, Pandas interagit parfaitement avec tous les modules Python dédiés à la data science comme Scikit-Learn ou Numpy. Nous allons justement présenter ce dernier qui agit dans l'ombre de Pandas.

2. Numpy, le pilier du calcul numérique

Numpy est une bibliothèque fondamentale pour le calcul scientifique en Python agissant comme une infrastructure de base pour de nombreuses autres bibliothèques. Prenons le temps de voir ensemble ce qui fait sa force et pourquoi elle est omniprésente dans les modules Python.

2.1 La structure ndarray

Le ndarray, abréviation de N-dimensionnal array signifiant tableau à n dimensions, est vraiment au cœur de la bibliothèque Numpy. Deux autres structures, les matrix et scalars, sont également proposées mais nous polariserons notre attention sur le ndarray tant son rôle est central.

Remarque

Attention, quantité de noms différents peuvent renvoyer à un ndarray. Ainsi, le terme tableau est la forme générique pour les qualifier mais nous pouvons rencontrer le terme vecteur pour des ndarray à une dimension ou matrice pour ceux en comprenant deux. Au-delà de deux dimensions, il est courant de rencontrer les appellations : array, NDArray, tenseur ou tensor.

Commençons par étudier les caractéristiques de cette structure fondamentale de Numpy.

2.1.1 Une structure homogène

Avant tout, c'est un tableau multidimensionnel homogène, c'est-à-dire qu'il peut prendre autant de dimensions que souhaité avec la contrainte que toutes les données soient du même type. Voici un moyen simple de créer un vecteur ndarray à partir d'une simple liste Python :

```
import numpy as np
a = [1, 2, 3, 4, 5]
array_numpy = np.array(a)
print(array_numpy.dtype)

# Output: int64
```

Quant à l'obligation d'homogénéité mentionnée plus haut, notons que l'introduction d'un membre de type float entraîne de facto le changement de type de l'ensemble du ndarray :

```
array_numpy_2 = np.array([1.0, 2, 3, 4, 5])
print(array_numpy_2.dtype)

# Output: float64
```

Cet exemple n'avait pour but que de créer un ndarray à partir d'une structure familière. Maintenant que nous avons pris nos marques, voici la façon de le créer directement :

```
debut = 0
fin = 10 #valeur non incluse
pas = 2
array_numpy = np.arange(debut, fin, pas) # Le pas peut être décimal

print(array_numpy)
# Output: [0 2 4 6 8]
```

Et l'éventail des possibilités ne s'arrête pas là. La commande `linspace`, par exemple, permet de créer un vecteur avec un certain nombre de valeurs d'intervalles identiques, très pratique lors de la création de graphiques :

```
array_linspace = np.linspace(0,100,15)
```

Cette simple ligne de code va ainsi générer un vecteur entre 0 et 100 inclus possédant 15 valeurs en tout à intervalles égaux.

Voyons maintenant par quels moyens créer des matrices à deux dimensions.

Cela peut se faire directement depuis une table Pandas de la façon la plus simple possible :

```
# df est un DataFrame pandas
array_from_pandas = df.values
```

Ici, nous avons récupéré tout le DataFrame mais nous n'aurions pu obtenir qu'une seule variable sous forme de vecteur de cette façon :

```
array_col_pandas = df['nom_colonne'].values
```

Ou plus :

```
array_cols_pandas = df[['nom_colonne1', 'nom_colonne2',
'nom_colonne3']].values
```

Il nous est aussi offert la possibilité de créer une matrice en fixant les dimensions et le contenu. Comme une matrice nulle, remplie de zéros, à l'aide de la commande `zeros` :

```
nb_rows = 10
nb_columns = 10

shape = (nb_rows, nb_columns)
matrice_zeros = np.zeros(shape)
```

Les commandes `ones` et `full` procèdent de la même façon mais avec respectivement des 1 ou une valeur spécifique définie pour toutes les cases :

```
# Matrice remplie de 1
matrice_1 = np.ones(shape)

# Matrice remplie de 100 (Tous les nombres sont possibles) :
matrice_100 = np.full(shape, 100)
```

Enfin, signalons les différentes possibilités offertes pour générer des nombres aléatoires, qu'ils soient entiers ou décimaux.

Pour les nombres décimaux, nous pouvons demander une matrice de nombres uniformes entre 0 et 1 grâce à `random.rand` :

```
# Exemple de matrice de 3 lignes sur 4 colonnes :
matrice_rand = np.random.rand(3,4)
```

L'utilisation nous permettra d'obtenir des nombres gaussiens, c'est-à-dire qui ont une moyenne de 0 et un écart-type de 1 :

```
# Exemple de matrice de 3 lignes sur 4 colonnes :
matrice_randn = np.random.randn(3,4)
```

La création des nombres entiers va nécessiter une opération supplémentaire : l'usage du mot-clé `reshape` car il n'est pas possible de définir directement les dimensions. Nous indiquerons ainsi dans `reshape` le nombre de lignes et de colonnes souhaitées :

```
# Création d'une liste de nombres entiers aléatoires
# random.randint(low, high=None, size=None)
randint_numpy = np.random.randint(0,50,100)

# Tranformation en matrice de 10 lignes sur 10 colonnes
randint_numpy = randint_numpy.reshape(10,10)
```

Cette matrice aurait pu être créée directement en une ligne en chaînant les commandes :

```
randint_numpy = np.random.randint(0,50,100).reshape(10,10)
```

Remarque

Attention à ce que le produit des dimensions de la nouvelle forme produite avec reshape corresponde exactement à la longueur de la liste de nombres aléatoires, sinon nous obtiendrons une erreur.

```
array([[ 0, 39, 24, 36, 35,  5,  6,  3, 34, 40],
       [33, 28,  4, 26, 32, 45,  9,  5, 33,  7],
       [30,  8, 20,  7,  3, 21, 27, 44,  3, 38],
       [20,  7, 19, 31,  0,  5, 27, 43, 30,  9],
       [19,  7, 21, 37, 28,  8, 43, 46,  0, 40],
       [38, 25, 10, 34, 23, 32, 19, 26, 14, 32],
       [ 6, 33, 44, 45, 41,  4, 29, 27, 17, 35],
       [ 2, 20, 45, 15, 36, 41,  4, 49, 13, 30],
       [45, 23, 34, 35,  9, 26, 26, 22, 12, 15],
       [34, 26, 38, 46, 16, 47, 40,  0, 10, 11]])
```

Avant de clôturer ce point, signalons une possibilité commune à toutes les structures Numpy : la possibilité de définir le dtype qui va impacter les performances. Ainsi, plus la place allouée en bits va être élevée, plus grande sera la précision mais au détriment du temps de calcul et vice versa.

Regardons ensemble la conséquence du changement de type sur une matrice :

```
# Définition de la graine d'aléa pour retrouver le même résultat
np.random.seed(0)

m1 = np.random.randn(3,4)
m2 = m1.astype(np.float16) # Modification du type de 64 à 16 bits

print("m1:\n",m1)
print("m2:\n",m2)
```

```
m1:
 [[ 1.76405235  0.40015721  0.97873798  2.2408932 ]
 [ 1.86755799 -0.97727788  0.95008842 -0.15135721]
 [-0.10321885  0.4105985   0.14404357  1.45427351]]
m2:
 [[ 1.764   0.4001  0.9785  2.24  ]
 [ 1.867  -0.977   0.95   -0.1514]
 [-0.1032  0.4106  0.144   1.454 ]]
```

Après cette exploration des différentes structures, regardons comment accéder aux données.

2.1.2 L'indexation

À l'instar des listes Python, les éléments des tableaux sont indexés. L'accès y est ainsi facilité. Mais la comparaison avec les simples listes Python s'arrête là car les ndarray offrent beaucoup plus de vitesse et d'options.

Commençons par découvrir comment accéder à nos données.

L'accès à un élément spécifique, dans le cadre d'une matrice à deux dimensions, consiste simplement à fournir le numéro de ligne et de colonne :

```
print(m2[0,1]) # Affichage de l'élément ligne 0 / colonne 1

# Output: 0.4001
```

Si nous souhaitons accéder à une ligne ou une colonne en particulier, il faut simplement recourir aux « : » indiquant d'inclure tous les éléments concernés :

```
print(m2[0, :]) # Affichage de la première ligne (d'index 0)

# Output: array([1.764 , 0.4001, 0.9785, 2.24  ], dtype=float16)
```

```
print(m2[ :, 1]) # Affichage de la deuxième colonne (d'index 1)

# Output: array([ 0.4001, -0.977 ,  0.4106], dtype=float16)
```

Nous pouvons aussi utiliser les « : » comme dans le cadre des listes pour afficher plusieurs lignes ou colonnes comme ici :

```
print(m2[ :, 1:3]) # Affichage des colonnes d'index 1 et 2


# Output:
array([[ 0.4001,  0.9785],
       [-0.977 ,  0.95  ],
       [ 0.4106,  0.144 ]], dtype=float16)
```

Pour finir, l'accès à des colonnes ou des lignes discontinues se fait en passant par une liste comme ici :

```
print(m2[:, [0,3]] # Affichage des colonnes d'index 0 et 3

# Output:
array([[ 1.764 ,  2.24  ],
       [ 1.867 , -0.1514],
       [-0.1032,  1.454 ]], dtype=float16)
```

Nous invitons celles et ceux qui ne sont pas encore très familiers avec ce type d'indexation à bien prendre le temps de les pratiquer car Numpy va revenir constamment au cours des différentes étapes.

Profitons de ce point sur les indexations pour aborder le sujet des manipulations des tableaux que nous allons rencontrer fréquemment.

2.1.3 La modification des structures

Il est courant de devoir agir sur la structure des données. En effet, certaines étapes nécessitent d'assembler des tables, d'autres de remettre le vecteur à plat ou de jouer sur ses dimensions. Une mise au point rapide s'impose en commençant par ce qu'il vaut mieux éviter : la fusion.

La fusion, également appelée merging, consistant à combiner deux tables en utilisant une clé primaire d'assemblage, est mieux réalisée avec Pandas, qui offre toutes les fonctionnalités nécessaires à cet effet. Nous verrons plus loin comment accomplir cette opération dans les meilleures conditions.