

La science des données

*Théorie et applications avec R et Python*

Édité par Daniel Lemire et Neila Mezghani

*avec Élodie Boissières, Robert Godin, Habib Louafi, Richmond  
Osei, Shadi Shuraida, Renée-Maria Schmitt et Dragos Vieru*

Première édition, mai 2024

<b>Avant propos</b>	<b>3</b>
<b>Chapitre 1</b>	<b>Le langage R pour la science des données 10</b>
<b>Chapitre 2</b>	<b>Introduction au langage Python 31</b>
<b>Chapitre 3</b>	<b>Traitement des données avec Python 60</b>
<b>Chapitre 4</b>	<b>Visualisation des données 103</b>
<b>Chapitre 5</b>	<b>Inférence statistique avec R 118</b>
<b>Chapitre 6</b>	<b>Apprentissage machine avec R et Python 141</b>
<b>Chapitre 7</b>	<b>Visualisation des données et analytique d'affaires 170</b>
<b>Chapitre 8</b>	<b>L'IA explicable - La panacée pour une IA éthique ? 205</b>
<b>Chapitre 9</b>	<b>Sécurité et vie privée en science des données 237</b>

## Avant propos

Depuis une quinzaine d'années, un domaine scientifique a émergé, celui de la science des données. Presque au même moment et en parallèle, le terme mégadonnées ou Big Data a émergé de travaux en sciences physiques où l'on capture et traite des volumes de données littéralement astronomiques.

La science des données est un domaine qui comprend notamment les statistiques et l'informatique en son cœur et qui est motivée essentiellement par des considérations pratiques. La science des données consiste à extraire des informations à partir de grandes quantités de données en utilisant diverses méthodes, algorithmes et processus scientifiques. Elle permet de découvrir des modèles cachés au sein des données brutes. Ce domaine a émergé grâce à l'évolution des statistiques mathématiques, de l'analyse des données et du *big data*.

### Pourquoi ce livre?

Quand nous nous sommes mis à la science des données, nous avons trouvé plusieurs livres, des cours en ligne, des vidéos, des blogs et d'autres ressources de grande qualité. Cependant, nous avons rencontré deux contraintes principales : (1) une contrainte linguistique puisque la majorité de ces ressources sont présentés en anglais et (2) une contrainte scientifique lié au fait que leurs contenus sont généralement ou bien très théorique ou bien trop pratique. Par conséquent, il était difficile de passer de la théorie à la pratique et vice versa. De telles ressources sont souvent destinés à des lecteurs ayant

des bases solides en mathématiques ou bien à des ingénieurs en informatique ayant de bonnes connaissances en programmation.

Or, pour comprendre les concepts théoriques en science des données et pouvoir les appliquer, il est nécessaire de traiter aussi bien les aspects théoriques que pratiques. Ainsi, outre la présentation de la théorie des divers aspects de la science des données, ce livre comprend de nombreux exemples de code en langage de programmation R et Python pour vulgariser et simplifier la compréhension de la théorie. Le choix de ces langages se justifie par leur simplicité et leur importance; il s’agit actuellement des langages les plus utilisés en science des données.

Nous avons donc voulu écrire un manuel qui permet aux lecteurs de se familiariser avec la science des données de différents points de vue. Il va des aspects techniques aux enjeux de sécurité et de vie privée. Le manuel traite de plusieurs sujets ayant trait à l’intelligence artificielle, tant du point de vue informatique que sociétal.

## **Plan du livre**

Ce livre est organisé en 9 chapitres.

- Les chapitre 1 et 2 sont consacrés aux langages R et Python respectivement. Nous croyons qu’une familiarité avec ces deux langages est un atout pour l’étude la science des données. D’ailleurs, la majorité des chapitres suivants prennent appui sur ces deux premiers chapitres.
- Le chapitre 3, est rédigé par Lemire et Godin. Dans ce chapitre, ils couvrent plusieurs aspects fondamentaux du

traitement des données soit leur extraction, leur transformation et leur chargement en Python.

- Le chapitre 4 est consacré à la visualisation des données qui consiste à transformer les chiffres bruts en graphiques, diagrammes ou autres représentations visuelles.
- Le chapitre 5 traite le processus par lequel on généralise les données d'un échantillon à une population : la statistique inférentielle. Au sein de ce chapitre Mezghani présente une des forces du langage de programmation R qui est l'analyse statistique.
- Le chapitre 6 porte sur un sous-domaine essentiel de l'intelligence artificielle : l'apprentissage machine. Dans ce chapitre, Lemire, Godin et Mezghani présentent de nombreux exemples de traitement de données et de construction de modèles avec R et Python.
- Le chapitre 7, a été rédigé par Shuraïda. Ce chapitre présente la visualisation des données dans le cadre de l'analytique d'affaires. L'objectif étant d'interpréter des données afin d'améliorer la prise de décision au sein de l'organisation
- Le chapitre 8 est rédigé par Vieru, Schmitt et Boissières. Ils font un tour d'horizon de l'intelligence artificiel explicable. En effet, la science des données utilise l'intelligence artificielle, mais parfois de manière opaque. Au sein de ce chapitre, un des objectifs visés est de concilier les facteurs éthiques avec le développement d'outils informatiques efficaces.

- Le chapitre 9 passe en revue les enjeux de sécurité et de vie privée en science des données. Au sein du huitième chapitre, Louafi et Osei illustrent leur propos avec plusieurs exemples, et ils fournissent des exemples de code en Python.

Le contenu de ce manuel est ambitieux et sa lecture sera sans aucun doute une aventure dans laquelle vous allez jongler entre la théorie et la pratique d'une part et entre Python et R d'autre part. Nous espérons que vous éprouverez autant de plaisir à lire ce manuel que nous en avons eu à l'écrire.

## Conventions

Ce livre est conçu avec des conventions typographiques et structurelles soigneusement choisies pour en faciliter la compréhension.

- ✓ Le code est encapsulé dans des encadrés identifiés selon le langage de programmation utilisé, soit « Code R » ou « Code Python ».
- ✓ Richement illustré d'exemples, ce livre recommande une progression séquentielle pour une meilleure appréhension. Il est ainsi conseillé d'aborder les exemples d'un même chapitre dans l'ordre.
- ✓ Le cas échéant, les chapitres comprennent leurs références bibliographiques afin de faciliter l'accès.

## Auteurs (en ordre alphabétique)

**Élodie Boissières** est directrice adjointe au Service technopédagogique de l'Université TELUQ. Elle détient une

Maitrise ès Sciences en commerce électronique de HEC Montréal et les certifications SCRUM I et PRINCE2. Elle s'intéresse à l'impact des nouvelles technologies en éducation à distance et aux enjeux de l'IA générative sur les ressources pédagogiques.

**Robert Godin** a été professeur au Département d'informatique de l'Université du Québec à Montréal de 1983 à 2014. Il est maintenant professeur associé à ce même département. Il a reçu un Ph.D. de l'Université de Montréal en 1986. Il compte à son actif plus de quatre-vingt-dix publications dans des revues et conférences avec comité de lecture. Il a publié plusieurs livres destinés à l'enseignement.

**Daniel Lemire** est professeur d'informatique à l'Université du Québec (TÉLUQ). Il est l'auteur de plus de 75 publications arbitrées, incluant plus de 50 articles parus dans des revues internationales. Il est éditeur de la revue *Software : Practice and Experience*, fondée en 1971. Ses logiciels sont utilisés par de grandes sociétés comme Google et Facebook. Il a reçu le prix d'excellence de l'Université du Québec 2020 en recherche et création pour une réalisation en recherche (tous secteurs confondus) concernant ses travaux sur l'accélération du traitement des fichiers JSON.

**Habib Louafi** est professeur en informatique à l'Université TÉLUQ depuis 2023. Il a aussi été professeur adjoint en cybersécurité au New York Institute of Technology et professeur adjoint en science informatique à l'Université de Régina. Il a réalisé deux post-doctorats dans le domaine de la cybersécurité :

Ericsson Canada Security research Group – Montréal de 2016-2018 et Synchronmedia Lab École des technologies supérieures de Montréal. Détenteur depuis 2013 d'un Ph.D. en génie informatique de l'École des technologies supérieures (ETS - Montréal), il est également diplômé depuis 2006 de la maîtrise en science informatique de L'Université du Québec à Montréal (UQAM). Il détient un B.Sc. en science informatique de l'Université d'Es-Senia d'Algérie.

**Neila Mezghani** est professeure en informatique à l'Université TÉLUQ où elle détient la chaire de recherche du Canada en analyse de données biomédicales. Elle s'intéresse à l'analyse et la classification de données en génie biomédical et à l'élaboration d'outils basés sur des méthodes d'intelligence artificielle pour le développement de système d'aide à la décision. Elle est professeure associée à l'École de technologie supérieure (ÉTS) et l'Institut National de Recherche Scientifique (INRS-EMT). Elle est, aussi, chercheuse au centre de recherche du LICEF, chercheuse au centre de recherche du CHUM et membre du laboratoire de recherche en imagerie et orthopédie (LIO).

**Richmond Osei** est analyste informatique principal à la Saskatchewan Health Authority. Il est titulaire d'une maîtrise en informatique de l'université de Regina. Il a également obtenu plusieurs certifications dans les domaines de l'exploration de données.

**Shadi Shuraida** est professeur de gestion des technologies de l'information à l'Université TÉLUQ. Il a obtenu son doctorat à



HEC Montréal, et ses intérêts de recherche et d'enseignement portent sur l'implémentation et l'utilisation de l'analytique d'affaires et de l'intelligence artificielle organisationnelle. Dans ces domaines, il s'intéresse principalement au rôle de l'apprentissage et de la prise de décision des utilisateurs et des développeurs.

**Renée-Maria Schmitt** est consultante en éthique d'intelligence artificielle. Elle détient un Baccalauréat ès Science de l'Université de Münster en Allemagne. Elle s'intéresse aux enjeux éthiques tels que la confiance dans l'intelligence artificielle et les biais algorithmiques.

**Dragos Vieru** est professeur titulaire à l'Université TELUQ. Il détient un doctorat en Administration de HEC Montréal. Ses recherches portent sur l'ambidextrie organisationnelle et l'éthique des technologies numériques. Ses travaux ont été publiés dans des revues telles que Information Systems Management, Journal of Knowledge Management et International Journal of Information Management. Il a plus de 15 ans d'expérience professionnelle dans la gestion des technologies de l'information dans le secteur public des soins de santé.

# Chapitre 1 : Le langage R pour la science des données

Neila Mezghani

## 1.1 Introduction

R est à la fois un langage de programmation et un environnement dédié au traitement statistique, à l'analyse et la manipulation de données et à la création de graphiques. Il a été créé en 1993 par Ross Ihaka et Robert Gentleman à l'Université d'Auckland, en Nouvelle-Zélande. À l'origine, R était conçu comme un outil pédagogique destiné à aider les étudiants à acquérir des compétences en analyse statistique, notamment en ce qui concerne des méthodes telles que la régression linéaire et logistique. Depuis lors, R a connu une évolution significative et rapide.

Les bases de R s'inspirent de divers langages de programmation, notamment du langage S et du Scheme (Abelson, 1996). Le langage S, à son tour, incorpore des éléments issus d'autres langages plus anciens, tels que l'APL (Iverson, 1962) et le langage Lisp (McCarthy, 1958). Comme ses prédécesseurs, R est un langage interprété, ce qui signifie qu'il nécessite un interprète pour exécuter ses scripts, à la différence des langages compilés tels que C ou C++, qui doivent être traduits en code machine par un compilateur avant d'être exécutés par un ordinateur.

La particularité de R réside dans ses origines académiques. Contrairement à certains logiciels statistiques commerciaux conçus

principalement pour des applications industrielles à grande échelle, comme SAS, R a été développé dans un environnement universitaire. Son objectif initial était de soutenir l'enseignement des statistiques et la recherche scientifique. Cette orientation académique a contribué à la création de fonctionnalités précieuses, notamment des outils de visualisation interactive qui facilitent l'exploration visuelle des données. Cette étape préliminaire est essentielle avant de procéder à des analyses statistiques plus avancées.

## **1.2 Popularité et utilisation de R dans la science des données**

Au cours de la dernière décennie, R a connu une évolution remarquable, se distinguant par l'enrichissement de ses fonctionnalités, la diversification de ses domaines d'application, et l'élargissement de sa communauté d'utilisateurs. Plusieurs facteurs ont contribué à cette évolution en puissance, parmi lesquels la disponibilité de R sous une licence open source (GNU General Public License), qui a stimulé l'intérêt et la croissance rapide de sa communauté. Les membres actifs de cette communauté ont enrichi le langage grâce à de nombreuses extensions et packages. Un package R est un ensemble cohérent de fonctions, de jeux de données et de documentation permettant de compléter les fonctionnalités du système de base ou d'en ajouter de nouvelles. Dans ce contexte, le Comprehensive R Archive Network (CRAN) s'est établi comme le dépôt central de ces extensions, regroupant des milliers de packages conçus pour diverses analyses statistiques, visualisations graphiques, et autres applications.

L'élan de R montre son positionnement comme un outil de référence pour l'analyse et la visualisation statistiques dans divers domaines de recherche. Sa popularité augmente grâce à sa flexibilité, son caractère open source et son écosystème riche en packages. Aujourd'hui, au-delà du milieu académique, R trouve sa place dans de nombreuses industries pour répondre à une multitude de besoins dans des domaines allant de la bio-informatique à la finance, en passant par le marketing et la santé.

### 1.3 Installation et configuration de R et RStudio

L'installation et la configuration de R est un processus assez simple. Pour le faire, il suffit de se rendre sur le CRAN puis de télécharger la version adaptée à votre système d'exploitation :

- **Windows** : Sous « Download R for Windows », cliquez sur "base" puis téléchargez et exécutez le fichier exécutable.
- **macOS** : Sur le CRAN, sous « Download R for macOS », téléchargez et installez-le package correspondant à votre version de macOS.

RStudio est un environnement de développement intégré (IDE) convivial conçu spécifiquement pour l'analyse de données et le développement de package avec R. Il est disponible à l'identique pour les plateformes Windows, macOS et Linux. Pour l'installer, il suffit de suivre les deux étapes suivantes :

1. Allez sur le site de RStudio pour télécharger la version gratuite de RStudio Desktop adaptée à votre système d'exploitation.<sup>1</sup>

---

<sup>1</sup> <https://posit.co/products/open-source/rstudio/>

## 2. Suivez les instructions d'installation.

L'interface de RStudio est composée de plusieurs panneaux qui peuvent être reconfigurés en fonction des préférences de l'utilisateur. Ces panneaux comprennent la console, un navigateur de fichiers et de graphiques, l'espace de travail et l'historique des commandes. La Figure 1-1 illustre la fenêtre d'édition des fichiers sources (utilisée pour éditer les fichiers de script), la console (pour saisir des commandes directes), la fenêtre de l'environnement (qui affiche la liste des objets créés), et la fenêtre multi-onglets. Cette dernière comprend plusieurs onglets principaux dont Files qui sert à naviguer dans le système de fichiers de votre ordinateur, Plots pour afficher les graphiques, Packages affiche les paquets disponibles et chargés (lorsque cochés) et Help qui donne de l'aide sur les fonctions.

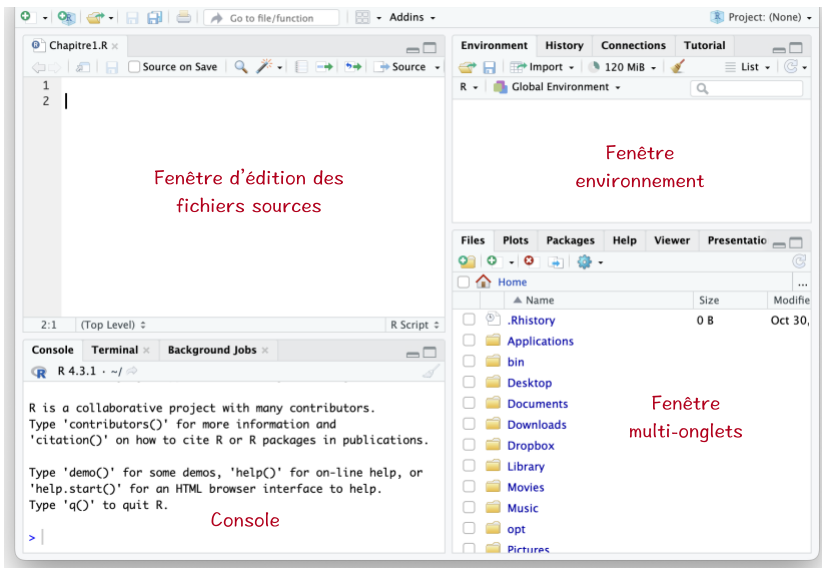


Figure 1-1: Fenêtre de RStudio sous macOS dans sa configuration par défaut. On y retrouve : la fenêtre d'édition des fichiers sources, la console, la fenêtre environnement et la fenêtre multi-onglets qui contient plusieurs

*onglets dont un pour les fichiers, un pour les graphiques et un pour les packages.*

Une fois l'installation de RStudio effectuée, vous pouvez utiliser les fonctions énumérées dans le script suivant pour connaître le répertoire de travail actuel (« working directory »), définir un répertoire de travail ou lister les fichiers et les répertoires dans le répertoire actuel. Ce script pourra être saisi dans la fenêtre d'édition des fichiers sources et exécuter par la suite ou bien il pourra être saisi et exécuté dans la console directement.

Code R
<pre>#connaître le répertoire de travail actuel getwd() #définir le répertoire de travail actuel setwd("/chemin/vers/votre/repertoire") #obtenir la liste des fichiers et répertoires list.files("/chemin/vers/votre/repertoire")</pre>

La documentation de RStudio se trouve entièrement en ligne. On y accède par le menu Help. L'onglet Help de la fenêtre multi-onglets de la Figure 1-1 offre également une interface pour accéder à l'aide de R et à celle de RStudio.

## 1.4 Bases du langage R

R est un langage interprété et non compilé ; cela signifie que les commandes tapées au clavier sont directement exécutées sans qu'il soit nécessaire de construire un programme complet. Les syntaxes de base de R sont simples et intuitives. Ainsi, pour les affectations, les symboles « <- » ou « = » sont utilisés. R prend en charge les opérations mathématiques de base, utilisant les signes classiques tels que "+" pour l'addition, "-" pour la soustraction, "\*" pour la multiplication et

`/"` pour la division. D'autres opérations plus avancées telles que l'exposant `"^"` et le modulo `"%%"` sont disponibles.

Les opérations logiques de base en R sont également intuitives. L'opérateur d'égalité est représenté par `"=="` tandis que l'opérateur de non-égalité est représenté par `"!="`. Ces opérateurs sont couramment utilisés pour créer des expressions conditionnelles dans les structures de contrôle telles que les instructions `if`, `else`, `while`, et les boucles `for`. Ils sont également utiles pour filtrer des données en fonction de certaines conditions dans le cadre de l'analyse de données en R.

Enfin, les fonctions statistiques de base portent des noms significatifs. Par exemple, `"mean()"` est utilisé pour calculer la moyenne, `"std()"` pour la variation standard, `"median()"` pour la médiane et `"sum()"` pour le calcul de la somme.

#### **1.4.1 Types de structures de données**

Les structures de données fondamentales en R comprennent les vecteurs, les matrices, les data frames, les ensembles de tableaux et les listes. La Figure 1-2 résume ces types de structures de données en incluant des exemples.



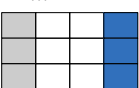
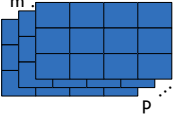
	Dimensions	Type (Mode)	Exemples
Vecteur	m ... 	Identique	<pre>vect eur_num &lt;- c(1, 2, 3, 4) vect eur_char &lt;- c("A l i c e", "B o b", "C h a r l e s", "D a v e") vect eur_log &lt;- c(TRUE, FALSE, TRUE, TRUE)</pre>
Matrice	m ... n ... 	Identique	<pre>ma_mat &lt;- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12), nrow = 3, ncol = 4, byrow = TRUE)</pre>
Data frame (Tableau de données)	m ... n ... 	Peut-être différent	<pre>mon_df &lt;- data.frame(   Identifiant = c(1, 2, 3, 4),   Nom = c("A l i c e", "B o b", "C h a r l e s", "D a v e"),   State = c(TRUE, FALSE, TRUE, TRUE)</pre>
Array (Ensemble de matrices)	m ... n ... 	Identique	<pre>mon_array &lt;- array(data = 1:48, dim = c(3, 4, 3))</pre>
Liste	$\left( \begin{array}{c} \text{Vecteur} \\ \text{Matrice} \\ \text{Data frame} \\ \text{Array} \end{array} \right)$	Peut-être différent	<pre>ma_liste = list(ma_mat, mon_df, mon_array)</pre>

Figure 1-2: Types de structure de données

## Les vecteurs

Le vecteur est l'une des structures de données les plus élémentaires et essentielles dans le langage de programmation R. Chaque élément d'un vecteur est accessible via un indice entre crochets. Dans un vecteur, tous les éléments sont du même type, également appelé mode. Les modes couramment rencontrés incluent : numérique (numeric), entier (integer), caractère (character), facteur (factor) et logique (logical). Lorsque nous combinons des éléments de différents types dans un vecteur en utilisant la fonction `c()`, R convertira automatiquement tous les éléments en un type commun en suivant une hiérarchie de types allant du moins flexible au plus flexible, c'est-à-dire le type logique, puis numérique et enfin caractère.



#### Code R

```
# Création d'un vecteur de valeurs numériques
vecteur_num <- c(1, 2, 3, 4)
# Création d'un vecteur de caractères
vecteur_char <- c("A", "B", "C", "D")
# Création d'un vecteur de valeurs logiques
vecteur_log <- c(TRUE, FALSE, TRUE, TRUE)
```

Pour confirmer ou tester le type des éléments d'un vecteur, vous pouvez utiliser la fonction `class` ou `typeof`. Vous pouvez également vous assurer s'il s'agit bien d'un vecteur en utilisant la fonction `is.vector`, qui retourne la valeur `True` si c'est bien un vecteur.

#### Code R

```
# tester le type des éléments d'un vecteur
class(vecteur_char)
# tester le type des éléments d'un vecteur
typeof(vecteur_char)
# tester s'il s'agit d'un vecteur
is.vector(vecteur_char)
```

## Les matrices

Les matrices sont des structures bidimensionnelles qui contiennent des éléments du même type. Chaque élément dans une matrice est identifié par deux indices, l'un pour les lignes et l'autre pour les colonnes. Pour créer une matrice en R, on utilise généralement la fonction `matrix()`. Cette fonction peut prendre plusieurs arguments, notamment le nombre de lignes (`nrow`) et le nombre de colonnes (`ncol`). On peut également spécifier si la matrice doit être remplie par ligne (`byrow = TRUE`) ou par colonne (`byrow = FALSE`, qui est le comportement par défaut). Les matrices sont particulièrement utiles pour effectuer des calculs mathématiques complexes, car elles peuvent être utilisées

avec des opérations arithmétiques standard telles que l'addition et la multiplication.

### Les array

En R, un array est une structure de données qui peut stocker des données dans plus de deux dimensions, contrairement aux matrices qui sont limitées à deux dimensions (lignes et colonnes). Les éléments d'un array doivent tous être du même type, tout comme les matrices. Les arrays sont utiles lorsque vous travaillez avec des données multidimensionnelles. Par exemple, si vous avez des données sur les ventes de différents produits, réparties dans plusieurs magasins et à travers plusieurs mois, un array peut être utilisé pour stocker ces données en trois dimensions.

### Les listes

Une liste en R est une structure de données qui peut contenir une combinaison de différents types de données, tels que des nombres, des chaînes, des vecteurs, des matrices, des data frames, et même d'autres listes. Cette structure est donc très flexible et utile dans différents contextes en sciences des données.

### Data frames

Les data frames sont les objets les plus courants dans le traitement de données habituel. Il s'agit de tableaux dont les lignes correspondent à des observations et les colonnes à des variables. Un data frame est une structure de données très utilisée dans R pour stocker des données de manière tabulaire, c'est-à-dire en lignes et en colonnes. Chaque colonne d'un data frame peut contenir des types de données différents

(numérique, caractère, etc.), mais chaque colonne doit avoir le même type de données. Les data frames sont particulièrement utiles pour les statistiques et les analyses de données car ils permettent de manipuler et d'analyser des ensembles de données avec des fonctions complexes, de réaliser des opérations par groupe, de filtrer des données, et de les résumer de différentes manières.

### 1.4.2 Les boucles

Les tests et les boucles répétitives sont des éléments essentiels en analyse de données. En R, les opérations vectorielles sont l'une des caractéristiques les plus puissantes, car elles permettent d'effectuer des calculs sur les différentes structures de données sans avoir à utiliser de boucles explicites.

**Exemple 1.1 :** Dans cet exemple, vous trouverez deux manières pour faire la somme numérique de deux vecteurs en utilisant une boucle for ou bien en utilisant une opération vectorielle.

#### Code R

```
x <- c(1, 2, 3, 4)
y <- c(5, 6, 7, 8)
n <- length(x) # Longueur des vecteurs

# Initialisation du vecteur résultat
result <- numeric(n)

# Boucle for pour l'addition élément par élément
for (i in 1:n) {
  result[i] <- x[i] + y[i]
}

# Addition élément par élément des vecteurs x et y
result <- x + y
```

**Exemple 1.2 :** Maintenant, vous trouverez deux versions de script pour rechercher la lettre « w » dans la liste `ma_liste` contenant des noms d'animaux, tout en comptant le nombre d'occurrences. La première version utilise une boucle `for` classique pour parcourir la liste et effectuer le comptage. La seconde version, est plus concise, se base sur des opérations vectorielles. Elle utilise avec `strsplit()` pour décomposer les chaînes de caractères, suivie de l'utilisation et `sum()` pour effectuer un comptage agrégé. Les deux versions produisent le même résultat.

Code R

```
# Création de la liste
ma_liste <- c("chat", "chien", "wombat", "poisson",
"wallaby")

# Version 1 : Utilisation d'une boucle for pour
rechercher la lettre "w" et compter les occurrences
nb_occurrences <- 0

for (mot in ma_liste) {
  lettres <- unlist(strsplit(mot, ""))
  for (lettre in lettres) {
    if (lettre == "w") {
      nb_occurrences <- nb_occurrences + 1
    }
  }
}

print(paste("En utilisant la boucle, La for lettre 'w'
apparaît", nb_occurrences, "fois dans la liste."))

# Version 2 : Utilisation d'une opération vectorielle pour
rechercher la lettre "w" et compter les occurrences
lettres <- unlist(strsplit(ma_liste, ""))
nb_occurrences <- sum(lettres == "w")

print(paste("En utilisant une opération vectorielle, La
for lettre 'w' apparaît", nb_occurrences, "fois dans la
liste."))
```

Les deux exemples précédents, illustrent différentes méthodes pour accomplir la même tâche. Ils mettent en évidence la facilité qu'apportent les opérations de R pour la manipulation de données.

## 1.5 Traitement des données avec R

Charger et sauvegarder des données sont des tâches fréquentes pour un scientifique des données. R propose plusieurs fonctions et méthodes pour importer et exporter des données depuis et vers différents formats de fichiers (fichier texte, fichier Excel, fichier JSON, etc). Cet aspect est essentiel car il vous permet d'acquérir des données à partir de sources externes, de les manipuler et de sauvegarder les résultats dans le format souhaité. Les principales fonctions de chargement (ou lecture) de données sont énumérées dans ce qui suit.

**Lecture et sauvegarde de fichiers textes :** plusieurs types de fichiers texte sont couramment utilisés en sciences des données, notamment les fichiers CSV (Comma-Separated Values), les fichiers TSV (Tab-Separated Values) et les fichiers TXT. R permet de lire ces fichiers à l'aide de fonctions telles que `read.csv()`, `read.table()`, ou `read.delim()`. De la même manière que pour la lecture, vous pouvez utiliser l'équivalent de ces fonctions pour sauvegarder des données dans des fichiers CSV, TSV ou d'autres formats de fichiers texte à l'aide de fonctions comme `write.csv()` ou `write.table()`. Le script suivant donne des exemples de lecture de fichiers.

Code R

```
# Lecture d'un fichier CSV
data <- read.csv("data.csv", header = TRUE, sep = ",")
# Lecture d'un fichier TSV avec tabulateur comme séparateur
data <- read.table("data.tsv", header = TRUE, sep = "\t")
```

```
# Lecture d'un fichier texte délimité par des tabulations
data <- read.delim("data.txt", header = TRUE)
```

L'attribut « `header = TRUE` » permet d'importer l'en-tête du fichier, ce qui permet de manipuler les colonnes avec leurs noms plutôt qu'avec des noms de variables générés automatiquement par R.

**Lecture et sauvegarde de fichiers Excel** : Les bibliothèques telles que `readxl` ou `openxlsx` offrent la possibilité de lire des fichiers Excel au format `.xls` ou `.xlsx` en utilisant des fonctions comme `read_excel()`. Il est important de noter que certaines de ces fonctions dépendent de packages spécifiques, donc il est essentiel de s'assurer que ces packages sont installés (si ce n'est pas déjà le cas) et de les charger dans votre environnement R avant de pouvoir utiliser ces fonctionnalités.

Code R

```
# Lecture de fichiers Excel
install.packages("readxl")
library(readxl)
data <- read_excel("data.xlsx", sheet = "nom_feuille")
```



La diversité de tous ces outils d'importation nous rappelle la nécessité de toujours vérifier que l'objet R dans lequel nous avons importé des données contient les bonnes données et qu'elles sont dans le bon format. Ainsi, les vérifications suivantes restent importantes : « Est-ce que le nombre de variables (colonnes) dans le data frame est le même que le nombre de variables dans le fichier ? » ou encore « Est-ce que le nombre d'observations (lignes) dans le data frame est le même que le nombre d'observations dans le fichier ? » De plus, il est essentiel de s'assurer que, pour chacune des

variables de l'ensemble de données, le type de données dans le data frame correspond au type de données dans le fichier.

### **1.5.1 Visualisation et transformation des données avec R**

À l'ère des données complexes, caractérisées par leur hétérogénéité, leur dimensionnalité et leurs variations, la capacité à extraire des informations de manière visuellement accessible est devenue cruciale. Le langage de programmation R, reconnu pour sa flexibilité et sa puissance, s'impose comme un outil incontournable dans le domaine de la visualisation de données.

R propose une bibliothèque riche et diversifiée de packages qui permettent aux scientifiques des données d'exprimer leurs découvertes. Parmi eux, ggplot2 se distingue comme l'un des packages les plus populaires de R pour la visualisation. Créé par Hadley Wickham et inspiré par le livre "The Grammar of Graphics" de Leland Wilkinson (Wilkinson, 2005), ggplot2 repose sur une grammaire de la visualisation des données. Cette approche permet aux utilisateurs de construire des graphiques en définissant des composants tels que les échelles, les axes, les données, et les couches de visualisation. La force de ggplot2 réside dans sa capacité à superposer plusieurs couches pour créer des visualisations complexes et élégantes.

Avec R, la manipulation et la transformation des données en vue de la visualisation sont également facilitées. Les packages tels que dplyr et tidyr simplifient la manipulation de données et la préparation préalable à la visualisation. Ils offrent des fonctionnalités pour filtrer, sélectionner, résumer et réorganiser les données de manière intuitive.

La visualisation de données avec R est une ressource précieuse en science des données, dont l'impact ne se limite pas à l'aspect visuel, mais qui contribue également à de nombreuses autres tâches d'un scientifique des données, telles que la modélisation et les analyses avancées.

### **1.5.2 Modélisation statistique avec R**

La modélisation statistique avec R bénéficie de sa capacité à gérer différents types de données et de modèles statistiques, ce qui en fait un outil puissant. En effet, R propose une variété d'outils pour la modélisation statistique, notamment des modèles linéaires et non linéaires, des modèles mixtes, des modèles de survie et des modèles bayésiens. Des packages tels que `lm` pour les modèles linéaires, `glm` pour les modèles linéaires généralisés, et `lme4` pour les modèles à effets mixtes illustrent la diversité et richesse des fonctionnalités de R. Cette diversité de packages et de fonctions permet aux utilisateurs de construire, d'évaluer et d'interpréter efficacement des modèles statistiques complexes.

De plus, comme mentionné précédemment, R offre une facilité de visualisation des données, ce qui facilite également la visualisation des résultats des modèles. Avec des packages comme `ggplot2`, les utilisateurs peuvent créer des graphiques clairs et informatifs pour l'analyse exploratoire des données, ainsi que pour la présentation des résultats des modèles.

### **1.5.3 R pour l'apprentissage machine**

La popularité de R a également touché le domaine de l'apprentissage machine, pour plusieurs raisons. Tout d'abord, R propose une vaste



collection de packages tels que caret, randomForest, xgboost et keras, qui permettent aux utilisateurs d'appliquer une grande variété de techniques d'apprentissage automatique. Ces techniques vont des régressions linéaires et logistiques à des méthodes plus complexes comme les forêts aléatoires, les machines à vecteurs de support et les réseaux de neurones profonds. Ces outils sont à la fois puissants et relativement simples à utiliser, ce qui rend R accessible même aux utilisateurs qui ne sont pas des experts en programmation.

Une fois de plus, la capacité de R à exceller dans l'analyse et la visualisation des données renforce son utilisation en apprentissage machine. Par exemple, les packages comme ggplot2 permettent de créer des visualisations de données complexes et informatives. Cette capacité à explorer facilement les données, à identifier des tendances et des anomalies, est cruciale pour la préparation des données et l'application des modèles d'apprentissage machine appropriés.

Enfin, R s'intègre bien avec d'autres outils et plateformes, ce qui facilite l'importation de données à partir de diverses sources et leur exportation vers des systèmes différents pour une utilisation ultérieure. Cette flexibilité fait de R un outil précieux pour l'apprentissage machine, capable de s'adapter à une multitude de scénarios et d'exigences de traitement des données. Le chapitre 6 – Apprentissage machine avec R et python, présente plusieurs techniques d'apprentissage dans les deux langages. L'objectif étant de démontrer les ressemblances et la facilité de navigation entre les deux.

## 1.6 Conseils et bonnes pratiques

Grâce à son écosystème riche en bibliothèque et en packages, R s'impose comme un outil incontournable pour la recherche scientifique et l'analyse de données. En tant que scientifique des données, il est fort probable que votre code soit partagé ou réexaminé ultérieurement. Il est donc nécessaire d'assurer une clarté optimale du code en adoptant un style cohérent, en choisissant des noms de variables significatifs et en commentant abondamment. Plusieurs guides de style visent à maximiser la lisibilité et la cohérence du code, parmi lesquels le guide de style du tidyverse se distingue par ses conventions et recommandations spécifiques à R. Ce dernier a été largement adopté par la communauté R et a influencé la façon dont de nombreux utilisateurs rédigent et organisent leur code.

### 1.6.1 Optimisation et performance

Face à des projets d'analyse sollicitant d'importants volumes de données, il est essentiel d'exploiter la puissance de la vectorisation de R afin d'améliorer les performances du code, en réduisant autant que possible l'utilisation de boucles comme décrit précédemment. Le script suivant montre des exemples d'utilisation des fonctions `rowSums()`, `colSums()`, `rowMeans()` et `colMeans()` pour effectuer des calculs sur des lignes et des colonnes d'un data frame.

#### Code R

```
# Créer un data frame exemple
data <- data.frame(
  A = c(1, 2, 3, 4),
  B = c(5, 6, 7, 8),
  C = c(9, 10, 11, 12)
)

# Calculer la somme des valeurs par ligne
row_sums <- rowSums(data)

# Calculer la somme des valeurs par colonne
col_sums <- colSums(data)

# Calculer la moyenne des valeurs par ligne
row_means <- rowMeans(data)

# Calculer la moyenne des valeurs par colonne
col_means <- colMeans(data)
```

L'optimisation du code en R peut se faire également en utilisant des packages dédiés à des opérations spécifiques. Par exemple, le package `data.table` est connu pour sa vitesse lors de la manipulation de grands ensembles de données. Par ailleurs, vous pouvez utiliser la fonction `system.time()` pour mesurer le temps d'exécution de votre code. Cela vous permettra d'identifier les parties du code qui prennent le plus de temps.

### 1.6.2 Gestion des erreurs et débogage

Les données peuvent parfois être imprévisibles. Il est recommandé d'implémenter des contrôles et des validations pour garantir la qualité des données en entrée. En cas de problèmes, utilisez `traceback()` pour localiser les erreurs et `browser()` pour le débogage interactif. Les assertions et les tests unitaires peuvent également prévenir de nombreux problèmes.

### 1.6.3 Ressources pour continuer à apprendre

Tout comme le domaine de la science des données, l'écosystème R est en perpétuelle évolution, et cette dynamique est en grande partie alimentée par une communauté active et variée. Cette communauté met à disposition une multitude de ressources pour l'apprentissage et l'amélioration continue. Parmi ces ressources, on y retrouve le dépôt officiel des packages R, des forums de discussion, des sites web spécialisés et des blogs dédiés.

D'autres ressources sont également disponibles, notamment la conférence useR!, un événement annuel organisé par et pour la communauté des utilisateurs du langage de programmation R. De même, l'organisation mondiale R-Ladies promeut la diversité de genre dans la communauté R à travers le monde, en encourageant et inspirant les personnes sous-représentées.

## 1.7 Conclusion

Le langage R a été initialement conçu pour la statistique et la recherche scientifique, mais grâce à la richesse de ses packages dédiés à la statistique, à la bio-informatique, à l'épidémiologie et à d'autres domaines scientifiques, il reste pertinent pour de nombreuses disciplines. De nombreuses universités et institutions continuent d'enseigner R dans leurs programmes de statistiques, d'épidémiologie et de sciences sociales, assurant ainsi un afflux continu de nouveaux utilisateurs.

R bénéficie d'une communauté dynamique et active. Les initiatives telles que R-Ladies ou les conférences useR! favorisent l'innovation et l'éducation autour du langage, garantissant ainsi sa pérennité. De

plus, des outils tels que RStudio, RShiny et RMarkdown ont permis d'intégrer R dans des environnements d'entreprise, favorisant son adoption pour la création de rapports, les tableaux de bord interactifs et d'autres applications professionnelles.

Finalement, bien que Python ait gagné en popularité dans le domaine de la science des données grâce à des bibliothèques telles que pandas, scikit-learn et TensorFlow, R conserve ses avantages, notamment en matière d'analyse statistique approfondie et de création de rapports. Le rapprochement de R avec d'autres langages de programmation, tels que Python, à travers des interfaces comme reticulate ou des plateformes comme Jupyter, témoigne de sa volonté d'intégration et d'adaptation aux évolutions du domaine de la science des données.

En résumé, bien que les technologies évoluent constamment, R maintiendra une place solide dans l'écosystème de la science des données pour les années à venir, grâce à sa communauté active et son évolution constante pour répondre aux besoins des utilisateurs.

## 1.8 Bibliographie

Abelson, H., & Sussman, G. J. (1996). Structure and interpretation of computer programs (p. 688). The MIT Press.

McCarthy, J. (1960). Recursive functions of symbolic expressions and their computation by machine, part I. Communications of the ACM, 3(4), 184-195.

McCarthy, J. "Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I." Communications of the ACM, Vol. 3, No. 4, 1958, pp. 184-195.

Wilkinson, L. (2012). The grammar of graphics (pp. 375-414).  
Springer Berlin Heidelberg

# Chapitre 2 : Introduction au langage Python

Daniel Lemire et Robert Godin

## 2.1 Introduction

Le langage de programmation Python est l'un des plus populaires en science des données. C'est aussi un langage de programmation versatile qui est notamment utilisé pour développer des sites web. Par exemple, YouTube fut en partie développé en Python. La popularité du Python tient au fait que c'est un langage relativement facile à lire. Il est possible d'écrire du code concis en Python pour résoudre des problèmes complexes. Si on le compare à d'autres langages comme C#, Java, C ou C++, le Python est généralement moins performant : un programme écrit en Python peut prendre plus de temps pour résoudre un problème donné. Cet inconvénient est moins important qu'il n'y paraît puisqu'il est facile d'appeler du code écrit en C ou C++ à partir du Python. Dans la pratique, il est donc possible de programmer en Python tout en bénéficiant de la performance d'autres langages. C'est notamment ainsi qu'on utilise le Python en science des données.

Guido van Rossum est l'inventeur du Python. Développé à partir de 1989, le Python a gagné en popularité au tournant du siècle avec la publication de la version 2.0 du langage. En 2008, la version 3 du langage Python fut rendue disponible et c'est la version que nous allons présenter. Les deux versions (2 et 3) sont incompatibles, mais assez similaires.

## 2.2 Installation

Vous pouvez exécuter des commandes Python directement dans votre navigateur : par ex., sur <https://www.python.org/shell/>. Néanmoins, il est parfois plus pratique de pouvoir exécuter Python directement sur votre ordinateur sans passer par un site web.

Il est possible que votre ordinateur dispose déjà d'un système capable d'exécuter des programmes Python. Néanmoins, il est important de vérifier que vous disposez bien de la version 3. Habituellement, Python annonce sa version lors du lancement de l'interpréteur. Il y a plusieurs versions de Python 3 : 3.1, ..., 3.11, etc. Le second nombre, après le premier point, est le numéro de version mineure (1, ..., 11, etc.) et il est souvent préférable de choisir une version mineure ayant un numéro plus élevé (par ex., 11 ou plutôt que 10). Nous vous recommandons d'installer la version 3.4 ou mieux.

L'installation de l'interpréteur Python est possible sur la plupart des ordinateurs. La stratégie à suivre diffère cependant selon le système d'exploitation.

**Windows.** Installez Python à partir de [python.org](https://python.org). Utilisez le bouton Télécharger Python (*Download*) qui apparaît sur la page web pour télécharger la dernière version. Vous pourrez ensuite exécuter le fichier d'installation. Si on vous en donne l'option, assurez-vous de demander que l'interpréteur soit ajouté à la variable PATH de votre système d'exploitation. Si vous ne disposez pas d'un accès administrateur, une autre option pour installer Python sur Windows consiste à utiliser le Microsoft Store :

<https://apps.microsoft.com/store/search/python>.



**macOS.** L'installation du système de Python sur macOS est similaire à Windows : il suffit de vous rendre sur [python.org](https://python.org). Cependant il est aussi possible d'installer Python en passant par brew (<https://brew.sh>). Une fois l'installation de Python complétée, nous vous invitons à installer l'éditeur *Visual Studio Code* de Microsoft. Il suffit de rendre sur le site <https://code.visualstudio.com/> et d'appuyer sur le bouton Télécharger (*Download*).

Une fois l'installation de Visual Studio Code effectuée, lancez l'application. Vous devriez voir une fenêtre qui peut ressembler à la Figure 2-1. L'environnement de Visual Studio Code comprend plusieurs éléments. À gauche, sur la figure, vous pouvez voir des icônes correspondant à la recherche (loupe) ou à l'ajout d'extensions (carrés).

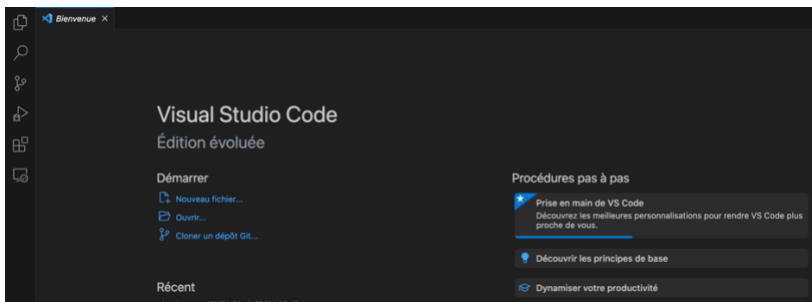


Figure 2-1. Fenêtre de Visual Studio Code au lancement (exemple)

Rendez-vous dans le menu de Visual Studio Code et choisissez « Nouveau Terminal ». Vous devriez alors voir apparaître une sous-fenêtre contenant une console vous permettant d'invoquer l'interpréteur Python. Une fois la console ouverte, tapez « python » et la touche de retour à la ligne (ou <fin de ligne>). Le résultat devrait ressembler à la Figure 2-2 : observez que la console (« terminal »)

peut apparaître à différents endroits, mais qu'elle apparaît souvent au bas de la fenêtre de l'éditeur. Au lancement de l'interpréteur, vous pourrez consulter la version de Python qui est disponible, assurez-vous qu'il s'agit bien de la version 3 (par ex., 3.11). Si la commande n'est pas reconnue, mais que vous avez bien installé Python sous Windows, nous vous suggérons de consulter le site *Utiliser Python sous Windows* <https://docs.python.org/fr/3/using/windows.html>. Consultez un technicien au besoin.

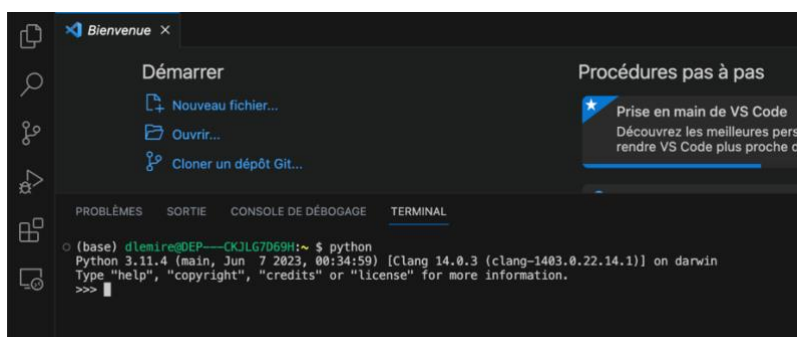


Figure 2-2. Fenêtre de Visual Studio Code comprenant une console (exemple)

Maintenant que vous avez accès à l'interpréteur Python, vous pouvez voir une invite de commande. L'invite de commande « >>> » indique que le console Python est en attente d'un énoncé du langage Python. Essayez de taper l'expression suivante :

```
Code Python
print('Hello, World!')
```

Le code Python peut être entré directement suivi de <fin de ligne>. Vous devriez voir une ligne apparaître avec le texte `Hello, World!` et l'apparition d'une nouvelle invite de commande.

Dans cet exemple, « print » est un exemple de fonction alors que « Hello World » est un exemple de chaîne de caractères. Une fonction accepte généralement un ou plusieurs paramètres.

Si vous obtenez un message d'erreur, assurez-vous d'avoir bien tapé l'expression correctement. Si vous faites du copier-coller ou que vous utilisez un traitement de texte, il est fréquent que les caractères d'un code informatique soient modifiés ce qui rend le code incorrect. Le plus souvent le code informatique doit se limiter aux caractères ASCII (apostrophe droite, etc.).

Bien qu'il soit pratique de pouvoir lancer l'interpréteur Python, il est souvent plus pratique d'enregistrer vos programmes Python au sein de fichiers dédiés et d'exécuter ces fichiers (parfois appelés *scripts*). Pour rendre l'opération aisée, installez l'extension Python (de Microsoft) au sein de Visual Studio Code. Choisissez dans le menu d'icônes de gauche l'icône « Extension » (composé généralement de 4 carrés). Une boîte de saisie devrait apparaître, tapez « Python ». Dans les premiers résultats, vous devriez trouver l'extension Python publiée par Microsoft. Après l'avoir sélectionnée, tapez « Installer » (*Install*).

Nous allons maintenant exécuter un fichier. Auparavant, il faut créer le fichier. Dans le menu de Visual Studio Code, choisissez « Nouveau fichier texte » ou l'équivalent. Une fenêtre vous permettant de saisir du texte devrait apparaître. Tapez l'expression Hello World (« print('Hello, World!') ») puis choisissez « Enregistrer » dans le menu de Visual Studio Code. Choisissez un nom pour votre fichier se terminant en « .py » comme « monprogramme.py ». Voir Figure 2-3.

Pour exécuter le nouveau programme, tapez F1 alors que vous êtes dans Visual Studio Code. Sur certains claviers, il peut être nécessaire de taper sur une touche fonction ou fn au même moment. Une boîte de saisie devrait alors apparaître, tapez « Python : Exécuter le fichier » (voir Figure 2-4). Sélectionnez avec votre souris la commande suggérée correspondant à votre saisie. Vous devriez alors voir votre programme s'exécuter dans la console.

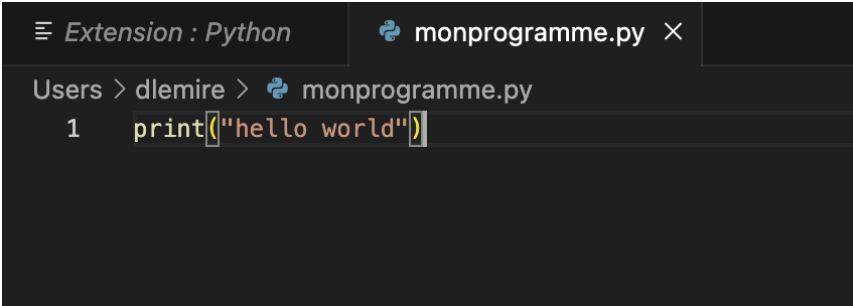


Figure 2-3. Fenêtre de Visual Studio Code comprenant un éditeur

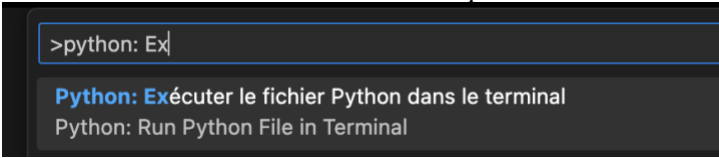


Figure 2-4. Fenêtre de Visual Studio Code comprenant l'exécution d'une commande

### 2.3 Les fondements du langage Python

La console Python peut être employée comme une calculatrice qui évalue une expression mathématique. Par exemple, si l'utilisateur entre l'expression « 3+2 » suivie de *<fin de ligne>*, le système répond en affichant le résultat de l'addition des deux entiers qui est 5.

Code Python
>>> 3+2
5

L'affichage du résultat est automatique lorsqu'une expression est saisie. La soustraction est exprimée par - :

Code Python
<pre>&gt;&gt;&gt; 5-4</pre>
<pre>1</pre>

La multiplication par \* :

Code Python
<pre>&gt;&gt;&gt; 3*4</pre>
<pre>12</pre>

La division par / :

Code Python
<pre>&gt;&gt;&gt; 20/5</pre>
<pre>4.0</pre>

L'opérateur // calcule la division entière dont le résultat est un entier et le % calcule le reste (modulo) :

Code Python
<pre>&gt;&gt;&gt; 11//2</pre>
<pre>5</pre>
<pre>&gt;&gt;&gt; 11%2</pre>
<pre>1</pre>

On peut donc vérifier qu'une valeur est divisible par une autre en vérifiant que le reste est zéro :  $12363 \% 3$  est zéro et on sait donc que 12363 est divisible par 3.

Dans les expressions plus complexes ( $4 * 3 + 1$ ), Python utilise la priorité des opérations (les multiplications et divisions, plus les additions et les soustractions, etc.). On peut aussi utiliser les parenthèses :  $4 * (3 + 1)$ . Quand il y a des parenthèses, l'expression au sein de la parenthèse est évaluée en premier.

On peut ajouter un commentaire au sein d’une expression en Python : le caractère dièse (#) en dehors d’une chaîne de caractères début une section commentaire qui se termine avec la fin de la ligne. Un commentaire est destiné au lecteur (humain) du code et il n’est pas interprété comme étant du code Python. Les commentaires peuvent être utiles si vous souhaitez vous assurer que le lecteur comprenne bien votre intention.

Code Python
>>> # ceci est un commentaire

## 2.4 Notion de variable

En informatique, nous stockons virtuellement les données au sein d’entités appelées *variables*. Une variable a généralement un nom et une valeur associée. En Python, il est possible d’assigner une valeur à une variable avec l’opérateur d’affectation « = » et d’utiliser le nom de variable pour désigner cette valeur par la suite.

Code Python
>>> a=2+3
>>> a
5
>>> print(a)
5
>>> 3*a
15
>>> a=a+1
>>> a
8

L’opérateur de mise à la puissance est \*\* :

Code Python
-------------

```
>>> 3**3
27
```

On peut combiner l'opérateur d'affectation avec les opérateurs arithmétiques. Par exemple, l'expression `a = a + 1` peut être remplacée par `a += 1`. Voici quelques exemples :

Code Python

```
>>> a+=1
>>> a
9
>>> a+=2
>>> a
11
>>> a-=10
>>> a
1
>>> a*=100
>>> a
100
```

La fonction [`type\(\)`](#) appliquée à une variable retourne le type de l'objet auquel la variable fait référence.

Code Python

```
>>> b = 5
>>> type(b)
<class 'int'>
>>> b = 3.4
>>> type(b)
<class 'float'>
```

Cet exemple illustre le fait qu'une variable Python peut faire référence à des objets de types différents au cours de l'exécution d'un programme.

## 2.5 Les chaînes de caractères

En Python, une séquence de caractères entre apostrophes (') ou guillemets (") est interprétée comme une chaîne de caractères, et le type d'une chaîne de caractères est appelé *str*. Les apostrophes triples (''') ou les guillemets triples (""") doivent être utilisés si la chaîne de caractères chevauche plusieurs lignes.

Code Python

```
>>> """longue
... chaîne
...
'longue\nchaîne\n'
```

On peut convertir une chaîne de caractère en nombre. Par exemple, on peut partir d'une chaîne de caractère et la convertir en nombre à virgule flottante :

Code Python

```
>>> s="0.4"
>>> type(s)
<class 'str'>
>>> s=float(s)
>>> s
0.40
```

On peut aussi convertir un nombre en chaîne de caractères :

Code Python

```
>>> s = str(3.1416)
>>> type(s)
```



```
<class 'str'>

>>> s

'3.1416'
```

On peut faire de l'arithmétique avec les chaînes de caractères :

Code Python

```
>>> s = "*"

>>> s * 5

*****

>>> s + s

***

>>> s + "-"

*_-
```

La longueur (*length*) d'une chaîne de caractères est obtenue avec la fonction *len* (par ex. *len(s)*). Par exemple, l'expression *len('12345')* a la valeur 5.

## 2.6 Structures de contrôle

Par défaut, Python interprète les expressions une à la suite de l'autre. Programmer en utilisant une longue liste d'expressions peut devenir pénible et c'est pourquoi on utilise des structures de contrôle qui permettent de revenir en arrière ou d'omettre une partie de l'exécution. En dehors de l'exécution séquentielle, il y a deux stratégies fondamentales de contrôle : la *répétition* et l'*alternative* (*choix, décision*).

### 2.6.1 La répétition avec l'énoncé *while*

Imaginons que l'on veuille afficher les entiers de 1 à 5. Le programme suivant peut produire ce résultat :

Code Python

```
# Afficher les entiers de 1 à 5
```

```
print(1)
print(2)
print(3)
print(4)
print(5)
```

S'il fallait afficher les entiers de 1 à 1 000 000 de cette manière, le programme serait long. Pour éviter de répéter les énoncés dans le programme, on peut employer une répétition (aussi appelée *boucle* ou *itération*). Pour l'essentiel, le programme répète le contenu de l'expression `while` tant qu'une condition est vraie :

```
Code Python
# Exemple d'une boucle while avec un compteur

compteur = 1
while compteur <=5:
    print(compteur)
    compteur = compteur + 1
```

Si vous tapez ce code dans Visual Studio Code, que vous enregistrez le fichier (avec l'extension « .py ») et que vous faites F1, *Python : Exécutez le fichier Python dans le Terminal*, vous devriez voir les nombres de 1 à 5 apparaître. Observez comment le commentaire est ignoré par l'interpréteur Python.

L'expression entre parenthèses (ici « `compteur <=5` ») doit être une *expression booléenne*, aussi appelée *condition*, dont la valeur est de type [bool](#) (*vrai* (*True*) ou *faux* (*False*)). Si cette condition est respectée (i.e. la valeur retournée par l'*expression* est *True*), le bloc après le *while* est répété en boucle jusqu'à ce que la condition ne soit plus respectée (i.e. la valeur retournée par l'*expression* est *False*). Python comporte plusieurs opérateurs permettant de faire des comparaison :

« == » égalité, « != » différent, « < » plus petit que, « <= » plus petit ou égal, « > » plus grand que, « >= » plus grand ou égal. Il est donc important que le code à l'intérieur de la boucle rende la condition *False* après un certain nombre d'itérations, sinon, la boucle continue de se répéter sans fin (*boucle infinie*).

Un *bloc de code* Python est une séquence d'un ou plusieurs énoncés indentés de manière identique par rapport à une entête qui se termine par « : ». Un entête peut-être un *while* ou d'autres types d'énoncés. La convention suggérée est d'indenter systématiquement avec 4 espaces mais ce n'est pas obligatoire. Il est aussi permis d'indenter avec les tabulations ou d'utiliser 2, 3 ou 10 espaces. Cependant, il ne faut pas mélanger les conventions.

Il est possible d'imbriquer un autre *while* dans le bloc et ceci à volonté en fonction des besoins. Par exemple, le programme suivant va afficher progressivement davantage d'astérisques sur chaque ligne :

#### Code Python

```
compteur = 1

while compteur <=5:
    s = ""
    while len(s) < compteur:
        s = s + "*"
    print(s)
    compteur = compteur + 1
```

L'utilisation d'une répétition avec compteur est très fréquente. La boucle *for* simplifie l'écriture de telles boucles. Le programme suivant affiche les entiers de 1 à 5.

#### Code Python

```
"""
Exemple d'une boucle for avec la fonction range()
"""

for compteur in range(1,6):
    print(compteur)
```

Après le mot-clé *for*, il faut spécifier une variable, ici *compteur*, qui prendra successivement les valeurs d'une séquence spécifiée après le mot-clé *in*. L'appel à [range](#)(1,6) génère la séquence 1, 2, 3, 4, 5. Le premier argument est la valeur initiale. La valeur du deuxième argument est la borne supérieure mais elle n'est pas incluse. L'incrément est 1 par défaut.

L'appel à [range](#)(5) génère la séquence 0,1,2,3,4. Par défaut, la valeur initiale est 0. Et [range](#)(4,16,3) génère 4,7,10,13 (le 16 est exclus).

L'incrément peut être négatif comme dans [range](#)(5,-10,-4) qui génère

5,1,-3,-7. Comme pour un incrément positif, la dernière valeur est exclue : ainsi `range(8,4,-2)` génère la séquence 8,6. Le 4 est exclu.

### 2.6.2 L'alternative (choix, décision) avec `if`

L'énoncé *if* permet au programme de prendre une décision au sujet des actions à exécuter en fonction d'une condition à évaluer. Le programme suivant affiche tous les entiers qui divisent l'entier 1113 :

Code Python

```
for i in range(1,1113+1):  
    if(1113 % i == 0):  
        print(i)
```

On peut aussi ajouter une clause *else* qui est invoquée quand la condition est fausse. Le programme suivant va afficher les entiers qui divisent 1113, en les précédant par des points, en utilisant un point par entier qui ne divise pas 1113 :

Code Python

```
for i in range(1,1113+1):  
    if(1113 % i == 0):  
        print(i)  
    else:  
        print(".", end="")
```

Dans cet exemple, la fonction *print* comprend un paramètre supplémentaire (*end*) afin d'éviter que chaque appel de la fonction *print* affiche une nouvelle ligne, le comportement par défaut. En Python, on peut faire référence à un paramètre par son nom (comme *end*).

Il est parfois utile d'interrompre le déroulement normal d'une répétition à l'intérieur du bloc de la répétition. La répétition est

interrompue par l'énoncé *break*. Le programme suivant va afficher le premier nombre plus grand que 1 et plus petit que 100 divisant 55 :

Code Python

```
for i in range(2,100):  
    if(55 % i == 0):  
        print(i)  
        break
```

L'énoncé *continue* permet de terminer l'itération en cours en passant directement à l'itération suivante sans arrêter la répétition comme le *break*. Le programme suivant affiche tous les multiples de 3 :

Code Python

```
for i in range(100):  
    if(i % 3 != 0):  
        continue  
    print(i)
```

## 2.7 Fonctions et modules

Nous avons utilisé une fonction (la fonction *print*) à laquelle on peut passer une chaîne de caractères devant être affichée à l'écran. La fonction *print* ne retourne rien, mais, en général, une fonction accepte des paramètres et retourne souvent une valeur. En Python, on définit une nouvelle fonction avec le mot-clé *def* suivi d'un bloc de code. Une fonction s'exécute généralement jusqu'à la fin du bloc ou jusqu'à ce que le mot-clé *return* soit rencontré. Le code suivant affiche la valeur 3 à l'écran :

Code Python

```
def mafonction(x,y):  
    return x + y
```

```
print(mafonction(1,2))
```

Les fonctions peuvent être organisées en modules. Python offre plusieurs modules prédéfinis. Le module [math](#) contient les fonctions mathématiques usuelles (arrondissement, logarithmes, trigonométrie, ...).

Voici un exemple d'importation du module [math](#) et d'utilisations de quelques fonctions courantes.

```
Code Python
>>> import math
```

La fonction *math.ceil(x)* du module [math](#) retourne le plus petit entier plus grand que *x*.

```
Code Python
>>> math.ceil(3.4)
4
```

Après avoir importé le module, il est possible d'importer le nom de la fonction explicitement pour éviter de spécifier le préfixe :

```
Code Python
>>> from math import ceil
>>> ceil(3.4)
4
```

La fonction *math.floor(x)* retourne le plus grand entier plus petit que *x*.

```
Code Python
>>> math.floor(3.4)
3
```

La fonction *math.sqrt(x)* retourne la racine carrée de *x*.

```
Code Python
>>> math.sqrt(16)
4.0
```

La fonction `math.log(x,[base])` retourne  $\log_{\text{base}}(x)$  et  $\ln(x)$  si l'argument *base* est absent.

Code Python

```
>>> math.log(8,2)
3.0
```

La variable `math.e` contient le nombre *e*. Ainsi l'appel suivante correspond à  $\ln(e)$ .

Code Python

```
>>> math.log(math.e)
1.0
```

La fonction `math.cos(x)` retourne le  $\cos(x)$  où *x* est en radian.

Code Python

```
>>> math.cos(1)
0.5403023058681398
```

La variable `math.pi` contient le nombre pi.

Code Python

```
>>> math.cos(math.pi)
-1.0
```

La fonction `math.sin(x)` retourne  $\sin(x)$  où *x* est en radian.

Code Python

```
>>> math.sin(math.pi/2)
1.0
```

## 2.8 Type list

Le type `str` permet de représenter une séquence de caractères alors que le type `list` permet de représenter une séquence d'éléments de type quelconque. Le traitement effectué sur une chaîne ou une liste est souvent similaire et plusieurs aspects se ressemblent. Ces deux types font partie des séquences Python. Une valeur de type `list` est composé



d'un crochet ouvrant « [ » suivi d'une liste d'éléments séparés par des virgules et terminé par un crochet fermant « ] ».

Code Python

```
>>> liste_de_int = [5,2,7,3,10,-4]

>>> liste_de_noms = ['Pierre','Jean','Jacques']

>>> liste_de_int

[5, 2, 7, 3, 10, -4]

>>> liste_de_noms

['Pierre', 'Jean', 'Jacques']
```

La fonction `len()` donne la taille de la liste.

Code Python

```
>>> len(liste_de_int)

6
```

La fonction `list()` permet de produire une liste à partir d'un autre type de séquence tel qu'un `range()` :

Code Python

```
>>> list(range(4))

[0, 1, 2, 3]
```

Une liste permet aussi de représenter une collection *hétérogène* formée d'éléments de types différents. En particulier, un élément d'une liste peut être lui-même une liste ! Ceci permet de construire des structures de données complexes avec des éléments enchâssés les uns dans les autres à volonté.

Code Python

```
>>> liste_melangee = ['Paul',150,30,['golf','tennis','hockey']]

>>> liste_melangee

['Paul', 150, 30, ['golf', 'tennis', 'hockey']]
```

Un tableau à deux dimensions peut être représenté par une liste de listes.

#### Code Python

```
>>> t = [[1,2,3],[4,5,6]]
>>> t
[[1, 2, 3], [4, 5, 6]]
```

Il est possible de répéter un élément à plusieurs reprises dans une liste.

#### Code Python

```
>>> liste_avec_repetition = [4,2,2,6,2,4]
>>> liste_avec_repetition
[4, 2, 2, 6, 2, 4]
```

Les opérations d'addition et de multiplication ont un effet analogue au cas des [str](#). Le littéral [] représente une liste vide. Pour ajouter des éléments d'une liste à une autre liste, l'opération d'addition peut être employée :

#### Code Python

```
>>> liste_fruits = []
>>> liste_fruits = liste_fruits + ['orange']
>>> liste_fruits
['orange']
>>> liste_fruits = liste_fruits + ['pomme']
>>> liste_fruits
['orange', 'pomme']
```

### 2.8.1 Accès par indice

Il est ainsi possible d'accéder à un élément avec la notation indiquée : *nom\_liste[indice]*. Le premier élément est à l'indice 0, le second à l'indice 1, et ainsi de suite.

#### Code Python

```
>>> liste_de_int[2]
7
```

```
>>> liste_de_noms[1]
'Jean'
>>> liste_melangee[3]
['golf', 'tennis', 'hockey']
```

On peut modifier les valeurs au sein d'une liste :

Code Python

```
>>> liste_de_int
[5, 2, 7, 3, 10, -4]
>>> liste_de_int[2] = 5
>>> liste_de_int
[5, 2, 5, 3, 10, -4]
```

### 2.8.2 Itération avec *for*

Le *for* permet de parcourir les éléments d'une liste de manière analogue à l'itération sur une chaîne. Dans l'exemple suivant, le *for* parcourt les éléments de la liste un par un et les affiche.

Code Python

```
>>> liste_de_int = [5,2,7,3,10,-4]
>>> for un_int in liste_de_int :
...     print(un_int)
5
2
7
1
3
10
-4
```

La réalisation d'une telle boucle *for* est intimement liée au mécanisme d'itérateur Python qui sera expliquée plus loin. La

méthode [enumerate\(\)](#) permet l'accès aux indices avec les éléments correspondants :

```
Code Python
>>> liste_de_int = [5,2,7,3,10,-4]
>>> for indice, un_int in enumerate(liste_de_int) :
...     print(indice, un_int)
0 5
1 2
2 7
3 3
4 10
5 -4
```

### 2.8.3 Test d'appartenance à la liste avec *in*

Le *in* permet de vérifier l'appartenance d'un élément à la liste :

```
Code Python
>>> liste_de_int = [5,2,7,3,10,-4]
>>> 3 in liste_de_int
True
>>> liste_de_noms = ['Pierre','Jean','Jacques']
>>> 'Paul' in liste_de_noms
False
```

### 2.8.4 Extraction d'une tranche d'une liste

On peut extraire une sous-liste avec les crochets :

```
Code Python
>>> liste_de_int = [5,2,7,3,10,-4]
>>> liste_de_int[2:4]
[7, 3]
>>> liste_de_int[3:]
[3, 10, -4]
>>> liste_de_int[:2]
[5, 2]
```

```
[5, 2]
>>> liste_de_int[0:5:2]
[5, 7, 10]
```

## 2.8.5 Méthodes et fonctions du type list

La méthode list.*index()* retourne la position de la première occurrence de l'élément désigné.

```
Code Python
>>> liste_de_int = [4,2,3,3,5,1,8,3,4,3]
>>> liste_de_int.index(3)
2
```

La méthode list.*count()* compte le nombre d'occurrences de l'élément désigné.

```
Code Python
>>> liste_de_int.count(3)
4
```

Comme les listes sont muables, plusieurs méthodes permettent de les modifier. La méthode list.*append()* ajoute un nouvel élément à la fin de la liste.

```
Code Python
>>> liste_de_int.append(2)
>>> liste_de_int
[4, 2, 3, 3, 5, 1, 8, 3, 4, 3, 2]
```

La méthode list.*pop()* supprime un élément à la position désignée et retourne l'élément supprimé.

```
Code Python
>>> liste_de_int.pop(4)
5
>>> liste_de_int
[4, 2, 3, 3, 1, 8, 3, 4, 3, 2]
```

La méthode `list.sort()` trie les éléments de la liste en fonction de l'ordre des objets. Les éléments sont triés sur place, et la liste est modifiée.

Code Python

```
>>> liste_de_int = [4,2,3,3,5,1,8,3,4,3]
>>> liste_de_int.sort()
>>> liste_de_int
[1, 2, 3, 3, 3, 3, 4, 4, 5, 8]
```

En revanche, la fonction `sorted()` trie la liste et retourne une nouvelle liste triée. La liste de départ demeure intacte.

Code Python

```
>>> liste_de_int = [4,2,3,3,5,1,8,3,4,3]
>>> liste_triee = sorted(liste_de_int)
>>> liste_de_int
[4, 2, 3, 3, 5, 1, 8, 3, 4, 3]
>>> liste_triee
[1, 2, 3, 3, 3, 3, 4, 4, 5, 8]
```

### 2.8.6 Liste en compréhension

Le mécanisme de liste en compréhension (*list comprehension*) en Python permet de produire le même effet mais avec une syntaxe simplifiée analogue avec la définition mathématique d'un ensemble en compréhension. La liste en compréhension est produite ici par l'indice d'un *for* :

Code Python

```
>>> liste_de_int_pairs = [indice for indice in range(0,11,2)]
>>> liste_de_int_pairs
[0, 2, 4, 6, 8, 10]
```

Une autre façon de produire le même résultat illustre la possibilité d'employer une expression formée à partir de l'indice du *for* :

Code Python

```
>>> liste_de_int_pairs = [2*indice for indice in range(6)]  
  
>>> liste_de_int_pairs  
  
[0, 2, 4, 6, 8, 10]
```

Il est possible d'inclure une clause *if* pour exclure des éléments du résultat. Ici, la liste retourne les entiers entre 0 et 9 qui ne sont pas des multiples de 3.

Code Python

```
>>> liste_non_mult3 = [i for i in range(10) if i%3 != 0]  
  
>>> liste_non_mult3  
  
[1, 2, 4, 5, 7, 8]
```

Enfin, il est permis de combiner plusieurs *for* imbriqués. Ici, les *for* imbriqués produisent la liste des paires d'entiers  $[i,j]$  où  $0 \leq i < 3$ ,  $0 \leq j < 3$  :

Code Python

```
>>> liste_de_paires = [[i,j] for i in range(3) for j in range(3) ]  
  
>>> liste_de_paires  
  
[[0, 0], [0, 1], [0, 2], [1, 0], [1, 1], [1, 2], [2, 0], [2, 1], [2, 2]]
```

## 2.9 Type dict

Dans plusieurs applications, il faut retrouver des données dans une collection à partir d'une clé d'accès. Par exemple, dans une application de contacts téléphoniques, on peut vouloir retrouver un numéro de téléphone dans la collection des contacts à partir d'un nom. Un [dict](#) peut être vu comme une généralisation du type [list](#) où l'accès aux éléments est effectué à partir d'une clé qui n'est pas limitée à un indice entier comme c'est le cas de [list](#). Plusieurs opérations internes de Python exploitent le type [dict](#). Un [dict](#) est un ensemble de couples (*clé*, *valeur*). L'implémentation est analogue avec celle du type [set](#)

sous forme d'une table de hachage. La fonction de hachage est appliquée à la clé pour déterminer l'adresse dans la table de hachage où se trouve la valeur. La clé doit être d'un type [hachable](#) comme un [str](#), un [int](#) ou un *float*. Plusieurs des opérations applicables à une liste sont aussi applicables à un [dict](#). Comme pour [list](#), un [dict](#) est muable. Un littéral [dict](#) est encadré par des accolades et contient une suite de couples *clé : valeur* séparés par des virgules. L'exemple suivant crée un répertoire téléphonique qui associe un numéro de téléphone (la valeur) à un nom (la clé) :

Code Python

```
>>> repertoire_telephonique = {'Pierre': '514-333-3333', 'Jean': '514-222-2222', 'Jacques': '514-555-5555'}

>>> repertoire_telephonique

{'Pierre': '514-333-3333', 'Jean': '514-222-2222', 'Jacques': '514-555-5555'}
```

Une valeur est sélectionnée par la syntaxe *nom\_dict[cle]* :

Code Python

```
>>> repertoire_telephonique['Jean']

'514-222-2222'
```

Le mot-clé *in* permet de vérifier l'appartenance d'un élément au [dict](#) :

Code Python

```
>>> 'Paul' in repertoire_telephonique

False
```

La méthode *keys()* retourne la liste des clés du [dict](#) sous forme d'une [vue](#) :

Code Python

```
>>> repertoire_telephonique.keys()

dict_keys(['Pierre', 'Jean', 'Jacques'])
```



La méthode `values()` retourne la liste des valeurs du [dict](#) sous forme d'une [vue](#) :

Code Python

```
>>> repertoire_telephonique.values()

dict_values(['514-333-3333', '514-222-2222', '514-555-5555'])
```

La fonction `len()` retourne la taille du [dict](#) :

Code Python

```
>>> len(repertoire_telephonique)

3
```

Pour ajouter un nouveau couple (*clé, valeur*) à un [dict](#), il suffit d'affecter la valeur à la nouvelle clé selon la syntaxe `nom_dict[cle] = valeur` :

Code Python

```
>>> repertoire_telephonique['Paul'] = '514-999-9999'

>>> repertoire_telephonique

{'Pierre': '514-333-3333', 'Jean': '514-222-2222', 'Jacques': '514-555-5555', 'Paul': '514-999-9999'}
```

Si la clé existe déjà, l'affectation modifie la valeur correspondant à la clé :

Code Python

```
>>> repertoire_telephonique['Jean'] = '514-111-1111'

>>> repertoire_telephonique

{'Pierre': '514-333-3333', 'Jean': '514-111-1111', 'Jacques': '514-555-5555', 'Paul': '514-999-9999'}
```

La syntaxe `del nom_dict[cle]` supprime le couple correspondant à la clé spécifiée :

Code Python

```
>>> del repertoire_telephonique['Jean']

>>> repertoire_telephonique

{'Pierre': '514-333-3333', 'Jacques': '514-555-5555', 'Paul': '514-999-9999'}
```

Le `for` permet d'itérer sur les clés du [dict](#) :

Code Python

```
>>> for cle in repertoire_telephonique :  
...     print(cle, ': ', repertoire_telephonique[cle])  
...  
Pierre : 514-333-3333  
Jacques : 514-555-5555  
Paul : 514-999-9999
```

## 2.10 Représentation du temps (time)

Les modules [time](#), [datetime](#) et [calendar](#) de Python contiennent des fonctions sophistiquées pour la représentation du temps. Le module [time](#) est basé sur les fonctions du langage C de la plate-forme sous-jacente. La fonction [time\(\)](#) du module [time](#) retourne le temps en secondes depuis l'*epoch* qui correspond au 1er janvier 1970 à 00 :00 :00 heures UTC ([Temps Universel Coordonné](#)) sous Windows et Unix, et ceci sous forme d'un nombre en point flottant. UTC est le temps de référence qui était autrefois appelé GMT (*Greenwich Mean Time*) :

Code Python

```
>>> import time  
  
>>> time.time()  
1541431583.7022107
```

## 2.11 Installation de modules additionnels

En installant Python, vous avez déjà plusieurs modules par défaut (par ex. `math`), mais il est facile d'ajouter de nouveaux modules spécialisés avec Python. Il suffit d'utiliser la command « `pip` ». Alors que vous êtes en ligne de commande (mais pas en sein de l'interpréteur Python), tapez la commande qui suit qui va installer au sein de l'environnement Python le module `fastrand` :

```
python -m pip install fastrand
```

Le module `fastrand` permet de générer rapidement des nombres aléatoires. Lancez ensuite l'interpréteur Python et entrez les commandes suivantes :

Code Python

```
>>> import fastrand  
>>> [fastrand.pcg32bounded(100) for i in range(10)]  
[97, 11, 64, 39, 3, 71, 90, 74, 27, 82]
```

Vous pouvez souvent installer rapidement de nouveaux modules Python ce qui est parfois essentiel pour être productif.

## 2.12 Conclusion

Le langage Python comprend des milliers de fonctions et de modules. Le langage est riche et permet de créer des serveurs web ou de faire de l'analyse statistique. Il est relativement facile d'ajouter de nouveaux modules. Il est impossible de tout couvrir en quelques pages, mais vous avez maintenant l'essentiel pour commencer à programmer en Python.

## Chapitre 3 : Traitement des données avec Python

Daniel Lemire et Robert Godin

### 3.1 Introduction

Le langage Python excelle quand vient le temps de manipuler des données. En effet, la disponibilité de modules comme NumPy et pandas fait en sorte que le traitement des données est particulièrement aisé. Naturellement, il faut d'abord se familiariser avec ces modules. Si vous n'avez pas une bonne maîtrise de l'algèbre linéaire (matrices et vecteurs) et des statistiques, il est possible que vous deviez passer outre à certaines sections de ce chapitre.

### 3.2 Le module NumPy

Le module [NumPy](#) (Numerical Python) permet de faire des calculs numériques et statistiques performants sur de grands ensembles de données. NumPy n'est pas lui-même écrit en Python, mais plutôt avec des langages plus performants comme le langage C.

Vous devriez installer NumPy dès maintenant en tapant en ligne de commande :

```
python -m pip install numpy
```

Une fois NumPy installé, vous pouvez écrire votre premier programme Python utilisant NumPy.

#### 3.2.1 Création de tableau *ndarray*

Le module NumPy comprend un type [ndarray](#) qui représente en mémoire une série de valeurs du même type. Nous pouvons l'utiliser pour représenter un vecteur ou une matrice dans le cadre de problèmes

d'algèbre linéaire. Plusieurs types sont permis au sein du [ndarray](#) dont le type *float64* (nombre à virgule flottante occupant 64 bits) ou le type *uint32* (entier non négatif occupant 32 bits). Par défaut, le module NumPy utilise le type *float64* qui permet de représenter plusieurs valeurs allant de  $-10^{308}$  jusqu'à  $10^{308}$  incluant toutes les valeurs entières de  $-2^{53}$  à  $2^{53}$ . Un [ndarray](#) peut être créé à partir d'autres types représentant des séquences en Python (par ex., *list*) par la fonction `numpy.ndarray()` ou `numpy.array()`. Dans l'exemple suivant, un vecteur (tableau à une dimension) de type [ndarray](#) est créé à partir d'une liste Python. Par défaut, les éléments sont convertis en *float64* afin de pouvoir accommoder les éléments de la liste. L'attribut *ndim* représente le nombre de dimensions du tableau, l'attribut *shape*, le nombre d'éléments de chacune des dimensions et l'attribut *dtype*, le type de chacun des éléments du tableau. Les programmeurs peuvent être paresseux et ils trouvent parfois qu'il est trop long de taper `numpy`. Il est donc commun de renommer dynamiquement le module « `numpy` » comme étant « `np` » avec l'expression « `import numpy as np` ».

Code Python

```
>>> import numpy as np
>>> v1 = np.array([4,2.5,3])
>>> type(v1)
<class 'numpy.ndarray'>
>>> v1
array([ 4. ,  2.5,  3. ])
>>> v1.ndim
1
```

```
>>> v1.shape
(3,)
>>> v1.dtype
dtype('float64')
```

L'attribut *size* est le nombre total d'éléments d'un tableau et *itemsize*, la taille de chacun des éléments en octets. La taille totale en mémoire des données en octets du tableau *v1* est donnée par l'expression « *v1.size \* v1.itemsize* ».

```
Code Python
>>> v1.size
3
>>> v1.itemsize
8
>>> v1.size * v1.itemsize
24
```

Il est possible de spécifier le type des éléments à la création du tableau avec le paramètre optionnel « *dtype* » :

```
Code Python
>>> v2 = np.array([1,2,3],dtype=float)
>>> v2
array([ 1.,  2.,  3.])
>>> v2.dtype
dtype('float64')
```

Une séquence de séquences est convertie en un tableau à deux dimensions, chacun des niveaux formant une dimension supplémentaire.

```
Code Python
>>> t1 = np.array([[3,2,6],[4,4,1]])
>>> t1
```

```
array([[3, 2, 6],
       [4, 4, 1]])
>>> t1.ndim
2
>>> t1.shape
(2, 3)
>>> t1.dtype
dtype('int32')
```

La fonction [zeros\(\)](#) prend un n-uplet en entrée qui définit les dimensions d'un tableau initialisé avec des 0 :

```
Code Python
>>> np.zeros(5)
array([ 0.,  0.,  0.,  0.,  0.])
>>> np.zeros((2,3))
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
>>> np.zeros((2,3,2))
array([[[ 0.,  0.],
        [ 0.,  0.],
        [ 0.,  0.]],
       [[ 0.,  0.],
        [ 0.,  0.],
        [ 0.,  0.]])])
```

La fonction [ones\(\)](#) crée un tableau initialisé avec des 1.

```
Code Python
>>> np.ones(4)
array([ 1.,  1.,  1.,  1.])
>>> np.ones((3,2))
array([[ 1.,  1.],
       [ 1.,  1.],
       [ 1.,  1.]])
```

```
[ 1.,  1.],  
[ 1.,  1.]])
```

Les fonctions `ones_like()` et `empty_like()` sont analogues.

Code Python

```
>>> t1 = np.array([[3,2,6],[4,4,1]])  
>>> np.zeros_like(t1)  
array([[0, 0, 0],  
       [0, 0, 0]])  
>>> np.ones_like(t1)  
array([[1, 1, 1],  
       [1, 1, 1]])
```

Il est possible que vous disposiez d'un objet qui a la forme d'un tableau (par ex., un objet de type *list*). En faisant une copie, la fonction `asarray()` convertit l'argument en `ndarray`. Si l'argument est déjà un `ndarray`, il est retourné sans en faire de copie.

Code Python

```
>>> n1 = [[1,2,3],[3,2,1]]  
>>> a1 = np.asarray(n1)
```

La fonction `identity(n)` ou `eye(n)` crée une matrice identité  $n$  par  $n$  (1 sur la diagonale et 0 ailleurs).

Code Python

```
>>> np.identity(3)  
array([[ 1.,  0.,  0.],  
       [ 0.,  1.,  0.],  
       [ 0.,  0.,  1.]])  
>>> np.eye(3)  
array([[ 1.,  0.,  0.],  
       [ 0.,  1.,  0.],  
       [ 0.,  0.,  1.]])
```



La fonction [arange\(\)](#) est analogue à [range\(\)](#) mais elle produit un [ndarray](#) plutôt qu'une liste.

Code Python

```
>>> np.arange(5)
array([0, 1, 2, 3, 4])
>>> np.arange(2,5)
array([2, 3, 4])
>>> np.arange(0,1,0.1)
array([ 0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9])
>>> np.arange(5,1,-1)
array([5, 4, 3, 2])
```

La fonction [linspace\(\)](#) est semblable à [arange\(\)](#) mais elle prend en paramètre le nombre d'éléments à produire plutôt que le pas.

Code Python

```
>>> np.linspace(0, 1, 11)
array([ 0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ])
```

La borne supérieure est incluse par défaut. Pour l'exclure, il faut spécifier *False* comme quatrième argument.

Code Python

```
>>> np.linspace(0,1,10,False)
array([ 0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9])
```

La fonction [random.rand\(\)](#) initialise les éléments d'un tableau avec des valeurs aléatoires uniformes dans l'intervalle [0,1). Les paramètres sont les tailles des dimensions.

Code Python

```
>>> np.random.rand(2,3)
array([[ 0.68373486, 0.09172521, 0.52205409],
       [ 0.78570633, 0.36038816, 0.35305615]])
```

Si aucun argument n'est spécifié, un scalaire *float* est retourné.

Code Python

```
>>> np.random.rand()
0.09117285595125935
```

La fonction [random.rand\(\)](#) initialise les éléments d'un tableau avec des valeurs aléatoires d'une [distribution normale](#)  $N(0,1)$ . Les paramètres sont les tailles des dimensions.

```
Code Python
>>> np.random.randn(3,2)
array([[ 1.1063802 ,  1.82941378],
       [ 0.43301485,  0.69286929],
       [-0.25160975, -1.11376667]])
```

La fonction [random.randint\(\)](#) permet de générer des entiers choisis aléatoirement dans un intervalle : l'expression `random.randint(1,10,4)` retourne un tableau de 4 entiers choisis au hasard dans l'intervalle [1,10).

```
Code Python
>>> np.random.randint(1,10,4)
array([8, 8, 7, 4])
```

La méthode [ndarray.reshape\(\)](#) et la fonction [reshape\(\)](#) modifient la forme du tableau pour se conformer aux dimensions spécifiées en paramètres. La taille du tableau doit être compatible avec la taille correspondant aux dimensions voulues.

```
Code Python
>>> np.arange(12).reshape(3,4)
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
>>> np.arange(12).reshape(2,3,2)
array([[[ 0,  1],
       [ 2,  3],
```

```
[ 4, 5]],  
  
[[ 6, 7],  
 [ 8, 9],  
 [10, 11]])
```

Une des dimensions du [reshape\(\)](#) peut être -1, ce qui signifie que la taille de la dimension est automatiquement ajustée pour que le résultat soit compatible avec le nombre d'éléments du tableau de départ. L'exemple précédent produit le même résultat que le précédent :

Code Python

```
>>> np.arange(12).reshape(2,-1,2)  
array([[[ 0, 1],  
        [ 2, 3],  
        [ 4, 5]],  
  
       [[ 6, 7],  
        [ 8, 9],  
        [10, 11]])
```

La méthode [flatten\(\)](#) convertit le tableau à  $n$  dimensions en un tableau à une dimension en juxtaposant bout à bout tous les éléments du tableau à  $n$  dimensions. Le paramètre *order* permet de préciser l'ordre des dimensions dans le processus d'aplatissement. Par défaut, si on passe le paramètre *order* = 'C', la fonction effectue l'aplatissement ligne par ligne.

Code Python

```
>>> a = np.arange(6).reshape(2,3)  
  
>>> a  
array([[0, 1, 2],  
       [3, 4, 5]])
```

```
>>> a.flatten(order='C')  
array([0, 1, 2, 3, 4, 5])
```

Par contre, si on passe le paramètre *Order* = 'F', la fonction ordonne les données par colonne (première colonne, deuxième colonne, etc.).

```
Code Python  
>>> a.flatten(order='F')  
array([0, 3, 1, 4, 2, 5])
```

Il est aussi possible de créer un tableau en concaténant plusieurs tableaux. La fonction [vstack\(\)](#) empile deux tableaux verticalement alors que [hstack\(\)](#) les juxtaposent horizontalement.

```
Code Python  
>>> t1=np.random.rand(2,2)  
  
>>> t1  
array([[0.43689455, 0.21828896],  
       [0.07543772, 0.07150816]])  
  
>>> t2=np.random.rand(2,2)  
  
>>> t2  
array([[0.21819298, 0.61907179],  
       [0.53598504, 0.4765252 ]])  
  
>>> np.vstack((t1,t2))  
array([[0.43689455, 0.21828896],  
       [0.07543772, 0.07150816],  
       [0.21819298, 0.61907179],  
       [0.53598504, 0.4765252 ]])  
  
>>> np.hstack((t1,t2))  
array([[0.43689455, 0.21828896, 0.21819298, 0.61907179],  
       [0.07543772, 0.07150816, 0.53598504, 0.4765252 ]])
```

### 3.2.2 Sélection par indice

La sélection par indice dans un [ndarray](#) est semblable au cas des séquences Python mais avec quelques différences. Voici un cas simple avec un indice :

Code Python
<pre>&gt;&gt;&gt; v = np.arange(5)  &gt;&gt;&gt; v  array([0, 1, 2, 3, 4])  &gt;&gt;&gt; v[2]  2</pre>

La sélection d'une tranche est analogue au mécanisme des séquences Python.

Code Python
<pre>&gt;&gt;&gt; v = np.arange(5)  &gt;&gt;&gt; v[2:4]  array([2, 3])  &gt;&gt;&gt; v[:2]  array([0, 1])  &gt;&gt;&gt; v[2:]  array([2, 3, 4])  &gt;&gt;&gt; v[0:4:2]  array([0, 2])  &gt;&gt;&gt; v[:]  array([0, 1, 2, 3, 4])</pre>

Contrairement aux séquences Python, une tranche d'un [ndarray](#) retourne une *vue* et non pas une copie de la tranche. Ceci permet de limiter la consommation de mémoire parce que la vue n'est qu'une référence aux données originales. Comme un [ndarray](#) est muable, une modification est permise dans la tranche mais il est important de

comprendre qu'elle est effectuée dans le tableau sous-jacent (et vice versa). Dans l'exemple suivant, la modification de l'élément de `v[0]` produit une modification de `v[2]`.

Code Python

```
>>> v1 = v[2:4]
>>> v1
array([2, 3])
>>> v1[0] = 7
>>> v1
array([7, 3])
>>> v
array([0, 1, 7, 3, 4])
```

Pour effectuer une copie d'une tranche, il faut employer la méthode `ndarray.copy()` ou encore la fonction `copy()`. Dans l'exemple suivant, la modification de `v[0]` ne modifie pas `v[2]`.

Code Python

```
>>> v = np.arange(5)
>>> v1 = v[2:4].copy()
>>> v1
array([2, 3])
>>> v1 = np.copy(v[2:4])
>>> v1
array([2, 3])
>>> v1[0] = 7
>>> v1
array([7, 3])
>>> v
array([0, 1, 2, 3, 4])
```

Un tableau à deux dimensions peut être vu comme un tableau de tableaux à une dimension.

Code Python

```
>>> t1 = np.array([[3,2,6],[4,4,1]])  
  
>>> t1  
array([[3, 2, 6],  
       [4, 4, 1]])
```

La sélection par un indice dans un tableau à deux dimensions retourne un tableau à une dimension.

Code Python

```
>>> t1[1]  
array([4, 4, 1])
```

Comme pour les séquences Python, il est possible d'employer une suite d'indices qui sont appliqués en séquence.

Code Python

```
>>> t1[1][2]  
1
```

Contrairement aux séquences Python, la notation plus usuelle d'une suite d'indices séparés par des virgules est aussi permise.

Code Python

```
>>> t1[1,2]  
1
```

Ces concepts se généralisent aux tableaux à  $n$  dimensions. Chacun des indices peut aussi être une tranche.

Un aspect important à noter est que la sélection d'une tranche préserve la dimension alors que l'emploi d'un indice diminue le nombre de dimensions. Par exemple, l'expression `t1[0:1,2]` retourne une vue à une dimension de `t1`.

Code Python

```
>>> t1[0:1,2]  
array([6])
```

L'expression `t1[0:1,2:3]` retourne une vue à deux dimensions avec le même élément.

Code Python

```
>>> t1[0:1,2:3]  
array([[6]])  
>>> t1[0:2,1]  
array([2, 4])  
>>> t1[0:2,1:3]  
array([[2, 6],  
       [4, 1]])
```

### 3.2.3 Indexation booléenne

[NumPy](#) permet de sélectionner les éléments d'un [ndarray](#) en exploitant un index qui est un autre [ndarray](#) de valeurs booléennes. Les éléments sélectionnés sont ceux dont la valeur correspondante d'index est *True*. Dans l'exemple suivant, *t1* est un tableau de deux lignes par trois colonnes. Le [ndarray](#) *filtre* est un tableau de même dimension formé de valeurs booléennes.

Code Python

```
>>> import numpy as np  
>>> t1 = np.array([[3,2,6],[4,4,1]])  
>>> t1  
array([[3, 2, 6],  
       [4, 4, 1]])  
>>> filtre=np.array([[False, True, False], [True, True, False]])  
>>> filtre  
array([[False, True, False],
```



```
[ True,  True, False]])
```

L'expression sélectionne les éléments de *t1* dont la valeur correspondante de *filtre* est *True*, sous forme d'un tableau à une dimension.

Code Python

```
>>> t2=t1[filtre]
>>> t2
array([2, 4, 4])
>>> t2=[0,0,0]
>>> t2
[0, 0, 0]
>>> t1
array([[3, 2, 6],
       [4, 4, 1]])
>>> t2=t1[filtre]
>>> t2
array([2, 4, 4])
```

### 3.2.4 Changement de dimensions

Il est possible d'insérer une nouvelle dimension dans le résultat d'une sélection. Par exemple, on peut faire passer un vecteur à un tableau 2D avec une colonne qui correspond au vecteur, en employant la constante *newaxis*.

Code Python

```
>>> v = np.arange(5)
>>> v.shape
(5,)
>>> t=v[0:3,np.newaxis]
>>> t
array([[0],
```

```
[1],  
[2]])  
>>> t.shape  
(3, 1)
```

La fonction [expand\\_dims\(\)](#) permet aussi d'ajouter de nouvelles dimensions.

```
Code Python  
>>> w=np.expand_dims(v,axis=1)  
>>> w  
array([[0],  
       [1],  
       [2],  
       [3],  
       [4]])  
>>> w.shape  
(5, 1)
```

À l'inverse, la fonction [squeeze\(\)](#) permet d'éliminer toutes les dimensions de taille 1.

#### Code Python

```
>>> t = np.random.rand(1,3,1)

>>> t
array([[[[0.93786784],
          [0.15052905],
          [0.65310908]]]])

>>> t.shape
(1, 3, 1)

>>> t1 = np.squeeze(t)

>>> t1
array([0.93786784, 0.15052905, 0.65310908])

>>> t1.shape
(3,)
```

Il est aussi possible de spécifier une dimension particulière à éliminer avec le paramètre *axis*.

#### Code Python

```
>>> t2=np.squeeze(t,axis=0)

>>> t2
array([[[[0.93786784],
          [0.15052905],
          [0.65310908]]]])

>>> t2.shape
(3, 1)

>>> t3=np.squeeze(t,axis=2)

>>> t3
array([[0.93786784, 0.15052905, 0.65310908]])

>>> t3.shape
(1, 3)
```

### 3.2.5 Indexation sophistiquée (*fancy*)

Une nouveauté par rapport à l'indexation habituelle des séquences Python est une forme d'indexation, dite *fancy*, plus sophistiquée par un indice qui est lui-même une séquence ou plus généralement un tableau d'indices. Chacune des valeurs de l'indice sélectionne un élément de la séquence d'entrée. On peut notamment utiliser cette approche pour permuter les éléments d'un tableau. Par exemple, on peut inverser les éléments d'un tableau :

Code Python

```
>>> v = np.array([4,12,7,9,62,8])
>>> v[[5,4,3,2,1,0]]
array([ 8, 62,  9,  7, 12,  4])
```

La forme du résultat est déterminée par la forme de l'indice.

Code Python

```
>>> v = np.array([4,12,7,9,62,8])
>>> v[[4,2,2,0,5]]
array([62,  7,  7,  4,  8])
>>> indices = np.array([[4,2,5],[2,0,1]])
>>> v[indices]
array([[62,  7,  8],
       [ 7,  4, 12]])
```

Cette notion est généralisable aux tableaux à plusieurs dimensions. Par exemple, dans le cas d'un tableau à deux dimensions comme entrée, chacun des éléments de l'indice sélectionne un élément du tableau en deux dimensions :

Code Python

```
>>> t=np.arange(12).reshape(3,4)
>>> t
array([[ 0,  1,  2,  3],
```

```
[ 4, 5, 6, 7],  
 [ 8, 9, 10, 11]])  
>>> t[[1,0],[3,1]]  
array([7, 1])
```

### 3.2.6 Opérations sur les tableaux (*ufuncs*) : vectorisation et diffusion

Une caractéristique avantageuse de [NumPy](#) est la possibilité d'effectuer des opérations (appelées [ufuncs](#)) sur des opérandes qui sont des tableaux sans devoir coder les itérations correspondantes. Comme le traitement est effectué en boucles du langage C, le résultat est obtenu plus rapidement. Cette transformation est parfois appelée *vectorisation* dans le jargon de [NumPy](#). D'autre part, les opérations entre opérandes de tailles différentes sont automatiquement transformées de la manière la plus naturelle possible par un processus de diffusion (*broadcasting*).

Lorsque deux opérandes sont de même taille, l'opération est appliquée élément par élément. Par exemple, la somme de deux vecteurs applique la somme élément par élément.

```
Code Python  
>>> v = np.arange(5)  
  
>>> v  
array([0, 1, 2, 3, 4])  
>>> u = np.arange(5,10)  
  
>>> u  
array([5, 6, 7, 8, 9])  
>>> v+u  
array([ 5, 7, 9, 11, 13])
```

Lorsque les opérandes sont de tailles différentes, le processus de *diffusion* (*broadcasting*) est appliqué. La somme d'un tableau et d'un scalaire applique automatiquement l'opération du scalaire à chacun des éléments du tableau.

Code Python
<pre>&gt;&gt;&gt; v array([0, 1, 2, 3, 4])  &gt;&gt;&gt; v + 1 array([1, 2, 3, 4, 5])</pre>

Conceptuellement, on peut imaginer que le scalaire 1 est étendu à un vecteur de taille 5 en répétant le 1 ([1,1,1,1,1]) afin de produire un opérande compatible avec l'autre opérande. Cependant, dans la mise en œuvre, la copie des éléments n'est pas réellement effectuée. Ce n'est qu'une conceptualisation du processus.

La diffusion est appliquée automatiquement à un grand nombre d'opérations binaires incluant les opérateurs de comparaison booléens.

Code Python
<pre>&gt;&gt;&gt; v*3 array([ 0,  3,  6,  9, 12])  &gt;&gt;&gt; v**2 array([ 0,  1,  4,  9, 16], dtype=int32)  &gt;&gt;&gt; v &gt; 3 array([False, False, False, False,  True], dtype=bool)</pre>

L'expression booléenne produit un tableau booléen qui peut servir pour une indexation booléenne. Dans l'exemple suivant, les éléments de *t1* supérieurs à trois sont sélectionnés par le tableau *t1>3*, et mis à 0.

Code Python

```
>>> t1 = np.array([[3,2,6],[4,4,1]])
>>> t1
array([[3, 2, 6],
       [4, 4, 1]])
>>> t1[t1>3]=0
>>> t1
array([[3, 2, 0],
       [0, 0, 1]])
```

Ce genre d'expression permet d'appliquer un traitement à des éléments d'un tableau qui respectent une condition. Les opérateurs Booléens, *and*, *or* et *not* ne sont pas applicables aux tableaux. Il faut employer les fonctions [numpy.logical\\_and\(\)](#), [numpy.logical\\_or\(\)](#), [numpy.logical\\_not\(\)](#).

Code Python

```
>>> t1 = np.array([[3,2,6],[4,4,1]])
>>> t1
array([[3, 2, 6],
       [4, 4, 1]])
>>> t1[np.logical_and(t1>2,t1<5)]=0
>>> t1
array([[0, 2, 6],
       [0, 0, 1]])
```

Un grand nombre de fonctions mathématiques (exponentielles, trigonométriques, ...) sont aussi automatiquement appliquées élément par élément sur les tableaux.

Code Python

```
>>> np.exp(v)
array([ 1. ,  2.71828183,  7.3890561 ,
```

```

20.08553692, 54.59815003])
>>> np.log(v)
array([-inf, 0. , 0.69314718,
       1.09861229, 1.38629436])
>>> np.sin(v)
array([ 0. , 0.84147098, 0.90929743,
       0.14112001, -0.7568025 ])
>>> np.cos(v)
array([ 1. , 0.54030231, -0.41614684,
      -0.98999925 , -0.65364362])
>>> np.sqrt(v)
array([ 0. , 1. , 1.41421356,
       1.73205081, 2. ])

```

Le produit avec `*` n'est pas un produit matriciel de deux tableaux mais il représente une multiplication qui opère élément par élément.

```

Code Python
>>> v
array([0, 1, 2, 3, 4])
>>> u
array([5, 6, 7, 8, 9])
>>> v*u
array([ 0, 6, 14, 24, 36])

```

Le produit scalaire entre deux vecteurs est exprimé par le `@` ou la méthode [numpy.dot\(\)](#). La fonction [numpy.matmul\(\)](#) produit le même effet sauf que les règles de diffusion de [numpy.dot\(\)](#) et [numpy.matmul\(\)](#) sont différentes.



#### Code Python

```
>>> v
array([0, 1, 2, 3, 4])

>>> u
array([5, 6, 7, 8, 9])

>>> v@u
80

>>> v.dot(u)
80

>>> np.matmul(v,u)
```

Le produit \* d'un tableau à deux dimensions (*matrice*) par un vecteur applique l'opération \* entre chacune des lignes du tableau à deux dimensions et le vecteur. C'est comme si le vecteur était dupliqué pour atteindre la même dimension que le tableau.

#### Code Python

```
>>> t1 = np.array([[3,2,6],[4,4,1]])

>>> v1 = np.array([1,2,3])

>>> t1*v1
array([[ 3,  4, 18],
       [ 4,  8,  3]])

>>> t2 = np.array([[1,2,3],[1,2,3]])

>>> t1*t2
array([[ 3,  4, 18],
       [ 4,  8,  3]])
```

Conceptuellement, c'est comme si le vecteur [1,2,3] est étendu à un tableau à deux dimensions par la diffusion, 2 lignes par 3 colonnes, de taille égale à l'autre opérande en répétant le vecteur [1,2,3] deux fois.

La diffusion est flexible et les règles générales sont relativement complexes. Le principe consiste à tenter d'étendre les opérandes, une dimension à la fois de la dernière à la première de manière à les rendre compatibles. Dans l'exemple suivant, le tableau *t2par1* a une colonne et trois pour *t1par3*. Le processus de diffusion triple la colonne de *t2par1* afin d'obtenir trois colonnes. On obtient ainsi conceptuellement le tableau de dimension 2 par 3 ([[10,10,10], [20,20,20]]). Ensuite, le tableau *t1par3* a une ligne, et *t2par1*, deux lignes. Le processus de diffusion duplique la ligne de *t1par3* afin que le nombre de lignes soit égal à celui de *t2par1*. On obtient conceptuellement le tableau de dimension 2 par 3 [[1,2,3], [1,2,3]]. Les deux matrices rendues compatibles sont ensuite additionnées élément par élément.

Code Python

```
>>> t1par3 = np.array([[1,2,3]])
>>> t1par3.shape
(1, 3)
>>> t2par1 = np.array([[10],[20]])
>>> t2par1.shape
(2, 1)
>>> t2par3 = t1par3+t2par1
>>> t2par3
array([[11, 12, 13],
       [21, 22, 23]])
>>> t2par3.shape
(2, 3)
>>> t2par1en2par3 = np.array([[10,10,10],[20,20,20]])
```

```
>>> t1par3en2par3 = np.array([[1,2,3],[1,2,3]])
>>> t2par1en2par3+t1par3en2par3
array([[11, 12, 13],
       [21, 22, 23]])
```

La multiplication plus usuelle d'un tableau par un vecteur est exprimée par `@` ou [numpy.dot\(\)](#) ou encore la fonction [numpy.matmul\(\)](#). Le résultat est un vecteur.

Code Python

```
>>> t1
array([[3, 2, 6],
       [4, 4, 1]])
>>> v1
array([1, 2, 3])
>>> t1@v1
array([25, 15])
>>> t1.dot(v1)
array([25, 15])
>>> np.matmul(t1,v1)
array([25, 15])
```

Le `@`, ou [numpy.dot\(\)](#) ou [numpy.matmul\(\)](#) sont aussi applicables à la multiplication matricielle.

Code Python

```
>>> t1
array([[3, 2, 6],
       [4, 4, 1]])
>>> t2 = np.array([[1,2],[3,4],[1,2]])
>>> t1@t2
array([[15, 26],
       [17, 26]])
>>> t1.dot(t2)
```

```
array([[15, 26],
       [17, 26]])
>>> np.matmul(t1,t2)
array([[15, 26],
       [17, 26]])
```

L'attribut `T` ou la fonction [transpose\(\)](#) retourne une vue de la matrice transposée.

```
Code Python
>>> t2
array([[1, 2], [3, 4]])
>>> t3
array([[0, 1], [2, 2]])
>>> t3.T
array([[0, 2], [1, 2]])
>>> np.transpose(t3)
array([[0, 2], [1, 2]])
```

### 3.2.7 Opérations d'agrégation

[NumPy](#) offre plusieurs opérations dites d'agrégation qui produisent un scalaire à partir d'un tableau. La méthode `ndarray.sum()` (ou la fonction [sum\(\)](#)) produit la somme des scalaires du tableau, peu importe sa forme.

```
Code Python
>>> a
array([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11])
>>> a.sum()
66
>>> sum(a)
66
>>> b = a.reshape(3,4)
```

```
>>> b
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
>>> b.sum()
66
```

L'agrégation est effectuée selon un axe si l'argument *axis* est spécifié. Si *axis*=0, la somme se fait selon l'axe des lignes, et produit le total pour chacune des colonnes. Si *axis*=1, la somme se fait selon l'axe des colonnes.

```
Code Python
>>> b.sum(axis=0)
array([12, 15, 18, 21])
>>> b.sum(axis=1)
array([ 6, 22, 38])
>>> t = np.arange(12)
>>> t1 = t.reshape(2,3,2)
>>> t1
array([[[ 0,  1],
        [ 2,  3],
        [ 4,  5]],
       [[ 6,  7],
        [ 8,  9],
        [10, 11]]])
>>> t1.sum()
66
>>> t1.sum(axis=0)
array([[ 6,  8],
```

```

[10, 12],
[14, 16]])
>>> t1.sum(axis=1)
array([[ 6,  9],
       [24, 27]])
>>> t1.sum(axis=2)
array([[ 1,  5,  9],
       [13, 17, 21]])

```

Pour comprendre ce que l'expression `sum(axis=0)` calcule, prenez en compte que le résultat est donné par les sommes  $0+6$ ,  $1+6$ ,  $2+8$ ,  $3+9$ ,  $4+10$ ,  $5+11$ . Pour l'expression `sum(axis=1)`, le résultat porte sur les sommes  $0+2+4$ ,  $1+3+5$ ,  $6+8+10$ ,  $7+9+11$ . Quant à l'expression `sum(axis=2)`, nous effectuons les sommes  $1+1$ ,  $2+3$ ,  $4+5$ ,  $7+7$ ,  $8+9$ ,  $10+11$ .

Si le paramètre *axis* est une liste, la somme est faite en regroupant tous les éléments des axes spécifiés.

```

Code Python
>>> t1.sum(axis=(0,1))
array([30, 36])
>>> t1.sum(axis=(0,2))
array([14, 22, 30])
>>> t1.sum(axis=(1,2))
array([15, 51])

```

Plusieurs autres fonctions d'agrégation sont disponibles telles que *min*, *max*, *mean* (moyenne), *var* (variance), *std* (écart-type).

### 3.3 Le module pandas

Le module [pandas](#) est populaire pour les manipulations de données typiques de la préparation et de l'analyse de données. Il est construit

à partir de [NumPy](#). Les étapes suivantes doivent typiquement être effectuées dans un processus d'analyse de données : chargement des données, transformation des données, production de statistiques exploratoires et visualisation. Le module [pandas](#) offre des outils versatiles à cet effet.

Une grande variété de sources de données peuvent être exploitées pour le chargement de données dont les fichiers CVS, JSON, HTML, et les requêtes SQL sur une base de données relationnelles. Les données sont chargées en exploitant les deux principales structures de données de [pandas](#) : le type [Series](#) et le type [Dataframe](#).

Une [Series](#) est un genre de tableau à une dimension avec un index qui permet de faire référence aux éléments. Par opposition à un tableau ou une séquence Python, l'index n'est pas limité à un intervalle d'entiers.

Un [Dataframe](#) est essentiellement un tableau à deux dimensions où chacune des colonnes correspond à une [Series](#). Contrairement à un tableau [NumPy](#), chacune des colonnes peut être de type différent à la manière d'une table d'une base de données relationnelle. Les lignes du [Dataframe](#) sont identifiées par un index de ligne et les colonnes par un index de colonnes. Les index permettent d'identifier les lignes et les colonnes afin d'y accéder directement. La sélection de tranches de lignes est permise avec la même syntaxe que pour les séquences Python.

Vous devriez installer le module pandas dès maintenant en tapant en ligne de commande :

```
python -m pip install pandas
```

Pour pouvoir utiliser NumPy et pandas dans vos programmes Python, nous vous suggérons de débiter ceux-ci par les lignes suivantes :

```
Code Python
import pandas as pd
import numpy as np
```

### 3.3.1 Le type Series de pandas

Une [Series](#) peut être créée à partir d'une séquence. Une valeur nulle est représentée par la valeur « not a number » *np.nan* de [NumPy](#).

```
Code Python
import pandas as pd
import numpy as np

serie1 = pd.Series([4.5,17,-2,np.nan,12.6])

serie1
```

Résultat :

```
0      4.5
1     17.0
2     -2.0
3      NaN
4     12.6
dtype: float64
```

L'attribut *values* retourne les valeurs de la [Series](#).

```
Code Python
serie1.values
```

Résultat :

```
array([ 4.5, 17. , -2. ,  nan, 12.6])
```

Le type des valeurs est un [ndarray](#) de [NumPy](#).

```
Code Python
type(serie1.values)
```

Résultat :

```
numpy.ndarray
```



L'attribut *index* retourne les éléments qui servent d'indices (étiquettes, *label*) pour les lignes.

Code Python
<code>serie1.index</code>

Résultat :

```
RangeIndex(start=0, stop=5, step=1)
```

Par défaut, l'index est une séquence d'entiers entre 0 et  $n-1$  comme pour un tableau [NumPy](#). La sélection par indice est analogue :

Code Python
<code>serie1[2]</code>

Résultat :

```
-2.0
```

La sélection par tranche est aussi analogue :

Code Python
<code>serie1[1:4]</code>

Résultat :

```
1    17.0
2    -2.0
3     NaN
dtype: float64
```

Au-delà de ces mécanismes offerts avec les séquences et les [ndarray](#), des mécanismes de sélection plus sophistiqués sont possibles.

Avec une valeur de type Series, un index n'est pas contraint à être un intervalle  $[0, n-1]$ . Dans l'exemple suivant, une liste de chaînes de caractères sert d'index. Le résultat est une structure semblable à un [dict](#) dont les attributs sont ordonnés.

Code Python
<code>serie2 = Series([25, 20, 32, 25], index=['Jean', 'Guy', 'Paul', 'Pierre'])</code>
<code>serie2</code>

Résultat :

```
Jean      25
```

```
Guy      20
Paul     32
Pierre   25
dtype: int64
```

La sélection par indice et par tranche est aussi permise pour les index non entiers :

Code Python
serie2['Guy']

Résultat :

```
20
```

On peut sélectionner un intervalle :

Code Python
serie2['Guy':'Pierre']

Résultat :

```
Guy      20
Paul     32
Pierre   25
dtype: int64
```

### 3.3.2 Le type Dataframe du module pandas

Un [Dataframe](#) est une structure tabulaire à deux dimensions, avec des lignes et des colonnes. Chacune des colonnes correspond à une [Series](#). Ainsi un [Dataframe](#) est semblable à un chiffrier ou encore une table d'une base de données relationnelle. Chacune des colonnes correspond habituellement à une caractéristique des observations à analyser. Les colonnes peuvent avoir des types différents les uns des autres. Il y a plusieurs façons de créer un [Dataframe](#). Par exemple, il est possible de partir d'une collection de collections et d'une liste d'étiquettes de lignes et de colonnes :

Code Python
donnees= [[30, 'Avocate'], [35, 'Pompier'], [25, 'Médecin'], [40, 'Ingénieur']]

```
df1 = pd.DataFrame(donnees, index=['Paule', 'Guy', 'Eve', 'Pierre'], columns=['age', 'profession'])
```

```
df1
```

Résultat :

	age	profession
Paule	30	Avocate
Guy	35	Pompier
Eve	25	Médecin
Pierre	40	Ingénieur

L'attribut *index* d'un [DataFrame](#) contient les étiquettes qui servent d'indices de ligne.

```
Code Python
```

```
df1.index
```

Résultat :

```
Index(['Paule', 'Guy', 'Eve', 'Pierre'], dtype='object')
```

L'attribut *columns* contient les étiquettes qui identifient les colonnes.

```
Code Python
```

```
df1.columns
```

Résultat :

```
Index(['age', 'profession'], dtype='object')
```

On peut sélectionner un intervalle de lignes par position avec la même syntaxe que pour les tranches (*slice*) :

```
Code Python
```

```
>>> df1[1:3]
```

	age	profession
Guy	35	Pompier
Eve	25	Médecin

Dans ce cas, la sélection spécifie une position entière. De manière un peu surprenante, la sélection avec un indice n'est pas permise...

```
Code Python
```

```
df1[1]
```

Résultat :

```

-----
-----
KeyError                                Traceback
  (most recent call last)
~\anaconda3\lib\site-
packages\pandas\core\indexes\base.py      in
get_loc(self, key, method, tolerance)
...

```

Il faut spécifier un intervalle :

Code Python
<pre>&gt;&gt;&gt; df1[1:2]</pre>
<pre>age profession</pre>
<pre>Guy 35  Pompier</pre>

Il est possible de sélectionner avec une tranche d'étiquettes comme pour une [Series](#) :

Code Python
<pre>&gt;&gt;&gt; df1['Guy':'Eve']</pre>
<pre>age profession</pre>
<pre>Guy 35  Pompier</pre>
<pre>Eve 25  Médecin</pre>

A noter que l'intervalle d'étiquettes inclut la borne supérieure ! La sélection par une étiquette n'est pas permise :

Code Python
<pre>df1['Guy']</pre>

Résultat :

```

-----
-----
KeyError                                Traceback
  (most recent call last)
~\anaconda3\lib\site-
packages\pandas\core\indexes\base.py      in
get_loc(self, key, method, tolerance)
...

```

Il faut spécifier un intervalle :

Code Python

```
>>> df1['Guy':'Guy']  
  
age profession  
Guy 35 Pompier
```

Une colonne d'un [Dataframe](#) est sélectionnée en spécifiant

l'étiquette de la colonne. On peut le faire avec l'attribut profession :

Code Python

```
df1['profession']
```

Résultat :

```
Paule      Avocate  
Guy        Pompier  
Eve        Médecin  
Pierre     Ingénieur  
Name: profession, dtype: object
```

Ou encore avec l'attribut « age » :

Code Python

```
df1['age']
```

Résultat :

```
Paule      30  
Guy        35  
Eve        25  
Pierre     40  
Name: age, dtype: int64
```

Le résultat est de type [Series](#).

Code Python

```
type(df1['age'])
```

Résultat :

```
pandas.core.series.Series
```

En revanche, une tranche de colonnes n'est pas permise :

Code Python

```
df1['age':'profession']
```

Résultat :

```
-----  
-----  
KeyError                                Traceback  
(most recent call last)  
~\anaconda3\lib\site-  
packages\pandas\core\indexes\base.py      in  
get_loc(self, key, method, tolerance)  
...
```

Il est aussi possible de sélectionner un sous-ensemble de lignes et de colonnes par les étiquettes en employant l'attribut *loc* :

Code Python
<pre>&gt;&gt;&gt; df1.loc['Guy':'Eve', ['age']]  age Guy  35 Eve  25</pre>

Le résultat est un [Dataframe](#). L'indice avec l'attribut *loc* est multidimensionnel. La première partie est une tranche de lignes alors que la seconde est une liste de colonnes :

Code Python
<pre>&gt;&gt;&gt; df1.loc['Guy':'Eve', ['age', 'profession']]  age profession Guy  35  Pompier Eve  25  Médecin</pre>

Comme les colonnes ne sont pas ordonnées, l'ordre peut être quelconque :

Code Python
<pre>&gt;&gt;&gt; df1.loc['Guy':'Eve', ['profession', 'age']]  profession age Guy  Pompier  35 Eve  Médecin  25</pre>

Il est possible de spécifier simplement une étiquette de ligne :

Code Python

```
df1.loc['Guy',['age']]
```

Résultat :

```
age    35
Name: Guy, dtype: object
```

Le résultat est de type [Series](#). Si le second paramètre est absent, toutes les colonnes sont sélectionnées :

Code Python

```
df1.loc['Guy']
```

Résultat :

```
age    35
profession    Pompier
Name: Guy, dtype: object
```

Si le second argument est aussi une étiquette, le résultat est un scalaire :

Code Python

```
df1.loc['Guy','age']
```

Résultat :

35

L'attribut `.at` produit aussi le même résultat avec une étiquette de ligne et une étiquette de colonne :

Code Python

```
df1.at['Guy','age']
```

Résultat :

35

Il est aussi possible faire une sélection par les positions de lignes et de colonnes en employant l'attribut *iloc* :

Code Python

```
>>> df1.iloc[1:3, 0:1]
```

```
age
Guy 35
Eve 25
```

Dans ce cas, les indices sont des entiers ou des tranches.

Si un des deux indices est un entier, la dimension est réduite à une [Series](#) :

Code Python
<code>df1.iloc[2,0]</code>

Résultat :

age	35
profession	Pompier
Name: Guy, dtype: object	

Si les deux sont entiers, le résultat est un scalaire :

Code Python
<code>df1.iloc[2,0]</code>

Résultat :

25
----

### 3.3.3 Lecture des données à partir d’une base de données

Une grande partie des données de nos entreprises se trouvent dans des systèmes de bases de données SQL : Oracle, MySQL, SQL Server, etc. Si vos données se trouvent dans une base de données SQL, il est possible de les charger directement dans un [Dataframe](#) de pandas. Si vous n’êtes pas familier avec les bases de données SQL, nous vous invitons à prendre connaissance de l’article Wikipédia correspondant :

[https://fr.wikipedia.org/wiki/Structured\\_Query\\_Language](https://fr.wikipedia.org/wiki/Structured_Query_Language)

Le système de base de données le plus simple et le plus populaire est sans doute SQLite : il permet de créer une base de données entièrement contenue dans un seul fichier, et il ne nécessite pas de serveur. Python inclus par défaut avec un module permettant d’accéder les bases de données SQLite. Bien que SQLite soit relativement simple en comparaison avec des systèmes de bases de données plus sophistiqués comme Oracle, SQL Server, etc., le



principe est le même du point de vue de l'utilisateur Python. On peut faire appel au module sqlite3 de la sorte :

Code Python

```
import sqlite3

with sqlite3.connect("img.db") as con:
    print(con.execute("SELECT 1 + 2").fetchall())
```

Dans cet exemple, l'accès à la connexion au moteur de base de données se fait au sein d'un bloc « with » : ce bloc ne s'exécute que si l'accès a été obtenu avec succès. En supposant que le fichier img.db n'existe pas, nous pouvons le créer et y mettre des données au format SQLite :

Code Python

```
with sqlite3.connect("img.db") as con:
    con.execute("CREATE TABLE IF NOT EXISTS accounts (account_number INTEGER, amount DECIMAL, PRIMARY KEY(account_number))")
    con.execute("INSERT INTO accounts (account_number, amount) values (12334, 1.50)")
```

Le nom de fichier est arbitraire et vous pouvez remplacer img.db par un autre nom. On peut consulter les données contenues dans le fichier img.db et les afficher à l'écran avec la fonction print :

Code Python

```
with sqlite3.connect("img.db") as con:
    print(con.execute("SELECT * FROM accounts").fetchall())
```

On peut aussi modifier les données contenues dans ce fichier :

Code Python

```
with sqlite3.connect("img.db") as con:
    con.execute("UPDATE accounts SET amount = amount + 10 WHERE account_number = 12334")
```

Le module pandas peut ensuite lire les données en question dans un Dataframe :

```
Code Python
import pandas as pd

import sqlite3

with sqlite3.connect("img.db") as con:
    df = pd.read_sql_query('SELECT * FROM accounts',con)

print(sum(df["amount"]))
```

### 3.3.4 Traitement de données ouvertes

La province de l'Ontario rend disponible les salaires de ses fonctionnaires les mieux payés. Il s'agit d'un exemple de données ouvertes : des données qui sont librement accessible et que vous pouvez utiliser au sein de vos projets. Commencez par récupérer les fichiers CSV (comma-separated-values) trouvés au sein du répertoire 'data' à l'adresse suivante :

<https://github.com/lemire/livreds>

Enregistrez les fichier CSV sur votre machine dans le même répertoire où vous exécutez vos programmes Python. Une fois les fichiers en place, vous devriez pour exécuter la commande suivante :

```
Code Python
dataset = pd.read_csv("tbs-pssd-compendium-salary-disclosed-2021-en-utf-8-2022-03-25.csv")

print(dataset.head(5))
```

Il peut être intéressant de consulter la taille du Dataframe que nous venons de charger :

```
Code Python
>>> print(dataset.shape)

(244390, 9)
```

```
>> print(dataset.keys())
Index(['Sector', 'Last Name', 'First Name', 'Salary', 'Benefits', 'Employer',
      'Job Title', 'Year', '_docID'],
      dtype='object')
```

Chacun des employés a son revenu total décrit en deux colonnes : salary (salaire) et benefits (bénéfices). Créons une nouvelle colonne représentant le revenu total des employés :

```
Code Python
dataset["total"] = dataset["Salary"].astype(float) + dataset["Benefits"].astype(float)
pd.options.display.float_format = '${:,.2f}'.format
```

On spécifie le format d’affichage des données pour être plus élégant (le signe de dollar et deux chiffres après la virgule). La fonction *groupby* d’un dataframe permet de calculer le salaire total moyen ainsi que le nombre d’employé par titre d’emploi.

```
Code Python
import numpy as np
salarypertitle = dataset.groupby("Job Title").agg({'total': [np.size, np.mean]})
```

On constate que certains titres d’emploi correspondent à peu d’employés. Nous pouvons placer une limite et ne considérer que les titres d’emploi correspondants à plus de 200 employés :

```
Code Python
largecount = salarypertitle[salarypertitle[("total", "size")]>200]
largecount.sort_values(("total", "mean"), ascending=False)
```

Au lieu du salaire moyen, nous pouvons plutôt calculer le salaire maximal :

```
Code Python
salarypertitle = dataset.groupby("Job Title").agg({'total': [np.size, np.max]})
salarypertitle = salarypertitle[salarypertitle[("total", "size")]>200].sort_values(("total", "a
max"), ascending=False)
```

```
print(salarypertitle.head(10))
```

On peut s'intéresser aux professeurs et trouver ceux qui ont la meilleure rémunération :

Code Python

```
>>> profsalary = dataset[dataset['Job Title'].str.contains("Prof")]
print(profsalary.sort_values("total",ascending=False)[["Last Name", "First Name", "Employer", "total"]].head(10))
```

Nous aimerions savoir si les hommes gagnent plus que les femmes. La base de données ne nous permet pas directement de distinguer les hommes et les femmes. Cependant, nous avons les prénoms et le prénom peut souvent être utilisé pour distinguer le sexe. À cette fin, nous pouvons utiliser le fichier *us-likelihood-of-gender-by-name-in-2014.csv* qui devrait être présent sur votre machine. Nous pouvons le charger dans un second Dataframe :

Code Python

```
>>> genderstat = pd.read_csv("us-likelihood-of-gender-by-name-in-2014.csv")
>>> print(genderstat[["sex", "name"]].head())
```

	sex	name
0	F	Elaine
1	F	Cathy
2	F	Heidi
3	F	Vicki
4	F	Melinda

Nous pouvons maintenant faire une jointure entre les deux Dataframes afin d'ajouter un attribut « sex » aux employés. Dans notre cas, on associe l'attribut « First Name » dans notre base de données de salaires avec l'attribut « name » dans la nouvelle base de données comportant un attribut « sex » : quand les deux valeurs (First name et name) correspondent, la rangée est fusionnée ce qui

correspond à l'ajout d'un attribut « sex » à la base de données des salaires. Par défaut, la fonction « merge » de pandas fait une jointure interne : si un prénom est absent de l'une des deux bases de données, les enregistrements sont omis du résultat. Dans tous les cas, le résultat ne sera pas parfait puisque les prénoms ne déterminent pas le sexe parfaitement, mais le résultat devrait être suffisamment correct pour répondre aux besoins de notre exercice :

```
Code Python
>>> datasetwithgender = pd.merge(dataset, genderstat, left_on="First Name", right_on="name")

>>> print(datasetwithgender.groupby(["sex"]).agg({'total': [np.mean, np.max, np.median]}))

      total
      mean    amax    median
sex
F  $118,530.40 $1,527,441.40 $107,026.77
M  $129,955.95 $1,635,785.84 $117,675.63
```

On constate donc que les hommes gagnent plus d'argent que les femmes dans cette base de données. Que pouvons-nous dire au sujet des professeurs d'une université en particulier ? Prenons l'Université Waterloo :

```
Code Python
waterlooprof = datasetwithgender[datasetwithgender["Employer"] == "University Of Waterloo"]

print(waterlooprof.groupby(["sex"]).agg({'total': [np.mean, np.max, np.median]}))

      total
      mean    amax    median
sex
F  $148,057.31 $340,825.95 $135,983.78
M  $160,769.90 $343,058.80 $154,489.34
```

On constate encore une fois un avantage financier pour les hommes. La base de données est vaste et vous pourriez faire le même exercice pour les médecins ou les avocats.

### **3.4 Conclusion**

On peut faire beaucoup plus de travail avec les modules NumPy et pandas. Il est possible de faire des analyses statistiques avancées. Il est aussi relativement aisé d'ajouter de la visualisation graphique à l'aide de modules supplémentaires qui s'intègrent bien avec NumPy et pandas. Par ailleurs, l'environnement de base de Python peut être remplacé par celui plus convivial de Jupyter. Néanmoins, la matière du chapitre vous donne les fondements nécessaires pour approfondir davantage vos connaissances.

## Chapitre 4 : Visualisation des données

Neila Mezghani et Daniel Lemire

### 4.1 Introduction à la visualisation des données

La visualisation de données est une étape essentielle de la science des données. Elle permet de transformer les chiffres bruts en graphiques, diagrammes ou autres représentations visuelles. La visualisation de données permet de découvrir des tendances cachées, de communiquer des connaissances complexes de manière intuitive et de prendre des décisions éclairées en exploitant le pouvoir de l'analyse visuelle. Dans ce chapitre, nous explorerons les principes fondamentaux de la visualisation de données, ainsi que les outils et techniques permettant de créer des visualisations informatives et percutantes. Deux types de visualisation sont traités soit la visualisation de la distribution et de la corrélation. La visualisation sera également traitée dans le chapitre 7 dans le cadre de l'analytique d'affaires, qui est le processus d'interprétation des données afin d'améliorer la prise de décision au sein de l'organisation.

Dans ce chapitre, nous utiliserons principalement Seaborn qui est une bibliothèque de visualisation de données en Python qui s'appuie sur Matplotlib et qui est intégrée avec pandas. Seaborn offre une interface de haut niveau pour la création de graphiques statistiques attrayants et informatifs. Elle permet également de créer des graphiques multi-plots avec des facilités pour comparer des sous-ensembles de données.

## 4.2 Visualisation de la distribution

La visualisation de la distribution offre un aperçu de la répartition et de la dispersion des valeurs au sein d'un ensemble de données. En examinant la distribution d'une variable, nous pouvons rapidement identifier des tendances et des anomalies que nous n'aurons pas pu identifier en examinant l'ensemble des données. Des outils tels que les histogrammes, les diagrammes en boîte et les diagrammes de densité nous permettent de visualiser la forme, la dispersion, la centralité et les queues de la distribution, fournissant ainsi des informations cruciales pour comprendre le comportement de nos données.

### Diagramme en boîte

Le diagramme en boîte (boxplot) est un graphique utilisé pour résumer cinq paramètres importants d'une variable quantitative: la valeur minimale, le premier quartile (Q1), la médiane (ou deuxième quartile Q2), le troisième quartile (Q3) et la valeur maximale. La ligne qui divise la boîte en deux parties représente la médiane.

**Exemple 4.1 :** La base de données `tips` de la bibliothèque `seaborn` contient des informations collectées à partir des pourboires donnés par les clients dans un restaurant. Dans cette base de données, vous trouvez des variables quantitatives telles que les montants des factures (`total_bill`) et le nombre de clients (`size`), et des variables qualitatives telles que le sexe (`sexe`) et les jours (`day`). Nous allons visualiser les tendances des montants des factures au sein du restaurant pour différents jours (`day`), ainsi que la distribution générale des montants des factures (`total_bill`). Pour ce faire, nous utilisons `sns.boxplot` pour créer le boxplot pour l'ensemble des données ou en spécifiant les colonnes `day`.



#### Code Python

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

# Charger le jeu de données
tips = sns.load_dataset('tips')

# Créer une figure et un arrangement de sous-graphiques
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

# Boxplot pour les montants des pourboires par jour
sns.boxplot(y='total_bill', data=tips, ax=axes[0], palette='viridis')
axes[0].set_title('(a)')
#axes[0].set_xlabel('Jour (day)')
axes[0].set_ylabel('Montant de la facture')

# Boxplot pour les montants des pourboires par jour
sns.boxplot(x='day', y='total_bill', data=tips, ax=axes[1], palette='viridis')
axes[1].set_title('(b)')
axes[1].set_xlabel('Jour (day)')
axes[1].set_ylabel('Montant de la facture')

# Afficher les graphiques
plt.tight_layout()
plt.show()
```

Nous obtenons les graphiques suivants. Le boxplot affiche la distribution des montants des factures (`total_bill`) ou pour chaque jour de la semaine (jeudi, vendredi, samedi, dimanche) (voir Figure 4-1). Vous remarquez que la distribution des montants totaux dépend des jours de la semaine (`day`). Aussi, la dispersion des montants est plus grande le week-end (vendredi, samedi et dimanche) par rapport à un jour de semaine (jeudi).

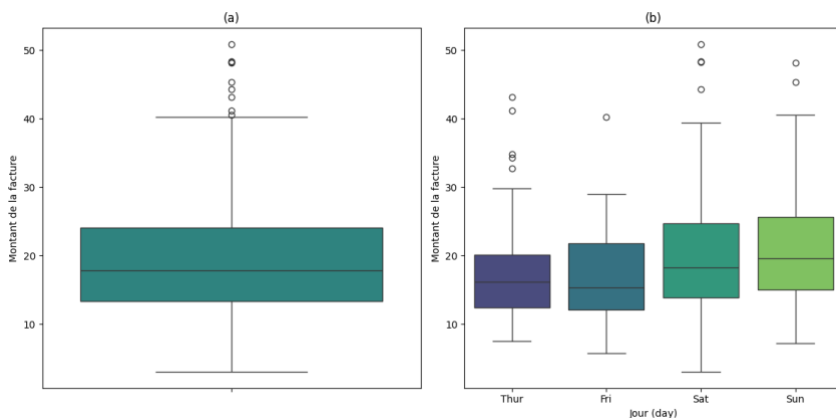


Figure 4-1: (a) Boxplot des montants des factures (*total\_bill*) et (b) Boxplot des montants des factures (*total\_bill*) par jour (*day*)

## Histogramme

Un histogramme représente une estimation de la densité d'une variable quantitative. La forme de l'histogramme est obtenue à la suite de la répartition des données selon un ensemble d'intervalles. De ce fait celle-ci peut être différente selon le nombre d'intervalles défini.

Le nombre de bins d'un histogramme a un impact important sur la visualisation de la distribution de la variable. Ainsi un nombre trop faible de bins peut masquer des caractéristiques importantes des données, tandis qu'un nombre trop élevé peut présenter trop de bruit, rendant difficile la détection de connaissance et de patterns significatifs.

Il existe deux méthodes couramment utilisées pour calculer le nombre de bins optimal:

- La règle de Sturge est une formule mathématique proposée par Herbert Sturges qui sert à découper une plage de valeurs en tranches pour en faire la description statistique. Soit un échantillon de  $N$  valeurs observées, la valeur approximative

pour le nombre de bins  $k$  en fonction de la taille  $N$  de l'échantillon :

$$k = 1 + \text{ceil}(\log_2 N)$$

où  $\log_2$  est le logarithme en base 2. La fonction `ceil` trouve la valeur entière la plus petite qui n'excède pas son paramètre.

- La règle de Freedman-Diaconis ne tient pas seulement compte de la taille de l'échantillon  $N$  mais aussi de sa dispersion :

$$k = \text{ceil}(\max(x) - \min(x) / I_w)$$

où

$$I_w = \frac{2(Q_3 - Q_1)}{\sqrt[3]{N}}$$

et où  $Q_1$  et  $Q_3$  sont respectivement le 1<sup>er</sup> et le 3<sup>e</sup> quartiles,  $N$  est le nombre d'échantillons et  $x$  est la variable à étudier.

Lors que les deux règles précédentes donnent des résultats différents, vous devez générer des histogrammes en utilisant les deux nombres de bins et les examiner visuellement. Vous pouvez par la suite choisir le nombre de bins qui donne l'histogramme le plus informatif et le plus clair sur la structure sous-jacente des données.

**Exemple 4.2 :** Dans cet exemple, nous allons déterminer le nombre de bins selon les deux règles.

Code Python

```
import seaborn as sns
import numpy as np

# Charger les données
tips = sns.load_dataset('tips')

# Extraire la colonne d'intérêt
data = tips['total_bill']

# Règle de Sturges
```

```

n_bins_sturges = int(np.ceil(np.log2(len(data)) + 1))

# Règle de Freedman-Diaconis
IQR = np.percentile(data, 75) - np.percentile(data, 25)
bin_width = 2 * IQR / len(data)**(1/3)
n_bins_fd = int(np.ceil((data.max() - data.min()) / bin_width))

# Afficher les résultats
print(f"Nombre de bins selon la règle de Sturges : {n_bins_sturges}")
print(f"Nombre de bins selon la règle de Freedman-Diaconis : {n_bins_fd}")

```

```

Nombre de bins selon la règle de Sturges : 9
Nombre de bins selon la règle de Freedman-Diaconis : 18

```

Nous obtenons en sortie deux valeurs différentes de nombres de bins qui aboutissent aux histogrammes de la 4-2. Vous remarquez que la largeur des intervalles est moins grande avec la règle de Freedman. Ceci est dû au nombre d'intervalle plus élevé (18 intervalles au lieu de 9). Pour la même raison, vous remarquez également que la fréquence est moins élevée puisque les observations sont réparties entre des intervalles voisins. La règle de Sturges donne un meilleur résultat visuellement puisque la structure est plus claire.

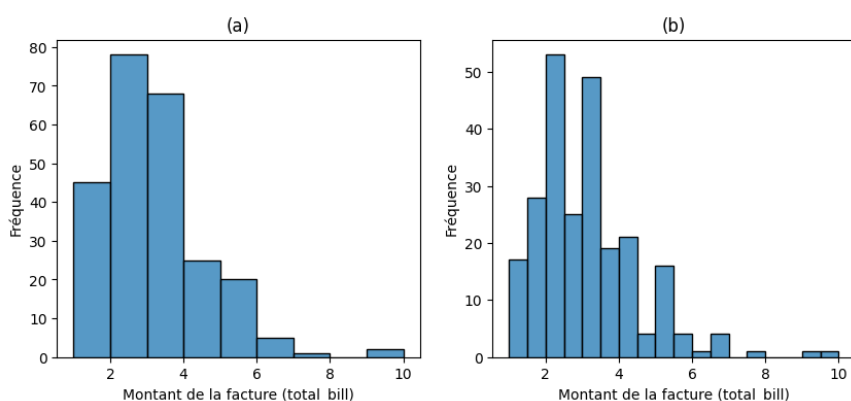


Figure 4-2: Histogrammes de la variable *Total\_bill* en se basant sur (a) la règle de Sturge et (b) la règle de Freedman-Diaconis

## Graphique de densité

Un graphique de densité est utilisé pour visualiser la distribution de probabilité d'une variable continue. Il est similaire à un histogramme, mais utilise une estimation de la densité de la probabilité pour représenter la distribution de manière plus lisse. Le graphique de densité est souvent tracé en superposant une estimation de densité de probabilité (courbe de densité) sur un histogramme ou directement comme une courbe lisse tel qu'illustré dans la Figure 4-3.

**Exemple 4.3 :** L'ajout du graphique de densité sur l'histogramme se fait facilement en spécifiant l'option `kde=True` dans la fonction `sns.histplot`

### Code Python

```
# Histogramme pour les montants des pourboires
sns.histplot(tips['total_bill'], bins=20, kde=True, ax=axes[1])
axes[1].set_title('Histogramme des montants des factures')
axes[1].set_xlabel('Montant de la facture')
axes[1].set_ylabel('Fréquence')
```

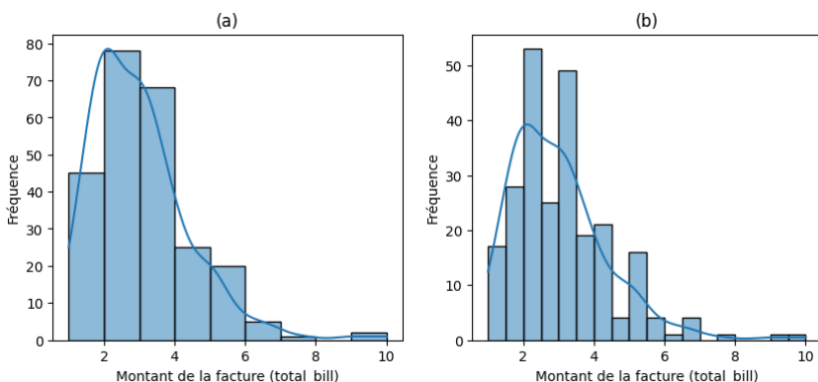


Figure 4-3: Superposition de la densité des variables sur les histogrammes

## 4.3 Visualisation de la corrélation

La visualisation de la corrélation permet de représenter graphiquement les associations entre différentes variables afin

d'identifier des tendances, des dépendances et des interactions potentielles qui peuvent échapper à une simple analyse numérique. Des outils tels que les graphiques de dispersion, les cartes de chaleur et les matrices de corrélation permettent de visualiser la force et la direction des relations entre les variables, facilitant ainsi l'identification de patterns et la prise de décisions informées.

Diagramme de dispersion

Le diagramme de dispersion est un nuage de point (*Scatterplot*) qui permet de représenter une variable numérique en fonction d'une autre variable numérique. Chaque point représente une observation. Lorsque les valeurs des observations sur l'axe des abscisses sont ordonnées, nous pouvons représenter un diagramme de dispersions connecté. Ce dernier est souvent utilisé pour les séries chronologiques où l'axe *X* représente le temps.

**Exemple 4.4 :** Pour visualiser la relation entre deux variables, nous pouvons utiliser un graphique de dispersion avec la fonction `sns.scatterplot`. La Figure 4-4 illustre le résultat.

Code Python
<pre>import seaborn as sns import pandas as pd import matplotlib.pyplot as plt  data = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6], 'C': [7, 8, 9]})  sns.heatmap(data)  plt.show()</pre>

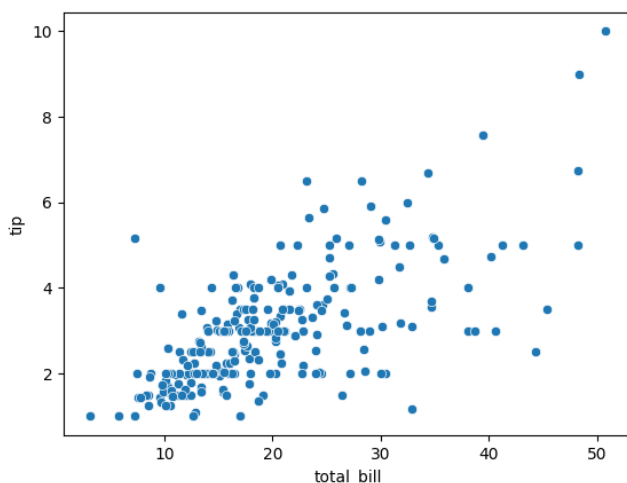


Figure 4-4 Graphique de dispersion des pourboires

### Diagramme de paires

Un diagramme de paires est une grille de nuages de points qui permet de comprendre la relation entre différentes. Chaque élément de la grille présente un nuage de point avec une différente variable sur l'axe des y et sur l'axes des x. Les graphiques diagonaux sont utilisés pour présenter la distribution des données.

**Exemple 4.5 :** Pour créer un diagramme de paires, nous utilisons la fonction `sns.pairplot` avec le jeu de données `tips`. La Figure 4-4 illustre le résultat. Sur la diagonale, on constate une certaine corrélation entre le pourboire (`tip`) et le montant de la facture (`total_bill`).

Code Python

```
import seaborn as sns
```

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

```
tips = sns.load_dataset("tips")
```

```
sns.pairplot(tips)
```

```
plt.show()
```

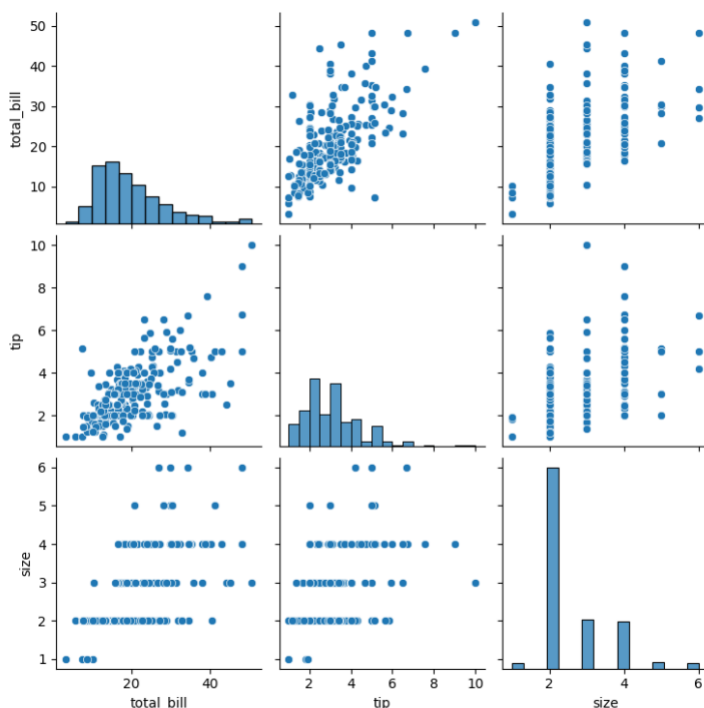


Figure 4-5 Diagramme de paires

#### Exemple 4.6: La

Figure 4-6 est un diagramme de paires portant sur les données IRIS (Longueur des sépales, Largeur des sépales, Longueur des pétales. Et Largeur des pétales). Sur la diagonale, vous observez l'histogramme de chaque caractéristique selon l'espèce de fleur. Les graphiques hors diagonale montrent des nuages de points pour chaque paire de caractéristiques, colorés selon l'espèce. Ces graphiques permettent de



visualiser les relations entre les différentes paires de caractéristiques. En occurrence, la case de la relation Longueur des sépales vs. Largeur des sépales, vous permet de voir si les sépales sont proportionnellement plus longs ou plus larges pour certaines espèces.

Code Python

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd # Import pandas
from sklearn.datasets import load_iris

iris = load_iris()
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['species'] = iris.target_names[iris.target]
pairplot = sns.pairplot(iris_df, hue='species', markers=["o", "s", "D"], palette='viridis')
pairplot.fig.suptitle('Pairplot des caractéristiques des données Iris par espèce', y=1.05)
plt.show()
```

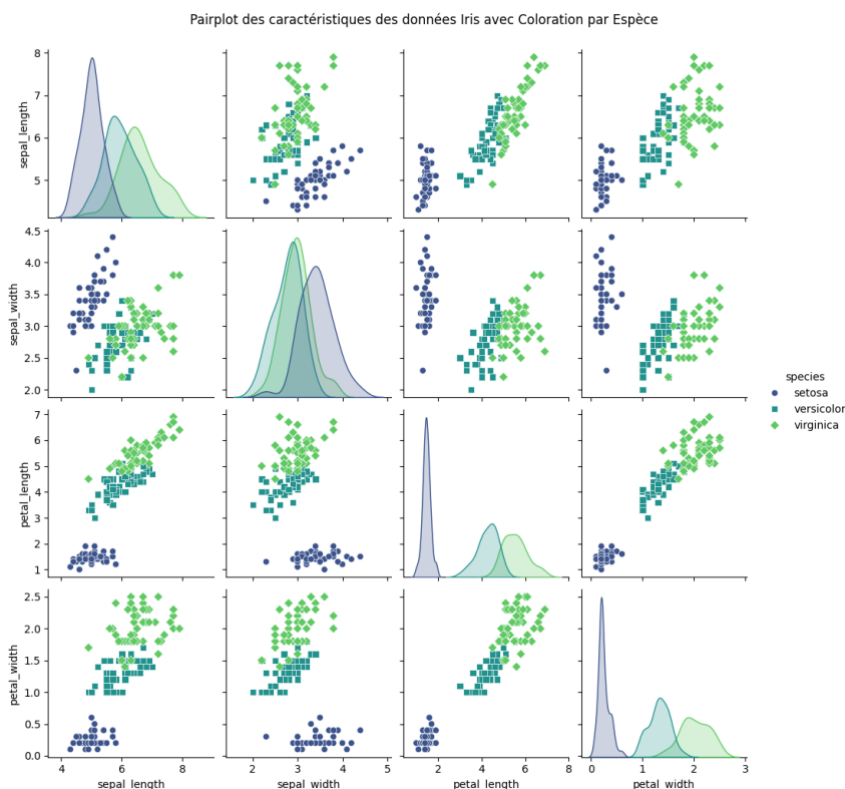


Figure 4-6 Diagramme de paires de l'ensemble de données IRIS

### Un graphique à bulles

Le graphique à bulles vient ajouter des connaissances au diagramme de dispersion. En effet, habituellement, chaque point représente une observation décrite par la variable y en fonction d'une variable x. Le diagramme à bulle représente l'observation sous forme de bulles où la taille de chaque bulle représente une troisième variable. Ainsi, il permet de visualiser des relations entre trois variables en même temps, en montrant à la fois la relation entre les variables sur les axes x et y, ainsi que la relation entre ces variables et la troisième variable représentée par la taille des bulles. Les graphiques à bulles sont

couramment utilisés dans l'analyse des données commerciales, la visualisation des données financières, etc.

**Exemple 4.7 :** Nous continuons avec la base de données `tips` qui contient des informations collectées à partir des pourboires donnés par les clients dans un restaurant. Dans cet exemple, nous allons représenter le pourboire `tip` en fonction du total des factures `total_bill`. Nous allons ajouter le nombre de personne (`size`). Pour ce faire, nous utilisons `sns.boxplot` pour créer le boxplot en spécifiant les colonnes `day` et `total_bill`. Nous utilisons `sns.scatterplot` en spécifiant les colonnes `total_bill` pour l'axe des abscisses, `tip` pour l'axe des ordonnées et '`size`' pour la taille des bulles.

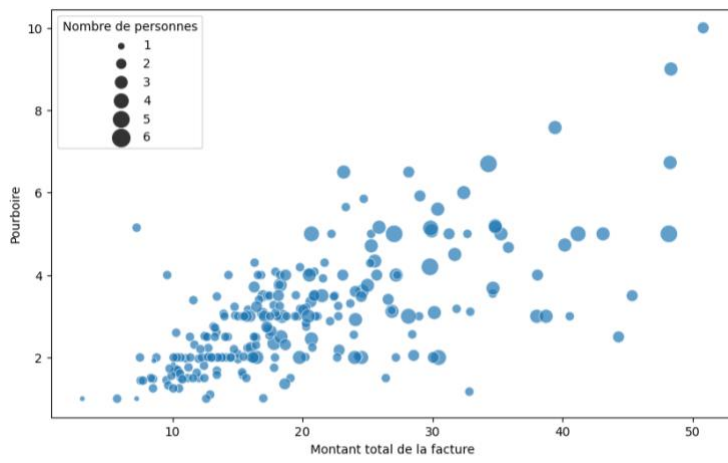


Figure 4-7 Graphique à bulles des pourboires

### Diagramme de chaleur

Le diagramme de chaleur (ou diagramme thermique) est utilisé pour visualiser la relation entre deux variables en affichant une matrice de valeurs sous forme de grille colorée. Les diagrammes de chaleur sont

couramment utilisés dans l'analyse de données, la visualisation de données géospatiales et la cartographie de la chaleur.

**Exemple 4.8 :** Pour créer un diagramme de chaleur, nous utilisons la fonction `sns.heatmap`. La Figure 4-8 donne un exemple de diagramme de chaleur. La carte de chaleur obtenue donne le nombre de passagers pour les vols de l'ensemble de données « flights ». Chaque cellule représente le nombre de passagers pour une combinaison année-mois spécifique. L'intensité de la couleur indique le nombre de passagers, les couleurs chaudes représentant des valeurs plus élevées. Les annotations indiquent le nombre exact de passagers.

Code Python

```
import seaborn as sns
import matplotlib.pyplot as plt
data = sns.load_dataset("flights")
data_matrix = data.pivot_table(index="month", columns="year", values="passengers")
plt.figure(figsize=(10, 6))
sns.heatmap(data_matrix, annot=True, cmap="coolwarm", fmt=".1f", linewidths=0.5)
plt.title("Passagers mensuels des compagnies aériennes en fonction de l'année")
plt.xlabel("Année")
plt.ylabel("Mois")
plt.show()
```

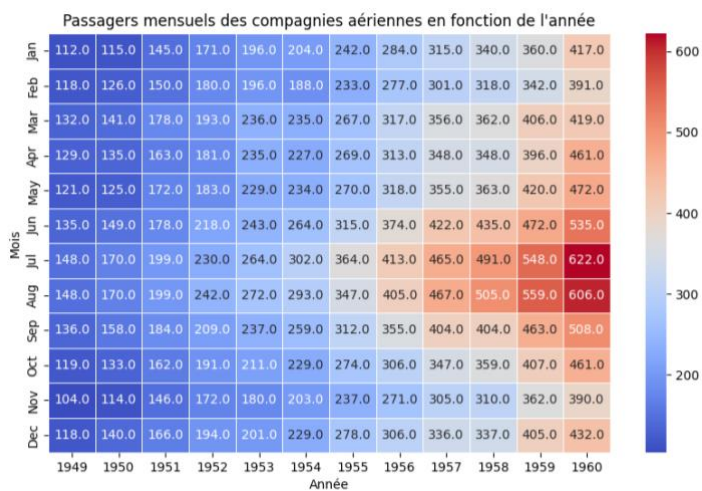


Figure 4-8 Diagramme de chaleur

# Chapitre 5 : Inférence statistique avec R

Neila Mezghani

## 5.1 Introduction

La statistique inférentielle (aussi appelée inférence statistique) est le processus de raisonnement par lequel on tire, à partir des observations réalisées sur un échantillon prélevé dans une population préalablement définie (population de référence), des conclusions concernant certaines caractéristiques quantitatives ou qualitatives de cette population. Le synonyme populaire de l'inférence statistique utilisé en intelligence artificielle est la « généralisation ».

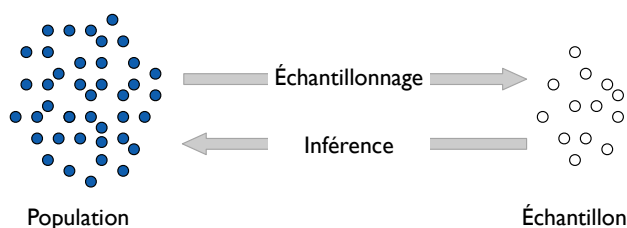
### 5.1.1 Terminologie

La définition de l'inférence statistique fait référence à une terminologie spécifique qui mérite d'être précisée.

- Population : Une population ou « population mère », est l'ensemble de toutes les entités (données ou individus) concernées par une étude particulière.
- Échantillon : Un échantillon est un sous-ensemble sélectionné de la population, représentatif de cette dernière.
- Individu : Un individu est un élément unique au sein de la population étudiée.

**Exemple 5.1 :** Vous désirez réaliser une étude sur le niveau de satisfaction des étudiants dans une grande université. Dans ce cas, la population cible serait l'ensemble de tous les étudiants inscrits à cette université. Cela comprend les étudiants de toutes les années d'études,

de différentes filières et de différentes nationalités. Étant donné que l'interrogation de chaque étudiant de l'université serait trop coûteuse et chronophage, vous opteriez pour un échantillon qui serait une partie plus petite mais représentative de la population totale. Cet échantillon pourrait être constitué d'un groupe d'étudiants sélectionnés aléatoirement, de manière à refléter la diversité présente au sein de l'université (incluant divers niveaux d'études et différentes filières et nationalités). Ainsi, chaque étudiant inclus dans cet échantillon représente un individu au sein de l'étude.



*Figure 5-1: Population et échantillon*

Les notions de populations et d'échantillons nous conduisent à distinguer deux autres concepts clés : les paramètres et les statistiques (voir Figure 5-1). Les paramètres sont des valeurs qui caractérisent une population entière. Ils sont souvent inconnus, car il est généralement impraticable, voire impossible, de mesurer chaque individu dans une population. Les paramètres incluent, par exemple, la moyenne de la population ( $\mu$ ), l'écart-type de la population ( $\sigma$ ) et la proportion de la population ( $\pi$ ).

En revanche, les statistiques sont des valeurs qui résument les caractéristiques d'un échantillon, c'est-à-dire un sous-ensemble d'une population. Les statistiques sont utilisées pour estimer les paramètres

de la population, car il est souvent irréalisable de mesurer l'intégralité de celle-ci. De ce fait, les statistiques peuvent varier d'un échantillon à l'autre, même lorsqu'ils proviennent de la même population, en raison de la variabilité d'échantillonnage. Les statistiques comprennent, par exemple, la moyenne de l'échantillon ( $\hat{\mu}$ ), l'écart-type de l'échantillon ( $\hat{\sigma}$ ) et la proportion de l'échantillon ( $\hat{\pi}$ ).

### 5.1.2 Objectif de l'inférence statistique

L'objectif de l'inférence statistique est de permettre une estimation précise des caractéristiques (paramètres) d'une population entière en utilisant les données issues d'un échantillon représentatif de cette population. En d'autres termes, l'inférence statistique vise à combler le fossé entre ce que nous pouvons observer directement à partir d'un échantillon limité et ce que nous souhaitons connaître sur la population globale.

La loi des grands nombres est un théorème mathématique fondamental en probabilités et statistiques. Dans le contexte de l'inférence statistique, cette loi stipule que les caractéristiques d'un échantillon aléatoire tendent à se rapprocher des caractéristiques réelles de la population à mesure que la taille de l'échantillon augmente. Ainsi, la loi des grands nombres justifie l'utilisation de statistiques d'échantillonnage (telles que la moyenne d'un échantillon) pour estimer les paramètres d'une population (comme la moyenne de cette population). En pratique, cela signifie que les estimations et les prédictions basées sur de grands échantillons sont généralement plus fiables que celles issues de petits échantillons.



Afin de réaliser ces estimations, l'inférence statistique repose sur trois grandes lignes ou principes fondamentaux : l'estimation ponctuelle, l'estimation par intervalle de confiance et les tests d'hypothèse. Ces trois principes feront l'objet de la suite de ce chapitre (Sections 2 à 4).

## **5.2 Domaines d'application de l'inférence statistiques**

La statistique inférentielle est un pilier essentiel de la science des données. Elle offre les outils et les techniques nécessaires pour analyser, interpréter et tirer des conclusions fiables à partir d'échantillons de données, plutôt qu'avec l'ensemble de la population. Elle joue un rôle crucial dans de nombreux domaines tels que :

- Médecine et épidémiologie : Dans le domaine de la santé, l'inférence statistique aide, à étudier les causes des maladies, à comprendre les facteurs de risque et à évaluer l'efficacité des traitements proposés aux patients.
- Sciences sociales : En sociologie, psychologie, et sciences politiques, l'inférence statistique permet d'analyser des enquêtes, des études comportementales, et des données démographiques pour comprendre les comportements et attitudes de l'humain.
- Marketing et recherche de marché : Elle est utilisée pour comprendre les préférences et comportements de la population des consommateurs.
- Éducation : Dans le domaine de l'éducation, l'inférence statistique aide à évaluer l'efficacité des méthodes

pédagogiques, à comprendre les facteurs influençant les performances de la population des étudiants.

- Environnement et écologie : En écologie l'inférence statistique est utilisée pour analyser les données environnementales, étudier la biodiversité, et évaluer l'impact des activités humaines sur l'environnement.
- Gouvernement et politique publique : L'inférence statistique aide à formuler des politiques publiques basées sur des données, à évaluer l'efficacité des programmes gouvernementaux et à mener des enquêtes sur la population.

### 5.3 Estimation ponctuelle

L'estimation ponctuelle est une méthode statistique qui consiste à utiliser une seule valeur, dérivée des données d'un échantillon, pour estimer un paramètre inconnu d'une population. Cette valeur unique est appelée « estimateur ponctuel » et vise à fournir la meilleure estimation possible du paramètre réel de la population. Si l'échantillon utilisé est de bonne qualité, l'estimateur ponctuel représente fidèlement la population.

L'estimation ponctuelle dépend du type du paramètre à estimer. Elle représente, sur la base des données observées dans un échantillon, la valeur la plus plausible du paramètre de la population, comme la moyenne, la médiane, la proportion, ou l'écart-type.

**Exemple 5.2 :** Nous désirons avoir une estimation du poids de la population des bébés québécois à la naissance. Pour obtenir ce paramètre par estimation ponctuelle, nous devons sélectionner un échantillon de bébés au moyen d'une technique aléatoire (ex. : 200

bébés) et calculer la moyenne de leur poids. Cette moyenne, noté  $\hat{\mu}$ , est un « estimateur ponctuel » du poids de la population des bébés québécois.

Un autre exemple, imaginons une enquête visant à déterminer une estimation de la proportion d'étudiants universitaires qui possèdent un ordinateur portable. Dans ce cas, nous considérons un échantillon de 200 étudiants et nous déterminons le nombre d'étudiants qui possèdent un ordinateur portable. L'estimateur ponctuel de la proportion (noté  $\hat{\pi}$ ) est calculé en divisant le nombre d'étudiants possédant un ordinateur portable par le nombre total d'étudiants dans l'échantillon.

### 5.3.1 Formalisation

Soit  $(x_1, x_2, \dots, x_n)$ , un échantillon aléatoire de taille  $n$  issu de la population. Nous souhaitons estimer un paramètre  $\theta$  d'une population à partir de cet échantillon. L'estimateur ponctuel  $\hat{\theta}$  peut-être sa moyenne  $\hat{\mu}$ , son écart-type  $\hat{\sigma}$  ou une proportion  $\hat{\pi}$ .

#### Estimateur de la moyenne

L'estimateur ponctuel de la moyenne de la population, noté  $\hat{\mu}$ , est défini comme la moyenne arithmétique de ces valeurs d'échantillon

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^n x_i.$$

#### Estimateur de la variance

La variance de l'échantillon, notée  $\hat{\sigma}^2$ , est un estimateur ponctuel de la variance de la population (notée  $\sigma^2$ ). Elle est calculée comme suit :

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \hat{\mu})^2.$$

### Estimateur d'une proportion

Dans le cas d'une variable qualitative, l'estimateur ponctuel de la proportion  $\hat{\pi}$  pour chaque modalité  $A_i$  est égal à :

$$\hat{\pi} = \frac{n_{A_i}}{n}$$

Où  $n_{A_i}$  est le nombre d'individus de l'échantillon de taille  $n$  qui possèdent la modalité (caractéristique)  $A_i$ .

**Exemple 5.3 :** La base de données mtcars est un ensemble de données classique en R, souvent utilisée pour des démonstrations et des analyses statistiques. mtcars contient des données sur les performances de 32 modèles de voitures, principalement des modèles nord-américains et européens, qui étaient populaires en 1973-1974. Les voitures sont décrites par plusieurs variables dont mpg (Miles/US gallon) qui est une mesure de l'efficacité énergétique en termes de distance parcourue par unité de carburant.

Le script suivant permet d'estimer la moyenne et la variance de la variable mpg.

```
Code R

data("mtcars")
# Estimation de la moyenne
sample.mean <- mean(mtcars$mpg)
sprintf("Estimation de la moyenne est: %f", sample.mean)

# Estimation de la variance
sample.sd <- sd(mtcars$mpg)
sprintf("Estimation de la variance est: %f", sample.sd)
```

Ces lignes de code renvoient les valeurs numériques suivantes qui correspondent à la valeur de  $\hat{\mu}$  et de  $\hat{\sigma}^2$ .

[1] "Estimation de la moyenne est: 20.090625"

[1] "Estimation de l'écart type est: 1.065424"

## 5.4 Inconvénients de l'estimation ponctuelle

Bien que largement utilisée en inférence statistique, l'estimation ponctuelle présente certains inconvénients :

- (1) Sensibilité aux valeurs extrêmes : les estimations ponctuelles peuvent être fortement influencées par des valeurs extrêmes. Cela peut conduire à des estimations qui ne sont pas représentatives de la population globale.
- (2) Manque de précision : un échantillon donné n'est pas toujours une image parfaitement fidèle de la population. De ce fait, en faisant une estimation ponctuelle, nous attribuons une valeur précise à un paramètre mais nous courrons le risque que la valeur attribuée soit éloignée de la réalité.
- (3) Dépendance de l'échantillon : l'estimation ponctuelle issue d'un échantillon donné peut être considérablement différente d'un échantillon à un autre .

Ainsi, au lieu de fournir une estimation ponctuelle, nous pouvons construire un intervalle de valeurs dans lequel se situerait la vraie valeur du paramètre avec une certaine probabilité d'y être.

## 5.5 Estimation par intervalle

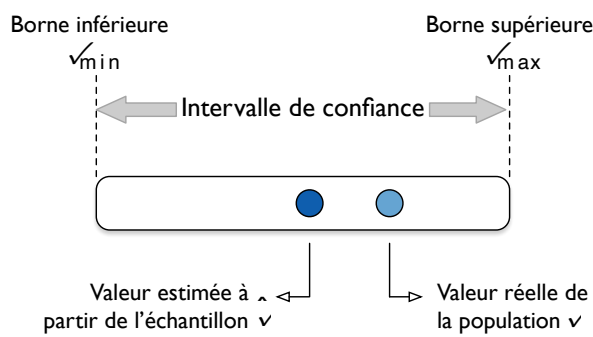
Un intervalle de confiance est une plage de valeurs calculée à partir des données d'un échantillon, qui est susceptible de contenir la valeur vraie d'un paramètre de la population. Cet intervalle est généralement exprimé avec une limite inférieure et une limite supérieure,

fournissant ainsi une estimation de la variabilité et de la précision de l'estimation.

L'intervalle de confiance donne non seulement une idée de l'estimation centrale du paramètre (comme la moyenne ou la proportion), mais aussi de la marge d'erreur associée à cette estimation. Il est souvent accompagné d'un niveau de confiance, indiquant la probabilité que l'intervalle contienne effectivement le paramètre de population estimé.

### 5.5.1 Formalisation

Soit  $\hat{\theta}$  une estimation ponctuelle de  $\theta$ , basée sur les données  $(x_1, x_2, \dots, x_n)$  de l'échantillon. Il est plus pertinent de fournir un intervalle  $\theta_{min} < \hat{\theta} < \theta_{max}$  plutôt que simplement une estimation ponctuelle. Un tel intervalle  $]\theta_{min}; \theta_{max}[$  s'appelle une estimation par intervalle de confiance du paramètre  $\theta$ . L'intervalle de confiance est associé à un niveau de confiance  $1 - \alpha$ , exprimé en pourcentage, qui indique le degré de certitude que le paramètre réel d'une population se situe à l'intérieur de l'intervalle de confiance. Pour souligner la dépendance entre l'intervalle de confiance et le niveau de confiance, l'intervalle de confiance est dénoté  $IC_{1-\alpha}(\theta)$ .



**Exemple 5.4 :** On s'intéresse au nombre de résidents permanents au Québec. Si ce nombre est à 50 000 personnes ( $\hat{\theta} = 50\,000$ ), l'intervalle de confiance peut être représenté par deux valeurs : 49 500 ( $\theta_{min}$ ) et 52 500  $\theta_{max}$ . Cet intervalle englobe la plage de valeurs possibles et exprime ainsi non seulement l'estimation centrale mais également l'incertitude associée à cette estimation.

### 5.5.2 Bornes de l'intervalles de Confiance

Les bornes de l'intervalle de confiance varient en fonction de la statistique que vous examinez (par exemple, la moyenne, la variance, la proportion, etc.) et du niveau de confiance choisi.

#### Intervalle de confiance de la moyenne

Les bornes de l'intervalle de confiance de la moyenne sont calculées en ajoutant et en soustrayant la marge d'erreur à la moyenne de l'échantillon. Cette marge d'erreur est déterminée par le niveau de confiance, la taille de l'échantillon, et l'écart-type. Si vous connaissez l'écart-type de la population ( $\sigma$ ), vous l'utilisez directement. Sinon, vous pouvez utiliser l'écart-type de l'échantillon ( $\hat{\sigma}$ ).

R vous permet d'obtenir facilement l'IC de la moyenne à partir d'un ensemble de données. A cet effet, vous pouvez utiliser la fonction `t.test()` pour des échantillons de petite taille ou lorsque la distribution n'est pas connue pour être normale.

**Exemple 5.5 :** Nous allons continuer avec l'ensemble de données `mtcars`, plus spécifiquement la variable `mpg` (Miles/US gallon). Pour calculer l'intervalle de confiance pour la moyenne, nous proposons les

lignes de codes suivants. Dans cet exemple le niveau de confiance est fixé à 0.95 à travers la variable `conf.level`.

Code R
<pre># Calcul de l'intervalle de confiance pour la moyenne resultat &lt;- t.test(mtcars\$mpg, conf.level = 0.95) # 95% par défaut resultat\$conf.int</pre>

Ces lignes de code renvoient les valeurs numériques 17.9 et 22.2 qui indiquent qu’avec un niveau de confiance de 95%, nous sommes assez sûrs que la moyenne réelle de l’efficacité énergétique `mpg` dans la population générale se situe quelque part entre 17.9 et 22.2 (Miles/US galon).

<pre>[1] 17.91768 22.26357 attr(,"conf.level") [1] 0.95</pre>
---

La Figure 5-2 illustre graphiquement la notion d’intervalle de confiance. L’IC de confiance pour la moyenne d’un échantillon illustre la plage dans laquelle la vraie moyenne de la population est estimée se situer avec un certain niveau de confiance. Les trois régions mentionnées représentent environ 68%, 95% et 99% des données, correspondant aux écarts-types dans une distribution normale.

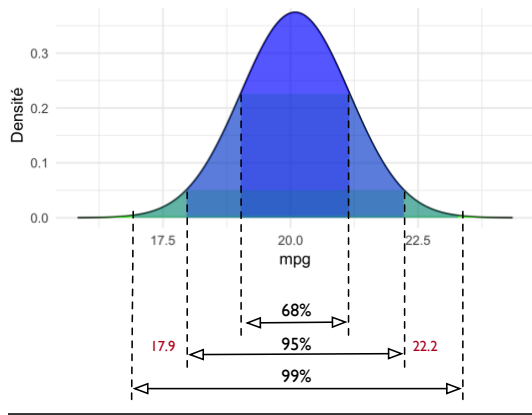


Figure 5-2: Distribution de la variable `mpg`



*et bornes de l'intervalle de confiance de sa moyenne*

Si vous choisissez un niveau de confiance de 95%, alors l'intervalle de confiance sera défini par les bornes inférieure et supérieure 17.9 et 22.2, indiquant que la moyenne réelle de la population est estimée se situer dans cet intervalle avec 95% de confiance.

### Intervalle de confiance de l'écart type

Le calcul de l'intervalle de confiance pour la variance d'une population à partir d'un échantillon se fait généralement en utilisant la distribution du Chi-carré (en anglais, Chi-square noté  $\chi^2$ ).

Les fonctions `qchisq(1 - alpha/2, df)` et `qchisq(alpha/2, df)` sont utilisées pour trouver les quantiles de la distribution du Chi-carré, nécessaires pour déterminer les limites de l'intervalle de confiance.

**Exemple 5.6 :** Pour calculer l'intervalle de confiance de la variance mpg (Miles/US gallon) de l'ensemble de données `mtcars`, nous proposons les lignes de codes suivants. Dans cet exemple le niveau de confiance  $\alpha$  est fixé à 0.05.

#### Code R

```
# Nombre d'observations
n <- length(mtcars$mpg)
# Calcul de la variance de l'échantillon
var_echantillon <- var(mtcars$mpg)
# Degrés de liberté
df <- n - 1
# Niveau de confiance
alpha <- 0.05
# Bornes inférieure et supérieure de l'intervalle de confiance
borne_inf <- (df * var_echantillon) / qchisq(1 - alpha/2, df)
borne_sup <- (df * var_echantillon) / qchisq(alpha/2, df)
# Affichage de l'intervalle de confiance
cat("L'intervalle de confiance à 95% pour la variance est \n [", borne_inf, ", ",
borne_sup, "]\n")
```

Ces lignes de code renvoient les valeurs numériques suivantes qui signifient qu'avec un niveau de confiance de 95%, vous pouvez être assez sûr que l'écart type réel de la population dont provient votre échantillon se situe entre 23.3 et 64.2.

L'intervalle de confiance à 95% pour la variance est  
[23.3465, 64.20343]

### Intervalle de confiance de la proportion

Si vous avez des données catégorielles et souhaitez calculer un intervalle de confiance pour une proportion, vous pouvez utiliser la fonction `prop.test()`. Cette fonction s'ajuste automatiquement pour de petites tailles d'échantillons et d'autres facteurs.

**Exemple 5.7 :** Nous avons réalisé dans l'exemple 5.1 une enquête visant à déterminer une estimation de la proportion d'étudiants universitaires qui possèdent un ordinateur portable. Dans ce cas, nous avons considéré un échantillon de 200 étudiants. Si parmi ces 200 étudiants nous avons identifié 120 étudiants qui possèdent un ordinateur portable. Dans ce cas, l'intervalle de confiance est donné par :

```
Code R

computers <- 120
students <- 200

# Calcul de l'intervalle de confiance pour la proportion
resultat <- prop.test(computers, students, conf.level = 0.95)
resultat$conf.int
```

Vous obtenez, alors, les valeurs numériques suivantes. Cet intervalle de confiance indique que, avec un niveau de confiance de 95%, vous pouvez être sûr que la véritable proportion de toutes les personnes qui

« possèdent un ordinateur » dans la population générale se situe entre 52.8% et 66.7%.

```
[1] 0.5283160 0.6677775  
attr(,"conf.level")  
[1] 0.95
```

## 5.6 Tests d'hypothèse

Le test d'hypothèse, également connu sous le nom de test statistique, est une hypothèse sur un paramètre de la population. Il joue un rôle crucial en statistique, agissant comme un complément à l'estimation ponctuelle. Après avoir obtenu une estimation d'un paramètre d'intérêt, il devient essentiel de comparer cette estimation à des valeurs de référence ou de comparer différentes populations en termes de ce paramètre.

Un test d'hypothèse est une procédure qui permet de choisir entre deux hypothèses complémentaires : l'hypothèse nulle, notée  $H_0$ , et l'hypothèse alternative, notée  $H_1$ . L'hypothèse nulle est souvent considérée comme l'hypothèse principale.

**Exemple 5.8 :** Considérons un exemple simple pour introduire les notions d'hypothèse nulle et d'hypothèse alternative. Imaginons que vous ayez besoin de vous rendre à la banque. Selon votre mémoire, vous pensez que la succursale devrait se trouver sur votre droite. Cette supposition, que vous considérez a priori comme vraie, est appelée l'hypothèse nulle, notée  $H_0$ . En revanche, l'hypothèse selon laquelle la succursale pourrait se trouver sur votre gauche est l'hypothèse alternative, notée  $H_1$ . L'objectif est alors de rechercher des indices ou repères autour de vous pour confirmer ou infirmer l'hypothèse nulle  $H_0$  : « La banque est à droite ». Si vous trouvez suffisamment

d'éléments indiquant que cette hypothèse est incorrecte, vous rejetez  $H_0$  au profit de  $H_1$ , concluant ainsi que la banque est à gauche.

La démarche adoptée dans un test d'hypothèse consiste à décider de « Rejeter » ou de « Ne pas rejeter » l'hypothèse nulle  $H_0$  en fonction de l'analyse des résultats obtenus à partir de l'échantillon de données. Quatre possibilités, détaillées dans le tableau suivant, sont envisageables:

	Décision	
	Ne pas rejeter $H_0$	Rejeter $H_0$
$H_0$ vraie	Décision correcte	Erreur de type I ( $\alpha$ )
$H_0$ fausse	Erreur de type II ( $\beta$ )	Décision correcte

Ici,  $\alpha$  représente le risque d'erreur de première espèce, c'est-à-dire le risque de rejeter à tort l'hypothèse nulle  $H_0$  lorsqu'elle est en réalité vraie ( $H_0$  vraie). À l'opposé,  $\beta$  représente le risque d'erreur de deuxième espèce, qui est le risque de ne pas rejeter  $H_0$  alors que l'hypothèse alternative  $H_1$  est vraie. Ces deux types d'erreurs sont antagonistes : plus le risque d'erreur  $\alpha$  est grand, plus le risque d'erreur  $\beta$  est petit, et vice versa. Idéalement, nous souhaiterions que ces deux erreurs soient nulles, mais cela n'est pas possible dans la pratique. Par conséquent, les décisions sont prises de manière à limiter l'erreur de première espèce  $\alpha$ . Le niveau de signification, souvent désigné par  $\alpha$  (alpha), est un seuil prédéterminé pour décider si un résultat est statistiquement significatif. Il représente la probabilité de

commettre une erreur de type I, c'est-à-dire de rejeter à tort l'hypothèse nulle.

Dans la pratique,  $\alpha$  est souvent fixé à 0,05 ou 5%, ce qui signifie qu'il y a 5% de chance de rejeter à tort l'hypothèse nulle.

### 5.6.1 Étapes d'un test d'hypothèse

Les tests d'hypothèses impliquent une série d'étapes qui aident à déterminer s'il y a suffisamment de preuves pour soutenir ou rejeter une hypothèse. Ces étapes peuvent être classées en quatre catégories :

#### Formuler les hypothèses.

La première étape du test d'hypothèse consiste à formuler l'hypothèse nulle et l'hypothèse alternative. L'hypothèse nulle suppose généralement qu'il n'y a pas de différence significative entre deux variables, tandis que l'hypothèse alternative suggère la présence d'une relation ou d'une différence.

La formulation de l'hypothèse vous amènera à choisir une des deux possibilités:

- Test d'hypothèse unilatéral lorsque la zone de rejet du test, c'est-à-dire la zone où l'on rejette l'hypothèse nulle, se trouve d'un seul côté de la distribution de l'échantillon. Par exemple si vous cherchez à savoir si un paramètre est supérieur (ou inférieur) à un autre ou à une valeur donnée  $\theta_0$  :

$$\begin{cases} H_0: & \theta = \theta_0 \\ H_1: & \theta \neq \theta_0 \end{cases}$$

- Un test d'hypothèse bilatéral lorsque les régions de rejet sont placées des deux côtés de la distribution de l'échantillon. Par

exemple, si vous cherchez une différence entre deux paramètres, ou entre un paramètre et une valeur donnée  $\theta_0$  sans se préoccuper du sens de la différence.

$$\begin{cases} H_0: \theta = \theta_0 \\ H_1: \theta < \theta_0 \end{cases} \quad \text{ou bien} \quad \begin{cases} H_0: \theta = \theta_0 \\ H_1: \theta > \theta_0 \end{cases}$$



Le choix de l'hypothèse nulle fait généralement intervenir un signe d'égalité. Ce choix permet une analyse statistique claire et objective, où l'on peut calculer la probabilité d'observer les données de l'échantillon si l'hypothèse nulle est vraie.

### Choisir une statistique

Une statistique est une fonction des variables aléatoires représentant l'échantillon. Le choix de la statistique pour un test d'hypothèse dépend de plusieurs facteurs clés relatifs à vos données et de l'objectif de votre étude. Parmi ces facteurs nous citons :

- **Type de Données:** Le type des données est le facteur principal. Pour les données quantitatives (comme les tailles ou les poids), des tests tels que le test t ou l'ANOVA sont utilisés. Pour les données catégorielles (comme les sexes ou les catégories professionnelles), le test du Chi-carré peut convenir.
- **Distribution des Données :** La distribution des données est importante pour choisir la statistique. Pour les données qui suivent une distribution normale, utilisez des tests paramétriques (tels que le test t pour des échantillons indépendants ou appariés, ou l'ANOVA). Si les données ne suivent pas une distribution normale, les tests non

paramétriques (tels que le test de Mann-Whitney, le test de Wilcoxon, ou le test de Kruskal-Wallis) sont préférés.

- **Taille de l'échantillon** : Les tests paramétriques sont généralement robustes pour de grands échantillons, même si les données ne suivent pas parfaitement une distribution connue.

### Définir le niveau de signification

Comme mentionné précédemment, un test d'hypothèse ne peut jamais rejeter l'hypothèse nulle avec une certitude absolue. Il existe toujours une certaine probabilité d'erreur que l'hypothèse nulle soit rejetée alors qu'en réalité, elle est vraie. Cette probabilité d'erreur est appelée le niveau de signification. Autrement dit, le niveau de signification est utilisé pour décider si l'hypothèse nulle doit être rejetée ou non. À ce niveau de signification correspond un seuil de signification, qui est le point de coupure ou la limite de la valeur  $p$  dans un test statistique, déterminant si un résultat est statistiquement significatif.

En recherche scientifique, un seuil de signification de 5 % (0,05) ou de 1 % (0,01) est généralement fixé. Ainsi, si un seuil de signification de 5 % est établi, cela signifie qu'il y a 5 % de chances de rejeter à tort l'hypothèse nulle. Il est à noter que la valeur de 5 % est communément utilisée, bien qu'elle ne repose sur aucune justification précise. Néanmoins, cette probabilité doit être adaptée au problème posé. Par exemple, pour prouver définitivement l'efficacité d'un vaccin lors d'un essai de grande envergure, des conditions plus strictes sont imposées, avec un seuil  $p < 1\%$  (voire plus faible). En revanche, lors d'un essai sur une maladie rare, sans traitement curatif, où l'on cherche

à obtenir une idée (plus ou moins objective) de l'efficacité d'un traitement, un seuil  $p < 10 \%$  peut être choisi.

### 5.6.1.1 Calcul de la statistique et prise de décision

Une fois, la statistique déterminée et le niveau de signification choisi, il s'agira de calculer à partir des données fournies la valeur de la statistique  $p$ . La décision concernant l'hypothèse posée et l'interprétation des résultats sont alors réalisées en fonction de la valeur de  $p$  :

$$\begin{cases} \text{Si } p > \alpha & \text{alors rejeter } H_0 \\ \text{Sinon} & \text{ne pas rejeter } H_0 \end{cases}$$

La Figure 5-3 illustre graphiquement les régions de rejet (régions critiques) des tests d'hypothèse. Ces régions dépendent du type de test, qu'il s'agisse d'un test unilatéral ou bilatéral. Si la statistique de test se trouve dans la région critique (c'est-à-dire, si elle est plus extrême que les valeurs critiques), alors vous devez rejeter l'hypothèse nulle. Si la statistique de test ne se situe pas dans la région critique, alors vous ne rejetez pas l'hypothèse nulle.

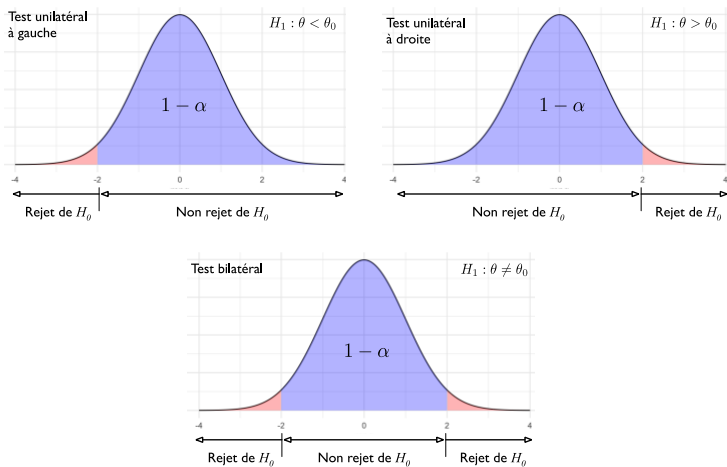




Figure 5-3: Régions de rejet ou de Non-rejet de  $H_0$



Souvent, nous disons, si  $p > \alpha$  alors nous rejetons l'hypothèse nulle  $H_0$ . Sinon, nous nous contentons de ne pas la rejeter l'hypothèse nulle mais nous ne pourrons jamais dire que nous l'acceptons.

**Exemple 5.9 :** Soit la base de données « Histoire de Vie » réalisée en France en 2003 par l'INSEE. Il comprend des données de 2000 personnes avec 20 variables différentes. Ce jeu de données est présenté sous la forme d'un dataframe avec 2000 lignes et 20 colonnes, fournissant une ressource précieuse pour l'analyse statistique et l'inférence dans des contextes variés tels que les sciences sociales et l'économie.

Supposons que vous voulez tester si l'âge moyen des individus dans l'échantillon (`hdv2003$age`) est supérieur à 35 ans.

Formulation des hypothèses :

- Hypothèse nulle ( $H_0$ ) : L'âge moyen est de 35 ans ( $\mu=35$ ).
- Hypothèse alternative ( $H_1$ ) : L'âge moyen est supérieur à 35 ans ( $\mu>35$ ).

## Calcul de la statistique :

### Code R

```
library(questionr)
data("hdv2003")
d <- hdv2003
# Hypothèse nulle: L'âge moyen est de 35 ans
mu0 <- 35

# Effectuer un test t unilatéral
t.test(hdv2003$age, mu = mu0, alternative = "greater")
```

Le paramètre `alternative = "greater"` indique que c'est un test unilatéral où l'hypothèse alternative est que la moyenne est plus grande que l'hypothèse nulle. L'exécution de ce code, fournit le résultat suivant :

### One Sample t-test

```
data: hdv2003$age
t = 34.731, df = 1999, p-value < 2.2e-16
alternative hypothesis: true mean is greater than 35
95 percent confidence interval:
 47.53359      Inf
sample estimates:
mean of x
 48.157
```

## Prise de décision et interprétation des résultats

La valeur de  $p$  ( $p$ -value) est inférieure au seuil de signification ( $p$ -value <  $2.2e-16$ ), donc l'hypothèse nulle est rejetée. Cela signifierait que l'âge moyen dans l'échantillon est significativement supérieur à 35 ans. Vous notez également que l'intervalle de confiance est  $]47.53359; +\infty[$ . Ceci démontre que la plage dans laquelle la vraie moyenne de la population est estimée se situer avec un certain niveau de confiance de 95% est  $]47.53359; +\infty[$ .

**Exemple 5.10 :** Vous souhaitez tester si la proportion de femmes dans la base de données hdv2003 est égale à 0.5. Dans ce cas, vous devez considérer la variable `sexe`.

Formulation des hypothèses :

- Hypothèse nulle ( $H_0$ ) : La proportion de femme de 50% ( $\mu=35$ ).
- Hypothèse alternative ( $H_1$ ) : La proportion de femme est différente de 50% ( $\mu>35$ ).

Calcul de la statistique :

Le code suivant permet de calculer la statistique

Code R

```
library(questionr)
data("hdv2003")
prop.test(x = sum(hdv2003$sexe == "Homme"), n = nrow(hdv2003), p = 0.5)
```

L'exécution de ce code affiche le résultat suivant.

1-sample proportions test with continuity correction

```
data: sum(hdv2003$sexe == "Femme") out of nrow(hdv2003), null probability
0.5
X-squared = 20.2, df = 1, p-value = 6.973e-06
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.5283722 0.5724319
sample estimates:
      p
0.5505
```

Prise de décision et interprétation des résultats

La valeur de  $p$  est inférieure à 0.05 ( $p\text{-value} = 6.973e-06$ ) et l'intervalle de confiance pour la proportion de femmes est de [0.52; 0.58].

En se basant sur ces résultats, vous rejetez l'hypothèse nulle, c'est-à-dire rejeter que la proportion de femme est de 50%. D'ailleurs, L'intervalle de confiance ne contient pas 0.5, ce qui renforce l'idée que

la proportion réelle de femmes est significativement différente de 50%.

**Exemple 5.11 :** Supposons que vous souhaitiez tester l'hypothèse selon laquelle la variance de l'âge des individus dans la base de données `hdv2003` est supérieure à une valeur spécifique, soit 200. Pour ce faire, vous pouvez utiliser le test chi-carré qui donne la probabilité que la variable aléatoire du chi-carré soit inférieure ou égale à une valeur donnée. `lower.tail=FALSE` spécifie que vous vous intéressez à la probabilité de l'extrémité supérieure de la distribution du chi-carré. Si `lower.tail = True` cela renvoie la probabilité de la partie inférieure de la distribution du chi-carré.

Code R

```
# Calcul de la variance échantillonnale
variance_echantillon <- var(hdv2003$age)
# Nombre d'observations
n <- length(hdv2003$age)

# Test du chi-carré unilatéral
test_variance <- (n - 1) * variance_echantillon / 200
p_value <- pchisq(test_variance, df = n - 1, lower.tail = FALSE)

# Afficher les résultats
cat("Variance de l'échantillon :", variance_echantillon, "\n")
cat("Statistique de test (Chi-carré) :", test_variance, "\n")
cat("Valeur-p : ", p_value, "\n")
```

L'exécution de ce code affiche le résultat suivant.

```
Variance de l'échantillon : 287.0249
Statistique de test (Chi-carré) : 2868.814
Valeur-p : 2.469468e-34
```

La valeur-p est inférieure à 0.05, vous pouvez conclure que la variance de l'âge des individus dans la base de données `hdv2003` est significativement supérieure à 200.

# Chapitre 6 Apprentissage machine avec R et Python

Daniel Lemire, Robert Godin et Neila Mezghani

## 6.1 Introduction

L'apprentissage machine, également connu sous le nom d'apprentissage automatique, est une discipline qui vise à découvrir des modèles automatiquement à partir des données. Ces modèles extraits sont utilisés pour effectuer pour prédire, classifier des données et analyser des structures au sein des données.

Les algorithmes d'apprentissage machine sont classés en fonction des techniques d'apprentissage qu'ils utilisent afin d'accomplir une tâche spécifique. Nous pouvons distinguer l'apprentissage supervisé, l'apprentissage non supervisé et l'apprentissage par renforcement (voir Figure 6-1).

Dans le cadre de l'apprentissage supervisé, également connu sous le nom de supervised learning en anglais, le système observe des paires d'entrées et de sorties, puis il apprend à créer une fonction ou un modèle permettant de produire la sortie à partir de l'entrée. Cette phase est appelée la phase d'apprentissage ou d'entraînement. L'apprentissage supervisé tire son nom de l'idée qu'un « professeur » supervise et enseigne au système la sortie attendue pour chaque entrée. Les données d'entrée et les données de sortie correspondantes sont connues, souvent appelées données labellisées ou étiquetées. Les

taches qui peuvent être accomplies par apprentissage supervisé sont la régression et la classification.

Contrairement à l'apprentissage supervisé, dans le cas de l'apprentissage non supervisé, les données de sortie ne sont pas étiquetées. Le système apprend alors de lui-même à organiser les données ou à découvrir des structures au sein des données. La tâche d'apprentissage la plus courante en apprentissage non supervisé est le regroupement, également appelé clustering en anglais, qui consiste à regrouper les données d'entrée en fonction de leurs caractéristiques communes. Ce type d'apprentissage est souvent utilisé pour la visualisation ou l'exploration de données pour assurer une tâche de regroupement ou d'analyse d'association entre les données.

L'apprentissage par renforcement repose sur des données d'entrée similaires à celles utilisées en apprentissage supervisé. Cependant, dans ce cas, l'apprentissage est guidé par l'environnement à travers des récompenses (positives ou négatives) calculées en fonction des erreurs commises pendant l'apprentissage. En robotique, l'apprentissage par renforcement a permis de développer des robots plus autonomes et adaptatifs à leur environnement.

Le choix de la technique d'apprentissage automatique à utiliser doit être faite par le scientifique des données en se basant sur la tâche à accomplir et les objectifs du projet.

Dans ce chapitre, nous présentons les rudiments de l'apprentissage machine (Partitionnement des données et validation des modèles) et de son application pratique en utilisant les langages de programmation Python et R. Nous explorerons quatre des principales tâches de

l'apprentissage machine : la régression, la classification, les règles d'association et le regroupement.

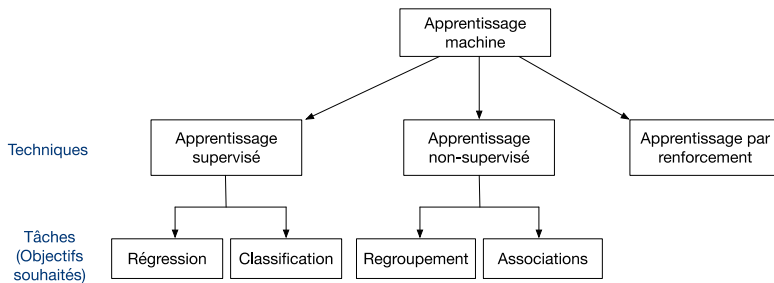


Figure 6-1: Choix des différentes techniques d'apprentissage machine en fonction de l'objectif souhaité

## 6.2 Partionnement des données

Indépendamment de la technique d'apprentissage machine à réaliser par le modèle développé, il importe aussi de mesurer la qualité de ce dernier. L'approche conventionnelle est de diviser les données en deux sous-ensembles. Le modèle est construit à partir d'un sous-ensemble (données d'entraînement) et testé sur le second sous-ensemble (données de validation). Cette division est importante puisqu'elle nous assure que le modèle n'a pas accès aux résultats du test lors de son entraînement. Par exemple, si un modèle à accès au dossier du professeur Smith lors de son entraînement et que le dossier du professeur Smith indique que celui-ci n'a pas pu faire ses paiements à temps, personne ne devrait se surprendre si le modèle arrive à prédire correctement que le professeur n'a pas pu faire ses paiements à temps. C'est une erreur commune chez les débutants en science des données d'entraîner et de tester un modèle sur les mêmes données.

Que ce soit avec R ou avec python, la partition des données se fait facilement avec les fonction `sample.split` ou `train_test_split`.

#### Code R

```
library(caTools)

#Pour avoir des partitions reproductibles
set.seed(1)

#Partition des données en 80% pour l'entrainement et 20% pour le test
sample <- sample.split(df$any_column_name, SplitRatio = 0.8)
train <- subset(df, sample == TRUE)
test <- subset(df, sample == FALSE)
```

#### Code Python

```
import numpy as np
from sklearn.model_selection import train_test_split

#Pour avoir des partitions reproductibles
np.random.seed(1)

#Partition des données en 80% pour l'entrainement et 20% pour le test
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.8, random_state=4 )
```

À la lecture de ces deux codes, vous pouvez constater les similarités de raisonnement dans les deux langages de programmation.

## 6.3 Validation des modèles

Pour un meilleur résultat, ce partitionnement peut être répété plusieurs fois. Une façon d'y arriver est la *validation croisée* qui consiste à partitionner les données d'entraînement en  $k$  groupes et à répéter l'entraînement  $k$  fois en prenant à chaque fois un des  $k$  groupes comme ensemble de validation, et en utilisant le reste des données pour l'entraînement. Cette approche permet de produire une évaluation de la capacité de généralisation du modèle en prenant la moyenne des



erreurs sur chacun des  $k$  groupes. Il est commun d'utiliser une valeur de  $k$  égale à 10 bien que d'autres valeurs soient valables.

Même la validation croisée doit être utilisée avec précaution : dans des conditions idéales, il est préférable de tester un algorithme avec des sources de données diverses, en plus de la validation croisée. Par exemple, un ingénieur pourra tester un algorithme en utilisant la validation croisée portant sur les données d'une première entreprise, et répéter ensuite l'exercice sur les données d'une autre entreprise.

#### Code R

```
# Charger la bibliothèque caret
library(caret)

# Charger un jeu de données, par exemple, le jeu de données iris
data(iris)

# Définir le nombre de folds pour la validation croisée (par exemple, 5 folds)
num_folds <- 5

# Créer un objet de contrôle pour la validation croisée
ctrl <- trainControl(method = "cv", number = num_folds)

# Spécifier le modèle à ajuster (par exemple, une régression logistique)
model <- train(Species ~ ., data = iris, method = "glm", trControl = ctrl)
```

#### Code Python

```
# Importer les bibliothèques nécessaires
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris

# Charger un jeu de données, par exemple, le jeu de données iris
iris = load_iris()
X, y = iris.data, iris.target

# Créer un modèle (par exemple, une régression logistique)
model = LogisticRegression()

# Définir le nombre de plis pour la validation croisée (par exemple, 5 plis)
num_folds = 5
```

```
# Réaliser la validation croisée
scores = cross_val_score(model, X, y, cv=num_folds)
```

Ces exemples illustrent comment effectuer une validation croisée en utilisant les bibliothèques `caret` pour R et `scikit-learn` pour Python

## 6.4 Bibliothèques populaires de R et de Python pour l'apprentissage machine

Les développements récents de l'apprentissage machine ont produit des avancées remarquables dans plusieurs domaines. Python et R sont devenus des langages populaires en apprentissage machine et en sciences des données parce qu'ils sont tous deux considérés comme relativement faciles à apprendre, en particulier pour les personnes ayant peu d'expérience en programmation. De plus, Les deux langages disposent d'écosystèmes de bibliothèques très étendus spécialement conçus pour l'apprentissage machine, la science des données et l'analyse statistique. Cela dit, R est traditionnellement considéré comme plus orienté vers les statistiques et l'analyse de données et Python comme un langage de programmation généraliste avec de fortes capacités en science des données. Ainsi, le langage Python a pris une petite longueur d'avance en apprentissage machine parce qu'il facilite l'expérimentation interactive pour le développement de modèles, la communication et le partage des résultats.

Il existe de nombreuses bibliothèques populaires en R et en Python pour l'apprentissage machine. Voici quelques-unes des bibliothèques les plus couramment utilisées dans ces langages :

## En R :

1. **caret** : caret est une bibliothèque polyvalente qui fournit des outils pour la préparation des données, la sélection de modèles, l'évaluation des modèles et la mise en œuvre de techniques d'apprentissage machine.
2. **randomForest** : Cette bibliothèque permet de créer et d'évaluer des modèles d'arbres de décision en utilisant la méthode de forêt aléatoire.
3. **glmnet** : glmnet est utilisé pour ajuster des modèles de régression logistique et linéaire régularisés (lasso et ridge).
4. **xgboost** : xgboost est une bibliothèque très populaire pour les arbres de gradient et est connue pour sa haute performance.
5. **tidyverse** : Le tidyverse est un ensemble de packages R, notamment dplyr, ggplot2 et tidyr, qui facilitent la manipulation et la visualisation des données.

## En Python :

1. **scikit-learn** : Scikit-learn est une bibliothèque incontournable pour l'apprentissage machine en Python. Elle offre de nombreux algorithmes pour la classification, la régression, le clustering, la réduction de dimensionnalité, etc.
2. **TensorFlow** et **Keras** : TensorFlow est une bibliothèque développée par Google pour le deep learning, tandis que Keras est une interface haut niveau qui facilite la création de réseaux de neurones avec TensorFlow.

3. **PyTorch** : PyTorch est une autre bibliothèque de deep learning qui gagne en popularité en raison de sa flexibilité et de sa facilité d'utilisation.
4. **pandas** : pandas est une bibliothèque essentielle pour la manipulation des données en Python. Elle permet de charger, nettoyer et préparer les données avant l'apprentissage machine.
5. **matplotlib** et **seaborn** : Ces bibliothèques sont utilisées pour la visualisation des données en Python, ce qui est important pour comprendre les données et évaluer les modèles.
6. **LightGBM** et **CatBoost** : Ce sont deux autres bibliothèques de gradient boosting populaires en Python, similaires à xgboost.

Ces bibliothèques constituent une base solide pour travailler sur des projets d'apprentissage machine en R et en Python. La plupart des bibliothèques et des packages d'apprentissage machine sont disponibles à la fois en R et en Python. Cependant, il existe quelques bibliothèques qui sont spécifiques à l'un ou l'autre langage en raison de leurs origines, de leurs fonctionnalités ou de leur popularité dans une communauté particulière. En occurrence, Seaborn est une bibliothèque de visualisation de données en Python qui se base sur matplotlib. Bien que matplotlib soit disponible en R, Seaborn est spécifique à Python pour ses fonctionnalités de visualisation de données de haute qualité.

## 6.5 La régression

Il existe deux principaux types de régression linéaire : la régression linéaire univariée et la régression linéaire multivariée.

### 6.5.1 Régression linéaire

La régression linéaire est une technique statistique couramment utilisée en science des données pour modéliser la relation entre une variable dépendante (ou cible) et une ou plusieurs variables indépendantes (ou caractéristiques) de manière linéaire. L'objectif de la régression linéaire est de trouver une équation linéaire qui décrit au mieux cette relation, ce qui permet de faire des prédictions ou d'analyser la relation entre les variables.

La régression linéaire univariée consiste à modéliser les valeurs de variables, dites variables dépendantes à partir d'une variable  $x$ , appelée variable indépendante, selon une équation linéaire. Pour déterminer l'équation appropriée, il faut exploiter un échantillon (ensemble) d'observations aussi appelées exemples ou simplement données. Il existe plusieurs formes de régression. Néanmoins, la régression linéaire univariée est un bon point de départ.

### 6.5.2 La régression linéaire univariée

Dans le cas où il y a qu'une seule variable dépendante ( $y$ ), le résultat de la régression linéaire est une formule de la forme  $y = ax + b$ . Il faut donc calculer à la fois la pente ( $a$ ) et l'ordonnée à l'origine ( $b$ ). En anglais, on fait souvent référence à la pente comme étant « *the coefficient* » et à l'ordonnée à l'origine comme étant « *the intercept* ». Il y a plusieurs manières de calculer une régression linéaire mais l'approche la plus commune est de minimiser l'écart entre la droite et

les données selon la somme des carrés : (1) pour chaque point de données, on calcule la valeur prédite par la droite en utilisant la valeur en  $x$  du point puis (2) on soustrait la valeur ainsi prédite de la valeur originale et (3) on met la différence au carré. En sommant toutes les erreurs, nous obtenons une valeur positive (l'erreur). Il suffit ensuite de calculer la droite qui minimise cette erreur. Une fois la droite déterminée, on utilise la mesure  $r^2$  qui est une mesure de la qualité du modèle ayant une valeur maximale de 1. Plus la valeur  $r^2$  est élevée, plus l'erreur est faible.

Vous pouvez faire ce calcul avec le module *sklearn* en Python. Vous devriez maintenant installer ce module en tapant la ligne suivante en ligne de commande :

```
python -m pip install sklearn
```

Pour visualiser le résultat, vous pouvez installer le module *matplotlib* qui permet de produire des graphiques :

```
python -m pip install matplotlib
```

**Exemple 6.1 :** Vous devriez enregistrer ce fichier CSV (*comma-separated values*) sur votre machine dans le répertoire où vous exécutez vos programmes Python :

[https://github.com/lemire/datasciencebook/raw/main/chapter5/global  
meantemp.csv](https://github.com/lemire/datasciencebook/raw/main/chapter5/global_meantemp.csv)

Le fichier en question comprend deux colonnes, la première colonne correspond à l'année alors que la seconde colonne correspond à la température globale moyenne en degrés Celsius selon la NASA (*NASA's Goddard Institute for Space Studies* ou *NASA GISS*).

Vous pouvez calculer la régression linéaire avec ce programme Python :

```
Code Python
import pandas as pd
from sklearn.linear_model import LinearRegression

modele = LinearRegression()
df = pd.read_csv("globalmeantemp.csv")
x = df["Year"].values.reshape(-1,1)
y = df["Nasa GISS"]
modele.fit(y=y,X=x)
print("intercept = ", modele.intercept_)
print("slope = ", modele.coef_)
r2 = modele.score(X=x,y=y)
print("regression score = ", r2)
```

Le programme Python affiche non seulement la pente et l'ordonnée à l'origine du modèle linéaire, mais aussi une mesure de qualité. Dans notre cas particulier, la mesure a comme valeur 0,76 ce qui signifie, intuitivement, que 76% de la valeur de la variable dépendante est déterminée par le modèle. Une mesure supérieure à 0,5 indique une bonne correspondance du modèle. Dans notre cas, une mesure de 0,76 indique une très bonne correspondance. La pente est de 0.00778 ce qui suggère un réchauffement de 0.00778 C par an, en moyenne, à partir du début des données (1880).

À partir du modèle, nous pouvons tenter de faire des prédictions. Quelle serait la température moyenne en 2100 ?

```
Code Python
print("temperature in 2100 = ",
modele.predict([[2100]]), " C")
```

Si vous exécutez ce code, vous constaterez que notre modèle prédit une température de 15,2 C en 2100. Il faut garder à l'esprit qu'il ne s'agit que d'un modèle : rien ne nous permet de croire que cette prédiction est correcte. Les modèles sont comme une carte routière :

ils peuvent être utiles mais l'écart entre la réalité et le modèle existe toujours.

Vous pouvez construire un modèle à partir d'un sous-ensemble des données. Par exemple, vous pouvez ne prendre en compte que des données à compter de 1960. Le segment de code suivant applique un tel filtre :

Code Python

```
x = df["Year"][df["Year"] > 1960].values.reshape(-1,1)
y = df["Nasa GISS"][df["Year"] > 1960]
modele.fit(y=y,X=x)
r2 = modele.score(X=x,y=y)
print("temperature in 2100 = ", modele.predict([[[2100]]], " C"))
```

Avec ce nouveau modèle, la température prédite en 2100 devient 16,3 C. Le coefficient de réchauffement est de 0.0125 C par an. Il semble que le modèle linéaire, à partir de 1880, ne prend pas en compte un réchauffement accéléré à partir de 1960. Il est relativement facile de construire un modèle plus riche.

### 6.5.3 La régression polynomiale

Au lieu d'une régression linéaire, vous pouvez opter pour une régression quadratique. Le principe est le même, mais cette fois-ci, vous cherchez la meilleure courbe ayant la forme quadratique  $ax^2 + bx + c$ .

**Exemple 6.2 :** Nous allons continuer avec le même ensemble de données. Vous pouvez obtenir le résultat souhaité avec le sous-module module *sklearn.preprocessing* et sa classe *PolynomialFeatures*. Il suffit d'ajouter le code suivant au programme pour faire le calcul automatiquement :

Code Python

```
from sklearn.preprocessing import PolynomialFeatures
```



```
x = df["Year"].values.reshape(-1,1)
y = df["Nasa GISS"]
poly = PolynomialFeatures(degree=2, include_bias=False)
poly_features = poly.fit_transform(x.reshape(-1, 1))
quadratic_model = LinearRegression()
quadratic_model.fit(y=y, X=poly_features)
r2 = quadratic_model.score(y=y, X=poly_features)
print("regression score = ", r2)
print("temperature in 2100 = ", quadratic_model.predict(poly.fit_transform([[2100]])),
"C")
```

Le coefficient de régression  $r^2$  a une valeur élevée (0,90). C'est normal : plus un modèle est riche, plus il comporte de coefficients, plus il est à même de bien correspondre aux données originales. Malheureusement, cela ne signifie pas pour autant que le modèle est préférable. En effet, un modèle comportant plus de paramètres est plus facilement victime de surajustement (*overfitting*) : il s'agit d'un phénomène où le modèle correspond mieux aux données que nous avons, mais peut-être moins bien aux données que nous n'avons pas. En d'autres termes, il n'est pas évident que ce nouveau modèle prédit mieux la température en l'an 2100 que la simple régression linéaire. Nous pouvons obtenir un coefficient de corrélation encore plus élevé entre le modèle et les données en utilisant un modèle plus riche, comportant un polynôme de degré 10. Le code Python est similaire :

#### Code Python

```
poly10 = PolynomialFeatures(degree=10, include_bias=False)
poly_features = poly10.fit_transform(x.reshape(-1, 1))
ten_model = LinearRegression()
ten_model.fit(y=y, X=poly_features)
```

Il est alors utile de représenter visuellement nos modèles afin de se faire une idée. Vous pouvez y arriver avec le module *matplotlib*. Ajoutez les lignes suivantes au programme qui enregistre sur votre machine un fichier graphique nommé « linearregression.pdf » :

#### Code Python

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib

X_modele = np.array(range(1880,2100,1)).reshape(-1,1)
Y_modele = modele.predict(X_modele)

matplotlib.rcParams['font.family'] = 'serif'
plt.scatter(x,y,label = 'données de départ')
plt.plot(X_modele,Y_modele, '-r',label = 'régression linéaire')
Y_modele = quadratic_model.predict(poly.fit_transform(X_modele))
plt.plot(X_modele,Y_modele, '-b',label = 'régression quadratique')
Y_modele = ten_model.predict(poly10.fit_transform(X_modele))
plt.plot(X_modele,Y_modele, '-g',label = 'régression degré 10')
plt.title("Regression univariée")
plt.xlabel('année')
plt.ylabel('température (C)')
plt.legend(loc='upper left', frameon=False)
plt.savefig('linearregression.pdf')
```

Le résultat est illustré à la Figure 6-2. Vous pouvez constater que les prédictions varient substantiellement selon le modèle choisi. Étant donné l'information dont nous disposons (un seul fichier de températures), il est impossible de déterminer lequel de nos modèles est le plus fiable, sans attendre à l'an 2100.

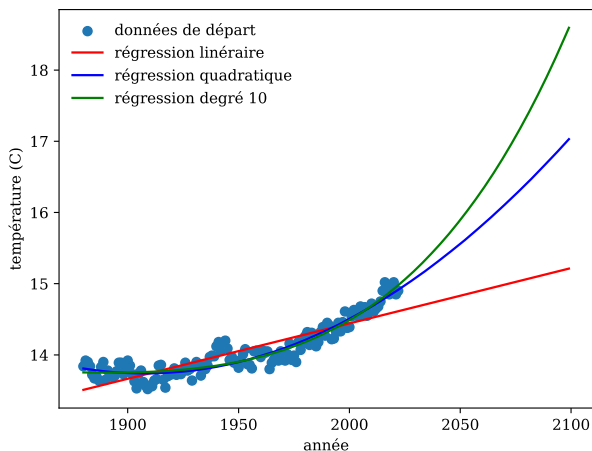


Figure 6-2. Exemples de régression

## 6.6 Classification

Dans le cas où les variables cibles sont binaires ou discrètes, leur prédiction est souvent appelée classification. Chacune des valeurs d'une variable dépendante est alors appelée une classe. Le cas le plus simple est la classification binaire où la variable à prédire comporte deux valeurs possibles. Par exemple, la classification binaire peut servir à prédire si un courriel est un pourriel ou non, si un client est risqué ou pas, si une image représente un hot-dog ou non, ou encore la présence d'une maladie dans des radiographies.

### 6.6.1 Arbres de décision

Une des techniques les plus simples pour la classification est l'arbre de décision. L'arbre de décision compare les valeurs des variables indépendantes et sur la base du résultat de la comparaison, d'autres comparaisons sont effectuées, jusqu'à ce qu'une prédiction puisse être faite. Par exemple, un arbre de décision permettant de prédire si un animal est un oiseau pourrait commencer par vérifier si l'animal est couvert de plumes. Dans le cas où il est couvert de plumes, l'arbre pourrait vérifier si l'animal a des ailes. Dans le cas où l'animal a des ailes et qu'il est couvert de plumes, l'arbre pourrait indiquer qu'il s'agit d'un oiseau. Dans tous les autres cas, l'arbre ferait la prédiction que l'animal n'est pas un oiseau. La construction manuelle d'un arbre de décision peut être laborieuse, mais les outils informatiques peuvent faire ce travail automatiquement.

Avant de construire votre premier arbre de décision, commencez par enregistrer sur votre machine le fichier CSV suivant dans le répertoire où vous exécutez vos programmes Python :

[https://github.com/lemire/datasciencebook/raw/main/chapter5/UCI\\_Credit\\_Card.csv](https://github.com/lemire/datasciencebook/raw/main/chapter5/UCI_Credit_Card.csv)

Vous pouvez charger le fichier dans un *dataframe* de pandas de ce code Python :

Code Python

```
import pandas as pd
df = pd.read_csv('UCI_Credit_Card.csv')
print(df.head())
```

Chaque ligne du fichier UCI\_Credit\_Card.csv comprend les données ayant trait à un individu. Le dernier attribut ("default.payment.next.month") indique si la personne n'a pu faire son paiement de carte de crédit à temps le mois suivant (1) ou si elle a pu le faire (0). Le fichier comporte plusieurs autres attributs dont le niveau d'éducation, le crédit restant, le statut marital, etc. La majorité des gens font leurs paiements à temps, mais une minorité (environ 22%) n'y arrive pas. Vous pouvez afficher les statistiques correspondantes :

Code Python

```
print(df["default.payment.next.month"][df["default.payment.next.month"]==1].count()
)
print(df["default.payment.next.month"][df["default.payment.next.month"]==0].count()
)
```

Quel serait le taux d'erreur d'un système qui prédirait systématiquement que les gens ne paient pas à temps ? Puisque c'est le cas de 78% des gens dans notre échantillon, la précision de l'algorithme qui prédit que tout le monde arrive à faire ses paiements devrait être de 78%. L'algorithme ne se trompe que 22% du temps. Ainsi, tout algorithme plus sophistiqué devrait avoir une précision d'au moins 78% : sinon, il ne vaut sans doute pas la peine d'être considéré.

Le fait que notre source de données est déséquilibrée, comportant plus de cas où les gens paient à temps que de cas où le paiement n'est pas fait, peut poser un problème. En effet, imaginons que les gens se mettent à compléter leurs paiements moins souvent. Il est possible alors que la précision d'un algorithme pourrait diminuer. Il est fâcheux d'avoir une mesure de précision qui dépend de la distribution des cas dans nos données. Les ingénieurs des données préfèrent normaliser les données. Ils souhaitent souvent avoir un nombre égal de cas dans chaque classe. Il est possible de retirer une partie des cas, mais une approche est la pondération. Notre logiciel peut pondérer les cas : un succès dans le cas où le paiement a été fait vaut moins que dans le cas où le paiement n'a pas été fait. La précision pondérée d'un algorithme qui prédit que tous font leur paiement à temps sera donc de 50%, peu importe la distribution des données. Un module comme *sklearn* permet de faire cette correction automatiquement en assignant la valeur « *balanced* » au paramètre *class\_weight*. Dans notre scénario, nous avons deux classes (paiement effectué ou non), mais dans les cas général où il y a  $k$  classes, un algorithme naïf devrait obtenir une précision pondérée de  $1/k$ .

Habituellement, un arbre de décision prend en compte plusieurs variables. Néanmoins, il est possible d'utiliser une seule variable. Un des attributs est SEX, qui indique si l'individu est de sexe féminin (1) ou masculin (0). Le code suivant construit un arbre de décision sur cet attribut :

Code Python

```

from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_text

x = df[["SEX"]]
y = df["default.payment.next.month"]
decision = DecisionTreeClassifier(class_weight = "balanced")
decision.fit(x, y)
print(export_text(decision))

```

L'arbre généré prend alors la forme suivante :

```

|--- feature_0 <= 1.50
|   |--- class: 1
|--- feature_0 > 1.50
|   |--- class: 0

```

En d'autres termes, dans le cas où la personne est un homme, l'arbre prédit qu'il y aura défaut de paiement, mais dans tous les cas où il s'agit d'une femme, l'arbre prédit que le paiement sera fait. Le résultat de l'arbre qui discrimine sur la base du sexe est une prédiction légèrement plus précise que le hasard (57% après pondération) :

```

Code Python
scores = cross_val_score(DecisionTreeClassifier(), x, y, cv=10)

print("précision %0.2f +/- %0.2f" % (scores.mean(), scores.std()))

scores = cross_val_score(DecisionTreeClassifier(class_weight = "balanced"), x, y,
cv=10)

print("précision %0.2f +/- %0.2f" % (scores.mean(), scores.std()))

```

Les variables PAY\_0, PAY\_2, PAY\_3, PAY\_4, PAY\_5, et PAY\_6 sont plus utiles. Elles indiquent pour chaque mois précédent, le retard de paiement : une valeur supérieure à 0 indique un retard de paiement d'un mois ou plus : la valeur 1 correspondant à un mois de retard, la valeur 2 à deux mois de retard, et ainsi de suite. Le résultat d'un modèle entraîné avec ces variables est une prédiction beaucoup plus précise que le hasard (78% après pondération) :

#### Code Python

```
x = df[["PAY_0", "PAY_2", "PAY_3", "PAY_4", "PAY_5", "PAY_6"]]

scores = cross_val_score(DecisionTreeClassifier(), x, y, cv=10)

print("précision %0.2f +/- %0.2f" % (scores.mean(), scores.std()))

scores = cross_val_score(DecisionTreeClassifier(class_weight = "balanced"), x, y,
cv=10)

print("précision %0.2f +/- %0.2f" % (scores.mean(), scores.std()))
```

La Figure 6-3 illustre un arbre de décision potentiel pour ce problème. Dans le cas, l'arbre de décision commence par vérifier la variable PAY\_2 : si celle-ci est plus grande que 1.5, l'individu est identifié comme un mauvais payeur. Dans le cas contraire, la variable PAY\_4 est vérifiée, si celle-ci est plus élevée que 0.5, alors le modèle prédit qu'il s'agit d'un mauvais payeur. Et ainsi de suite. Il est sans doute possible faire mieux. Vous êtes invités à essayer !

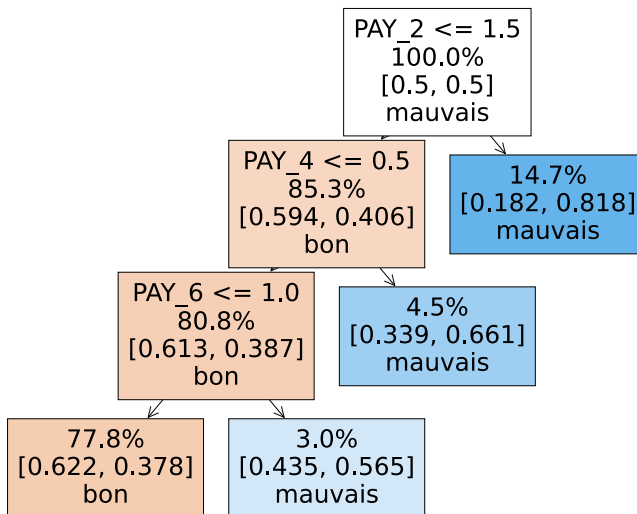


Figure 6-3. Exemple d'arbre de décision

## 6.7 Le regroupement

Le regroupement, aussi appelé agrégation (en anglais, *clustering*), est une méthode d'apprentissage machine non supervisée qui a pour objectif de former des groupes ou agrégats (en anglais, *clustering*) d'objets similaires à partir d'un ensemble hétérogène d'objets (aussi appelées individus ou points).

Le regroupement trouve son application dans plusieurs domaines. Par exemple, en écologie des populations, il permet de regrouper des espèces en fonction de leurs caractéristiques écologiques ou de leur niche. Dans le domaine du marketing, le regroupement permet de regrouper des produits pour optimiser la mise en rayon et les inventaires selon les comportements d'achat. Finalement, en sciences sociales, il permet de regrouper les répondants selon leurs réponses pour identifier des modèles dans les opinions ou les comportements sociaux.

Formellement, soit un ensemble de données formé de  $N$  individus décrits par des variables. Le regroupement consiste à trouver une répartition de ces données en  $K$  groupes de sorte que les individus à l'intérieur d'un même groupe soient similaires.

Les méthodes de regroupement varient en fonction de l'objectif et de la nature des données. Les algorithmes couramment utilisés incluent le K-means, la classification ascendante hiérarchique et le DBSCAN. Chaque algorithme a ses avantages et ses inconvénients, et le choix de la méthode appropriée dépend souvent de la nature des données et des questions de recherche spécifiques.



### 6.7.1 Algorithme des K-moyennes

L'algorithme des K-moyennes utilise une technique d'affinement itératif qui consiste à améliorer progressivement la qualité des groupes selon un indice de similarité entre les différents individus. L'hyperparamètre d'entrée de l'algorithme des K-moyennes est le nombre de groupes désiré K. La distance euclidienne est souvent utilisée pour mesurer le rapprochement des groupes.

**Exemple 6.3 :** Nous allons considérer l'ensemble de données USArrests qui est inclus dans R. Cet ensemble contient des statistiques sur les arrestations pour meurtre (**Murder**), agression (**Assault**) et viol (**Rape**) par 100 000 habitants dans chaque état des États-Unis, ainsi que le taux de criminalité urbaine en 1973 (**UrbanPro**). L'objectif ici est de faire un regroupement des différentes régions en se basant sur les 4 statistiques, qui sont en d'autres termes nos caractéristiques.

```
Code R
library(tidyverse) # data manipulation
library(cluster)  # clustering algorithms

# Lecture des données et suppression des données manquantes
df <- USArrests
df <- na.omit(df)

# Normalisation des données
df <- scale(df)

# Regroupement avec k=2
k2 <- kmeans(df, centers = 2, nstart = 25)
str(k2)
```

L'exécution de ce programme vous donne en sortie une liste contenant plusieurs informations. Les plus importantes sont (1) cluster, un vecteur d'entiers qui prend dans notre cas les valeurs « 1 » et « 2 »

indiquant le cluster auquel chaque point est attribué (2) centres, la matrice des centres de clusters et (3) totss, withinss, tot.withinss et betweenss qui sont respectivement la somme totale des carrés, le vecteur de la somme des carrés intra-cluster, la somme totale des carrés intra-cluster, c'est-à-dire la somme des withinss et la somme des carrés inter-clusters. Le nombre d'observations dans chaque cluster est donné par size. Ainsi dans votre cas, vous avez un groupe qui contient 30 observations nommé groupe « 1 » et un groupe contenant 20 observations noté « 2 ».

```
List of 9
 $ cluster   : Named int [1:50] 2 2 2 1 2 2 1 1 2 2 ...
 ..- attr(*, "names")= chr [1:50] "Alabama" "Alaska"
 "Arizona" "Arkansas" ...
 $ centers    : num [1:2, 1:4] -0.67 1.005 -0.676 1.014 -
 0.132 ...
 ..- attr(*, "dimnames")=List of 2
 ...$ : chr [1:2] "1" "2"
 ...$ : chr [1:4] "Murder" "Assault" "UrbanPop" "Rape"
 $ totss      : num 196
 $ withinss   : num [1:2] 56.1 46.7
 $ tot.withinss: num 103
 $ betweenss  : num 93.1
 $ size       : int [1:2] 30 20
 $ iter       : int 1
 $ ifault     : int 0
 - attr(*, "class")= chr "kmeans"
```

Pour visualiser les résultats d'une analyse de clustering, la fonction `fviz_cluster` fait partie du package `factoextra`. Si les données originales ont plus de deux ou trois dimensions, elle applique une réduction de dimensionnalité pour projeter les données dans un espace bidimensionnel ou tridimensionnel, facilitant ainsi la visualisation.

```
Code R
library(factoextra) # clustering algorithms & visualization

# Visualisation des regroupements
```

```
fviz_cluster(k2, data = df)
```

Vous obtenez ainsi le graphique de la Figure 6-4. Les deux axes principaux, Dim1 et Dim2, représentent les deux premières composantes principales qui expliquent 62% et 24,7% de la variance dans les données. Deux clusters délimités par des couleurs et des formes différentes : le cluster 1 (en rouge) et le cluster 2 (en bleu). Chaque point ou forme triangulaire représente une observation c'est-à-dire un état des États-Unis. Les états sont groupés en fonction de leur similitude selon les critères utilisés pour le clustering soit les 4 statistiques sur les arrestations pour meurtre (Murder), agression (Assault) et viol (Rape) par 100 000 habitants dans chaque état des États-Unis, ainsi que le taux de criminalité urbaine en 1973 (UrbanPro).

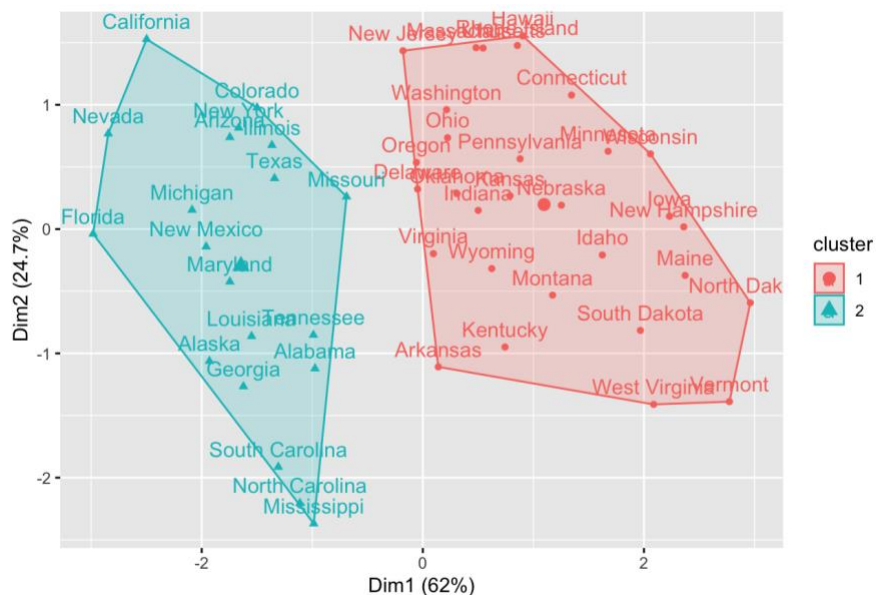


Figure 6-4. Visualisation des regroupements

## 6.7.2 Le regroupement hiérarchique

Contrairement au k-means, où il est nécessaire de spécifier à l'avance le nombre de clusters à former, le clustering hiérarchique ne nécessite pas la fixation du nombre de cluster à l'avance. Cet algorithme permet de produire une hiérarchie ou une structure d'arbre de clusters appelée dendrogramme. Il existe deux approches principales en clustering hiérarchique. L'approche agglomérative, aussi appelé regroupement hiérarchique ascendant, consiste à considérer chaque observation comme un cluster individuel. Ensuite, les paires de clusters les plus proches sont progressivement fusionnées selon une certaine métrique de distance (comme la distance euclidienne) et une méthode de liaison (ou "linkage"). À l'opposé, l'approche divisive, aussi appelé regroupement hiérarchique descendant, consiste à considérer un seul cluster qui contient toutes les observations et à chaque étape, le cluster existant est divisé en deux clusters plus petits.

**Exemple 6.4 :** Nous allons considérer l'ensemble de données USArrests décrit précédemment. Pour réaliser un regroupement hiérarchique en considérant une méthode de liaison, vous pouvez utiliser la fonction `hclust`. Comme vous pouvez remarquer le nombre de clusters choisi n'est fixé dès le départ mais plutôt à l'étape découpage du dendrogramme pour créer deux clusters avec `cutree`.

#### Code R

```
# Clustering hiérarchique utilisant la méthode de "complete linkage"
hc_complete <- hclust(dist(USArrests_norm), method = "complete")

# Affichage du dendrogramme
plot(hc_complete)

# Découpage du dendrogramme pour créer deux clusters
groups <- cutree(hc_complete, k = 2)

# Visualisation des clusters avec les étiquettes des États
plot(hc_complete, labels = rownames(USArrests), cex = 0.6)
# Ajoute des rectangles pour montrer les clusters
rect.hclust(hc_complete, k = 2, border = "red")
```

La sortie est dans ce cas est un dendrogramme dont le plus long segment vertical qui, si coupé horizontalement, diviserait le dendrogramme en deux groupes distincts. Ainsi dans votre cas, vous avez un groupe qui contient 23 observations et un groupe contenant 27 observations. Voir Figure 6-5.

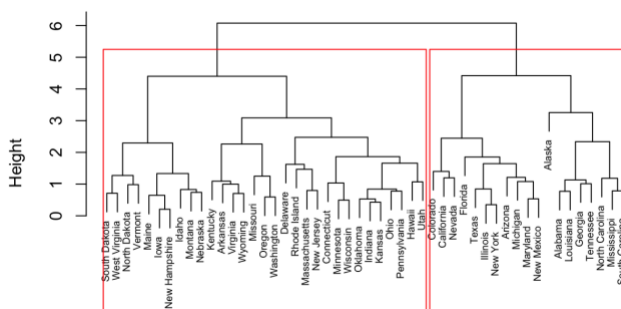


Figure 6-5. Dendrogramme

### 6.7.3 Comparaisons de deux algorithmes de regroupement

Comme vous l'avez certainement noté, les deux algorithmes K means et hiérarchique donnent des résultats de regroupement différent. La question qui se pose est alors « comment comparer les résultats de deux algorithmes de clustering exécutés sur le même jeu de données? ». Pour comparer, vous pouvez utiliser des métriques qui

évaluent la qualité des clusters comme le coefficient de silhouette, l'indice de Davies-Bouldin et l'indice Dunn. La seule difficulté consiste à pouvoir interpréter ces métriques. En occurrence, le coefficient de silhouette prend des valeurs entre -1 et 1. Une moyenne élevée du score de silhouette sur tous les échantillons indique un clustering robuste et bien séparé. Les indices de Davies-Bouldin et de Dunn sont positifs. En revanche, lors d'une comparaison de plusieurs clusters ou modèles de clustering, nous chercherons à minimiser l'indice de Davies-Bouldin alors que nous cherchons à maximiser l'indice de Dunn.

**Exemple 6.5 :** Vous pouvez comparer la silhouette des deux algorithmes K means et hiérarchique sur l'ensemble de données USArrests décrit précédemment.

Code R

```
b# Clustering K-means
set.seed(123) # Pour la reproductibilité des résultats
km_res <- kmeans(USArrests_norm, centers = 2)

# Clustering hiérarchique agglomératif
hc_res <- hclust(dist(USArrests_norm), method = "complete")
hc_clusters <- cutree(hc_res, k = 2)

# Calcul des scores de silhouette pour K-means
silhouette_km <- silhouette(km_res$cluster, dist(USArrests_norm))

# Calcul des scores de silhouette pour le clustering hiérarchique
silhouette_hc <- silhouette(hc_clusters, dist(USArrests_norm))

# Moyennes des scores de silhouette
avg_silhouette_km <- mean(silhouette_km[, "sil_width"])
avg_silhouette_hc <- mean(silhouette_hc[, "sil_width"])

# Affichage des résultats
cat("Moyenne de la silhouette pour K-means: ", avg_silhouette_km, "\n")
cat("Moyenne de la silhouette pour le regroupement agglomératif: ", avg_silhouette_hc,
"\n")
```

## 6.8 Règles d'association

Les règles d'association sont un concept fondamental dans le domaine de la science des données et de l'apprentissage automatique. Elles consistent à extraire des informations à partir des corrélations fréquentes ou des co-occurrences entre des éléments d'un ensemble de données. Ceci permet d'identifier des relations implicites cachées dans d'importante quantité de données.

Les règles d'association ont été particulièrement utiles dans le contexte de l'analyse de panier de marché (market basket analysis), où l'objectif est de trouver des ensembles d'articles achetés fréquemment ensemble. Aujourd'hui, elles trouvent leurs applications dans plusieurs domaines. En occurrence les règles d'association sont très utiles pour le diagnostic médical. Elles permettent d'identifier des combinaisons de symptômes qui sont fréquemment associées à certaines maladies. Dans le domaine des télécommunications, les règles d'association permettent d'analyser les schémas de trafic et optimiser la capacité du réseau. Les règles d'association peuvent être appliquées dans le domaine d'éducation pour recommander du contenu éducatif basé sur les préférences et les performances des élèves.

Formellement, une règle d'association est généralement exprimée sous la forme  $A \Rightarrow B$ , où  $A$  et  $B$  sont des ensembles d'items différents. La règle signifie que si  $A$  est réalisé, alors  $B$  a également une forte probabilité d'être réalisée. Les ensembles  $A$  et  $B$  sont souvent appelés l'antécédent (ou le préalable) et le conséquent de la règle, respectivement.

Les règles d'association sont générées et évaluées en utilisant des métriques qui décrivent la force et la pertinence d'une règle d'association. Parmi ces métriques nous retrouvons le support, la confiance et le lift.

**Exemple 6.6 :** Pour mieux comprendre ce concept, nous allons générer des règles d'association sur la base de données Groceries qui est fréquemment utilisée pour démontrer et tester des algorithmes de génération de règles d'association. L'objectif est de trouver des modèles d'achat fréquents parmi les clients, c'est-à-dire des ensembles d'articles qui tendent à être achetés ensemble. Par exemple, une règle d'association trouvée dans ces données pourrait indiquer que les clients qui achètent du pain et du lait sont également susceptibles d'acheter des œufs.

Code R
<pre>library(arules) data("Groceries") summary(Groceries)  rules &lt;- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8)) rules_sorted &lt;- sort(rules, by = "lift", decreasing = TRUE) inspect(head(rules_sorted, n = 3))</pre>

Ces lignes de codes vous permettent d'obtenir les trois règles d'associations les plus importantes en termes de lift. Ainsi l'achat de {liquor, red/blush wine} implque l'achat de {bottled beer}

<pre>{liquor, red/blush wine} =&gt; {bottled beer} {citrus fruit,other vegetables,soda,fruit/vegetable juice} =&gt; {root vegetables} {tropical fruit,other vegetables,whole milk,yogurt,oil} =&gt; {root vegetables}</pre>
---



## 6.9 Bibliographie

Chabot, P. (2022). « Déguster les vins de Bordeaux au Grand Marché », Le Soleil, <<https://www.lesoleil.com/2022/08/31/deguster-les-vins-de-bordeaux-au-grand-marche-252ed6e8-896ca4efb608819761868a89>>, consulté le 31 août 2022.

Leroux, M., I. Vivegnis et D. Laflamme (2017). Cercle pédagogique et analyse de cas : dispositifs de réflexivité collective au cœur de la formation initiale des enseignants, communication présentée dans le cadre du 85<sup>e</sup> congrès de l'ACFAS, mai, Montréal, Université McGill.

Bouchard, P. (2014). Les soins disponibles pour les chats en fin de vie : enquête sur le terrain, mémoire de maîtrise, Québec, Université Laval.

Germain, M.-N. et R. Tremblay (2018). « Les clés du changement », *Nouvelle gestion publique*, 2(3), p. 3-8.

Germain, M.-N. et R. Tremblay (2018). « Le contenu, la présentation et les références d'un ouvrage », dans J. Gagnon (dir.), *La présentation d'une bibliographie*, Québec, Presses de l'Université du Québec, p. 21-44.

Gagnon, J. (2018). *La présentation d'une bibliographie*, Québec, Presses de l'Université du Québec.

# Chapitre 7 : Visualisation des données et analytique d'affaires

Shadi Shuraïda

## 7.1 Introduction

Dans le cadre de l'analytique d'affaires, qui est le processus d'interprétation des données afin d'améliorer la prise de décision au sein de l'organisation, la visualisation des données commence seulement à être appréciée à sa juste valeur pour les avantages qu'elle peut apporter aux organisations. La visualisation des données, c'est-à-dire l'utilisation d'images pour représenter des informations, est utilisée dans toutes les organisations par les employés qui ont besoin d'interagir avec les données et de les analyser pour y déceler des modèles significatifs, des tendances et des anomalies (Stodder, 2013). En effet, face au déluge de données, la visualisation des données est devenue essentielle pour la compréhension et la prise de décision au sein des organisations. Comme l'a déclaré Edward Tufte dans son ouvrage séminal, *The Visual Display of Quantitative Information*, « les graphiques révèlent les données » (Tufte, 2001, p.13). Cela dit, comme nous le verrons dans les sections suivantes, si de bonnes visualisations de données sont essentielles pour prendre de meilleures décisions, de mauvaises visualisations de données peuvent embrouiller ou induire en erreur les utilisateurs.

Le rôle des technologies de l'information (TI) dans ce domaine a été crucial pour faire progresser la visualisation des données. Ces

technologies ont fourni aux utilisateurs des outils puissants qui leur permettent d'interpréter les données et de communiquer leurs connaissances à d'autres personnes.

Ce chapitre est structuré comme suit. La section suivante présente un bref historique de l'évolution des visualisations de données, suivi d'une discussion sur l'importance et les avantages de l'utilisation des visualisations de données. Ensuite, une vue d'ensemble de la littérature concernant ce qui rend les visualisations de données efficaces est présentée. Vient ensuite un résumé de l'utilisation des visualisations de données d'affaires, y compris les tableaux de bord. Enfin, ce chapitre se termine par une discussion et quelques démonstrations de visualisations trompeuses.

## **7.2 Brève histoire de la visualisation de données**

La visualisation des données est souvent considérée comme un développement relativement moderne dans le domaine de l'analyse. Mais en fait, la représentation graphique des informations est profondément enracinée dans les premières cartes, graphiques et lignes de temps (Sharda et al., 2018).

Plus précisément, l'histoire de la visualisation des données remonte à au moins 200 ans av. J.-C. Les visualisations de cette époque ont pris la forme des tableaux décrivant la position des étoiles et d'autres astres, ainsi que lors de l'élaboration de cartes destinées à faciliter la navigation et l'exploration (Few, 2007). L'idée de coordonnées a été utilisée par les géomètres égyptiens de l'Antiquité pour le tracé des villes, où les positions terrestres et célestes ont été localisées par quelque chose qui s'apparente à la latitude et à la longitude, au moins

jusqu'en 200 av. J.-C. Plus tard, le mathématicien et astronome Claude Ptolémée (a créé une projection cartographique d'une terre sphérique en latitude et longitude à Alexandrie qui a servi de référence pendant plus de mille ans (McCluskey, 1998).

Au 16<sup>e</sup> siècle, les techniques et les instruments d'observation et de mesure précises, tels que la triangulation pour cartographier les lieux avec exactitude, ont marqué le début du domaine de la visualisation des données (Friendly et Wainer, 2021). Avec l'expansion territoriale au XVII<sup>e</sup> siècle, les questions de temps, de distance et d'espace sont devenues primordiales pour l'élaboration des cartes et la navigation. Ce siècle est également marqué par les progrès de la géométrie (par exemple les coordonnées cartésiennes), de la mesure et de la statistique (Friendly et Wainer, 2021). C'est ainsi qu'au 17<sup>e</sup> siècle que la représentation visuelle de données quantitatives par rapport à des échelles de coordonnées bidimensionnelles, la forme la plus courante de ce que nous appelons les graphiques est apparue. Mais la première visualisation de données statistiques peut être attribuée à l'astronome et cartographe Michael Florent van Langren, en 1644, qui a utilisé une seule dimension pour montrer les 12 estimations connues à l'époque de la différence de longitude entre Tolède et Rome présenté dans Figure 7-1 (Friendly et al., 2017).

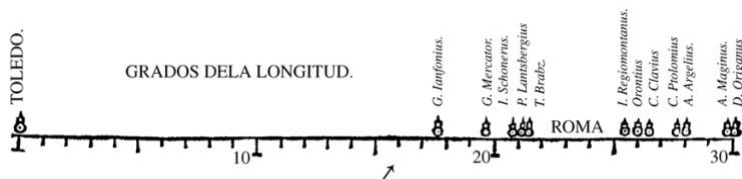


Figure 7-1. Le graphique de van Langren de 1644 sur les déterminations de la distance en longitude de Tolède à Rome. Source : Friendly et al. (2017)

Bien que ces visualisations aient été à l'origine de la présentation graphique d'informations quantitatives, la plupart des développements ont eu lieu au cours des deux derniers siècles et demi. Plus précisément, on attribue à William Playfair l'invention des graphiques modernes, y compris les diagrammes circulaires, et les graphiques à barres et à lignes. L'ingénieur et économiste politique écossais a déclaré la guerre aux tableaux. Le graphique présenté dans la Figure 7-2 était publié dans son livre *Commercial and Political Atlas* en 1786, et montre la balance commerciale entre l'Angleterre d'un côté, et le Danemark et la Norvège de l'autre, entre 1700 et 1780.

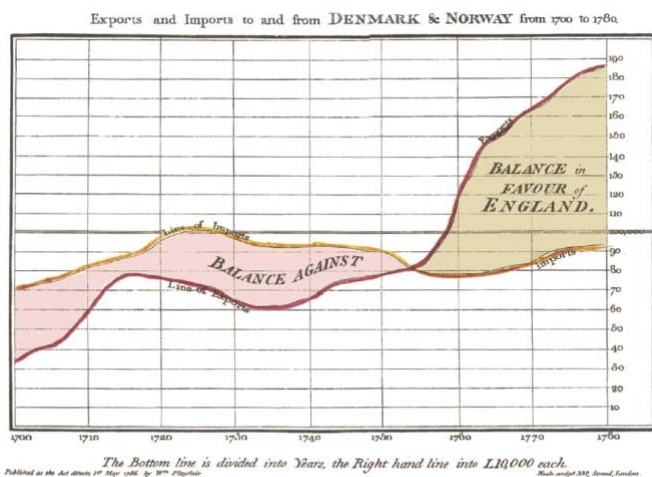


Figure 7-2 Balance commerciale entre l'Angleterre et le Danemark et la Norvège de 1700 à 1780. Source :

<https://www.historyofinformation.com/detail.php?entryid=2929>

La seconde moitié du 19<sup>e</sup> siècle est considérée comme l'âge d'or des graphiques statistiques. Cette période a été marquée par des innovations et une croissance rapide des visualisations. Ces innovations ont également été rendues possibles par la création d'offices statistiques dans toute l'Europe, qui ont collecté des données quantitatives dans le but de la planification sociale, le commerce et les transports (Friendly et al., 2017).

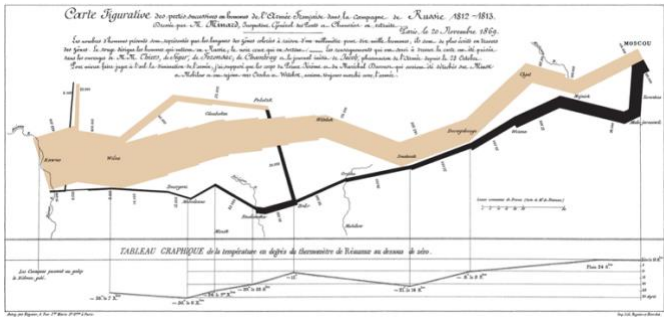
Un excellent exemple de visualisations de cette ère ayant conduit à des découvertes qui n'auraient peut-être pas été faites facilement autrement est le graphique populaire créé par le médecin John Snow dans les années 1850. La visualisation cartographique présentée dans la Figure 7-3 montre l'emplacement des décès dus au choléra dans le centre de Londres en septembre 1854. Dans cette visualisation, les décès sont marqués par des points et les onze pompes à eau de la région sont marquées par des croix (Tufte, 2001).



Figure 7-3 Les flambées de choléra lors de l'épidémie londonienne de 1854.  
Source : Tufte (2001, p. 24)

À l'époque, les germes n'étaient pas encore compris et les gens croyaient que le choléra se propageait par ce que l'on appelle les miasmes dans l'air. L'épidémie soudaine et grave de choléra dans le Soho de Londres était donc un mystère (Rogers, 2013). Cette cartographie a permis d'observer que les cas étaient regroupés autour de la pompe à eau de rue Broad (aujourd'hui Broadwick), ce qui a amené John Snow à penser que le choléra était dû à l'eau contaminée plutôt qu'à l'air (Rogers, 2013). À la suite de cette observation, Snow a fait retirer la poignée de la pompe, mettant ainsi fin à la pandémie qui a coûté la vie à plus de 500 personnes (Tufte, 2001).

Mais l'un des graphiques multidimensionnels les plus populaires de cette époque est peut-être celui créé par Charles Joseph Minard en 1869 (voir Figure 7-4). Dans ce graphique, Minard décrit la décimation de l'armée napoléonienne lors de la campagne de Russie de 1812. Ce qui est impressionnant dans ce graphique, c'est qu'il présente six types de données en deux dimensions : le nombre de soldats de Napoléon, la distance parcourue, la température, la latitude et la longitude, la direction et la position de l'armée par rapport à des dates spécifiques.





La renaissance de la visualisation des données est apparue au milieu des années 1960 avec la confluence du développement des méthodes d'analyse statistique des données, et de l'avènement de la technologie de l'information et des appareils d'affichage (Friendly et al., 2017). En 1977, John Tukey, professeur de statistiques à l'université de Princeton, a créé une approche graphique simple et efficace pour explorer les données et leur donner un sens dans ce qu'il a appelé l'analyse exploratoire des données (Few, 2007).

Avant l'avènement des ordinateurs personnels, la création de graphiques visuels était un processus à forte charge de travail qui nécessitait des compétences et des matériaux spéciaux. Mais les progrès réalisés depuis dans le domaine du hardware, ainsi que l'accessibilité et l'omniprésence des logiciels, ont entraîné une explosion des méthodes de visualisation des données et des types de données. À l'aide d'applications informatiques avancées, les organisations aujourd'hui peuvent traiter de grandes quantités de données, qui leur fournissent des visualisations qui les guident dans leur prise de décision.

Cependant, cet intérêt soulève des questions essentielles : pourquoi un tel intérêt pour les visualisations de données (importance) et comment peuvent-elles être utiles à la prise de décision au sein de l'organisation (forme) ?

### **7.3 Pourquoi visualiser les données ?**

Les avantages des visualisations de données sont soulignés dans de nombreuses publications et sont souvent présentés comme l'un des principaux chapitres des livres d'introduction à l'analyse des données

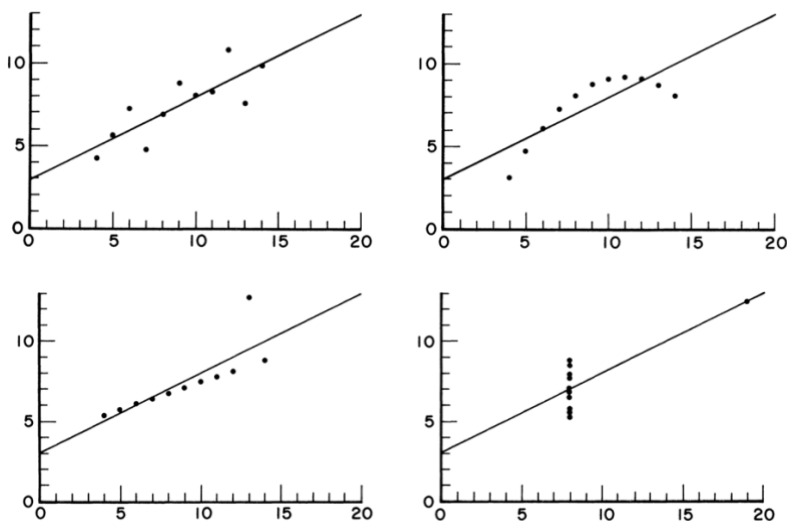
(par exemple, Sharda et al., 2018, Jaggia et al., 2023). Cependant, l'importance des visualisations est présentée de manière remarquable par l'exemple du quartet d'Anscombe (1973) présenté dans la Figure 7-5 (Tufté, 2001). À l'époque où le statisticien anglais a publié son article en 1973, les gens débattaient encore de la nécessité des graphiques de données (Schneider et Dineen, 2013). Alors il a créé un quartet d'ensemble de données fictives. Si on examine les statistiques sommaires des données présentées dans la Figure 7-5, ces quatre ensembles de données semblent identiques : les quatre ensembles de données sont décrits par exactement le même modèle linéaire. Toutefois, ces ensembles de données sont très différents. Voyez-vous la différence dans les données présentées dans la Figure 7-5 ?

I		II		III		IV	
X	Y	X	Y	X	Y	X	Y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

N = 11  
 mean of X's = 9.0  
 mean of Y's = 7.5  
 equation of regression line:  $Y = 3 + 0.5X$   
 standard error of estimate of slope = 0.118  
 $t = 4.24$   
 sum of squares  $X - \bar{X} = 110.0$   
 regression sum of squares = 27.50  
 residual sum of squares of Y = 13.75  
 correlation coefficient = .82  
 $r^2 = .67$

Figure 7-5 Quartet d'Anscombe. Source : Tufté (2001, p. 13).

Lorsque les quatre ensembles de données sont visualisés (voir Figure 7-6), elles révèlent des schémas de relations remarquablement différents entre les dimensions x et y. L'idée du quartet d'Anscombe est simple et puissante : les différences peuvent être clairement révélées par la représentation graphique des données, et les statistiques sommaires traditionnelles peuvent être trompeuses.

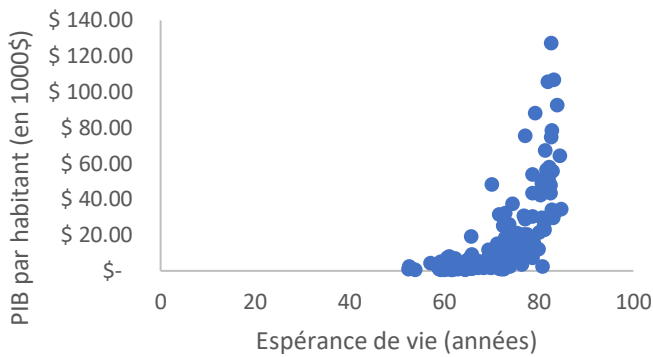


*Figure 7-6 Graphique du quartet d'Anscombe (I - IV) ensembles de données.  
Source : Anscombe (1973)*

Avec cet exemple, Anscombe (1973) a pu contrer l'idée de l'époque selon laquelle « les calculs numériques sont exacts, mais les graphiques sont approximatifs » et que « faire des calculs complexes est vertueuse, alors que regarder réellement les données est une tricherie » (p. 17). Mais outre le fait que la visualisation des données constitue une partie importante de l'analyse des données, cet exercice démontre également d'autres raisons pour lesquelles la visualisation des données est précieuse : ils révèlent les données (Tufté, 2001). En d'autres termes, les données présentées dans la Figure 7-6 sont exactement celles présentées dans la Figure 7-5. Pourtant, la Figure 7-6 nous a aidés à mieux comprendre. Comment expliquer cela ?

Les spécialistes des sciences cognitives et les professionnels s'accordent à dire que la présentation graphique des données

augmente la cognition (Hegarty, 2018) en soutenant la compréhension et le jugement de plusieurs manières (Zheng, 2017 ; Dudycz, 2010). La première est que les visualisations organisent l'information dans une perspective spatiale. Les visualisations relationnelles organisent souvent l'information de manière que les représentations des entités liées soient proches, ce qui facilite la recherche et l'intégration de l'information (Chen et al., 2009), c'est-à-dire la compréhension. Les graphiques organisent les entités en les plaçant dans un espace défini par les axes x et y. Par conséquent, les entités similaires sont visualisées comme étant proches les unes des autres (Hegarty, 2018). Par exemple, le diagramme de dispersion en Figure 7-7 démontre la relation entre l'espérance de la vie et le PIB par habitant de 177 pays.



*Figure 7-7 Relation entre PIB par habitant et l'espérance de la vie à la naissance. Source de données : Données des Nations Unies sur l'espérance de vie, de <https://population.un.org/wpp/Download/Standard/Mortality/>, et les données de la Banque mondiale sur le PI PIB par pays, <https://data.worldbank.org/indicator/NY.GDP.MKTP.CD>, récupérées le 13 décembre 2023.*

Dans ce graphique, on peut voir que des éléments similaires sont regroupés, par exemple le groupe des pays dont le PIB est inférieur à

20 000\$ ou ceux dont le PIB est supérieur à 60 000\$. Il faudrait une incroyable capacité cognitive humaine pour identifier cette organisation à partir des données textuelles des 177 pays.

Ainsi, l'organisation visuelle des informations réduit la charge cognitive nécessaire au traitement des informations et facilite la mémorisation, le rappel et la compréhension des informations (Borkin et al., 2013 ; Zheng, 2017). Cette réduction de la charge cognitive libère des ressources cognitives pour d'autres aspects de la réflexion, par exemple l'analyse de la relation entre les variables (Hegarty, 2018). Si nous examinons par exemple la relation entre l'espérance de vie et le PIB dans la Figure 7-7, nous pouvons ensuite voir comment des éléments similaires tels que l'espérance de vie changent en fonction des différents niveaux du PIB (faible, moyen et élevé). C'est ainsi que les visualisations efficaces donnent un aperçu des structures, des relations et des tendances dans les données, ce qui nous permet d'interpréter et de comprendre facilement des ensembles de données potentiellement complexes (Zheng, 2017). Les visualisations nous permettent également d'identifier les zones de différence telles que les anomalies qui peuvent être obscures dans les textes ou les tableaux (Dudycz, 2010). Ceci est illustré dans la Figure 7-7, où l'on peut facilement déterminer si un pays dispose d'un PIB élevé, mais d'une espérance de vie plus faible, par exemple.

En tant que telle, une représentation graphique efficace des données simplifie l'analyse et la prise de décision dans le domaine de l'analyse des affaires. Cela dit, la question se pose de savoir ce qui constitue

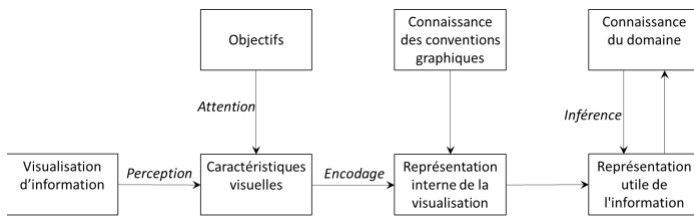
une visualisation efficace (Borkin et al., 2013; Alhadad, 2018; Hegarty, 2018).

#### **7.4 Qu'est-ce qui rend une visualisation efficace ?**

Une visualisation de données efficace est une visualisation qui améliore la compréhension de l'information par l'individu (Hegarty, 2018 ; Alhadad, 2018). Ainsi, pour comprendre ce qui fait une visualisation efficace, on se réfère aux modèles cognitifs qui permettent de comprendre les visualisations d'informations. En ce sens, deux conclusions importantes et solides de la recherche sur la psychologie cognitive et la compréhension des graphiques peuvent nous aider à identifier ce qui constitue des visualisations de données efficaces.

1. La première est que l'efficacité de la visualisation d'un ensemble de données doit tenir compte de la tâche à effectuer avec ce visuel et de l'utilisateur de ce visuel (Hegarty, 2018; Alhadad, 2018).
2. La seconde, qui concerne l'utilisateur du visuel, est que la capacité humaine de traitement de l'information est limitée (Alhadad, 2018).

La Figure 7-8 présente un modèle de compréhension de la visualisation d'informations adapté de Hegarty (2018). Conformément aux modèles antérieurs (par exemple, Carpenter et Shah, 1998), la compréhension des graphes implique l'interaction entre les processus de visualisation et de connaissance individuelle.



*Figure 7-8 Modèle des différentes représentations (indiquées par des cases) et processus (indiqués par des flèches) impliqués dans la compréhension d'une visualisation d'information. Source : Hegarty (2018, p. 21, traduction libre).*

Plus précisément, la capacité limitée de la personne à traiter l'information fait que seule une petite quantité d'informations visuelles peut être traitée à un moment donné (Alhadad, 2018). En conséquence, lorsqu'il est confronté à des entrées multiples (visuels dans ce cas), le système cognitif donne la priorité à certains éléments pour la suite du traitement, tout en ignorant d'autres. Comme le montre la Figure 7-8, l'attention est soit focalisée volontairement par un mécanisme descendant dirigé par les objectifs de l'observateur (processus descendant), soit par ce qui est visuellement saillant dans le graphique (processus ascendant) (Kahneman, 2003).

Par conséquent, l'information que les individus reconnaissent lorsqu'ils font passer du visuel à leur système de mémoire (encodage) dépend 1) de l'attention qu'ils portent à certains éléments du graphique (Carpenter et Shah, 1998), et 2) de leurs connaissances préalables des conventions du schéma du graphique (Alhadad, 2018; Hegarty, 2018). Ceci est important pour la conception graphique, car la première étape pour comprendre les informations importantes et pertinentes d'un visuel est liée à leur saillance dans le graphique et à la connaissance par l'utilisateur des conventions graphiques. Dans ce cas, des

informations très saillantes, mais non pertinentes pour la tâche, telles que des images, peuvent détourner l'attention de l'utilisateur de l'encodage des informations pertinentes pour la tâche. Selon Tufte (2001), c'est le principe fondamental de la conception des graphiques : « avant tout, montrez les données » (p. 92, traduction libre de la citation).

De même, un utilisateur peut ne pas encoder les informations importantes d'un graphique s'il ne connaît pas la convention graphique présentée. Une convention consiste par exemple à représenter la variable indépendante sur l'axe des x et la variable dépendante sur l'axe des y, ou à considérer qu'une ligne droite représente une relation linéaire. Afin de comprendre pleinement la signification du visuel, l'utilisateur de l'information doit également faire d'autres déductions basées sur la connaissance du domaine. Dans la Figure 7-7, par exemple, un décideur politique du gouvernement du Québec peut avoir besoin de connaître des informations non présentées dans le graphique, comme l'éventail moyen des valeurs du PIB et de l'espérance de vie au Canada. En effet, certaines études ont montré que les spectateurs commettaient des erreurs systématiques dans l'interprétation des graphiques lorsque leurs connaissances préalables étaient incompatibles avec les informations présentées dans le graphique (Shah et Hoeffner, 2002).

En substance, ce modèle montre que les caractéristiques visuelles de la visualisation des données influencent la perception, l'attention et l'encodage, tandis que l'expérience et les connaissances guident l'attention et la compréhension visuelles, et ce de manière concertée



(Alhadad, 2018; Hegarty, 2018). Alors que les experts ou les novices en visualisation peuvent choisir intuitivement certaines caractéristiques de visualisation pour représenter efficacement les données, des études cognitives humaines suggèrent que ces caractéristiques sont incompatibles avec les connaissances préalables des utilisateurs et peuvent donc détourner l'attention des utilisateurs et les empêchent de comprendre les informations importantes (Zacks et al, 1998; Shah and Hoeffner, 2002).

#### **7.4.1 Principes de conception de visualisations efficaces**

Sur la base des théories du traitement de l'information et des recherches antérieures sur la compréhension des visualisations, un ensemble de principes pour la conception de visualisations efficaces a été proposé (Tufte, 2001; Kosslyn, 2006; Alhadad, 2018; Hegarty, 2018). Ces principes visent essentiellement à réduire la charge cognitive (surcharge d'informations) et à permettre l'allocation de processus cognitifs pour comprendre les données et en tirer des conclusions. Il convient ici de souligner qu'il n'existe pas de meilleur visuel, sans prise en compte de la tâche et de l'utilisateur de la visualisation (Hegarty, 2018). Voici un résumé de certains principes importants tirés de cette littérature :

1. Principe de pertinence et saillance. La communication est plus efficace lorsque l'on présente suffisamment d'informations et pas plus (Tufte, 2001; Kosslyn, 2006; Hegarty, 2018). Les visualisations efficaces présentent à l'utilisateur les informations pertinentes nécessaires à l'objectif visé, sans le submerger d'encombrements visuels et de détails inutiles. En supprimant les éléments non

pertinents tels que les détails excessifs, les embellissements visuels ou la densité de la mise en page, l'utilisateur alloue davantage de capacités cognitives au traitement des informations essentielles (Alhadad, 2018). Tufte (2001) considère qu'il s'agit là du principe fondamental des bons graphiques de données : « avant tout, montrer les données » (p. 92, traduction libre de la citation). Cela dit, le type d'informations contenues nécessaires à la compréhension dépend du graphique et de la tâche. Néanmoins, de ce principe découlent les conseils généraux de fournir des informations contextuelles importantes à partir desquelles les utilisateurs peuvent interpréter les données, par exemple des titres clairs et précis pour les graphiques (Kosslyn, 2006) et des étiquettes d'axes (Alhadad, 2018).

En outre, en présentant les informations nécessaires, le principe de saillance stipule que les visuels doivent être conçus pour signaler les informations les plus importantes. Ce principe est important pour attirer l'attention des utilisateurs et les aider à comprendre ce qui est important (Kosslyn, 2006).

2. Principe d'organisation perceptuelle (regroupement). Étant donné les capacités limitées des humains en matière de traitement de l'information et de mémoire, l'un des mécanismes permettant d'améliorer la mémoire et la compréhension est le regroupement perceptif (Gobet et al., 2001). Il consiste à regrouper des éléments en unités plus grandes ou plus larges en fonction de leur signification ou des associations apprises (Alhadad, 2018). Un exemple de regroupement est la façon dont nous nous souvenons de nos indicatifs téléphoniques en Amérique du Nord ; l'indicatif à trois chiffres est

mémorisé comme une seule unité, plutôt que comme trois nombres distincts. Les stratégies de regroupement perceptif comprennent l'utilisation de paramètres visuels communs tels que la couleur, la forme ou l'emplacement pour faciliter le "regroupement" des informations dans des groupes compatibles avec les objectifs de la tâche. Par exemple, le regroupement de données basé sur la similitude (par l'utilisation de couleurs ou de symboles communs), la proximité (distance) ou la continuité (traçage de lignes entre les données) permet à l'utilisateur d'associer différents points de données en tant que groupes partagés. La littérature suggère que la redondance dans la combinaison de ces caractéristiques (comme la combinaison de la couleur et de la forme ; par exemple, cercles bleus vis-à-vis carrés orange) peut renforcer la segmentation pour un traitement plus efficace (Alhadad, 2018).

Dans la Figure 7-9, des données fictives sont fournies concernant l'effet de trois stratégies différentes sur quatre trimestres. Cette figure utilise le regroupement des stratégies en utilisant des couleurs et des symboles différents pour chaque stratégie (similarité). Dans ce cas, en supposant que l'utilisateur veuille décider d'une stratégie en fonction de son impact dans le temps, il faut distinguer deux éléments, la stratégie et son effet à travers le temps. L'organisation de l'information peut donc être améliorée si nous regroupons également les stratégies en utilisant la continuation (ligne regroupant chaque stratégie) comme dans la partie à droite de la Figure 7-9.

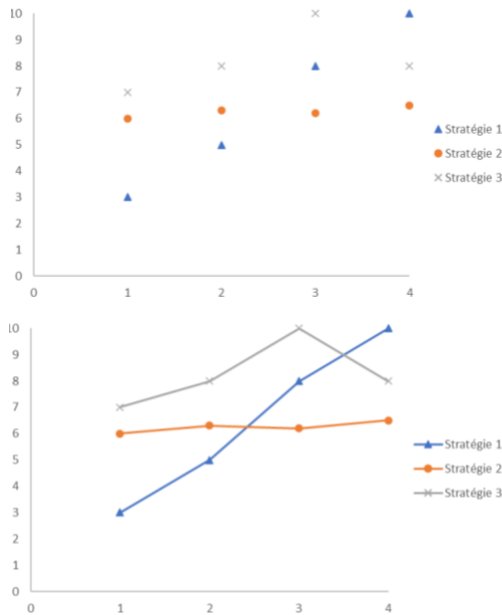


Figure 7-9 Comparaison des caractéristiques visuelles d'un diagramme à points avec codage redondant (forme/couleur) entre les catégories, avec des lignes en pointillés pour faciliter le suivi au sein de chaque stratégie.

3. Principe de compatibilité tâche-affichage. En rapport avec ce qui précède, la réflexion sur la manière dont les éléments d'un affichage visuel peuvent être regroupés en entités psychologiquement significatives peut soutenir l'attention et la cognition (Pinker, 1990) d'une manière compatible avec la tâche. Par exemple, le graphique linéaire est mieux adapté à l'intégration d'informations sur la relation entre la stratégie et le temps qu'un diagramme à barres ou un simple diagramme à points.

Ce principe s'étend également à l'utilisation de variables visuelles qui véhiculent une signification cohérente avec des suppositions spatiales et culturelles communes, telles que le haut est bon, le bas est mauvais et les éléments graphiques plus grands représentent une plus grande

quantité de quelque chose. D'autres suppositions communes incluent que les lignes indiquent des connexions et que la dimension horizontale est naturellement associée au temps (Kosslyn, 2006). Par exemple, la partie gauche de la Figure 7-10 va à l'encontre de trois conceptions courantes qui sont corrigées dans sa partie droite : la première est qu'un axe horizontal doit être utilisé pour indiquer le temps (les quarts), la deuxième est que les valeurs ou la hauteur sur l'axe vertical indiquent la quantité, et la troisième est que la couleur rouge est utilisée pour indiquer les valeurs négatives.

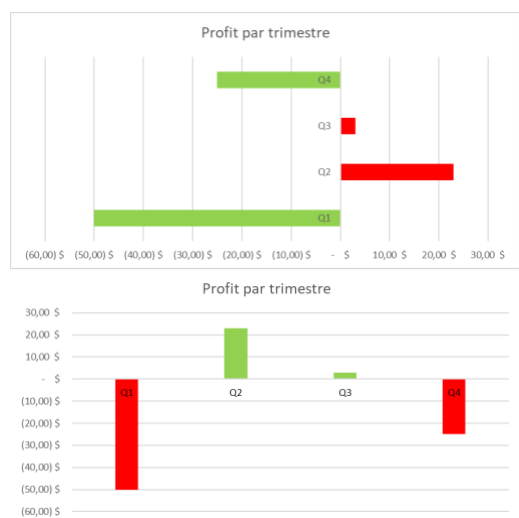


Figure 7-10 Comparaison de deux diagrammes à barres qui diffèrent dans l'affichage des caractéristiques de temps, de couleur et de valeur (positive ou négative).

4. Principe de la connaissance appropriée. Toute conception de visualisation efficace doit tenir compte des connaissances des utilisateurs. En fait, tous ces principes ne sont pas utiles si l'utilisateur de la visualisation ne dispose pas des connaissances nécessaires pour interpréter le graphique présenté. Ces connaissances comprennent les

conventions d'affichage visuel et les connaissances du domaine (Kosslyn, 2006; Hegarty, 2018). Par exemple, si la personne qui utilise le visuel n'est pas familiarisée avec le type de représentation utilisé (un diagramme en boîte à moustaches, par exemple), elle ne peut pas comprendre le contenu de l'information. De même, des connaissances dans un domaine tel que les statistiques sont nécessaires pour interpréter les lignes de régression linéaire, ou des connaissances en météorologie pour comprendre les cartes météorologiques. En tant que telle, bien que de bons visuels présentent de nouvelles informations, les concepts utilisés doivent être familiers aux utilisateurs et le format d'affichage doit être compréhensible pour eux.

## **7.5 Visualisation des données d'affaires**

Le terme « visualisation des données d'affaires » a une signification plus large que les seules activités commerciales, et se réfère généralement à de nombreuses activités humaines et organisationnelles qui assurent le fonctionnement d'un système. Il peut s'agir d'activités commerciales, non lucratives, éducatives, sportives, de divertissement, gouvernementales et bien d'autres encore. Dans ces activités, des données sont produites et enregistrées pour refléter tous les aspects des activités organisationnelles, puis elles sont analysées et font l'objet de rapports à différents niveaux. Les données d'affaires sont généralement abstraites, quantitatives, structurées ou semi-structurées et multidimensionnelles (Dudycz, 2010 ; Zheng, 2017). Ainsi, ces données sont transformées en informations significatives et utiles qui sont consommées par les

humains. La transformation de ces données sous forme de graphiques est la visualisation des données.

Les objectifs de la visualisation des données d'affaires se concentrent principalement sur l'exploration des données et la prise de décision (Stodder, 2013; Zheng, 2017). L'exploration des données permet aux utilisateurs de l'organisation de comprendre les données en révélant des tendances, en localisant les valeurs aberrantes et en montrant les relations entre différents facteurs. Supposons, par exemple, qu'un responsable des opérations de transport public ait pour objectif d'explorer les facteurs ou les tendances susceptibles d'entraîner une augmentation de la fréquentation. Les observations faites par le gestionnaire à partir des visualisations de données peuvent par exemple conduire à des idées intéressantes sur les tendances, les valeurs aberrantes ou les relations dans le temps, les données démographiques ou les lieux géographiques. Celles-ci peuvent à leur tour conduire à une exploration plus poussée et à d'autres objectifs.

De même, la visualisation des données fournit aux responsables des informations qui les aident à prendre des décisions au sein de l'organisation. Par exemple, les visualisations de données peuvent contribuer à améliorer l'affectation des ressources de sécurité publique en fonction des concentrations de criminalité (Nepomuceno & Cabral Seixas Costa, 2019) ou aider les médecins à évaluer l'état de santé de leurs patients et à agir en conséquence (Wanderer, Nelson, Ehrenfeld, Monahan, & Park, 2016).

En tant que telles, les visualisations dans les organisations servent des buts et des objectifs multiples. Une fois que vous avez déterminé les

données dont vous avez besoin, la question est de savoir quel format de visualisation utiliser ?

### **7.5.1 Choisir la bonne visualisation pour la tâche**

Il n'existe pas de graphique idéal à utiliser. Le choix du graphique dépend de l'information transmise, de l'objectif de la visualisation et du public visé. Mais une fois que vous avez identifié les informations que vous souhaitez transmettre et que vous avez réfléchi à votre public, vous devez choisir un format. Ainsi, les graphiques les plus familiers et les plus utilisés sont notés dans le tableau 6.1 (Kosslyn, 2006 ; Stodder, 2013; Jaggia et al., 2023), avec des lignes directrices du contexte de leur utilisation en termes d'objectif et de type de données (voir Kosslyn, 2006, et Jaggia et al., 2023, et Sharda et al., 2018 pour des suggestions de design plus détaillées). Il convient de noter ici que les cas d'utilisations conseillés sont normatifs et ne s'appliquent pas à tous les instances et les contextes.



Tableau 6.1. Les graphiques les plus familiers et leurs cas d'utilisation conseillés.

Graphique	Cas d'utilisation conseillés (type de données)			Cas d'utilisation conseillés (objectifs)
	Variables nominales	Données quantitatives et données de rangs	Données en pourcentage et en proportion	
Graphique en barres ou en colonne	✓	✓		Pour montrer les valeurs relatives des points, et particulièrement utiles pour visualiser les changements dans les données.
Graphique linéaire		✓		Lorsque les conventions définissent des tendances significatives. Par exemple, pour suivre les changements ou les tendances dans le temps.
Graphique circulaire	✓		✓	Pour indiquer les quantités relatives approximatives.
Graphique à barres empilées	✓	✓	✓	Pour donner une impression précise des parties d'un ensemble.
Diagramme de dispersion		✓		Pour donner une impression générale de la relation entre deux variables. Particulièrement utile pour afficher la relation entre deux grands ensembles de données.
Carte thermique (heat map)	✓			Pour identifier les combinaisons de variables nominales qui ont une signification économique.

Il existe également de nombreuses variantes et extensions de ces graphiques plus familiers, comme le diagramme à bulles (extension du diagramme de dispersion), et l'histogramme (variante du diagramme à barres). En outre, les logiciels spécialisés et les nouveaux besoins ont donné lieu à une explosion du nombre de représentations visuelles disponibles utilisées par les organisations. Par exemple, les visualisations de nuages de mots sont souvent

utilisées par les organisations pour analyser les messages des employés et des clients sur les médias sociaux ou les flux en ligne, ou des graphiques de réseau (ou nœuds-liens) pour analyser les relations dans les réseaux sociaux. Cela dit, bien qu'il existe de nombreux types de représentations de données utilisées dans les entreprises, Kosslyn (2006) suggère que l'utilisation et la survie des graphiques familiers depuis leur invention par William Playfair en 1786 jusqu'à aujourd'hui témoignent de leur utilité.

### **7.5.2 Tableaux de bord en analytique d'affaires**

Les tableaux de bord sont devenus des éléments courants des efforts analytiques de la plupart des organisations (Few, 2007; Sharda et al., 2018). Leur conception vise à imiter celle d'un tableau de bord de voiture qui fournit au conducteur des informations sur la vitesse actuelle du véhicule, le niveau de carburant et la température du moteur, afin qu'il puisse évaluer les conditions de fonctionnement actuelles et prendre les décisions appropriées. De même, un tableau de bord de données fournit aux gestionnaires les informations les plus importantes sur un seul écran de qui leur permet d'évaluer les performances de leur organisation afin de prendre des mesures efficaces (Dudycz, 2010).

Souvent, les gestionnaires sont confrontés au défi d'être inondés d'informations, de ne pas avoir assez d'informations ou de ne pas avoir les informations dont ils ont besoin au bon moment. Le but des tableaux de bord est de fournir la bonne information, au bon moment, à la bonne personne qui peut prendre une décision informée. En tant que tels, les tableaux de bord efficaces offrent aux

organisations trois fonctionnalités principales: surveiller, analyser et gérer (Eckerson, 2010).

La première est qu'il permet aux responsables d'entreprise de surveiller les processus et activités critiques de l'entreprise à l'aide d'indicateurs de performance qui déclenchent des alertes en cas de problèmes potentiels. À l'aide d'indicateurs visuels (tableaux, graphiques, cadrans, etc.), les tableaux de bord doivent présenter les mesures de performance pertinentes pour l'utilisateur. Par exemple, un tableau de bord destiné à un directeur des opérations doit fournir des informations relatives à ses décisions, telles que le stock disponible et l'utilisation des capacités, tandis qu'un tableau de bord destiné à un directeur financier doit fournir des informations sur la situation financière de l'entreprise, telle que les liquidités disponibles, les obligations courantes liées à la dette, etc.

Deuxièmement, des tableaux de bord efficaces permettent à l'utilisateur d'analyser la cause profonde des problèmes mis en évidence par la fonction de surveillance, en explorant des informations pertinentes et opportunes à partir de perspectives multiples et à différents niveaux de détail. Enfin, des tableaux de bord efficaces permettent aux utilisateurs de gérer les personnes et les processus afin d'améliorer les décisions, d'optimiser les performances et d'orienter l'organisation dans la bonne direction. Plus précisément, les tableaux de bord traduisent la stratégie de l'organisation en mesures et en objectifs concrets, adaptés aux utilisateurs visés. Cela facilite la communication avec les différents groupes au sujet des objectifs de l'organisation et de ce qu'ils doivent faire dans leurs

domaines respectifs pour atteindre les buts fixés. D'ailleurs, grâce à la possibilité de surveiller les performances, les tableaux de bord permettent aux utilisateurs d'évaluer leurs actions afin d'affiner la stratégie ou d'aligner leurs actions avec l'orientation stratégique de l'organisation.

En fournissant ces trois fonctionnalités, le défi de créer un tableau de bord efficace est d'afficher toutes les informations requises sur un seul écran (Sharda et al.,2018). En tant que tel, si le respect des principes de conception susmentionnés est nécessaire, il n'est pas suffisant. Compte tenu de la grande quantité d'informations fournies, il peut être fastidieux pour un décideur d'évaluer si un graphique présente des valeurs ou des tendances acceptables. Pour faciliter le traitement de l'information, des objets ou attributs visuels spécialisés sont parfois utilisés pour fournir le contexte d'évaluation (c'est-à-dire bon, mauvais, acceptable, etc..). Comme le montre l'exemple d'un tableau de bord des ressources humaines (Figure 7-11), certains de ces attributs comprennent l'utilisation de molettes, de feux de signalisation ou de codes de couleur.



Figure 7-11 Exemple de tableau de bord des ressources humaines.

Outre le fait qu'un tableau de bord soit bien conçu, un aspect très important est son utilité. Les données doivent être fiables, actualisées et utiles pour que l'utilisateur puisse prendre des décisions informées.

## 7.6 Visualisations trompeuses et conclusion

Même si les visualisations peuvent rendre l'information plus accessible, plus compréhensible et plus convaincante, elles peuvent aussi être facilement mal utilisées et mal comprises, même par leurs concepteurs. La distorsion de l'information via les visualisations de données est un problème depuis longtemps (Tuft, 2001), et sa prévalence a augmenté avec la popularité accrue des visualisations de données dans les médias populaires et sur Internet (Pandey, 2015).

Les concepteurs et les communicateurs qui utilisent des visuels peuvent, intentionnellement ou non, présenter de l'information déceptive. Si certaines visualisations trompeuses peuvent être le résultat de l'inattention ou de l'ignorance de leurs créateurs quant aux bonnes pratiques en matière de création visuelle, d'autres sont délibérément conçues pour induire en erreur et influencer le public

cible (Pandey et al., 2015). Toutes les visualisations sont sujettes à la tromperie par le biais de diverses techniques qui peuvent inclure la couleur (voir Figure 7-10), des axes tronqués ou disproportionnés, ou des étiquettes manquantes. Pandey et al. (2015) ont démontré que les visualisations trompeuses ont un effet important sur la compréhension qu'ont les gens d'une question particulière, et ils identifient deux catégories principales dans lesquelles les visualisations peuvent être trompeuses.

La première est l'exagération ou la sous-estimation du message. Ce type de déception se produit lorsque l'information n'est pas déformée, mais lorsque l'étendue de l'information présentée est modifiée, c'est-à-dire que les données sont exagérées ou sous-estimées. L'axe des ordonnées tronqué en est un exemple. Prenons l'exemple de la Figure 7-12, qui compare le nombre de personnes bénéficiant de l'aide sociale à celui des personnes ayant un emploi à temps plein. Ce graphique déforme la proportionnalité, laissant entendre à première vue qu'il y a presque cinq fois plus de personnes bénéficiant de l'aide sociale que de personnes ayant un emploi à temps plein. Les chiffres indiqués sont exacts, mais lorsqu'ils sont présentés sur un axe des ordonnées complet qui commence à 0 dans la partie droite de la Figure 7-12, ils sont moins spectaculaires.

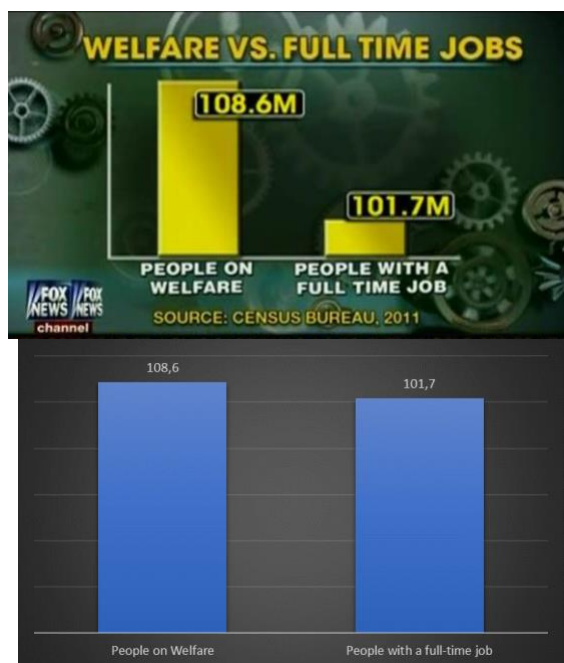


Figure 7-12 Exemples de visualisation trompeuse en utilisant l'axe des ordonnées tronqué. Source : <https://www.mediamatters.org/fox-friends/dishonest-fox-chart-overstates-comparison-welfare-full-time-work-500-percent>, consulté le 9 janvier 2024.

La comparaison des quantités dans les visualisations peut également être trompeuse en cartographiant les données dans une zone graphique. Au lieu de faire figurer la valeur des données dans la zone graphique (Figure 7-13), certaines visualisations trompeuses peuvent utiliser des variables autres que la surface, telles que le rayon. Cela fausse considérablement la représentation graphique des données.

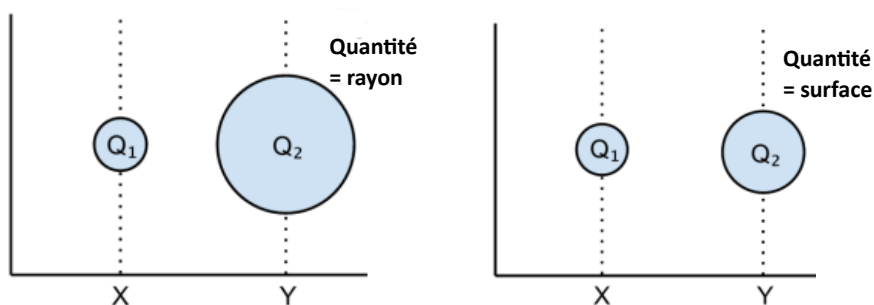


Figure 7-13 Exemples de visualisation trompeuse montrant que la surface d'une visualisation pourrait être une distorsion de la quantité. Source : Pandey et al. (2015, p. 1472).

La deuxième catégorie proposée par Pandey et al. (2015) est l'inversion du message. Comme mentionné dans le principe 3, les gens associent les directions à des tendances telles que l'augmentation à la hausse et la diminution à la baisse. Cette interprétation humaine fait de l'axe inversé l'une des techniques de distorsion les plus courantes susceptibles d'induire les individus en erreur. Comme le montre la Figure 7-14, le lecteur pourrait penser que les décès par arme à feu en Floride ont diminué (comme le suggère la tendance) à la suite de la mise en œuvre de la loi *stand your ground*. Cependant, en regardant de plus près le graphique, on constate que l'axe des ordonnées est inversé et que les chiffres augmentent vers le bas.



## Gun deaths in Florida

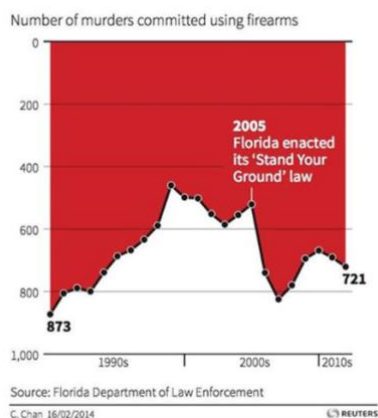


Figure 7-14 Exemple de visualisation trompeuse en utilisant l'inversion du message. Source : <https://policyviz.com/2023/02/07/10-ways-to-mislead-with-data-visualization/>, consulté le 9 janvier 2024.

En conclusion, bien que ces techniques soient les principales utilisées dans la visualisation trompeuse, il en existe beaucoup d'autres, moins courantes. Il incombe donc aux concepteurs des visualisations de respecter les principes et normes de conception des graphiques afin de transmettre des informations exactes. Néanmoins, les études en sciences cognitives suggèrent que tous les problèmes actuels d'interprétation des visualisations ne peuvent pas être résolus uniquement par une bonne conception visuelle. Ces études suggèrent également que les parties prenantes doivent être mieux informées sur les conventions et principes visuels, afin d'être de meilleurs consommateurs de visualisations de données.

## 7.7 Bibliographie

Alhadad, S. S. (2018). Visualizing data to support judgement, inference, and decision making in learning analytics: Insights from cognitive psychology and visualization science. *Journal of Learning Analytics*, 5(2), 60-85.

Anscombe, F. J. (1973). Graphs in statistical analysis. *The American Statistician*, 27(1), 17-21.

Borkin, M. A., Vo, A. A., Bylinskii, Z., Isola, P., Sunkavalli, S., Oliva, A., & Pfister, H. (2013). What makes a visualization memorable? *IEEE Transactions on Visualization and Computer Graphics*, 19(12), 2306-2315.

Carpenter, P. A., & Shah, P. (1998). A model of the perceptual and conceptual processes in graph comprehension. *Journal of Experimental Psychology: Applied*, 4(2), 75.

Chen, M., Ebert, D., Hagen, H., Laramée, R. S., van Liere, R., Ma, K. L., Ribarsky, W., Scheuermann, G. & Silver, D. (2009). Data, information, and knowledge in visualization. *IEEE Computer Graphics and Applications*, 29(1), 12-19.

Dudycz, H. (2010). Visualization methods in Business Intelligence systems—an overview. *Business Informatics* (16). *Data Mining and Business Intelligence*, 104, 9-24.

Eckerson, W. W. (2010). Performance dashboards: measuring, monitoring, and managing your business. John Wiley & Sons.

Few, S. (2007). Data visualization: past, present, and future. *IBM Cognos Innovation Center*.

Friendly, M., & Denis, D. (2005). The early origins and development of the scatterplot. *Journal of the History of the Behavioral Sciences*, 41(2), 103-130.

Friendly, M., Sigal, M., & Harnanansingh, D. (2017). The milestones project: a database for the history of data visualization. Dans *Visible Numbers* (pp. 219-234). Routledge.

Friendly, M., & Wainer, H. (2021). A history of data visualization and graphic communication. Cambridge, MA: Harvard University Press.

Gobet, F., Lane, P. C., Croker, S., Cheng, P. C., Jones, G., Oliver, I., & Pine, J. M. (2001). Chunking mechanisms in human learning. *Trends in Cognitive Sciences*, 5(6), 236-243.

Hegarty, M. (2018). Advances in cognitive science and information visualization. Dans Score reporting research and applications, ed. D. Zapata-Rivera. Routledge.

Jaggia, S., Kelly, A., Lertwachara, K., & Chen, L. (2023). Business analytics: Communicating with numbers. 2e ed. McGraw Hill LLC.

Kahneman, D. (2003). Maps of bounded rationality: Psychology for behavioral economics. *The American Economic Review*, 93(5), 1449–1475.

Kosslyn, S. M. (2006). Graph design for the eye and mind. New York, NY: Oxford University Press

McCluskey, S. C. (1998). Astronomies and cultures in early medieval Europe. Cambridge University Press.

Nepomuceno, T. C. C., & Costa, A. P. C. S. (2019). Spatial visualization on patterns of disaggregate robberies. *Operational Research*, 19, 857-886.

Pandey, A. V., Rall, K., Satterthwaite, M. L., Nov, O., & Bertini, E. (2015, April). How deceptive are deceptive visualizations? An empirical analysis of common distortion techniques. *Proceedings of the 33rd annual ACM conference on human factors in computing systems*. 1469-1478.

Pinker, S. (1990). A theory of graph comprehension. Dans *Artificial intelligence and the future of testing*, R. Freedle (Ed.), 73–126. Hillsdale, NJ: Erlbaum.

Rogers, S. (2013, 15 mars). John Snow's data journalism: the cholera map that changed the world. *The Guardian*. <https://www.theguardian.com/news/datablog/2013/mar/15/john-snow-cholera-map>

Schneider, E., & Dineen, C. (2013). Adding a dimension to Anscombe's quartet: Open source, 3-D data visualization. *Proceedings of the American Society for Information Science and Technology*, 50(1), 1-3.

Shah, P., & Hoeffner, J. (2002). Review of graph comprehension research: Implications for instruction. *Educational Psychology Review*, 14, 47-69.

Sharda, R., Delen, D., & Turban, E. (2018). Business intelligence, analytics, and data science: a managerial perspective. Pearson.

Stodder, D. (2013). Data visualization and discovery for better business decisions. *TDWI Research*.

Tufte, E.R. (2001). The visual display of quantitative information. 2e éd. Graphics Press

Wanderer, J. P., Nelson, S. E., Ehrenfeld, J. M., Monahan, S., & Park, S. (2016). Evolving health informatics semantic framework and metadata-driven architectures. *Journal of Medical Systems*, 40(12), 1-9.

Zacks, J., Levy, E., Tversky, B., & Schiano, D. J. (1998). Reading bar graphs: Effects of extraneous depth cues and graphical context. *Journal of Experimental Psychology: Applied*, 4(2), 119.

Zheng, J. G. (2017). Data visualization for business intelligence. Dans *Global business intelligence* (ed. J.M. Munoz). Routledge. 67-82.

# Chapitre 8 : L'IA explicable - La panacée pour une IA éthique ?

Dragos Vieru, Renée-Maria Schmitt et Élodie Boissières

## 8.1 Introduction

Le domaine de l'intelligence artificielle (IA) et de l'apprentissage automatique (AA ou Machine Learning) a fait l'objet d'une attention particulière ces derniers temps, notamment avec l'émergence de modèles d'IA génératifs tels que ChatGPT. L'IA est un terme général qui englobe divers domaines de recherche tels que le traitement du langage naturel, la parole, la vision, la robotique et l'AA (Mukhamediev et al., 2022). Bien qu'il n'existe pas encore de définition uniforme de l'IA, il est clair qu'il s'agit d'un domaine passionnant qui évolue rapidement (Ali et al., 2023).

Adoptés dans divers domaines et contextes, tels que l'éducation, le droit, les soins de santé, la finance et les transports (Agarwal et al., 2022; Jia et Zhang, 2022), les systèmes d'IA deviennent omniprésents et conduisent à une transition vers une société plus algorithmique (Adadi et Berrada, 2018). Ils remodelent nos valeurs, nos propriétés et nos interactions quotidiennes (Floridi et al., 2018; Zhou et al., 2022).

Cependant, il est actuellement controversé et difficile de mesurer l'influence et les impacts, positifs ou négatifs des systèmes d'IA, sur les vies et les environnements humains (Floridi et al., 2018). En effet,

même si les systèmes d'IA impactent de nombreux domaines, on peut penser que les répercussions seront moindres lorsque l'on parle de recommandations musicales et plus critiques lorsque l'on parle du traitement médical ou du transport autonome (Arrieta et al., 2020). Ainsi, les principaux défis et risques qui émergent des systèmes d'IA sont principalement issus des décisions négatives ou néfastes de ces systèmes (Henriksen et al., 2021).

L'un de ces principaux défis concerne le manque de transparence des systèmes d'IA, notamment à cause de leur complexité algorithmique. Il devient très difficile, voire impossible, pour les humains de comprendre les systèmes d'IA opaques, leur fonctionnement interne et leur raisonnement pour la prise de décision. Or, la transparence et/ou la compréhension des systèmes d'IA sont essentielles pour garantir leur comportement et leur fonctionnement corrects, particulièrement en ce qui concerne leur alignement sur les valeurs et les principes éthiques. Par conséquent, il devient crucial de comprendre le raisonnement qui sous-tend les décisions prises par les systèmes d'IA, et les humains doivent être en mesure de recueillir des informations approfondies sur les résultats de ces systèmes et sur la manière dont ils ont été générés (Lipton, 2018; Chazette et al., 2019). L'intelligence artificielle explicable (en anglais EXplainable Artificial Intelligence - XAI) représente un ensemble de processus et de méthodes qui visent à produire des systèmes d'IA plus transparents, compréhensibles et explicables, sans diminuer leurs performances et leur précision (Adadi et Berrada, 2018; Arrieta et al., 2020; Langer et al., 2021). Elle aborde ainsi la responsabilité numérique et les aspects

sociaux, éthiques et écologiques de l'utilisation des systèmes d'information (Sovrano et al., 2022). Au moment de la rédaction de ce document (fin 2023), l'IA allait bientôt être réglementée dans l'Union européenne (UE) par le biais de la loi sur l'IA et au Canada par le biais de la loi sur l'intelligence artificielle et les données, qui fait partie du projet de loi C-27 plus large, déposé par le gouvernement du Canada en juin 2022. La réglementation européenne contiendra des exigences en matière d'IA explicable (IAE). Par son souci de transparence, la IAE viendrait impacter positivement les enjeux éthiques des systèmes d'IA. En effet, plusieurs enjeux de l'IA ont pu être identifiés (cf. Fjeld et al., 2020; Hagendorff, 2020) et nous nous concentrerons ici sur les trois principes éthiques les plus discutées, comme le suggère Hagendorff (2020) : 1) la protection de la vie privée; 2) l'équité; et 3) la responsabilité. En mettant l'accent sur ces trois principes et en considérant l'impact potentiellement positif de la IAE sur l'éthique dans les systèmes d'IA, nous répondrons à la question suivante : La IAE peut-elle mitiger les enjeux éthiques liés à la protection de la vie privée, à l'équité et à la responsabilité dans les systèmes d'IA ?

Notre analyse suggère que la IAE représente un catalyseur d'une IA plus éthique en identifiant et en localisant les violations de la vie privée et de l'équité ainsi que les parties responsables. Cependant, une action humaine supplémentaire est nécessaire pour traiter et atténuer les enjeux identifiés.

## **8.2 La prochaine génération de l'IA**

Un programme lancé par la Defense Advanced Research Projects (DARPA) des États-Unis en mai 2017 a suscité l'intérêt et la recherche

dans le domaine de la IAE (Adamson, 2023; Rawal et al., 2022). Le programme intitulé « Broad Agency Announcement : Explainable Artificial Intelligence » avait pour objectif d'encourager la recherche et le développement dans le domaine de la IAE (DARPA, 2016; Gunning et Aha, 2019). Cela a déclenché un intérêt mondial (Arrieta et al., 2020) dans un domaine de recherche multidisciplinaire (Langer et al., 2021). La recherche sur la IAE couvre un large éventail de disciplines, allant de l'IA, de l'AA, de la science des données et de l'interaction homme-machine aux sciences sociales et cognitives, à la psychologie et à la philosophie (Adadi et Berrada, 2018; Miller, 2019; Brunotte et al., 2022a) et cherche à rendre les systèmes IA, ses processus et décisions compréhensibles pour les humains (Gunning et Aha, 2019; Langer et al., 2021).

Aujourd'hui, la pertinence de la IAE est reconnue par de nombreuses parties prenantes, notamment des experts en AA, des régulateurs, des avocats, des philosophes et des futurologues (Mittelstadt et al., 2019). Elle englobe bien plus que quelques méthodes technologiques individuelles et est considérée comme un mouvement et une partie de la prochaine génération de développement de la IAE (Adadi et Berrada, 2018; Adamson, 2023). Les techniques de IAE sont développées et employées dans de nombreux secteurs cruciaux, notamment les soins de santé, l'armée, les transports, le droit, la sécurité et la finance (Adadi et Berrada, 2018; Arrieta et al., 2020).



### 8.3 La terminologie et les principes de base de l'IA explicable (IAE)

Afin de bien comprendre les principes de base et la terminologie de l'IA explicable, les définitions des termes et leurs limitations sont cruciales pour la suite de cette analyse. (Gilpin et al., 2018; Rawal et al., 2022). Les définitions données s'inscrivent dans le contexte de l'IA.

Le terme compréhension désigne « l'action de comprendre le sens, le fonctionnement, la nature, etc. de quelque chose » (Dictionnaire Larousse, s.d.). En termes d'IA, la compréhension fait référence à la connaissance, par exemple, du fonctionnement interne des mécanismes d'un modèle d'IA et de la prédiction des résultats. La compréhensibilité décrit la capacité d'un modèle à permettre la compréhension par les humains (Arrieta et al., 2020).

La transparence fait référence à la visibilité des informations (Turilli et Floridi, 2009). Un système d'IA est dit transparent si son fonctionnement interne, ses compositions et ses données d'entraînement sont visibles pour les humains, c'est-à-dire que l'on peut "voir à travers" (Chazette et al., 2019). La transparence permet aux humains de comprendre intrinsèquement les informations visibles (Rawal et al., 2022), qu'il s'agisse de l'ensemble du système ou de certains de ses composants (Lipton, 2018).

L'explicabilité est une autre façon de permettre la compréhension (Chazette et al., 2019; Miller, 2019). Elle s'exprime activement en fournissant et en échangeant des raisons et des informations sur le fonctionnement d'un système ou son comportement (Chazette et al.,

2019; Mittelstadt et al. 2019). Ainsi, une explication agit comme un pont entre les humains et un décideur, leur permettant de communiquer efficacement (Guidotti et al., 2018).

Enfin, le concept d'interprétabilité fait référence à la mesure dans laquelle un agent est capable d'acquérir et d'utiliser les informations véhiculées par les explications fournies par le système, ainsi que les informations rendues disponibles par le niveau de transparence du système (Tomsett et al., 2018). Par conséquent, l'information fournie à l'humain doit être expliquée et transparente (Miller, 2019; Arrieta et al., 2020).

#### **8.4 Principales approches de l'IA explicable (IAE)**

Selon la DARPA (2016), l'idée derrière la IAE est de développer des techniques avancées d'apprentissage automatique qui génèrent des modèles facilement explicables. Ces modèles, associés à des techniques d'explication efficaces, permettront aux utilisateurs de comprendre, de faire confiance et de gérer plus efficacement la nouvelle génération de systèmes d'IA. Pour atteindre ces objectifs, une grande variété de techniques et d'outils IAE ont été développés au cours des dernières années (Adadi et Berrada, 2018; Arrieta et al., 2020). Pour catégoriser ces approches, plusieurs classifications existent, qui ne sont ni mutuellement exclusives ni exhaustives (Adadi et Berrada, 2018). Les spécialistes de l'IA font une distinction entre les systèmes d'IA qui peuvent être conçus de manière transparente, et ceux qui ne le peuvent pas (sans perte de performance) (Arrieta et al., 2020). Cette classification a été introduite par Lipton (2018) et est fréquemment reprise dans la littérature.

D'une part, un système peut être conçu de manière à être transparent en lui-même - on parle alors de transparence (Lipton, 2018; Rawal et al., 2022). Les systèmes transparents sont comme des boîtes de verre (Rai, 2020) dont le fonctionnement interne et les composants sont visibles pour les humains (Lipton, 2018; Tomsett et al., 2018). Les humains peuvent extraire des informations directement afin d'améliorer leur compréhension (Langer et al., 2021). Toutefois, la complexité et les performances des modèles transparents sont généralement limitées (Adamson, 2022).

D'autre part, un système et ses décisions peuvent être rendus compréhensibles par l'application d'un modèle ou d'un composant IAE externe, moins complexe, qui fournit une sorte d'explication pour l'aspect en question - ce que l'on appelle l'explicabilité post-hoc (Lipton, 2018; Arrieta et al., 2020; Rawal et al., 2022). Cela est particulièrement nécessaire pour les systèmes d'IA/AA très profonds et puissants qui ne peuvent pas être rendus transparents sans perte de performance ou de précision (Adadi et Berrada, 2018; Adamson, 2022). Il existe plusieurs méthodes d'explication qui peuvent être appliquées à un éventail de systèmes d'IA complexes (Rawal et al., 2022), telles que les explications textuelles, visuelles et basées sur des exemples (Arrieta et al., 2020; Rawal et al., 2022).

## **8.5 L'éthique dans l'IA**

L'émergence et les progrès des systèmes d'IA s'accompagnent d'enjeux éthiques croissants dans de nombreux domaines critiques tels que la santé ou le droit (Zhou et al., 2022) en impactant directement la vie humaine (Floridi et al., 2018; Zhou et al., 2022). Le

besoin d'ajouter de l'éthique dans les systèmes d'IA se fait donc sentir de plus en plus (Arrieta et al., 2020; Zhou, 2019).

L'éthique est une sous-discipline de la philosophie (Schlagwein et Willcocks, 2023) qui se réfère à un système de lignes directrices et de principes moraux sur la manière de se comporter dans des situations critiques et dilemmatiques - clarifiant ce qui est bon pour les individus et le grand public (cf. Anderson et Anderson, 2007; Santosh et Wall, 2022). L'éthique en tant que science comporte plusieurs domaines majeurs tels que la métaéthique (qui s'intéresse à la nature et à la signification de l'éthique), l'éthique normative (qui détermine les normes et les valeurs de l'action morale) et l'éthique appliquée (qui applique l'éthique à des situations spécifiques, parfois historiquement nouvelles, ou à des domaines d'actions possibles) (Melchert, 2014).

L'éthique de l'IA est une forme d'éthique appliquée, qui s'intéresse aux enjeux éthiques qui se posent dans le contexte des systèmes d'IA (Hagendorff, 2020). L'éthique doit être intégrée dans les systèmes d'IA et les enjeux éthiques doivent y être abordés (Zhou, 2019; Baldi et Oliveira, 2022). Des lignes directrices éthiques doivent être établies pour définir les principales exigences et principes éthiques importants pour garantir un comportement éthique dans les systèmes d'IA (cf. Jia et Zhang, 2022). Il existe un certain nombre d'approches techniques et non techniques pour permettre des systèmes d'IA éthiques. Certaines approches techniques sont, par exemple, les méthodes d'explication comme la IAE ou l'éthique par la conception (AI HLEG, 2019). Selon Jia et Zhang (2022), une approche non technique largement adoptée consiste à définir des lignes directrices éthiques

claires. Dans son examen systématique de la littérature dans le domaine de l'éthique de l'IA, Hagendorff (2020) analyse et compare vingt-deux lignes directrices éthiques différentes et leurs principes. Son analyse a montré que trois principes, la protection de la vie privée, l'équité/non-discrimination/justice et la responsabilité, apparaissent dans 80 % de la littérature analysée (Hagendorff, 2020). D'autres méta-études aboutissent à des conclusions similaires (Fjeld et al., 2020; Khan et al., 2022). Compte tenu de leurs effets négatifs potentiels sur l'humanité, la nécessité de systèmes d'IA éthiques est plus importante que jamais (Guidotti et al., 2018).

### **8.5.1 La vie privée**

En général, le concept de vie privée signifie avoir le contrôle de ses propres données personnelles, de leur accès et de leur utilisation (Brunotte et al., 2023; Santosh et Wall, 2022). La définition de la vie privée dans le contexte des systèmes d'IA est cruciale, car elle reconnaît le droit d'une personne à contrôler ses données. En tant que propriétaire des données, l'individu peut fixer des limites et décider des aspects de la vie privée qu'il souhaite partager avec les collecteurs de données et les utilisateurs. Il s'agit notamment de déterminer qui peut accéder aux données et à quel moment, et de veiller au respect de la vie privée (Brunotte et al., 2023).

La protection de la vie privée est progressivement entravée, par exemple, par les applications pour les maisons intelligentes (la configuration d'une maison dans laquelle les appareils et dispositifs peuvent être automatiquement contrôlés à distance depuis n'importe quel endroit disposant d'une connexion internet, à l'aide d'un appareil

mobile ou d'un autre appareil en réseau) ou par les technologies liées à la santé (Brunotte et al., 2022b), car les données à caractère personnel font partie des opérations de fusion, de traitement et d'interprétation (IEEE, 2017). Souvent, les personnes concernées ne sont même pas conscientes des risques qu'elles encourent en matière de protection de la vie privée, ni du type de données collectées, par qui, dans quel but et où elles sont stockées. Les applications de géolocalisation collectent généralement des informations sur les lieux fréquemment visités, tels que le domicile ou le lieu de travail. L'alimentation en données des systèmes d'intelligence artificielle présente de réelles menaces pour la vie privée et, par conséquent, la vie privée est une question d'éthique sérieuse (Rawal et al., 2022). Ainsi, pour garantir le principe éthique de la protection de la vie privée, une série d'exigences doivent être remplies par les systèmes informatiques et les systèmes d'IA. Les personnes concernées doivent donner leur autorisation et être pleinement informées de l'utilisation et des pratiques relatives à leurs données, tout en conservant un contrôle total sur celles-ci (Abrassart et al., 2018; Brunotte et al., 2022b). Ils doivent pouvoir accéder à leurs données personnelles, les rectifier ou les supprimer à tout moment (Abrassart et al., 2018; Fjeld et al., 2020; Hagendorff, 2020). L'accès aux données doit être limité aux personnes ayant des raisons d'accès appropriées, tandis que la qualité et l'intégrité des données doivent être garanties, y compris un traitement correct, transparent et équitable des données (Abrassart et al., 2018; AI HLEG, 2019).

### 8.5.2 L'équité

En association avec les systèmes d'IA, l'équité est caractérisée comme l'absence de toute forme de partialité ou de favoritisme envers des individus ou des groupes sur la base d'attributs personnels sensibles et indifférents au contexte (Mehrabi et al., 2022) ou comme une condition d'impartialité envers une personne ou un groupe (Agarwal et al., 2022). Garantir l'équité des décisions des systèmes d'IA est un autre défi crucial de l'éthique de l'IA (Saxena et al., 2019; Zhou, 2019).

Pour définir l'équité, on évalue une large série de notions, d'interprétations et de mesures qui peuvent être différentes selon la subjectivité personnelle et le contexte (Franklin et al., 2022). Cependant, en raison de l'absence d'une définition unifiée, de l'ambiguïté entre les notions et les mesures, et de la quantité de littérature existante, l'équité est difficile à mesurer (Franklin et al., 2022). Les systèmes d'IA peuvent intégrer divers biais (Franklin et al., 2022). Ces biais peuvent apparaître à n'importe quel stade du cycle de vie d'un système d'IA (Agarwal et al., 2022). Les trois sources potentielles de biais sont les données, les individus et le système d'IA lui-même (Stinson, 2022).

Les quantités de données utilisées pour la formation des systèmes d'IA peuvent être biaisées, défectueuses ou non représentatives (Fjeld et al., 2020). Il est important d'en être conscient lors de l'analyse et l'interprétation des données pour s'assurer que les conclusions sont exactes et impartiales (Chakraborty et al., 2021). D'autant plus que les données reproduisent inévitablement les jugements humains et les

modèles sociaux (Jia et Zhang, 2022). Par conséquent, les décisions prises par les systèmes d'IA peuvent non seulement recréer ces biais, mais aussi les aggraver (Langer et al., 2021). Par exemple, le fait de ne pas utiliser des ensembles de données totalement représentatifs ou de ne pas prendre en compte des caractéristiques dites « sensibles » ou « protégées », telles que la race ou le sexe, peut donner lieu à des résultats injustes ou discriminatoires (Saxena et al., 2019; Zhou 2019).

### **8.5.3 La responsabilité**

Selon les lignes directrices en matière d'IA éthique, à ce stade, seuls les êtres humains peuvent être tenus pour responsables des dommages, et non les systèmes d'IA (Abrassart et al., 2018; Wieringa, 2020). Selon Loi et Spielkamp (2021), le concept de responsabilité lui-même et ses différentes formes et dimensions manquent de clarté. Bovens (2007) définit la responsabilité, ou être responsable, comme l'action, de la part de la partie responsable d'une situation préjudiciable, d'expliquer et de justifier ses actions auprès des personnes concernées ou de leurs représentants. Les parties affectées ont le droit de poser des questions et de porter des jugements, et la partie responsable peut être confrontée à des conséquences. Selon cette définition, la responsabilité est de nature relationnelle et comprend cinq éléments clés : l'acteur, le forum, la relation, le contenu et les critères du compte rendu, et les conséquences (Wieringa, 2020). En appliquant la responsabilité relationnelle au contexte de l'IA, un système d'IA ainsi que d'autres parties prenantes, peuvent représenter les acteurs, tandis que les décisions prises par un système d'IA peuvent être considérées comme le contenu et les critères du compte rendu;



c'est ce que l'on appelle la « responsabilité algorithmique » (Wieringa, 2020).

Les trois mêmes phases sont pertinentes pour tout compte rendu efficace : information, explication/justification et conséquences (Bovens, 2007). Dans la phase d'information, les informations pertinentes sont mises à la disposition du forum par l'acteur qui l'informe de son comportement. La deuxième phase comprend l'interrogation de l'acteur par le forum et la demande d'explications et de justifications concernant le comportement. Ces explications et justifications doivent être claires et compréhensibles (Busuioc, 2021). Dans cette étape, la responsabilité est étroitement liée au concept de l'obligation de rendre des comptes, la responsabilité faisant référence à la fois à la capacité et à la volonté de l'acteur de fournir au forum les raisons et les explications de ses actions, ainsi qu'au droit du forum de demander ces raisons (Bovens, 2007; Busuioc, 2021). Enfin, la troisième phase identifie les conséquences qui doivent être imposées ou du moins possibles (Busuioc, 2021).

La littérature suggère que les systèmes autonomes comme les véhicules ou les applications de diagnostic médical peuvent prendre des décisions qui doivent être prises en compte par quelqu'un (cf. Santosh et Wall, 2022). Par exemple, en 2018, une voiture autonome Uber a provoqué un accident mortel (Wakabayashi, 2018). Bien qu'il y ait eu un conducteur dans la voiture, celle-ci conduisait de manière autonome au moment de l'accident. Cet exemple souligne à lui seul la nécessité d'établir des mécanismes pour garantir la responsabilité

legale (Cooper et al., 2022; Henriksen et al., 2021) et la responsabilité algorithmique.

Néanmoins, avec la prévalence des systèmes d'IA, il devient de plus en plus problématique de localiser et d'attribuer les responsabilités (Langer et al., 2021). Nissenbaum (1996) identifie quatre obstacles qui compliquent l'attribution des responsabilités avec l'avènement des systèmes informatisés : 1. Le problème de la multiplicité des mains - il s'agit du fait que le nombre de parties impliquées passe de quelques unes à un système complexe de parties; 2. l'apparition de bogues informatiques; 3. le fait de blâmer les systèmes informatiques pour leurs décisions et de les utiliser comme « boucs émissaires »; et 4. le défi de la « propriété sans responsabilité légale » (en anglais - *ownership without liability*). Le défi de la « propriété sans responsabilité légale » se réfère au problème des droits de propriété des systèmes et de leurs composants.

Avec les progrès des systèmes d'IA par rapport aux systèmes informatiques normaux, des défis supplémentaires se posent (Henriksen et al., 2021). La capacité des systèmes d'IA opaques à apprendre continuellement à partir des données plutôt que d'avoir un code écrit explicite aggrave l'attribution de la responsabilité (Bovens, 2007; Henriksen et al., 2021). Un « déficit de responsabilité » apparaît (Lima et al., 2022). L'une des conséquences directes possibles est la prise de décisions autonomes ayant des conséquences négatives dont personne n'est directement responsable - car trop de personnes peuvent avoir contribué (de manière intracçable) au préjudice (cf. Cooper et al., 2022).

En conclusion, afin de répondre aux trois enjeux éthiques majeurs, l'IA explicable doit donc prendre en considération la protection de la vie privée, l'équité et la responsabilité et aider à diminuer les biais issus des systèmes d'IA.

## **8.6 Comment l'IA explicable (IAE) aborde les enjeux éthiques**

Comme mentionné précédemment, les enjeux éthiques sont étroitement liés et exacerbés par l'opacité des systèmes d'IA et la compréhension limitée qui en résulte. Nous rappelons notre principale question de recherche La IAE peut-elle mitiger les enjeux éthiques liés à la protection de la vie privée, à l'équité et à la responsabilité dans les systèmes d'IA ? Pour répondre à cette question, nous analysons comment les deux principales techniques de la IAE, la transparence et l'explicabilité a posteriori, ont un impact sur les enjeux éthiques. Ces deux approches contribuent généralement de manière différente à l'atténuation des enjeux éthiques. Par conséquent, les implications potentielles de la transparence et de l'explicabilité sur les trois enjeux éthiques sont abordées ultérieurement.

### **8.6.1 La transparence**

La transparence peut sembler à première vue l'approche la plus évidente pour résoudre le problème de l'opacité puisque la transparence est décrite comme le contraire de l'opacité. La création de systèmes transparents peut être un outil précieux pour découvrir des enjeux éthiques lorsque les informations nécessaires à l'examen des enjeux éthiques sont rendues visibles (Turilli et Floridi, 2009). En

rendant visibles les informations sur le système d'IA, les humains peuvent mieux comprendre et localiser les menaces potentielles pour la vie privée, l'équité et/ou la responsabilité. Toutefois, la création de systèmes d'IA transparents s'accompagne de difficultés. Tout d'abord, la conception et le développement transparents des systèmes d'IA ne sont généralement pas réalisables sans perte de performance et/ou de précision, à moins de considérer des modèles simplifiés à l'extrême. Réaliser des systèmes d'IA efficaces et transparents est actuellement un objectif contradictoire. Deuxièmement, même si les systèmes d'IA peuvent être rendus transparents, les informations rendues visibles, comme le code, les données ou les processus algorithmiques, sont souvent dénuées de sens et non compréhensibles pour les néophytes (cf. Doran et al., 2017; Langer et al., 2021) et parfois même pour les ingénieurs et les experts (Köhl et al., 2019). Cela est dû, d'une part, au manque d'expertise et de connaissances et, d'autre part, à la complexité des systèmes d'IA (Adadi et Berrada, 2018; Busuioc, 2021; Santosh et Wall, 2022). Par exemple, les néophytes comme les utilisateurs peuvent être en mesure d'inspecter le code d'un système d'IA transparent mais peuvent ne pas comprendre, à l'étape suivante, ce que le code sous-jacent signifie, ce que le système d'IA fait et comment il prend des décisions. Au contraire, les décideurs peuvent disposer d'une expertise suffisante, mais l'énorme quantité de données et de paramètres introduits dans un système d'IA et créés dans ses propres représentations rend souvent impossible la saisie de la complexité des interactions entre les caractéristiques, même

lorsqu'elles sont totalement transparentes (Adadi et Berrada, 2018; Busuioc, 2021).

Dans ces deux cas, la transparence seule ne permet pas implicitement la compréhension. Par conséquent, la création de systèmes transparents sans perte de caractéristiques importantes n'est généralement pas réalisable et/ou pas suffisante. Ainsi, ni les violations de la vie privée ni les injustices existantes ne peuvent nécessairement être identifiées ou atténuées, et le code transparent ne permet pas nécessairement de rendre des comptes. La transparence n'est pas (encore) prometteuse pour réaliser des systèmes d'IA plus privés, plus équitables ou plus responsables, ou seulement dans des cas exceptionnels. Mais supposons que la transparence soit théoriquement possible, car il n'est pas exclu que des systèmes d'IA très complexes, mais transparents, soient développés à l'avenir. Alors, la transparence pourrait contribuer différemment à la compréhension et à l'atténuation des enjeux éthiques, selon l'un des trois niveaux de transparence : la simulabilité, la décomposabilité ou la transparence algorithmique.

Pour illustrer les différents effets de la transparence sur l'éthique dans les systèmes d'IA, nous prenons l'exemple d'un système de recrutement par IA formé à partir de données historiquement biaisées concernant le sexe. Dans cet exemple, la caractéristique du sexe prise en compte dans le processus décisionnel du système d'IA est sensible et pertinente d'un point de vue éthique. Par conséquent, les décisions du système sont injustes d'un point de vue éthique.

Du point de vue de la simulabilité, un haut niveau de transparence global permettrait aux humains de simuler l'ensemble du système de recrutement, y compris tous les composants, paramètres et données d'entrée en une seule fois (cf. Lipton, 2018). Sur la base des données d'un candidat donné en entrée et des paramètres visibles, les humains seraient en mesure de simuler la décision du système de recrutement concernant le candidat. On pourrait alors identifier que la caractéristique du sexe a une influence majeure sur le processus de décision (à supposer que l'expertise nécessaire soit présente). Cette information pourrait être utilisée pour corriger et éliminer la caractéristique du sexe du processus de décision. Par conséquent, la simulabilité pourrait contribuer à atténuer la question éthique de l'équité en la rendant perceptible.

La décomposabilité permettrait de rendre visible séparément chacun des composants du système de recrutement (cf. Lipton, 2018). Par exemple, les données d'entrée, les calculs et les paramètres seraient individuellement transparents. Ce niveau de transparence ne mènerait pas directement à l'identification de l'enjeu d'équité (biais de la donnée sexe) car les paramètres transparents sont considérés séparément, et non dans un contexte logique avec le processus de calcul et les données d'entrée. Dans le cas où des composants individuels du modèle peuvent être combinés par des humains pour comprendre la logique cohérente globale, par exemple en cartographiant le paramètre de sexe à la décision algorithmique, l'injustice sous-jacente pourrait être identifiée. Cela nécessiterait cependant une interprétation humaine plus approfondie des

composants visibles. Par conséquent, l'effet de la décomposabilité sur les systèmes d'IA éthiques dépend de la possibilité de combiner les informations des composants individuellement transparents pour comprendre la logique sous-jacente globale et mitiger l'enjeu éthique. L'activation de la transparence algorithmique seule permettrait de divulguer le processus algorithmique qui sous-tend la décision de recrutement. Le fait de pouvoir inspecter le processus algorithmique seul ne rend pas visibles les paramètres ou les données d'entrée. Par conséquent, le seul fait d'assurer la transparence du processus ne permet pas de découvrir que la caractéristique du sexe est fortement pondérée dans la décision de recrutement. La transparence algorithmique ne suffirait pas à identifier la pertinence de cette caractéristique et pourrait ne pas conduire à des systèmes d'IA plus éthiques puisqu'elle ne rend visible que l'algorithme et ne contient pas d'informations sur les données d'entrée ou les paramètres.

Par conséquent, si elle est réalisable et comprise, la transparence totale pourrait être une condition préalable importante pour atténuer les problèmes éthiques. Plus les parties des systèmes d'IA sont rendues transparentes, plus il y a d'informations visibles qui peuvent être inspectées; les violations de la vie privée et de l'équité ainsi que les parties responsables peuvent être identifiées. Les problèmes éthiques deviennent apparents avec la transparence. Néanmoins, ils doivent être résolus par les humains.

Étant donné qu'il est peu probable que la transparence soit disponible rapidement ou qu'elle permette la compréhension, la technique IAE de transparence n'est que dans des cas exceptionnels une solution

réalisable pour obtenir des systèmes d'IA plus privés, plus équitables et plus responsables. Cela équivaut à la nécessité d'expliquer (cf. Doran et al., 2017; Langer et al., 2021; Santosh et Wall, 2022). Les explications peuvent être nécessaires soit comme substitut dans le cas où la transparence n'est pas réalisable, soit en complément de la transparence dans le cas où la transparence ne permet pas implicitement la compréhension. C'est pourquoi la plupart des modèles IAE actuels sont basés sur des techniques d'explicabilité a posteriori (Adamson, 2022).

### **8.6.2 L'explicabilité a posteriori**

L'explicabilité a posteriori est considérée comme une technique pratique pour favoriser la compréhension et satisfaire aux exigences éthiques d'un système (Chazette et al., 2021). Mais comment ? En ce qui concerne la protection de la vie privée, l'équité et la responsabilité, les explications peuvent révéler des informations explicatives utiles d'un système d'IA (autrement) opaque concernant, par exemple, son fonctionnement, ses processus ou ses entrées et sorties. Les humains peuvent utiliser toutes ces informations pour comprendre les systèmes d'IA, un peu comme un mode d'emploi. Les explications et la compréhension qui en résultent sont utiles pour contrôler la conformité des systèmes d'IA avec les trois grands principes éthiques. Les humains peuvent contrôler ou justifier directement les décisions des systèmes d'IA ou peuvent indirectement tirer des conclusions causales ou identifier des modèles à partir des explications (Langer et al., 2021) pour obtenir d'autres informations et détails sur le système d'IA et ses décisions.



Pour déterminer comment et dans quelle mesure les explications contribuent à atténuer les problèmes éthiques liés à la protection de la vie privée, à l'équité et à la responsabilité, il est essentiel d'examiner la compréhension réelle obtenue (Langer et al., 2021). Un aspect doit non seulement être expliqué, mais aussi compris pour contribuer éventuellement à des systèmes d'IA plus privés, équitables et responsables. Par exemple, si une explication peut révéler des informations sur une caractéristique sensible prise en compte dans le processus décisionnel du système d'IA, mais que l'inspecteur humain ne comprend pas les informations explicatives, l'injustice de l'utilisation de la caractéristique sensible ne peut pas être identifiée et l'explication elle-même ne contribue pas à des systèmes d'IA plus équitables.

Les explications a posteriori peuvent révéler des informations nécessaires pour reconnaître l'injustice, les atteintes à la vie privée et les personnes responsables du préjudice causé. Toutefois, l'atténuation des enjeux liés à la protection de la vie privée ou à l'équité doivent toujours être pris en charge par les humains. Par conséquent, fournir une explication peut aider à justifier et à contrôler un comportement éthique ou à identifier des problèmes éthiques concernant la vie privée, l'équité et la responsabilité, mais ne peut pas directement atténuer ces problèmes. Néanmoins, l'utilisation d'explications permet de prévenir ou d'atténuer les comportements contraires à l'éthique.

En général, plus les informations sont accompagnées d'explications adéquates, plus les humains peuvent comprendre, contrôler et

améliorer les systèmes d'IA, et plus les questions et enjeux éthiques peuvent être identifiés et diminués par les humains. Cependant, dans la pratique de la IAE, les explications sont le plus souvent données avec des termes très techniques qui ne peuvent être compris qu'avec un certain niveau d'expertise (Floridi et al., 2018). La manière de rendre ces explications compréhensibles pour tous reste à déterminer à l'avenir (Cortese et al., 2022).

## 8.7 Conclusion

Les systèmes d'IA, qui deviennent de plus en plus omniprésents dans notre société, redéfinissent nos valeurs, nos propriétés et nos interactions quotidiennes. Cependant, cette intrusion apporte son lot d'enjeux éthiques. Dans notre analyse, nous avons introduit l'approche de l'intelligence artificielle explicable (IAE) comme pouvant être un catalyseur d'une IA plus éthique. En effet, nous avons déterminé que la IAE peut contribuer à mitiger les enjeux éthiques liés à la vie privée, à l'équité et à la responsabilité, mais elle ne peut pas les modérer à elle seule. Pour tirer pleinement bénéfice de la IAE, les informations divulguées doivent être adaptées et comprises. Il faut choisir le bon type et le bon niveau de technique de IAE et divulguer le bon aspect au(x) bon(s) destinataire(s) pour contribuer à la protection de la vie privée, à l'équité et à la responsabilité. Le Tableau 8-1 résume les avantages et les inconvénients de la IAE pour la protection de la vie privée, l'équité et la responsabilité.

La première contribution que la IAE peut apporter, se trouve dans le domaine de la transparence. En utilisant l'approche de transparence et ses trois niveaux (simulabilité, décomposabilité et transparence

algorithmique), la IAE permet aux systèmes d'IA d'être plus compréhensibles aux humains et d'identifier les menaces qu'ils représentent sur les enjeux éthiques. Malheureusement la création de systèmes transparents sans perte de caractéristiques importantes n'est généralement pas réalisable et/ou pas suffisante.

Donc, pour aller plus loin, la deuxième contribution que la IAE aborde, est celle de l'explicabilité à posteriori. Avec cette approche, les systèmes d'IA peuvent être contrôler, les décisions justifier et il est possible aux humains de tirer des conclusions causales et identifier des explications des comportements. En expliquant et en rendant plus visible, lisible et compréhensible l'information, la IAE permet d'atténuer de manière indépendante les causes des problèmes éthiques.

Malgré tout, le dernier enjeu, la responsabilité, pourrait être mieux prise en compte par la IAE si l'ensemble du processus de création des systèmes d'IA, y compris leur conception, leur développement et leur mise en œuvre, était documenté. La IAE pourrait utiliser ces informations pour localiser plus précisément les parties responsables tout au long du processus de création des systèmes d'IA.

En conclusion, la IAE n'en est qu'à ses balbutiements et il reste une grande place à l'amélioration. Comme il n'existe pas encore d'accord commun sur la définition d'une « explication » (Lipton, 2018; Köhl et al., 2019), ses caractéristiques requises (Guidotti et al., 2018) ou son déploiement dans la pratique (Hafermalz et Huysman, 2021), une grande confusion règne entre une IA « explicable » et une IA réellement « explicative » (Sovrano et al., 2021). La IAE sera-t-elle

en mesure d'unifier les recherches entre les sciences de l'explication, qui comprennent les sciences cognitives et sociales, la philosophie et le droit ? En attendant, la IAE représente une belle approche pour apporter plus de transparence dans des systèmes complexes et opaques.

Tableau 8-1 Effets de la IAE sur la vie privée, l'équité et la responsabilité

Avantages IAE	Principes éthiques	Inconvénients de la IAE
<ul style="list-style-type: none"> <li>+ Identification des violations de la vie privée et localisation de leurs causes</li> <li>+ Justification de la protection de la vie privée des systèmes d'IA</li> <li>+ + Sensibilisation au respect de la vie privée</li> </ul>	Vie privée	<ul style="list-style-type: none"> <li>– Exposition des données personnelles</li> <li>– Risque de classification incorrecte des systèmes d'IA comme étant privés</li> </ul>
<ul style="list-style-type: none"> <li>+ Identification de l'iniquité et localisation de ses causes</li> <li>+ Justification de l'équité des systèmes d'IA</li> </ul>	Équité	<ul style="list-style-type: none"> <li>– Risque de classification incorrecte des systèmes d'IA comme étant équitables</li> </ul>
<ul style="list-style-type: none"> <li>+ Aide à l'identification des parties responsables</li> <li>+ Fournit des informations significatives pour la reddition de comptes</li> </ul>	Responsabilité	<ul style="list-style-type: none"> <li>– La transparence et la documentation du processus de création des systèmes d'IA ne sont pas suffisantes.</li> </ul>

## 8.8 Bibliographie

Abrassart, C., Bengio, Y., Chicoisne, G., de Marcellis-Warin, N., Dilhac, M.-A., Gambs, S., ... et Voarino, N. (2018). « La Déclaration de Montréal pour un développement responsable de l'intelligence artificielle », Université de Montréal.

(<https://declarationmontreal-iaresponsable.com/la-declaration/>, consulté le 16 octobre 2023).

Adadi, A., et Berrada, M. (2018). « Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI) », *IEEE Access* (6), p. 52138–52160.

Adamson, G. (2023). Ethics and the explainable artificial intelligence (XAI) movement. *Authorea Preprints*.

Agarwal, A., Agarwal, H., et Agarwal, N. (2022). « Fairness Score and process standardization: framework for fairness certification in artificial intelligence systems », *AI and Ethics* (3), 267–279.

AI HLEG. (2019). « Ethics guidelines for trustworthy AI », Report for the European Commission by the High-Level Expert Group on Artificial Intelligence (AI HLEG). Report no. B-1049. Brussels, Belgium. (<https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai>, consulté le 16 octobre 2023).

Ali, S., Abuhmed, T., El-Sappagh, S., Muhammad, K., Alonso-Moral, J. M., Confalonieri, R., ... et Herrera, F. (2023). « Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence ». *Information Fusion*, 99, 101805.

Anderson, S. L., et Anderson, M. 2007. « Machine Ethics: Creating an Ethical Intelligent Agent », *AI Magazine* (28), p. 15-26.

Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., ... et Herrera, F. (2020). « Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI », *Information Fusion* (58), p. 82-115.

Baldi, V., et Oliveira, L. (2022). « Challenges to incorporate accountability into artificial intelligence », *Procedia Computer Science* (204), p. 519-523.

Bovens, M. (2007). « Analysing and Assessing Accountability: A Conceptual Framework », *European Law Journal* (13:4), p. 447-468.

Brunotte, W., Chazette, L., Klös, V., et Speith, T. (2022a). « Quo Vadis, Explainability? – A Research Roadmap for Explainability Engineering », in *Requirements Engineering: Foundation for Software Quality*, V. Gervasi and A. Vogelsang (eds.), Springer International Publishing, p. 26–32.

Brunotte, W., Chazette, L., Kohler, L., Klunder, J., et Schneider, K. (2022b). « What About My Privacy? Helping Users Understand Online Privacy Policies », *Actes de conférence de l'International Conference on Software and System Processes and International Conference on Global Software Engineering*, Pittsburgh PA USA, p. 56-65.

Brunotte, W., Specht, A., Chazette, L., et Schneider, K. (2023). « Privacy Explanations - A Means to End-User Trust », *Journal of Systems and Software* (195), 111545.

Busuioc, M. (2021). « Accountable Artificial Intelligence: Holding Algorithms to Account », *Public Administration Review* (81:5), p. 825-836.

Chakraborty, J., Majumder, S., et Menzies, T. (2021). « Bias in machine learning software: Why? How? What to do? », *Actes de conférence de 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, Athens, Greece, p. 429-440.

Chazette, L., Karras, O., et Schneider, K. (2019). « Do End-Users Want Explanations? Analyzing the Role of Explainability as an Emerging Aspect of Non-Functional Requirements », *Actes de conférence de l'IEEE 27th International Requirements Engineering Conference*, Jeju Island, Korea, p. 223-233.

Chazette, L., Brunotte, W., et Speith, T. (2021). « Exploring Explainability: A Definition, a Model, and a Knowledge Catalogue

», Actes de conférence de l'IEEE 29th International Requirements Engineering Conference, Notre Dame, IN, USA, p. 197-208.

Cooper, A. F., Moss, E., Laufer, B., et Nissenbaum, H. (2022). « Accountability in an Algorithmic Society: Relationality, Responsibility, and Robustness in Machine Learning », Actes de conférence de l'ACM Conference on Fairness, Accountability, and Transparency, Seoul, Korea, p. 864-876.

Cortese, J. F. N. B., Cozman, F. G., Lucca-Silveira, M. P., et Bechara, A. F. (2022). « Should explainability be a fifth ethical principle in AI ethics? », *AI and Ethics* (3), p. 123-134.

DARPA (Defense Advanced Research Projects Agency). (2016). Broad Agency Announcement Explainable Artificial Intelligence (XAI), Arlington, VA. (<https://www.darpa.mil/attachments/DARPA-BAA-16-53.pdf>, consulté le 10 octobre 2023).

Doran, D., Schulz, S., et Besold, T. R. (2017). « What Does Explainable AI Really Mean? A New Conceptualization of Perspectives », arXiv preprint arXiv:1710.00794.

Fjeld, J., Achten, N., Hilligoss, H., Nagy, A., et Srikumar, M. (2020). « Principled Artificial Intelligence: Mapping Consensus in Ethical and Rights-based Approaches to Principles for AI », Berkman Klein Center Research Publication, (2020-1) Center for Internet & Society, (<http://nrs.harvard.edu/urn-3:HUL.InstRepos:42160420>, consulté le 14 octobre 2023).

Floridi, L., Cowls, J., Beltrametti, M., Chatila, R., Chazerand, P., Dignum, ... et Vayena, E. (2018). « AI4People-An Ethical Framework for a Good AI Society: Opportunities, Risks, Principles, and Recommendations », *Minds and Machines* (28:4), p. 689-707.

Franklin, J. S., Bhanot, K., Ghalwash, M., Bennett, K. P., McCusker, J., and McGuinness, D. L. 2022. « An Ontology for Fairness Metrics », Actes de conférence de l'AAAI/ACM



Conference on AI, Ethics, and Society, Oxford, United Kingdom, p. 265-275.

Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., et Kagal, L. (2018). « Explaining Explanations: An Overview of Interpretability of Machine Learning », Actes de conférence de l'IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA), Turin, Italy, p. 80–89.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., and Pedreschi, D. (2018). « A Survey of Methods for Explaining Black Box Models », *ACM Computing Surveys* (51:5), p. 1-42.

Gunning, D., et Aha, D. 2019. « DARPA's Explainable Artificial Intelligence (XAI) Program », *AI Magazine* (40:2), p. 44-58.

Hafermalz, E., et Huysman, M. 2021. « Please Explain: Key Questions for Explainable AI research from an Organizational perspective », *Morals & Machines* (1:2), p. 10–23.

Hagendorff, T. 2020. « The Ethics of AI Ethics: An Evaluation of Guidelines », *Minds and Machines* (30:1), p. 99-120.

Henriksen, A., Enni, S., et Bechmann, A. (2021). « Situated Accountability: Ethical Principles, Certification Standards, and Explanation Methods in Applied AI », Actes de conférence de l'AAAI/ACM Conference on AI, Ethics, and Society, Online Event, Association for Computing Machinery, p. 574-585.

IEEE (The Institute of Electrical and Electronics Engineers) (2017). « Ethically Aligned Design: A Vision for Prioritizing Human Well-being with Autonomous and Intelligent Systems », The IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems, Version 2. ([https://standards.ieee.org/wp-content/uploads/import/documents/other/ead\\_v2.pdf](https://standards.ieee.org/wp-content/uploads/import/documents/other/ead_v2.pdf), consulté le 10 octobre 2023).

Jia, K., et Zhang, N. (2022). « Categorization and eccentricity of AI risks: a comparative study of the global AI guidelines », *Electronic Markets* (32:1), p. 59–71.

Khan, A. A., Badshah, S., Liang, P., Waseem, M., Khan, B., Ahmad, A., Fahmideh, M., Niazi, M., et Azeem Akbar, M. (2022). « Ethics of AI: A Systematic Literature Review of Principles and Challenges », Actes de conférence de l'International Conference on Evaluation and Assessment in Software Engineering 2022, Gothenburg, Sweden, p. 383–392.

Köhl, M. A., Baum, K., Langer, M., Oster, D., Speith, T., et Bohlender, D. (2019). « Explainability as a Non-Functional Requirement », Actes de conférence de l'IEEE 27th International Requirements Engineering Conference, Jeju Island, SKorea, p. 363–368.

Langer, M., Oster, D., Speith, T., Hermanns, H., Kästner, L., Schmidt, E., Sesing, A., et Baum, K. (2021). « What Do We Want From Explainable Artificial Intelligence (XAI)? -- A Stakeholder Perspective on XAI and a Conceptual Model Guiding Interdisciplinary XAI Research », Artificial Intelligence (296), 103473.

Lima, G., Grgić-Hlača, N., Jeong, J. K., et Cha, M. (2022). « The Conflict Between Explainable and Accountable Decision-Making Algorithms », Actes de conférence de l'ACM Conference on Fairness, Accountability, and Transparency, Seoul, Korea, p. 2103–2113.

Lipton, Z. C. (2018). « The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery », Queue (16:3), p. 31–57.

Loi, M., et Spielkamp, M. (2021). « Towards Accountability in the Use of Artificial Intelligence for Public Administrations », Actes de conférence de l'AAAI/ACM Conference on AI, Ethics, and Society, Virtual Event, USA, p. 757–766.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., et Galstyan, A. (2022). « A Survey on Bias and Fairness in Machine Learning », ACM Computing Surveys (54:6), p. 1-35.

Melchert, N. (2014). *The Great Conversation: A Historical Introduction to Philosophy* [7ed.]. Oxford University Press.

Miller, T. (2019). « Explanation in artificial intelligence: Insights from the social sciences », *Artificial Intelligence* (267), p. 1-38.

Mittelstadt, B., Russell, C., et Wachter, S. (2019). « Explaining Explanations in AI », *Actes de conférence de la Conference on Fairness, Accountability, and Transparency, Atlanta, USA*, p. 279–288.

Mukhamediev, R. I., Popova, Y., Kuchin, Y., Zaitseva, E., Kalimoldayev, A., Symagulov, A., ... et Yelis, M. (2022). « Review of Artificial Intelligence and Machine Learning Technologies: Classification, Restrictions, Opportunities and Challenges », *Mathematics*, (10:15), 2552.

Nissenbaum, H. (1996). « Accountability in a computerized society », *Science and Engineering Ethics* (2:1), p. 25-42.

Rai, A. (2020). « Explainable AI: From Black Box to Glass Box », *Journal of the Academy of Marketing Science* (48:1), p. 137-141.

Rawal, A., McCoy, J., Rawat, D. B., Sadler, B. M., et Amant, R. St. (2022). « Recent Advances in Trustworthy Explainable Artificial Intelligence: Status, Challenges, and Perspectives », *IEEE Transactions on Artificial Intelligence* (3:6), p. 852-866.

Santosh, K., et Wall, C. (2022). *AI, Ethical Issues and Explainability - Applied Biometrics*, Springer Nature Singapore, p. 1-46.

Saxena, N. A., Huang, K., DeFilippis, E., Radanovic, G., Parkes, D. C., et Liu, Y. (2019). « How Do Fairness Definitions Fare?: Examining Public Attitudes Towards Algorithmic

Schlagwein, D., et Willcocks, L. (2023). « ‘ChatGPT et al.’: The Ethics of Using (Generative) Artificial Intelligence in Research and Science », *Journal of Information Technology*, (38:3), p. 232-238.

Sovrano, F., Vitali, F., et Palmirani, M. (2021). « Making Things Explainable vs Explaining: Requirements and Challenges Under the GDPR », dans *AI Approaches to the Complexity of Legal Systems XI-XII*, V. Rodríguez-Doncel, M. Palmirani, M. Araszkiewicz, P. Casanovas, U. Pagallo and G. Sartor (eds.), Springer International Publishing, p. 169-182.

Stinson, C. (2022). « Algorithms are not neutral: Bias in collaborative filtering », *AI and Ethics* (2:4), p. 763-770.

Tomsett, R., Braines, D., Harborne, D., Preece, A., et Chakraborty, S. (2018). « Interpretable to Whom? A Role-based Model for Analyzing Interpretable Machine Learning Systems », dans *ICML Workshop on Human Interpretability in Machine Learning (WHI 2018)*, Stockholm, Sweden.

Turilli, M., et Floridi, L. (2009). « The ethics of information transparency », *Ethics and Information Technology* (11:2), p. 105-112.

Wakabayashi, D. (2018). « Self-Driving Uber Car Kills Pedestrian in Arizona, Where Robots Roam », *The New York Times*. (<https://www.nytimes.com/2018/03/19/technology/uber-driverless-fatality.html>, consulté le 16 octobre 2023).

Wieringa, M. (2020). « What to account for when accounting for algorithms: a systematic literature review on algorithmic accountability », *Actes de conférence de la Conference on Fairness, Accountability, and Transparency*, Barcelona, Spain, p. 1-18.

Zhou, A. (2019). « The intersection of ethics and AI », *AI Matters* (5:3), p. 64-69.

Zhou, J., Chen, F., & Holzinger, A. (2020, July). Towards explainability for AI fairness. In *International Workshop on Extending Explainable AI Beyond Deep Models and Classifiers* (pp. 375-386). Cham: Springer International Publishing.

# Chapitre 9 : Sécurité et vie privée en science des données

Habib Louafi et Richmond Osei

## 9.1 Introduction

Le paysage contemporain des données a connu une transformation remarquable ces dernières années, avec une explosion de la production de données, soulignant le besoin critique d'une prise de décision fondée sur les données. Dans une ère caractérisée par l'innovation numérique et les avancées technologiques, la dynamique des données a évolué à un rythme sans précédent. La numérisation rapide de notre monde a entraîné une explosion de la création de données. Chaque clic, chaque balayage, chaque achat et chaque interaction avec des services en ligne contribue à l'augmentation constante du volume de données. Alors que nous nous dirigeons vers un avenir de plus en plus interconnecté, cette explosion de la production de données ne montre aucun signe de ralentissement (Reinsel et al., 2018).

Cette augmentation est due à la convergence de plusieurs sources, les interactions avec les médias sociaux, les appareils de l'Internet des Objets (IdO) et la prolifération du contenu numérique jouant le rôle de catalyseurs clés. Les médias sociaux sont devenus une force mondiale, avec des milliards d'utilisateurs contribuant à un flux continu de données, offrant des informations précieuses sur le comportement humain et les tendances. Simultanément, les appareils IoD génèrent des flux de données en temps réel, qu'il s'agisse

d'appareils ménagers ou de capteurs industriels, ce qui alimente encore davantage le déluge de données. En outre, l'explosion du contenu numérique, allant des vidéos aux livres électroniques, modifie la façon dont l'information est diffusée.

### **9.1.1 Importance et rôle de la science des données**

Les implications de ce déluge de données sont considérables et touchent pratiquement tous les aspects de notre vie. Des secteurs tels que la santé, la finance et le commerce ont été profondément influencés par la disponibilité de vastes ensembles de données. Dans le domaine de la santé, par exemple, l'analyse des dossiers des patients et des données d'imagerie médicale a permis d'établir des diagnostics plus précis et des plans de traitement personnalisés, sauvant ainsi des vies (Obermeyer et Emanuel, 2016). Dans les entreprises, les données sont devenues un atout stratégique, permettant de mieux connaître les consommateurs, d'optimiser la chaîne d'approvisionnement et d'innover.

La nécessité d'une prise de décision fondée sur les données est bien documentée. Par exemple, dans leur livre (Chavez et al., 2019), les auteurs analysent le potentiel de transformation de l'analyse des données et affirment que les entreprises doivent exploiter les données pour acquérir un avantage concurrentiel, améliorer l'expérience de leurs clients et optimiser leurs opérations. Dans le paysage concurrentiel féroce d'aujourd'hui, les données permettent aux entreprises de prendre des décisions éclairées, de repérer les tendances émergentes et d'adapter rapidement leurs stratégies en fonction de l'évolution de la dynamique du marché.

Dans une publication de McKinsey & Company (Henke et al., 2016), les auteurs soulignent comment la science des données est en train de remodeler les industries. Ils expliquent comment les organisations qui exploitent la puissance de la science des données sont plus performantes que leurs concurrents, soulignant la demande croissante de professionnels de la science des données.

Les scientifiques des données et les analystes jouent un rôle essentiel dans la transformation des données brutes en informations exploitables, permettant aux organisations de prendre des décisions fondées sur les données à grande échelle. Ce changement d'orientation vers les sciences des données a entraîné une demande accrue de personnes compétentes en analyse de données, en apprentissage automatique et en intelligence artificielle.

### **9.1.2 Sécurité et protection de la vie privée**

Cette ère d'abondance de données apporte ses propres défis. La sécurité et la protection de la vie privée sont devenues des préoccupations majeures dans le paysage des données. Par conséquent, une demande accrue de gestion responsable des données est apparue, avec au premier plan les questions de vie privée, de sécurité et d'utilisation éthique. Pour naviguer dans ce paysage riche en données, il faut trouver un équilibre entre le potentiel de progrès et le besoin de protection et d'équité.

La sécurité et la confidentialité sont devenues importantes dans la science des données en raison des récentes violations de données et des problèmes de confidentialité. En 2018, le scandale Cambridge Analytica (Cadwalladr et Graham-Harrison, 2018) impliquant

Facebook a montré les graves conséquences d'une mauvaise manipulation des données des utilisateurs. Cet incident, comme beaucoup d'autres, a souligné l'urgence de mesures rigoureuses de protection des données.

Les organisations collectent et stockent de grandes quantités de données sensibles afin d'offrir des services personnalisés. Garantir la sécurité et la confidentialité des données collectées et partagées n'est pas seulement une obligation légale et éthique, mais aussi une mesure de confiance vis-à-vis des clients et des parties prenantes. La sécurité implique la confidentialité des données collectées, tandis que la protection de la vie privée concerne l'information des utilisateurs et l'obtention de leur consentement concernant l'ensemble du processus par lequel les données transitent.

### **9.1.3 Objectifs**

À la lumière de ces développements, ce chapitre vise à fournir aux lecteurs un aperçu de l'évolution du paysage des données et de ses implications en matière de sécurité et de protection de la vie privée. Les lecteurs peuvent s'attendre à acquérir une compréhension globale des défis et des opportunités liés aux données dans le monde contemporain, du rôle important de la science des données et du besoin critique de mesures solides en matière de sécurité et de protection de la vie privée. En outre, ce chapitre explore l'évolution des solutions éthiques et réglementaires liées à la manipulation des données.



## 9.2 Qu'est-ce que la science des données ?

Dans le monde des affaires d'aujourd'hui, où le rythme est rapide et la concurrence féroce, le terme « Science des Données » est devenu omniprésent, mais sa véritable essence et son évolution historique sont souvent méconnues.

La science des données s'est imposée comme un domaine essentiel dans le monde d'aujourd'hui, dominé par les données. Cette discussion vise à fournir une vue d'ensemble de la science des données, de ses racines, de sa portée dans divers secteurs et de son importance dans notre vie quotidienne.

### 9.2.1 Définition

La science des données peut être formellement définie comme le domaine interdisciplinaire qui utilise des méthodes, des processus, des algorithmes et des systèmes scientifiques pour extraire des connaissances et des idées à partir de données structurées et non structurées. La science des données peut être formellement définie comme suit :

*« L'étude systématique des données structurées et non structurées pour en extraire des connaissances significatives, des idées et des informations exploitables » (Dhar, 2013).*

Cette définition résume l'essence de la science des données, qui implique l'application de diverses techniques et méthodologies pour transformer les données en informations utiles. Elle transcende l'analyse traditionnelle des données en englobant un champ d'application plus large, impliquant souvent le concept des « données

volumineuses » (*big data*), l'apprentissage automatique (AA) et l'intelligence artificielle (IA). Dans son article, Chris Anderson affirme que la science des données s'efforce d'extraire des connaissances utiles des données sans avoir besoin d'hypothèses prédéfinies (Anderson, 2008).

### 9.2.2 Racines et applications de la science des données

Les racines de la science des données remontent aux statistiques et à l'exploration des données. Les statisticiens sont depuis longtemps impliqués dans l'analyse des données afin d'en extraire des informations significatives. Dans (Breiman, 2001), l'auteur reconnaît le passage des statistiques traditionnelles aux approches fondées sur les données et souligne l'importance de la modélisation prédictive dans l'analyse des données.

Les applications de la science des données touchent de nombreux domaines, soulignant sa remarquable polyvalence et son potentiel de transformation. Trois exemples notables sont :

- **L'intelligence économique** : La science des données est aujourd'hui appliquée à la veille stratégique, qui implique la collecte et l'analyse de données pour prendre des décisions commerciales stratégiques. Elle permet aux organisations de comprendre la dynamique du marché, les préférences des clients et les tendances émergentes, améliorant ainsi leur compétitivité.

Dans le monde de l'entreprise, la science des données joue un rôle indéniable dans la découverte des tendances du marché, du comportement des clients et dans l'optimisation des

stratégies commerciales. Davenport et Harris montrent l'importance de l'analyse dans l'obtention d'un avantage concurrentiel (Davenport et Harris, 2007).

- **Soins de santé** : Dans le domaine de la santé, la science des données a permis des avancées révolutionnaires. Par exemple, l'analyse prédictive est utilisée pour prévoir les épidémies, et les algorithmes d'apprentissage automatique contribuent à la détection précoce des maladies. Ces applications permettent non seulement de sauver des vies, mais aussi de réduire considérablement les coûts des soins de santé (Gligorijević et al., 2016; Chen et al., 2019).

En outre, la science des données a joué un rôle important dans l'amélioration des soins aux patients et l'offre d'une médecine personnalisée. En analysant de vastes ensembles d'informations sur les patients, les prestataires de soins de santé peuvent adapter les traitements et les interventions à chaque patient, ce qui permet de fournir des soins de santé plus efficaces et efficients (Obermeyer and Emanuel, 2016).

- **Détection d'anomalies** : La détection des anomalies est cruciale dans des domaines tels que la détection des fraudes, où l'identification de comportements ou d'événements inhabituels peut éviter des pertes importantes. Avec l'avènement des techniques avancées de science des données, telles que les algorithmes d'apprentissage profond, la précision et l'efficacité de la détection d'anomalies se sont considérablement améliorées. Ces algorithmes peuvent

analyser de vastes ensembles de données en temps réel, ce qui permet d'identifier rapidement des anomalies qui, autrement, passeraient inaperçues et risqueraient d'entraîner l'arrêt des systèmes informatiques (Ravinder et Kulkarni, 2023).

### **9.3 La sécurité dans la science des données**

La sécurité est une préoccupation réelle dans le domaine de la science des données, en particulier en raison de la dépendance croissante à l'égard des données en tant qu'atout précieux pour les organisations dans divers secteurs. Les données ne sont pas seulement un actif, elles peuvent aussi devenir un handicap si elles ne sont pas protégées de manière adéquate. Nous examinons l'importance cruciale de la sécurité dans la science des données, les types de menaces de sécurité, les meilleures pratiques pour assurer la sécurité des données, et des études de cas réels qui mettent en évidence des implémentations de sécurité réussies et non réussies.

#### **9.3.1 Les données en tant qu'atout et handicap**

Les données sont souvent considérées comme le « nouveau pétrole » dans le paysage numérique contemporain, ce qui montre leur importance dans le monde d'aujourd'hui, qui est axé sur les données. Les organisations ont exploité la puissance des données pour obtenir des informations, éclairer la prise de décision stratégique et favoriser l'innovation. Qu'il s'agisse de comprendre les préférences des consommateurs ou d'analyser les tendances du marché, les données sont devenues un atout inestimable qui alimente l'efficacité opérationnelle et l'avantage concurrentiel. Son potentiel de transformation est illustré par l'utilisation d'analyses de données

avancées et de l'intelligence artificielle, qui permettent aux organisations de découvrir des informations auparavant inaccessibles et de stimuler l'innovation dans divers secteurs. Toutefois, il faut reconnaître que ce bien précieux peut rapidement se transformer en un handicap réel s'il n'est pas protégé adéquatement (Consulting, 2018).

Les données, en tant qu'atout, ne doivent pas occulter le fait qu'elles comportent des risques inhérents, mettant en péril la sécurité des données et la protection de la vie privée. En effet, des mesures de protection inadéquates contre les violations de données, l'accès non autorisé et l'utilisation abusive des données peuvent rapidement transformer cet atout précieux en un handicap aux conséquences considérables. Ces conséquences englobent les pertes financières, l'atteinte à la réputation d'une organisation et les ramifications juridiques potentielles. Il est donc impératif que les organisations adoptent des mesures complètes de sécurité des données et de protection de la vie privée. En donnant la priorité à la protection des données, les organisations peuvent continuer à les exploiter comme une ressource précieuse, tout en atténuant les risques associés à leur mauvaise utilisation.

### **9.3.2 Conséquences d'une mauvaise sécurité**

Une mauvaise sécurité des données peut avoir de lourdes conséquences financières pour les organisations. Les violations de données et les cyberattaques peuvent entraîner des pertes financières directes sous la forme de vols d'actifs, de fraudes, mais aussi de coûts liés à l'investigation et à l'atténuation des incidents de sécurité. En

outre, les organisations peuvent également encourir des dépenses importantes en termes d'amendes réglementaires et de frais juridiques s'il s'avère qu'elles ne respectent pas les lois sur la protection des données. Ces charges financières peuvent être écrasantes, en particulier pour les petites et moyennes entreprises, et peuvent même menacer la survie de certaines d'entre elles.

L'atteinte à la réputation est une autre conséquence d'une mauvaise sécurité qui peut être tout aussi dévastatrice. En cas de violation de données, la confiance du public dans une organisation peut être gravement affectée. Les clients et les partenaires peuvent perdre confiance dans la capacité de l'organisation à protéger les informations sensibles. Cette perte de confiance peut entraîner une baisse des activités, un désintéressement de la clientèle et une détérioration des relations à long terme. Reconstruire une réputation ternie peut être un processus difficile et long, qui nécessite des efforts considérables en matière de transparence, de communication et d'amélioration des pratiques de sécurité.

La protection de la vie privée est un sujet qui exige des efforts considérables en matière de transparence, de communication et d'amélioration des pratiques de sécurité. En outre, les atteintes à la vie privée sont une préoccupation majeure associée à une sécurité des données inadéquate. Lorsque des informations personnelles sensibles sont exposées, le droit à la vie privée des individus est violé, ce qui peut entraîner des préjudices, des usurpations d'identité et des troubles émotionnels. La protection de la vie privée n'est pas seulement une obligation légale, mais aussi un impératif moral, et les organisations

qui ne respectent pas cet impératif risquent de se heurter à l'indignation et à la condamnation de l'opinion publique (Institute, 2021).

Outre les conséquences financières et les atteintes à la réputation, une mauvaise sécurité des données peut également entraîner des conséquences juridiques. Diverses réglementations relatives à la protection des données, telles que le Règlement Général sur la Protection des Données (RGPD) dans l'Union européenne, imposent des exigences strictes aux organisations en ce qui concerne la sauvegarde des données à caractère personnel. Le non-respect de ces réglementations peut entraîner des amendes substantielles et des sanctions juridiques. Les organisations qui négligent la sécurité des données peuvent se retrouver empêtrées dans des batailles juridiques coûteuses et prolongées, ce qui ne fait qu'aggraver leurs difficultés financières (Consulting, 2018).

### **9.3.3 Menaces de sécurité sur les données**

À l'ère numérique, les menaces de sécurité sont devenues de plus en plus fréquentes et sophistiquées, posant des risques importants sur les organisations et les individus. Dans ce chapitre, nous explorons trois grands types de menaces de sécurité: les violations de données, les menaces internes et les cyber-attaques. Nous mettons en lumière leur nature, leur impact et la nécessité de prendre des mesures proactives.

#### Violation de données

Les violations de données constituent une préoccupation majeure dans le monde interconnecté d'aujourd'hui. Elles impliquent un accès non autorisé à des données sensibles, entraînant l'exposition ou le vol

d'informations confidentielles. Les violations de données peuvent avoir de graves répercussions financières et sur la réputation. Un exemple frappant est la violation d'Equifax (Puig, 2019), qui a exposé les informations personnelles et financières de millions de personnes. De tels incidents soulignent l'importance d'implémenter des mesures de sécurité solides pour protéger les données.

Pour réduire le risque de violation de données, les organisations doivent donner la priorité au chiffrement des données, aux contrôles d'accès et à des audits de sécurité réguliers.

En outre, le respect des réglementations relatives à la protection des données, telles que le GDPR et le CCPA (California Consumer Privacy Act), est essentiel pour garantir un traitement responsable des données sensibles (Consulting, 2018; Bonta, 2023).

Dans le domaine de la science des données, lorsque les failles de sécurité existantes, également connues sous le nom de vulnérabilités, ne sont pas corrigées, elles peuvent affecter les ensembles de données (datasets) collectées ainsi que les modèles formés. En effet, si des acteurs malveillants parviennent à introduire des données fausses ou trompeuses dans les données collectées ou dans le trafic reniflé, les modèles générés ne sont plus représentatifs et sont donc trompeurs. D'autre part, les failles de sécurité peuvent conduire à des fuites de données, ce qui affecte la confidentialité des données et la vie privée des individus. Dans ce cas, des informations sensibles sur les clients et les utilisateurs peuvent être révélées, ce qui est évidemment préjudiciable aux utilisateurs et aux clients, mais aussi à la réputation de l'entreprise, à ses revenus et à sa réputation. Il n'est pas nécessaire



de mentionner l'impact d'un modèle d'apprentissage automatique trompeur, lorsqu'il est déployé. Dans des contextes tels que les soins de santé ou l'armée, des modèles mal entraînés peuvent avoir un impact grave sur des vies humaines et des missions critiques.

### Menaces internes

Les menaces internes constituent un défi unique car elles proviennent de l'intérieur d'une organisation. Elles impliquent des employés ou des personnes de confiance qui abusent de leurs privilèges d'accès à des fins malveillantes. Les statistiques révèlent que les menaces internes, actions involontaires et malveillantes, constituent un problème important conduisant à des failles de sécurité. Cela montre l'importance non seulement des mesures de protection technologiques, mais aussi des programmes de formation et de sensibilisation des employés.

Pour faire face aux menaces internes, les organisations devraient mettre en œuvre un modèle d'accès au moindre privilège, former régulièrement leurs employés aux meilleures pratiques de sécurité et établir des politiques et des procédures claires pour le signalement d'activités suspectes. La vigilance des employés et une culture de la sécurité sont essentielles pour prévenir et détecter les menaces internes (CISA, 2020).

Dans le domaine de la science des données, des acteurs internes malveillants ou des employés mal formés peuvent créer des failles de sécurité. Ces dernières exposent l'ensemble des données de l'organisation et les modèles statistiques sur lesquels elle s'appuie. Comme nous l'avons expliqué précédemment, ces types de failles de

sécurité sont sensibles et doivent être atténuées par tous les moyens, en particulier dans les services critiques, tels que les systèmes de santé et les missions militaires.

### Cyberattaques

Les cyberattaques englobent un large éventail de tactiques et de stratégies visant à compromettre la sécurité des données. Ces attaques comprennent les tentatives d'hameçonnage, par lesquelles des acteurs malveillants trompent les individus en leur faisant révéler des informations sensibles, les attaques par rançongiciel qui chiffrent les données jusqu'au paiement d'une rançon, et les attaques par déni de service distribué (DDoS) qui perturbent les services en ligne.

La nature évolutive des cyberattaques oblige les organisations à rester vigilantes et à adapter leurs stratégies de sécurité. Cela implique la mise en œuvre de systèmes avancés de détection des menaces, d'un filtrage robuste des courriels pour lutter contre les tentatives d'hameçonnage et de plans de reprise après incidents pour atténuer l'impact des cyberattaques. Les organisations devraient également collaborer avec des experts en cybersécurité pour se tenir informées des nouvelles menaces et vulnérabilités.

La science des données n'est pas une exception, car elle traite des données collectées auprès des utilisateurs et des clients, et des systèmes formés, basés sur l'apprentissage automatique et les algorithmes d'IA. Lorsque la sécurité de l'organisation est compromise, les données ainsi que les modèles entraînés peuvent l'être également. Cela souligne l'importance de protéger, en particulier, les ressources critiques de l'organisation et les serveurs, où

les données sensibles sont stockées et où les systèmes et les processus critiques sont exécutés.

#### 9.3.4 Techniques de sécurité

Les organisations traitent de grandes quantités d'informations sensibles et il est essentiel de garantir la confidentialité et l'intégrité de ces données. Plusieurs techniques de sécurité peuvent être utilisées, chacune ayant son champ d'application et ses limites. Dans ce chapitre, nous nous concentrons sur trois des meilleures techniques de sécurité qui peuvent être appliquées à la science des données, à savoir le chiffrement, la transmission sécurisée des données et le contrôle d'accès.

##### Chiffrement

Le chiffrement est une mesure de sécurité fondamentale qui joue un rôle central dans la protection des données. Il consiste à convertir les données en un format inintelligible, ce qui les rend inutilisables pour quiconque ne dispose pas de la clé de déchiffrement appropriée. Il est essentiel d'utiliser des méthodes de chiffrement puissantes pour protéger les informations sensibles d'un accès non autorisé. Le chiffrement garantit que même si un acteur malveillant accède aux données, il ne peut pas en déchiffrer le contenu sans la clé de déchiffrement. En mettant en œuvre le chiffrement, les organisations ajoutent une couche supplémentaire de sécurité à leurs données, atténuant ainsi les risques associés aux violations de données et à l'accès non autorisé (Stallings, 2017). Des algorithmes de chiffrement avancés, tels que l'AES (*Advanced Encryption Standard*) (Rijmen et Daemen, 2001) et le RSA (*Rivest-Shamir-Adleman*) (Rivest et al., 1978), offrent une protection

solide et sont largement adoptés dans divers secteurs, tels que la finance et les soins de santé.

Les algorithmes de chiffrement peuvent être classés en deux catégories principales, à savoir les algorithmes symétriques (également appelés systèmes à clé secrète/partagée) et les algorithmes asymétriques (également appelés systèmes à clé publique). Dans le premier cas, une seule clé est utilisée et doit être gardée secrète pour les deux parties impliquées dans le chiffrement et le déchiffrement. Dans le second, deux clés, privée et publique, sont utilisées. La clé publique est utilisée pour le chiffrement du message, tandis que la clé privée est utilisée pour son déchiffrement. La Figure 9-1 et la Figure 9-2 illustrent, respectivement, le chiffrement symétrique à l'aide de l'algorithme AES et le chiffrement asymétrique à l'aide de l'algorithme RSA. En termes de mise en œuvre, le Code source 9.1 montre comment l'algorithme AES peut être utilisé en langage Python. Il est à noter que différentes implémentations, utilisant différentes bibliothèques, sont disponibles dans la littérature. Pour plus de détails sur le chiffrement symétrique et asymétrique, le lecteur est invité à consulter le livre de Stallings et Brown (Stallings and Brown, 2017).

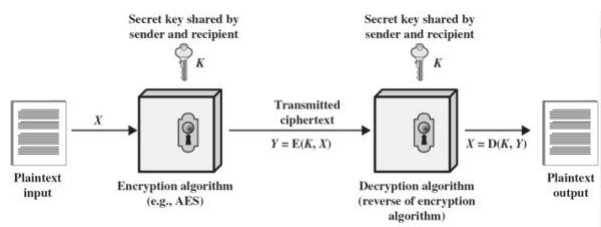


Figure 9-1. Chiffrement symétrique (Stallings and Brown, 2017).

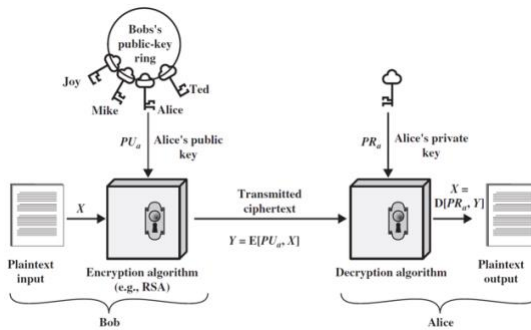


Figure 9-2. Chiffreage asymétrique (Stallings and Brown, 2017).

#### Code Python

```
# Importer les bibliothèques nécessaires
import os
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes

# Générer une clé aléatoire pour le chiffrement et un vecteur initial aléatoire
key = os.urandom(32)
iv = os.urandom(16)
# Initialiser le chiffrement, en utilisant l'algorithme AES
# et le mode de fonctionnement CBC
cipher = Cipher(algorithms.AES(key), modes.CBC(iv))
# Créer un chiffreur
encryptor = cipher.encryptor()
# Données à chiffrer
data_to_encrypt = b"a secret message"
# Imprimer les données à chiffrer
print("Data to be encrypted: ", data_to_encrypt)
# Chiffrer les données
ct = encryptor.update(data_to_encrypt) + encryptor.finalize()
# Imprimer les données chiffrées
print("Encrypted data: ", ct)
# Définir un déchiffreur
decryptor = cipher.decryptor()
# Déchiffrer des données
dt = decryptor.update(ct) + decryptor.finalize()
# Imprimer les données chiffrées
print("Decrypted data: ", dt)
```

Code source 9.1. Chiffrement/déchiffrement avec l'algorithme AES.

### Transmission sécurisée des données

La sécurisation des données pendant leur transmission est un autre aspect crucial de la sécurité des données. Lorsque les données sont en

transit, elles sont susceptibles d'être interceptées par des acteurs malveillants. Pour éviter cela, les organisations utilisent généralement des protocoles tels que « Secure Sockets Layer (SSL) » et « Transport Layer Security (TLS) » pour sécuriser la transmission des données. Les protocoles SSL et TLS établissent des connexions chiffrées entre les utilisateurs et les serveurs, garantissant que les données échangées entre eux restent confidentielles et protégées contre les écoutes clandestines. Ces protocoles sont largement utilisés dans les transactions en ligne, la communication par courrier électronique et la navigation sur le web. En mettant en œuvre SSL/TLS, les organisations ne protègent pas seulement les données sensibles, mais instaurent également la confiance avec leurs utilisateurs, car les mécanismes de transmission sécurisée sont la marque d'un traitement responsable des données (Dierks et Rescorla, 2008).

En science des données, les données stockées sur des serveurs ou collectées en direct doivent également être transférées d'un serveur à l'autre. Par exemple, dans les réseaux traditionnels ou IoT, les appareils sont dispersés sur de vastes zones, et le trafic qu'ils génèrent est donc transféré de ces appareils vers des serveurs centralisés ou infonuagiques pour être traité. Par conséquent, des mesures de sécurité (par exemple, SSL ou TLS) doivent être mises en œuvre pour sécuriser les données transférées. Sinon, le trafic capturé peut être altéré, ce qui peut induire en erreur les algorithmes d'apprentissage automatique lors de la création de systèmes prédictifs.

En outre, dans les systèmes distribués basés sur la science des données, comme l'apprentissage fédéré, les systèmes prédictifs sont

formés sur différents sites du réseau. Ensuite, ces modèles entraînés sont transférés d'un serveur à l'autre afin de capturer l'ensemble du comportement du système. Cela rend les modèles vulnérables aux attaques si des mesures de sécurité appropriées ne sont pas mises en œuvre sur les systèmes de transmission de données.

### Contrôle d'accès

La mise en œuvre de mécanismes de contrôle d'accès est essentielle pour garantir que seules les personnes autorisées peuvent accéder à des données spécifiques. Le contrôle d'accès basé sur les rôles « Role-based Access Control (RBAC) » et les méthodes d'authentification puissantes sont des outils efficaces pour limiter l'accès aux informations sensibles. Le contrôle d'accès basé sur les rôles (RBAC), qui est le concept le plus largement déployé, attribue des rôles et des autorisations aux utilisateurs en fonction de leurs fonctions et de leurs responsabilités, ce qui garantit que les personnes n'ont accès qu'aux données nécessaires à leurs tâches, réduisant ainsi le risque d'accès non autorisé. Les méthodes d'authentification, telles que l'authentification multifactorielle (AMF), ajoutent une couche supplémentaire de sécurité, obligeant les utilisateurs à fournir plusieurs formes de vérification, telles qu'un mot de passe et un balayage d'empreintes digitales. Il est ainsi beaucoup plus difficile pour les acteurs malveillants d'obtenir un accès non autorisé à des systèmes et des données sensibles (Stallings, 2017).

De même, les données, au repos ou en transit, ne doivent être manipulées que par des utilisateurs autorisés. Dans le cas contraire, des acteurs malveillants peuvent altérer les données ou les modèles

entraînés. Ces acteurs malveillants peuvent être des attaquants extérieurs, mais aussi internes, comme nous l'avons expliqué précédemment. Il n'est pas nécessaire de mentionner l'impact de telles actions sur les systèmes affectés, tels que les établissements de soins de santé et les installations militaires.

### 9.3.5 Études de cas

Dans le paysage en constante évolution de la science des données, une mise en œuvre réussie de mesures de sécurité robustes est primordiale pour sauvegarder des données précieuses. Deux études de cas notables illustrent les résultats variables de la mise en œuvre de la sécurité au sein d'organisations de premier plan.

#### Une mise en œuvre réussie de la sécurité

Des géants technologiques renommés, comme Amazon et Google, ont fait preuve d'une réussite exemplaire dans la mise en œuvre de mesures de sécurité complètes. Leurs stratégies englobent une approche à multiples facettes qui comprend le chiffrement, la surveillance continue et la détection proactive des menaces. Ces organisations ont constamment investi dans des technologies et des pratiques de sécurité de pointe pour protéger leurs ressources de données.

L'utilisation intensive du chiffrement par Amazon pour protéger les informations sensibles des clients, et la surveillance continue des menaces et des brèches potentielles par Google, ont contribué de manière significative à leur succès en matière de sécurité des données.



Les organisations qui accordent la priorité à des stratégies de sécurité globales protègent non seulement leurs informations sensibles, mais renforcent également la confiance de leurs clients et parties prenantes.

#### Une mise en œuvre de la sécurité qui n'a pas abouti

En revanche, des organisations comme Facebook et Equifax ont fait l'objet d'un examen minutieux de la part du public et ont subi des conséquences juridiques en raison de lacunes dans la sécurité des données. Le scandale Facebook-Cambridge Analytica est un excellent exemple d'utilisation abusive de données et de mesures de sécurité inadéquates, qui a conduit à une rupture de la confiance des utilisateurs. La violation massive des données d'Equifax a exposé les informations personnelles sensibles de millions de personnes, entraînant des répercussions juridiques et financières.

Ces deux cas servent d'exemples d'avertissement, soulignant la nécessité de mesures de sécurité proactives et de pratiques responsables en matière de traitement des données. Ils montrent que même les organisations disposant d'importantes ressources en matière de données ne sont pas à l'abri des attaques contre la sécurité et la vie privée. Elles doivent donner la priorité à la sécurité afin d'éviter les dommages financiers et de réputation associés aux violations de données (Hinds et al., 2020).

### **9.4 La protection de la vie privée dans la science des données**

La protection de la vie privée est une préoccupation fondamentale dans le domaine de la science des données, où la collecte, le stockage

et le partage des données sont des éléments essentiels du processus analytique. Les scientifiques des données et les organisations doivent faire face à des problèmes complexes de protection de la vie privée afin de protéger les droits des individus et de respecter les normes éthiques, tout en extrayant des informations précieuses des données. Ce chapitre explore les différents aspects de la protection de la vie privée dans les sciences des données, y compris les définitions, les problèmes de protection de la vie privée, la législation et les lignes directrices pertinentes, les techniques de préservation de la vie privée et les études de cas. Nous soulignons également les cas où la vie privée a été compromise et les leçons qui en ont été tirées.

#### **9.4.1 Définition de la vie privée**

La vie privée, en tant que droit humain fondamental, a évolué au fil des siècles. Toutefois, à l'ère numérique contemporaine, la définition de la vie privée a subi d'importantes transformations en raison des progrès rapides de la technologie, qui ont brouillé les frontières entre les domaines public et privé. Elle englobe la sauvegarde des données personnelles, le contrôle des flux d'informations et la préservation de l'autonomie des individus dans un monde où chaque clic, chaque glissement et chaque frappe laisse une empreinte numérique.

Ce paysage dynamique de la vie privée touche à la fois les espaces physiques et virtuels, ce qui rend son analyse encore plus complexe. C'est pourquoi un examen méticuleux et une réflexion approfondie sont nécessaires pour le traitement des données (Becker, 2019).

### 9.4.2 Catégories de protection de la vie privée

La protection de la vie privée dans le domaine de la science des données peut être classée en deux catégories principales :

- **Vie privée individuelle** : Cette dimension concerne la sauvegarde des informations personnelles d'un individu et la préservation de son droit à contrôler l'utilisation de ses données. Elle implique la protection des données sensibles, telles que les informations personnelles identifiables « Personally Identifiable Information (PII) », contre l'accès ou la divulgation non autorisés. La protection de la vie privée est un aspect fondamental de l'éthique des données et implique l'obtention d'un consentement éclairé avant la collecte et le traitement des données personnelles. Par exemple, la collecte et l'analyse d'informations personnellement identifiables (PII) sans consentement constituent une violation de la vie privée, ce qui souligne l'importance de pratiques éthiques en matière de traitement des données (Consulting, 2018).
- **Vie privée collective** : En revanche, la protection collective de la vie privée se concentre sur les droits à la vie privée des groupes, communautés ou organisations. Il s'agit de protéger les données qui, lorsqu'elles sont agrégées ou analysées collectivement, peuvent révéler des schémas ou des caractéristiques sensibles ayant une incidence sur la vie privée d'un groupe. La protection collective de la vie privée souligne l'importance de préserver la confidentialité des données liées à un groupe, telles que les informations démographiques ou

les tendances comportementales. Par exemple, l'agrégation de données sur les dossiers médicaux d'un quartier spécifique pourrait révéler des tendances sensibles en matière de santé, ce qui souligne la nécessité de préserver la confidentialité collective (Bhave et al., 2020). Ces informations, si elles sont révélées, peuvent être exploitées par exemple par les compagnies d'assurance pour modifier leurs primes d'assurance, ce qui pourrait avoir un impact négatif et menacer la santé et le budget du groupe de population ciblé.

#### **9.4.3 Questions relatives à la protection de la vie privée**

Les questions relatives à la protection de la vie privée se posent à différents stades du cycle de vie des processus des sciences des données. La collecte, le stockage et le partage des données sont les deux phases critiques au cours desquelles ces questions se posent.

##### Collecte des données

La collecte des données est une phase cruciale de la science des données, mais elle est lourde d'implications éthiques liées à la protection de la vie privée. La collecte de données sans le consentement éclairé des personnes ou sans qu'elles en soient informées peut potentiellement porter atteinte à leur droit à la vie privée. Le consentement éclairé est la pierre angulaire d'une collecte de données éthique, exigeant que les personnes soient pleinement conscientes des données collectées, de leur finalité et de la manière dont elles seront utilisées (Mertens, 2018).

## Stockage des données

Les décisions relatives aux emplacements et à la manière dont les données sont stockées jouent un rôle important dans la protection de la vie privée. Les techniques de chiffrement et d'anonymisation des données (qui sont détaillées plus loin dans ce chapitre), sont essentielles pour protéger les données stockées contre tout accès non autorisé. Ces mesures de sécurité permettent de garantir que, même en cas de violation des données, les données exposées restent sécurisées et inintelligibles (par exemple, chiffrées) pour les utilisateurs non autorisés.

## Partage de données

Le partage des données joue un rôle essentiel dans divers domaines, facilitant la recherche collaborative, les partenariats commerciaux et l'échange d'informations précieuses. Cependant, l'acte de partager des données avec des tiers soulève également d'importantes questions en matière de protection de la vie privée. En l'absence d'autorisations et de contrôles appropriés, le partage de données peut potentiellement exposer des informations sensibles, mettant ainsi en péril la vie privée des individus.

L'un des principaux défis du partage de données est le risque de partage non autorisé ou non contrôlé. Lorsque les organisations partagent des données sans mesures de protection adéquates, elles risquent de divulguer par inadvertance des PII ou des données commerciales sensibles à des destinataires involontaires. Il peut en résulter des atteintes à la vie privée, des conséquences juridiques et des atteintes à la réputation.

Pour répondre à ces préoccupations, les organisations doivent établir de solides politiques de partage des données et mettre en place des contrôles d'accès stricts. Le respect de la législation et des lignes directrices pertinentes est la voie à suivre pour garantir des politiques de partage des données responsables et totalement transparentes (Sweeney, 2002). À cette fin, les guides RGPD et CCPA, par exemple, peuvent être mis en œuvre dans les organisations en ce qui concerne le flux de données.

#### **9.4.4 Cadres éthiques et réglementaires**

Dans le sillage des scandales liés à la protection de la vie privée et des violations de données, le cadre éthique et réglementaire entourant les données a également évolué de manière significative. Le règlement général sur la protection des données (RGPD) de l'Union européenne, mis en œuvre en 2018 (Consulting, 2018), a établi une norme mondiale pour la confidentialité et la protection des données. Il a obligé les organisations du monde entier à réévaluer leurs pratiques de traitement des données, en mettant au premier plan les droits des individus à contrôler leurs données. Par ailleurs, la loi Californienne sur la protection de la vie privée des consommateurs « California Consumer Privacy Act (CCPA) » (Bonta, 2023), partage certaines des bases de la protection des données personnelles. Parmi l'ensemble des droits accordés à l'utilisateur, nous pouvons citer les suivants :

- L'utilisateur a le droit de savoir comment ses informations personnelles sont collectées, utilisées et partagées par les entreprises.

- L'utilisateur a le droit, dans une certaine mesure, de supprimer les informations personnelles qui ont été collectées par les entreprises.
- L'utilisateur a le droit de refuser la vente ou le partage de ses informations personnelles.
- L'utilisateur a le droit d'être protégé contre toute forme de discrimination dans l'exercice de ses droits au titre de la CCPA.

En outre, de nombreux pays et régions ont introduit ou mis à jour leurs lois sur la protection des données afin de s'aligner sur l'évolution des préoccupations en matière de protection de la vie privée. Ce paysage en évolution témoigne de la nécessité pour les organisations d'adopter des pratiques solides de gouvernance des données et des mesures de conformité.

#### **9.4.5 Protection différentielle de la vie privée**

La confidentialité différentielle (*Differential Privacy*) fait référence à la formulation mathématique de la vie privée (Dwork, 2008). Elle consiste à introduire un bruit contrôlé ou une perturbation dans les données, ce qui protège efficacement les identités individuelles tout en permettant une analyse pertinente des données. Dans le domaine des données du trafic réseau par exemple, où les détails les plus fins peuvent être particulièrement sensibles, la confidentialité différentielle offre une méthode robuste de préservation de la confidentialité. Elle permet d'extraire des informations précieuses des données sans compromettre l'anonymat des individus ou des entités.

Un algorithme qui analyse l'ensemble des données pour évaluer certaines statistiques, par exemple, est considéré comme différentiellement privé si les données individuelles ne peuvent pas être déduites en regardant simplement la sortie de l'algorithme. En d'autres termes, le comportement de l'algorithme ne change pas de manière significative si l'on ajoute ou supprime simplement des données individuelles à l'ensemble de données (entrée de l'algorithme). Cela garantit formellement que les données individuelles ne peuvent pas être divulguées.

**Exemple 9.1 :** Supposons que nous disposions d'une base de données de dossiers médicaux contenant des informations sur le fait que chaque patient est atteint d'une maladie spécifique, telle que le diabète. Nous voulons déterminer le pourcentage de patients atteints de diabète sans révéler les données individuelles des patients.

Soit  $D$  l'ensemble de données originale et  $D'$  un ensemble de données qui diffère de  $D$  par les données d'un patient. Nous définissons une fonction d'interrogation  $f$  comme suit :

$f(D) =$  *Pourcentage des patients dans  $D$  qui ont le diabète*

Pour obtenir une confidentialité différentielle, nous introduisons du bruit dans le résultat de la requête. Soit  $\epsilon$  le paramètre de confidentialité, qui contrôle la quantité de bruit ajoutée. Nous pouvons ajouter par exemple un « bruit de Laplace » à la requête, comme suit :

$$f(\hat{D}) = f(D) + \text{Laplace}\left(\frac{1}{\epsilon}\right).$$



La distribution de Laplace a une fonction de densité de probabilité qui est donnée par :

$$Laplace(x | \mu, b) = \frac{1}{2b} e^{-\frac{|x-\mu|}{b}}$$

où  $\mu$  est le paramètre de localisation et  $b$  le paramètre d'échelle. Dans notre cas,  $\mu = 0$  et  $b = 1/\epsilon$ .

La confidentialité différentielle garantit que pour deux ensembles de données voisines  $D$  et  $D'$  (différant par les données d'un individu), la probabilité (notée par  $P$ ) d'observer un résultat particulier (noté par  $r$ ) de requête est borné comme suit :

$$\frac{P(f(D)=r)}{P(f(D')=r)} \leq e^\epsilon.$$

#### 9.4.6 Techniques de préservation de la vie privée

Comme expliqué précédemment, le traitement des données et l'analyse sont essentiels pour offrir des services personnalisés et aider à la prise de décision. En revanche, toutes les entreprises ne disposent pas de l'expertise nécessaire pour mener et mettre en œuvre des techniques de science des données afin de prendre des décisions à partir des données. Pour cette raison, les entreprises ont recours à l'externalisation des données à des tiers pour obtenir des informations sur leurs données et des recommandations. Cependant, en raison de la présence d'informations sensibles, les entreprises sont réticentes à partager leurs données. La question est donc de savoir comment partager les données tout en protégeant les informations sensibles et en respectant les normes réglementaires et éthiques.

Pour protéger les droits des personnes et maintenir les normes éthiques, il est essentiel de mettre en œuvre les meilleures pratiques

qui garantissent la confidentialité des données. Deux pratiques fondamentales à cet égard sont : la minimisation et l'anonymisation des données.

Minimisation des données

Le principe de la minimisation des données est de limiter la quantité d'informations sensibles détenues par les organisations, réduisant ainsi le risque d'exposition en cas de violation ou d'accès non autorisé. En ne collectant que ce qui est nécessaire, les organisations renforcent non seulement la protection de la vie privée, mais rationalisent également les processus de gestion des données. La minimisation des données est conforme au principe fondamental « collecter moins, protéger moins », qui fait partie intégrante d'un traitement responsable des données (Klosek, 2018).

**Exemple 9.2 :** Le code source 9.2 présente un exemple simple de code Python qui montre comment la minimisation des données peut être appliquée à un ensemble de données.

<p>Code Python</p> <pre>import pandas as pd # Créer un échantillon de données data = {     'Name': ['Alice', 'Bob', 'Charlie'],     'Email': ['alice@example.com', 'bob@example.com', 'charlie@example.com'],     'Phone': ['123-456-7890', '555-555-5555', '999-999-9999'],     'Address': ['123 Main St', '456 Elm St', '789 Oak St'], } # Convertir les données en DataFrame Pandas df = pd.DataFrame(data) # Imprimer l'ensemble de données original print("Original Dataset: \n", df)  # Minimisation des données : Supprimer les champs sensibles (Email, Téléphone) df_minimized = df.drop(['Email', 'Phone'], axis=1)</pre>
---

```
# Afficher l'ensemble de données minimisé  
print("Minimized Dataset:", df_minimized)
```

*Code source 9.2. Minimisation des données.*

### Anonymisation des données

L'anonymisation consiste à transformer des données sensibles en une forme qui dissimule l'identité des individus, tout en conservant l'utilité des données pour l'analyse. Elle peut s'appliquer à toute information sensible, qui peut être différente d'un contexte à l'autre.

Des méthodes telles que le masquage et la symbolisation des données sont couramment employées pour y parvenir. Par exemple, les techniques de masquage permettent de remplacer les noms, les adresses ou toute autre information sensible par des pseudonymes, ce qui garantit que les données anonymisées ne peuvent pas être reliées à des personnes spécifiques. Plusieurs méthodes d'anonymisation sont proposées dans la littérature, et les plus utilisées sont dans ce chapitre. Avant d'explorer ces méthodes, examinons l'utilité des données et sa relation avec la confidentialité des données. Les techniques d'anonymisation établissent un équilibre entre l'utilité des données et la protection de la vie privée, permettant aux organisations d'extraire des informations utiles tout en respectant la confidentialité des personnes (Solove, 2006). L'utilité des données fait référence à la qualité des données nécessaires à l'analyse, qui peut être totalement ou partiellement perdue lorsque les données sont anonymisées de manière agressive.

## Vie privée et utilité des données

En science des données, où la collecte, l'analyse et l'utilisation de volumes importants d'informations soutiennent la prise de décision, l'équilibre délicat entre la protection de la vie privée et l'utilité des données n'a jamais été aussi prononcé. D'une part, les individus et les organisations s'inquiètent de plus en plus de l'utilisation abusive des informations personnelles, compte tenu de l'augmentation constante de l'empreinte numérique. D'autre part, la valeur intrinsèque des données dans l'élaboration des politiques, l'avancement de la recherche et le développement des entreprises reste indéniable. En réponse à ce défi majeur, les techniques d'anonymisation sont devenues des outils indispensables, permettant une utilisation responsable des données tout en préservant la vie privée des individus. Le thème omniprésent de la protection de la vie privée par rapport à l'utilité des données souligne le dilemme fondamental auquel se heurte l'anonymisation des données. D'une part, on ne saurait trop insister sur l'importance primordiale de la protection de la vie privée des personnes et des entités, en particulier dans le contexte du trafic réseau, qui peuvent contenir des informations confidentielles sur les personnes et les entités. Inversement, les données présentent une immense utilité dans divers domaines, notamment la sécurité des réseaux, l'amélioration des performances et la recherche avancée. Le cœur du problème apparaît lorsque l'exploitation de l'utilité des données entre en conflit avec la protection de la vie privée.

**Vie privée :** La sensibilisation croissante au droit à la vie privée et aux violations de données a suscité des inquiétudes légitimes quant à

la collecte et à la diffusion des informations personnelles. Les individus s'inquiètent de plus en plus de la manière dont leurs données sont utilisées, partagées et potentiellement exploitées sans leur consentement. En outre, des réglementations strictes en matière de protection de la vie privée, telles que les guides RGPD et CCPA, soulignent l'importance de protéger la vie privée des individus dans le cadre du traitement des données.

**Utilité des données :** Sans accès à des données pertinentes et complètes, les progrès dans divers domaines, seraient gravement entravés, notamment dans les domaines de la santé, de la finance et de l'éducation. L'équilibre entre le besoin d'utilité des données et le respect de la vie privée est un défi complexe et évolutif.

Dans les traces de réseaux, par exemple, les adresses IP sont des informations sensibles qui doivent être dissimulées. En effet, si ces informations sont révélées, via l'externalisation du trafic réseau, elles peuvent être exploitées par des acteurs malveillants pour identifier les nœuds les plus faibles du réseau (ceux qui ont un trafic important, par exemple). Dans ce cas, les criminels peuvent cibler ces nœuds avec plus de trafic pour accentuer le trafic réseau et ainsi perpétrer des attaques DoS/DDoS. Toutefois, si les adresses IP sont dissimulées, la relation entre elles, qui représente l'utilité des données dans cet exemple, est perdue. Par conséquent, tout modèle entraîné sur ces données sera trompeur.

Dans les systèmes de soins de santé, les identités des patients sont sensibles et doivent être dissimulées. Cependant, la relation parentale

entre les patients est perdue. Dans ce cas, tout diagnostic reposant sur la relation parentale sera incomplet et donc trompeur.

Il est donc très important de trouver un compromis entre la protection de la vie privée (quantité de données à dissimuler) et l'utilité des données (quantité de données à révéler). À cette fin, diverses techniques d'anonymisation sont proposées dans la littérature. Les plus importantes d'entre elles sont présentées dans les sections suivantes.

### Méthodes d'anonymisation

Dans les systèmes en réseau par exemple, les techniques d'anonymisation sont essentielles pour garantir que les traces du réseau sont correctement anonymisées, empêchant ainsi l'identification d'individus ou d'entités dans les données. En général, l'anonymisation est appliquée dans d'autres domaines aussi. Dans cette section, nous explorons ces techniques, en commençant par les plus élémentaires.

### **La randomisation**

La randomisation consiste à remplacer les informations sensibles par des valeurs aléatoires. Il s'agit d'une méthode relativement simple à mettre en œuvre, mais qui a des effets importants sur l'utilité. Cette dernière diminue de manière significative après l'anonymisation. Par exemple, l'utilisation de chaînes aléatoires pour les noms de villes ne permet pas d'effectuer des analyses significatives. En effet, si deux villes appartiennent au même pays, cette information est perdue après l'anonymisation. En outre, si la fonction (ou le système) de randomisation n'est pas hautement sécurisée, la méthode

d'anonymisation sera vulnérable aux attaques par fréquence statistique, comme tout générateur de nombres aléatoires. En d'autres termes, la sécurité de la méthode d'anonymisation basée sur la randomisation dépend de la sécurité du générateur de nombres aléatoires sous-jacent utilisé.

**Exemple 9.3 :** Considérons un ensemble de données contenant une liste d'individus avec leurs informations sensibles et non sensibles, comme indiqué dans le Tableau 9-1.

Tableau 9-1. Ensemble de données avant l'anonymisation.

	Nom	Age	Ville	NomDeRue	NoCivique	Salaire
0	John	30	Montréal	Saint-Denis	10	60.000
1	Albert	45	Toronto	King's college	88	75.000
2	Oliver	25	Calgary	20 avenue west	34	50.000
3	Pamela	33	Vancouver	Island Highway	45	65.000

En appliquant la méthode d'anonymisation basée sur la randomisation à cet ensemble de données, les champs « Nom », « Ville » et « NomDeRue » peuvent être remplacés par des chaînes de caractères générés de manière aléatoire, comme le montre le Tableau 9-2.

Tableau 9-2. Ensemble de données après anonymisation, à l'aide de la méthode de randomisation. Dans cet exemple, les champs « Nom », « Ville » et « NomDeRue » sont remplacés par des chaînes de caractères générés aléatoirement.

	Nom	Age	Ville	NomDeRue	NoCivique	Salaire
0	OUIjk8VN4w	30	ZsNeY1liD8	yKan2tsXp4	10	60.000
1	vMsTYWsgam	45	iMGqadzRdp	yvc3X6nzLC	88	75.000
2	8rYb3vjSm4	25	44TRZoWa3v	NRvSgOG85y	34	50.000
3	GKD9LXeq67	33	nVRkYBD4dO	eSCFyZScMz	45	65.000

En utilisant Python, l'anonymisation basée sur la randomisation peut être mise en œuvre, comme le montre le Code source 9.3. Dans ce

morceau de code, les trois champs « Nom », « Ville » et « NomDeRue » sont remplacés par des valeurs aléatoires.

```
Code Python

# Importer les bibliothèques nécessaires
import pandas as pd
import random
import string

# Fonction d'anonymisation qui opère sur une base de données
def randomization_anony(df):
    for column in df.columns:
        # vérifie si les valeurs des colonnes sont des chaînes de caractères
        if df[column].dtype == 'O':
            for i in range(len(df)):
                # générer une chaîne aléatoire
                random_string = random.choices(string.ascii_letters + string.digits, k=10)
                # Remplacer le champ original par la chaîne aléatoire
                df[column][i] = ".join(random_string)
    return df

# Chargement de l'ensemble de données original
df = pd.read_csv('original_data.csv')
# Imprimer le jeu de données original
print("Original Dataset")
print(df)
# Appeler la fonction d'anonymisation
df_randomized = randomization_anony(df)
# Imprimer l'ensemble de données anonymisées
print("Anonymized dataset using randomization")
print(df_randomized)
```

*Code source 9.3. Anonymisation par la méthode de randomisation.*

## **La permutation**

L'anonymisation par la méthode de permutation n'est qu'une variante de la méthode de randomisation. Au lieu de remplacer les données sensibles par des valeurs aléatoires, les données sensibles sont permutées. La permutation peut se faire par colonne ou par ligne. Comme la méthode de randomisation, il s'agit d'une méthode relativement simple à mettre en œuvre, mais qui réduit l'utilité des données. Par exemple, si l'anonymisation est effectuée par colonne, la



corrélation, les idées et les relations entre les colonnes seront perdues. En termes de confidentialité, la fonction de permutation (ou le système) n'est pas sécurisée et la méthode d'anonymisation sera donc vulnérable aux attaques par fréquence statistique, comme la méthode de randomisation.

**Exemple 9.4 :** Si on reprend l'exemple de l'ensemble de données présenté dans le Tableau 9-1, l'ensemble de données peut être anonymisé à l'aide d'une permutation par colonne, comme indiqué dans le Tableau 9-3.

*Tableau 9-3. Ensemble de données après anonymisation, à l'aide de la méthode de permutation par colonne. Dans cet exemple, les valeurs des colonnes « Ville », « NomDeRue », « NoCivique » et « Salaire » sont permutées.*

	Nom	Age	Ville	NomDeRue	NoCivique	Salaire
0	John	30	Vancouver	Island Highway	45	65.000
1	Albert	45	Calgary	20 avenue west	34	50.000
2	Oliver	25	Montreal	Saint-Denis	10	60.000
3	Pamela	33	Toronto	King's college	88	75.000

**Agrégation/généralisation**

L'agrégation (également appelée généralisation) consiste à regrouper des données provenant de plusieurs champs (par exemple, des individus) afin de créer de nouvelles vues groupées des données. En d'autres termes, au lieu de stocker et d'externaliser les données de chaque champ séparément, les données des champs sensibles peuvent être regroupées (agrégées) en différents groupes ou pages.

**Exemple 9.5 :** Considérons l'ensemble de données présenté dans le Tableau 9-1. En appliquant la méthode d'anonymisation basée sur l'agrégation à cet ensemble de données, les valeurs des champs « Age » et « Salaire » peuvent être remplacées par des fourchettes d'âge et des fourchettes de salaire, respectivement, comme le montre le Tableau 9-4.

*Tableau 9-4. Ensemble de données après anonymisation, en utilisant la méthode d'agrégation. Dans cet exemple, les valeurs des champs « Age » et « Salaire » sont remplacées par des tranches d'âge et des tranches de salaire, respectivement.*

	Nom	Age	Ville	NomDeRue	NoCivique	Salaire
0	John	21-30	Montreal	Saint-Denis	10	41.000-60.000
1	Albert	31-50	Toronto	King's college	88	61.000-80.000
2	Oliver	21-30	Calgary	20 avenue west	34	41.000-60.000
3	Pamela	31-40	Vancouver	Island Highway	45	61.000-80.000

Après l'application de cette méthode d'anonymisation, il n'est plus possible pour les adversaires d'obtenir un âge ou un salaire identifiable. Il convient de noter que l'utilité des données dépend de la granularité utilisée pour remplacer les données sensibles (par exemple, l'âge ou le salaire) par leurs plages respectives. Plus les plages de données utilisées sont larges, moins l'utilité des données est persévérée, et vice versa. Par exemple, dans le champ « Salaire », nous avons deux valeurs (deux plages) dans l'ensemble de données anonymisées, au lieu de quatre valeurs dans l'ensemble de données original. Pour le champ « Age », nous avons trois valeurs (plages) au

lieu de quatre. Cela améliore la confidentialité et réduit les types d'attaques par fréquence statistique.

En utilisant Python, l'anonymisation basée sur l'agrégation peut être mise en œuvre, comme le montre la Code source 9.4. Dans ce morceau de code, les valeurs de deux champs, « Age » et « Salaire », sont regroupées dans des plages.

Code Python
<pre>import pandas as pd df = pd.read_csv('original_data.csv') print("Original Dataset"); print(df) # Anonymisation du champ Age en regroupant ses valeurs dans des intervalles bins = [0, 20, 30, 40, 50, 60, 70, 80, 90, 100] labels = ['0-20', '21-30', '31-40', '41-50', '51-60', '61-70', '71-80', '81-90', '91-100'] df['Age'] = pd.cut(df['Age'], bins=bins, labels=labels) # Anonymisation du champ Salaire en regroupant ses valeurs dans des intervalles bins = [0, 20000, 40000, 60000, 80000, 100000] labels = ['0-20.000', '21.000-40.000', '41.000-60.000', '61.000-80.000', '81.000-100.000'] df['Salary'] = pd.cut(df['Salary'], bins=bins, labels=labels) # Imprimer l'ensemble de données anonymisées print("Anonymized dataset using aggregation of Age and Salary fields"); print(df)</pre>

*Code source 9.4. Anonymisation à l'aide de la méthode d'agrégation.*

Il est clair que la méthode d'agrégation seule n'est pas sûre et qu'elle doit donc être combinée à d'autres méthodes d'anonymisation, telles que la randomisation. Par exemple, en appliquant conjointement les méthodes de randomisation et d'agrégation à l'ensemble de données présenté dans le Tableau 9-1, l'ensemble de données anonymisées est présenté dans le Tableau 9-5.

Tableau 9-5. Ensemble de données après anonymisation, à l'aide des méthodes de randomisation et d'agrégation.

	Nom	Age	Ville	NomDeRue	NoCivique	Salaire
0	OUIjk8VN4w	21-30	ZsNeY1IiD8	yKan2tsXp4	10	41.000-60.000
1	vMsTYWsgam	31-50	iMGqadzRdp	yvc3X6nzLC	88	61.000-80.000
2	8rYb3vjSm4	21-30	44TRZoWa3v	NRvSgOG85y	34	41.000-60.000
3	GKD9LXeq67	31-40	nVRkYBD4dO	eSCFyZScMz	45	61.000-80.000

**Masquage/Pseudonymisation**

L'anonymisation par masquage des données remplace les informations sensibles par des pseudonymes ou des jetons, ce qui garantit que les données originales ne peuvent pas être reliées à des personnes spécifiques. On parle également de « pseudonymisation » ou de « tokenisation ». Dans le secteur financier, par exemple, les noms et adresses des clients peuvent être remplacés par des identifiants uniques, ce qui permet d'effectuer des analyses sans compromettre la protection de la vie privée. Ces identifiants de remplacement (pseudonymes ou jetons) peuvent être empruntés à d'autres ensembles de données existants ou générés aléatoirement.

**Exemple 9.6 :** Considérons l'ensemble de données présenté dans le Tableau 9-1. En appliquant la méthode d'anonymisation basée sur le masquage à cet ensemble de données, les données sensibles (c'est-à-dire « Nom », « Ville » et « NomDeRue ») peuvent être remplacées par des données similaires mais non sensibles (des pseudonymes pour les noms et d'autres noms de ville et adresses à la place des noms originaux), comme le montre le Tableau 9.6.

Tableau 9-6. Ensemble de données après anonymisation, à l'aide de la méthode de masquage. Dans cet exemple, les valeurs des champs « Nom », « Ville » et « NomDeRue » sont remplacées par des données non sensibles.

	Nom	Age	Ville	NomDeRue	NoCivique	Salaire
0	Bono	30	London	Abbey Road	10	60.000
1	Kafu	45	Paris	Rue de Rivoli	88	75.000
2	Deco	25	New York	Central Park	34	50.000
3	Gimax	33	Halifax	Anchor Drive	45	65.000

En utilisant Python, l'anonymisation basée sur le masquage peut être mise en œuvre, comme le montre le Code source 9.5. Dans ce morceau de code, les valeurs des champs « Nom », « Ville » et « NomDeRue » sont remplacées par des pseudonymes, d'autres noms de ville et d'autres noms de rue, respectivement.

Code Python

```
# Importer les bibliothèques nécessaires
import pandas as pd

# Chargement du jeu de données original
df = pd.read_csv('original_data.csv')

# Imprimer le jeu de données original
print("Original Dataset")
print(df)

# Masquage des valeurs sensibles par des pseudonymes pour
# les noms, les villes et les noms de rues
masked_df = df.copy()
masked_df['Name'] = ['Buno', 'Kafu', 'Deco', 'Gimax']
masked_df['City'] = ['London', 'Paris', 'New York', 'Halifax']
masked_df['StreetName'] = ['Abbey Road', 'Rue de Rivoli', 'Central Pak', 'Anchor Drive']

# Imprimer l'ensemble de données anonymisées
print("Anonymized dataset using masking of the Name, City and StreetName fields")
print(masked_df)
```

Code source 9.5. Anonymisation par la méthode du masquage.

Le masquage des données est efficace et facile à mettre en œuvre pour les petits ensembles de données. Cependant, lorsque la taille des données augmente, ce qui est le cas dans les scénarios de science des données, la mise en œuvre du masquage des données devient très difficile. En effet, nous ne pouvons pas imaginer tous les pseudonymes et les différentes adresses, par exemple, qui doivent être encodés pour remplacer les données sensibles. En ce qui concerne la protection de la vie privée, le masquage des données n'est pas très efficace, car il est également vulnérable aux attaques par fréquence statistique. En d'autres termes, les mêmes pseudonymes sont utilisés pour les mêmes valeurs de données sensibles. Par exemple, si le nom d'une ville est répété dans l'ensemble de données original, le même pseudonyme sera utilisé pour toutes les occurrences de ce nom dans l'ensemble de données anonymisées, et ces informations peuvent être exploitées pour effectuer une attaque de fréquence statistique. Par conséquent, l'application de la méthode de masquage seule n'est pas suffisante et doit être combinée à d'autres méthodes d'anonymisation.

### **Perturbation**

L'anonymisation par perturbation des données consiste à ajouter un bruit aléatoire aux données sensibles pour les rendre plus difficiles à identifier. Par exemple, l'âge d'un individu peut être modifié en ajoutant ou en déduisant une petite quantité. Cette quantité ajoutée/déduite est le bruit, qui peut être ajusté pour trouver un compromis entre la protection de la vie privée et l'utilité des données. Ce bruit peut être généré de manière aléatoire et contrôlé par l'écart type afin d'équilibrer la vie privée et l'utilité des données.

**Exemple 9.7 :** En appliquant la méthode d'anonymisation basée sur les perturbations à l'ensemble de données présenté dans le Tableau 9-1, les valeurs du champ « Salaire », par exemple, peuvent être modifiées en y ajoutant un bruit aléatoire, comme le montre le Tableau 9-7.

*Tableau 9-7. Jeu de données après anonymisation, à l'aide de la méthode de perturbation. Dans cet exemple, les valeurs des champs « Age » et « Salaire » sont mises à jour en ajoutant un bruit aléatoire basé sur l'écart-type.*

	Nom	Age	Ville	NomDeRue	NoCivique	Salaire
0	Bono	23.0	London	Abbey Road	10	60.145
1	Kafu	29.0	Paris	Rue de Rivoli	88	75.007
2	Deco	24.0	New York	Central Park	34	50.015
3	Gimax	25.0	Halifax	Anchor Drive	45	65.052

En appliquant la méthode de perturbation de cette manière, en utilisant l'écart-type comme bruit par exemple, on préserve la distribution des données, ce qui contribue à préserver l'utilité des données. Le respect de la vie privée n'est pas visible dans ce petit ensemble de données qui ne comprend que quatre lignes. Toutefois, pour les grands ensembles de données, la méthode de perturbation représente un meilleur outil d'anonymisation qui permet d'équilibrer la vie privée et l'utilité des données.

En utilisant Python, l'anonymisation basée sur la perturbation peut être mise en œuvre, comme le montre le Code source 9.6. Dans ce morceau de code, les valeurs de deux champs, « Age » et « Salaire », sont mises à jour en ajoutant du bruit calculé à l'aide de l'écart type.

Code Python

```

# Importer les bibliothèques nécessaires
import pandas as pd
import numpy as np
import math

# Fonction d'anonymisation qui opère sur une colonne du dataframe
def add_noise(df, column, noise = None):

    # Si le bruit n'est pas défini, l'écart-type (std) est utilisé à la place
    if noise == None:
        noise = df[column].std()

    # Calculer la colonne perturbée
    col_perturb = df[column].add(np.random.normal(0, noise, df.shape[0]))

    # Arrondir les valeurs vers le bas (pas nécessaire)
    col_perturb = col_perturb.apply(lambda x: np.ceil(x))

    # Créer une copie du dataframe et la mettre à jour avec la colonne
    # perturbée
    df_copy = df.copy()
    df_copy[column] = col_perturb

    return df_copy

# Charger de l'ensemble de données original
df = pd.read_csv('original_data.csv')

# Imprimer le jeu de données original
print("Original Dataset")
print(df)

# Ajouter du bruit au Salaire
df_perturb = add_noise(df, 'Salary', noise = 100)

# Ajouter du bruit à l'âge
df_perturb = add_noise(df_perturb, 'Age', noise = 10)

# Imprimer l'ensemble de données anonymisées
print("Anonymized dataset using perturbation of the Age and Salary fields")
print(df_perturb)

```

*Code source 9.6. Anonymisation à l'aide de la méthode de perturbation.*

## **k-anonymat**

Le k-anonymat est une technique d'anonymisation fondamentale qui garantit la protection des informations sensibles tout en préservant



l'utilité globale des données. Le  $k$ -anonymat garantit que chaque enregistrement de l'ensemble de données ne peut être distingué d'au moins  $k - 1$  autres enregistrements, en ce qui concerne certains attributs. Cela signifie que les données d'un individu ne peuvent pas être distinguées parce qu'elles sont regroupées avec au moins  $k-1$  autres individus qui partagent les mêmes attributs. En effet, en regroupant les comportements individuels similaires, cette technique renforce efficacement la protection de la vie privée, en rendant beaucoup plus difficile pour les adversaires de discerner des individus ou des entités spécifiques dans les données anonymisées.

Les scientifiques des données doivent d'abord classer les caractéristiques en trois catégories: les identifiants, les non-identifiants et les quasi-identifiants. La première classe représente les caractéristiques qui doivent être supprimées de l'ensemble de données, car elles peuvent être utilisées pour identifier directement les individus, comme les noms. La deuxième classe, celle des non-identifiants, contient toutes les caractéristiques qui doivent rester dans l'ensemble de données, car elles sont nécessaires à l'analyse et ne révèlent pas directement l'identité des personnes. La dernière classe, celle des quasi-identifiants, représente les caractéristiques qui doivent être traitées. Elles doivent être traitées de manière que chaque combinaison distincte de ces caractéristiques désigne au moins  $k$  enregistrements (lignes) (Sweeney, 2002). Le  $k$ -anonymat peut être obtenu par suppression, généralisation ou les deux.

- **Suppression** : Cette méthode consiste à supprimer les attributs et à les remplacer par « \* » par exemple. En d'autres termes,

tout ou une partie des valeurs des attributs peuvent être supprimées et remplacées par « \* ».

- Généralisation** : Cette méthode est similaire à celle présentée précédemment et consiste à remplacer les valeurs des caractéristiques par des valeurs plus larges (c'est-à-dire des fourchettes). Par exemple, si la valeur de l'attribut « Age » est « 43 », elle peut être remplacée par « 40-50 ».

**Exemple 9.8 :** Supposons que nous disposions d'un ensemble de données contenant des dossiers médicaux contenant des informations sensibles, notamment le nom, l'âge, le sexe, la religion et l'état de santé, comme indiqué dans le Tableau 9-8.

Tableau 9-8. Ensemble de données avant l'anonymisation.

	Nom	Age	Sexe	Taille (cm)	Poids (kg)	Province	Religion	État de santé
0	Michelle	25	Femme	173	75	Québec	Christianisme	Arthrite
1	Amelia	21	Femme	182	79	Ontario	Christianisme	Grippe
2	Oliver	30	Homme	185	80	Manitoba	Judaïsme	Pneumonie
3	Nora	22	Femme	176	71	Ontario	Autre	Diabète
4	Ryan	20	Homme	170	81	Manitoba	Bouddhisme	Digestif
5	William	17	Homme	180	85	Ontario	Christianisme	Arthrite
6	Charlotte	29	Femme	185	68	Québec	Islam	Digestif
7	Johnson	16	Homme	165	70	Ontario	Autre	Diabète
8	Henry	27	Homme	188	69	Manitoba	Christianisme	Diabète
9	Logan	15	Homme	178	82	Ontario	Autre	Grippe

Nous voulons partager ces données à des fins de recherche tout en préservant la vie privée des personnes figurant dans l'ensemble de données. À cette fin, nous pouvons appliquer un 2-anonymat ( $k = 2$ ) pour anonymiser les données, en ce qui concerne les quasi-identifiants « Age », « Sexe » et « Province ». Pour ce faire, nous pouvons supprimer les attributs « Nom » et « Religion », qui sont très sensibles, et généraliser l'attribut « Age ». L'ensemble de données obtenu après l'application du 2-anonymat est présenté dans le Tableau 9-9.

Tableau 9-9. Ensemble de données après anonymisation, à l'aide d'une méthode 2-anonymat, en ce qui concerne les attributs « Age », « Sexe » et « Province ».

	Nom	Age	Sexe	Taille (cm)	Poids (kg)	Province	Religion	État de santé
0	*	21-30	Femme	173	75	Québec	*	Arthrite
6	*	21-30	Femme	185	68	Québec	*	Digestif
1	*	21-30	Femme	182	79	Ontario	*	Grippe
3	*	21-30	Femme	176	71	Ontario	*	Diabète
5	*	00-20	Homme	180	85	Ontario	*	Arthrite
7	*	00-20	Homme	165	70	Ontario	*	Diabète
9	*	00-20	Homme	178	82	Ontario	*	Grippe
2	*	21-30	Homme	185	80	Manitoba	*	Pneumonie
4	*	21-30	Homme	170	81	Manitoba	*	Digestif
8	*	21-30	Homme	188	69	Manitoba	*	Diabète

Dans ce tableau, les enregistrements sont réordonnés pour montrer les groupes créés par l'anonymisation. Il convient de noter que, selon le contexte, d'autres attributs peuvent être considérés comme quasi-identifiants dans l'anonymisation.

Ce tableau montre que chaque ligne de l'ensemble de données représente au moins deux individus, en ce qui concerne les quasi-identifiants sélectionnés, ce qui permet d'atteindre le 2-anonymat.

### ***l*-diversité**

La méthode *l*-diversité est une autre méthode d'anonymisation utilisée pour préserver la confidentialité des ensembles de données en réduisant la granularité de la représentation des données. Elle étend le *k*-anonymat en garantissant qu'avec chaque groupe d'enregistrements indiscernables (groupe *k*-anonymat), il y a au moins *l* valeurs différentes pour les attributs sensibles.

La méthode *l*-diversité répond à certaines des limites de la méthode *k*-anonymat, dans la mesure où la protection des identités, en les regroupant en *k* individus, ne garantit pas la protection des valeurs sensibles correspondantes, qui peuvent présenter une certaine

homogénéité. En d'autres termes, elle ajoute une diversité intra-groupe supplémentaire pour les valeurs sensibles.

**Exemple 9.9 :** En continuant avec l'ensemble de données des dossiers médicaux, nous voulons nous assurer que même si plusieurs dossiers partagent les mêmes attributs (grâce au 2-anonymat), il y a une diversité dans les valeurs sensibles au sein de chaque groupe. À partir de l'ensemble de données anonymisées à l'aide du 2-anonymat, nous observons qu'il existe quatre groupes, en ce qui concerne les quasi-identifiants sélectionnés (« Age », « Sexe » et « Province »). Il est clair qu'il y a au moins trois valeurs différentes de « État de Santé » dans chaque groupe. Cela signifie que l'ensemble de données anonymisées est déjà 2-diversité, selon la définition de la *l*-diversité avec *l* = 2. Nous pouvons essayer de traiter davantage l'ensemble de données afin d'augmenter la diversité et d'atteindre, par exemple, une 3-diversité. Dans cet exemple, nous supprimons les attributs « Age », « Taille » et « Poids » et généralisons l'attribut « Province », ce qui permet d'obtenir une 3-diversité, comme le montre le Tableau 9-10. Ici, on obtient trois groupes, chacun possédant au moins trois enregistrements différents (3-diversité).

*Tableau 9-10. Ensemble de données après anonymisation, en utilisant une anonymisation à 3-diversité sur l'ensemble de données déjà basé sur 2-anonymat, en ce qui concerne l'attribut sensible « État de santé ».*

	Nom	Age	Sexe	Taille (cm)	Poids (kg)	Province	Religion	État de santé
0	*	*	Femme	*	*	Quebec/Ontario	*	Arthrite
1	*	*	Femme	*	*	Quebec/Ontario	*	Grippe
3	*	*	Femme	*	*	Quebec/Ontario	*	Diabète
6	*	*	Femme	*	*	Quebec/Ontario	*	Digestif
2	*	*	Homme	*	*	Manitoba/Alberta	*	Pneumonie
4	*	*	Homme	*	*	Manitoba/Alberta	*	Digestif
8	*	*	Homme	*	*	Manitoba/Alberta	*	Diabète
5	*	*	Homme	*	*	Quebec/Ontario	*	Arthrite
7	*	*	Homme	*	*	Quebec/Ontario	*	Diabète
9	*	*	Homme	*	*	Quebec/Ontario	*	Grippe

**Comparaison des méthodes d'anonymisation**

Dans cette section, nous présentons une brève comparaison entre les différentes méthodes d'anonymisation présentées précédemment. Les méthodes qui sont composées d'autres méthodes, telles que le  $k$ -anonymat et la  $l$ -diversité, ne sont pas incluses afin d'éviter toute redondance. La comparaison, présentée dans le Tableau 9-11, se concentre sur quatre critères, à savoir la réidentification, les statistiques des attributs, la corrélation des caractéristiques et la performance des modèles ML.

Tableau 9-11. Comparaison entre les différentes méthodes d'anonymisation.

Méthode	Réidentification	Statistiques des attributs	Corrélation des attributs	Performance des modèles ML
Généralisation	Faible	Faible	Faible	Faible
Randomisation	Élevé	Moyen	Faible	Faible
Permutation	Élevé	Élevé	Très faible	Très faible
Masquage	Très élevé	Élevé	Élevé	Élevé

**Anonymisation multi-vues**

Cette technique innovante exploite plusieurs perspectives ou vues des données pour trouver un équilibre délicat entre la protection de la vie privée et l'utilité des données (Mohammady et al., 2018, 2021). En prenant en compte divers angles de présentation des données, l'anonymisation multi-vues améliore la protection de la vie privée tout en préservant l'utilité des données.

Le concept d'anonymisation multi-vues consiste à créer plusieurs versions (c'est-à-dire des vues) de l'ensemble de données. Ces vues sont générées à partir d'une vue originale, de telle sorte que la vue originale ne puisse pas être révélée par des adversaires en observant simplement les différentes vues. Différentes techniques sont proposées pour créer les vues, telles que le partitionnement de l'ensemble de données en différentes partitions et les mélanger.

Ensuite, des techniques d'anonymisation de base sont appliquées aux vues. Les adversaires ont accès aux vues et à l'algorithme d'anonymisation (ils peuvent l'utiliser sans en connaître la structure). Ils attaquent l'algorithme en injectant des données spécifiques et en observant le comportement des vues, en essayant de révéler la structure de l'algorithme ou la vue originale.

**Exemple 9.10**: Imaginons que nous menions une recherche impliquant deux ensembles de données : un ensemble de données sur les soins de santé et un ensemble de données financières. L'objectif est d'effectuer des analyses sur les deux ensembles de données tout en préservant la vie privée des personnes. Définissons ces deux ensembles de données comme suit :

1. *Ensemble de données sur les soins de santé (Santé)* : Contient les dossiers médicaux des patients avec des champs tels que « ID du patient », « Diagnostic » et « Traitement ».
2. *Ensemble de données financières (Finance)* : Contient des informations sur les revenus des personnes, avec des champs tels que « ID de l'individu » et « Revenu ».

L'anonymisation multi-vues comprend les étapes suivantes :

- *Généralisation des données* : Dans chaque vue de données, les attributs sensibles sont généralisés ou transformés pour assurer la protection de la vie privée. Par exemple, dans l'ensemble de données « Santé », les diagnostics spécifiques peuvent être généralisés à des catégories plus larges. Dans l'ensemble de données « Finance », des fourchettes de revenus

sont utilisées à la place des valeurs exactes de revenus pour protéger la confidentialité des revenus individuels. Le

- *Tableau 9-12* présente un exemple de ces transformations.

*Tableau 9-12. Exemples de généralisation des données dans l'anonymisation multi-vues.*

Ensemble de données	Attribut original	Attribut généralisé
Santé	Hypertension Diabète	Cardiovasculaire Métabolique
Finance	45.000 \$ 65.000 \$	40.000 \$ – 50.000 \$ 60.000 \$ – 70.000 \$

- *Prévention des liens* : Des méthodes sont appliquées pour empêcher l'établissement de liens entre les enregistrements anonymisés dans différentes vues. Cela est essentiel pour garantir que la même personne ne puisse pas être identifiée en combinant des informations provenant de différentes vues. Dans notre exemple, des méthodes sont appliquées pour empêcher l'établissement de liens entre les enregistrements anonymisés dans les différentes vues des ensembles de données « Santé » et « Finance ». Par exemple, des identifiants pseudonymes peuvent être utilisés au lieu d'utiliser les ID des patients et les IDs des individus.
- *Garantie de la confidentialité* : Des méthodes, telles que le  $k$ -anonymat et/ou la  $l$ -diversité, sont appliquées pour s'assurer que les données anonymisées répondent aux exigences de confidentialité. Par exemple, nous pouvons nous assurer qu'il y a au moins  $k$  individus ayant les mêmes attributs généralisés

dans chaque ensemble de données, « Santé » et « Finance », pour obtenir un  $k$ -anonymat.

- *Analyse* : Une fois les données rendues anonymes, elles peuvent être confiées à des tiers à des fins d'analyse ou de recherche à travers des vues multiples sans révéler d'informations sensibles (par exemple, le diagnostic du patient et son revenu). Ainsi, les analystes peuvent étudier la corrélation entre certaines conditions médicales et les tranches de revenus sans identifier des individus spécifiques.

#### 9.4.7 Études de cas

L'importance de la protection de la vie privée dans la science des données est soulignée par plusieurs affaires très médiatisées qui ont mis en lumière les défis éthiques et sécuritaires associés au traitement des données. Dans cette section, trois cas bien connus liés à la protection de la vie privée sont expliqués.

- **Le scandale Facebook-Cambridge Analytica** : Une affaire de protection de la vie privée, appelée « le scandale Facebook-Cambridge Analytica », a été révélée en 2018 (Cadwalladr et Graham-Harrison, 2018). Ce scandale a révélé la collecte et l'utilisation abusive et contraire à l'éthique de données personnelles d'utilisateurs de Facebook à des fins politiques. Cambridge Analytica, une société de conseil politique, a obtenu l'accès aux données personnelles de millions d'utilisateurs sans leur consentement, ce qui a conduit à une sensibilisation accrue du public aux questions de confidentialité des données. À la suite de ce scandale, des



changements réglementaires ont été mis en œuvre pour renforcer la protection des données et le droit à la vie privée, illustrant les conséquences considérables des atteintes à la vie privée à l'ère numérique. Cette affaire est un rappel saisissant du besoin critique de pratiques éthiques en matière de données et de l'importance de protéger les informations personnelles des individus.

- **La fuite de données de recherche d'AOL** : Un autre incident notable qui souligne l'importance de la dépersonnalisation des données est la fuite des données de recherche d'AOL (Anderson, 2006). En 2006, AOL a publié par inadvertance un vaste ensemble de données contenant les requêtes de recherche effectuées par ses utilisateurs. Alors que cet ensemble de données était destiné à des fins de recherche, sa publication sans anonymisation appropriée a entraîné une violation importante de la vie privée. Les utilisateurs individuels pouvaient être potentiellement identifiés grâce à leurs requêtes de recherche, ce qui a suscité des inquiétudes quant à l'atteinte à la vie privée. Cet incident a mis en évidence la nécessité de recourir à des techniques rigoureuses de dépersonnalisation des données pour protéger l'anonymat des utilisateurs lors du partage de données à des fins de recherche ou d'analyse. Il s'agit d'une mise en garde contre les conséquences potentielles d'un manque d'anonymisation des données sensibles.

- **Problèmes de confidentialité liés à Zoom** : L'utilisation massive de Zoom en 2020 a mis en lumière les vulnérabilités et les problèmes de protection de la vie privée associés aux plateformes de vidéoconférence (Brodkin, 2020). La pandémie COVID-19 ayant entraîné une évolution vers le travail à distance et les réunions en ligne, la popularité soudaine de Zoom a donné lieu à un examen plus approfondi concernant la vie privée. En effet, des inquiétudes concernant la protection de la vie privée et la sécurité sont apparues, notamment en ce qui concerne les pratiques de partage des données. Par exemple, les attaques de type « *Zoom bombing* » étaient possibles, permettant à des personnes non autorisées d'accéder à des réunions privées et de les perturber. Cet incident a non seulement incité l'entreprise à prendre des mesures immédiates, mais a également sensibilisé les utilisateurs et les organisations à l'importance des technologies sécurisées et respectueuses de la vie privée dans le paysage de la communication numérique.

## 9.5 Discussion

La sécurité et la protection de la vie privée sont des concepts étroitement liés mais distincts. La sécurité englobe les mesures prises pour protéger les données et les systèmes contre les accès non autorisés, les violations ou les dommages. Elle implique la mise en œuvre de mesures de sauvegarde, de chiffrement, de contrôles d'accès et d'autres mécanismes pour garantir la confidentialité, l'intégrité et la disponibilité des données. D'autre part, la protection de la vie privée

concerne le droit d'un individu à contrôler ses informations personnelles et à décider de la manière dont elles sont collectées, utilisées et partagées. Elle implique le respect des limites et du consentement. La Figure 9-3 montre comment ces deux concepts sont liés.

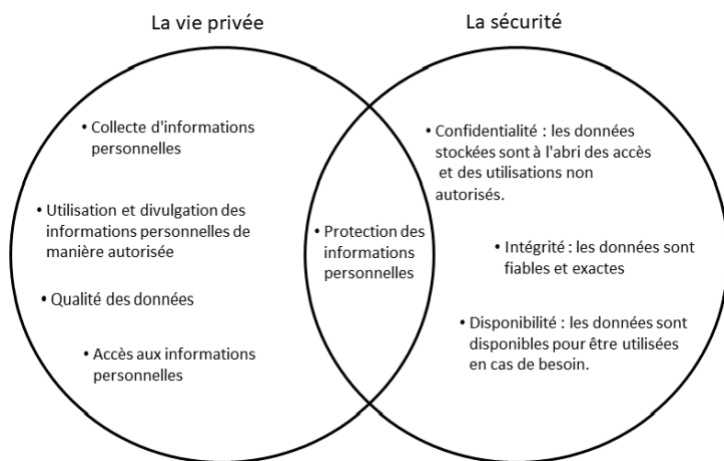


Figure 9-3. Vie privée vs. Sécurité, adaptée du (Croughan, 2021).

### Sécurité : Protection des données et des systèmes

La sécurité englobe un ensemble complet de mesures et de pratiques visant à protéger les données, les systèmes et les actifs contre diverses menaces. Ces menaces peuvent être des accès non autorisés, des cyberattaques, des violations de données, des vols ou des dommages physiques. Les mesures de sécurité impliquent souvent la mise en œuvre des systèmes de sécurité telles que le chiffrement, les pare-feux, les systèmes de détection d'intrusion, les contrôles d'accès et les audits de sécurité réguliers.

Les objectifs primordiaux de la sécurité sont d'assurer la confidentialité, l'intégrité et la disponibilité des données. La

confidentialité implique de restreindre l'accès aux données aux seules personnes autorisées, l'intégrité garantit que les données restent inaltérées et fiables, et la disponibilité garantit que les données sont accessibles en cas de besoin. La sécurité est importante pour prévenir les violations de données et maintenir la confiance des individus et des organisations dans un monde de plus en plus numérisé et connecté.

### Vie privée : Droits individuels et contrôle des données

La protection de la vie privée, quant à elle, est centrée sur les droits des individus à contrôler leurs informations personnelles. Elle s'articule autour des principes d'autonomie, de consentement et de limites concernant la collecte, l'utilisation et le partage des données personnelles. Le respect de la vie privée implique l'obtention d'un consentement éclairé, la transparence des pratiques en matière de données et la possibilité pour les individus de faire des choix concernant leurs données.

La protection de la vie privée n'est pas seulement une obligation légale, c'est aussi un droit humain fondamental. Il permet aux individus de conserver leur dignité, leur autonomie et leur espace personnel à une époque où les données sont une ressource précieuse. Il répond aux préoccupations concernant la collecte de données, le profilage, la surveillance et l'utilisation potentiellement abusive des informations personnelles.

#### **9.5.1 L'interaction entre la sécurité et la vie privée**

Bien que la sécurité et la protection de la vie privée aient des objectifs distincts, elles sont souvent liées. Des mesures de sécurité robustes

peuvent renforcer la protection de la vie privée en protégeant les données à caractère personnel contre les violations et les accès non autorisés. Par exemple, le chiffage peut protéger les informations sensibles pendant la transmission, et les contrôles d'accès peuvent limiter les personnes autorisées à consulter et à modifier les données. Inversement, le respect des principes de protection de la vie privée peut conduire à de meilleures pratiques en matière de sécurité. En faisant preuve de transparence sur le traitement des données, les organisations peuvent instaurer un climat de confiance avec les utilisateurs et faire preuve d'une gestion responsable des données. Le consentement d'une personne à la collecte et au traitement des données s'aligne sur le traitement éthique des données, ce qui contribue à la fois à la protection de la vie privée et à la sécurité.

### Compromis entre sécurité et protection de la vie privée

Le défi consiste à trouver un équilibre délicat entre la sécurité et le respect de la vie privée. Une focalisation excessive sur les mesures de sécurité peut involontairement porter atteinte au droit à la vie privée. Par exemple, une surveillance trop invasive ou la collecte de données sans consentement peuvent porter atteinte à la vie privée. Inversement, une importance extrême accordée à la vie privée au détriment de la sécurité peut rendre les systèmes vulnérables aux attaques menaçant la protection des données.

## **9.6 Recommandations**

Sur la base des enseignements tirés de la discussion de la section 9.5.1, nous proposons quelques recommandations à l'intention des particuliers et des entreprises:

- **Pour les entreprises** : Établir une politique globale de sécurité et de protection de la vie privée qui s'aligne parfaitement sur les principes éthiques. Mettre continuellement à jour les mesures de sécurité et s'engager de manière proactive dans des évaluations externes de piratage éthique. Veiller à obtenir le consentement clair et éclairé des personnes lors de la collecte de leurs données.
- **Pour les particuliers** : Faire preuve de prudence et d'une conscience accrue lors du partage d'informations personnelles en ligne. Acquérir une compréhension approfondie de la manière dont vos données sont utilisées et envisager l'adoption d'outils et de services de préservation de la vie privée. Face à l'évolution constante des technologies émergentes, il est important de comprendre l'importance de trouver un équilibre délicat entre la sécurité et la protection de la vie privée. Trouver le juste équilibre entre la sécurité et la vie privée reste un défi fondamental qui exige une recherche, une innovation et une pratique éthique permanentes.

Dans les deux contextes, individuel et professionnel, la réalisation de cet équilibre est un défi permanent qui nécessite une recherche constante, des solutions innovantes et un engagement ferme en faveur des pratiques éthiques.

## 9.7 Conclusion

Le paysage contemporain des données témoigne d'une transformation remarquable provoquée par l'ère numérique. L'explosion de la production de données a mis en évidence la nécessité cruciale d'une

prise de décision fondée sur les données, façonnant ainsi l'évolution de divers secteurs et industries. Ce paysage en évolution a fait de la science des données une pièce incontournable dans la prise de décision moderne, où les praticiens sont à l'avant-garde de l'extraction d'informations précieuses à partir d'un ensemble de données en constante expansion.

La croissance exponentielle de la création de données, alimentée par les interactions humaines et la prolifération des appareils IdO, projette un avenir interconnecté marqué par des volumes de données sans précédent. Les implications sont profondes et touchent pratiquement tous les aspects de notre vie. Qu'il s'agisse des soins de santé, où les données permettent d'améliorer les diagnostics et les traitements, ou des industries qui bénéficient de l'analyse des données dans un large éventail d'applications, l'influence des données est omniprésente.

Toutefois, à l'ère de l'abondance des données, il est impératif de relever les défis pressants de la sécurité et de la protection de la vie privée. Les violations de données et les problèmes de protection de la vie privée qui ont été révélés nous rappellent clairement la nécessité d'adopter des mesures rigoureuses de protection des données. À mesure que nous avançons, il est de notre responsabilité de trouver un équilibre délicat entre l'exploitation de la puissance des données au service du progrès et de l'innovation et la protection de la vie privée et de la sécurité des personnes.

Dans ce chapitre, les origines et l'application de la science des données ont été présentées, en soulignant l'importance des données et de l'analyse des données dans nos vies. Ensuite, les défis et les solutions

en matière de sécurité et de protection de la vie privée ont été présentés afin de garantir une utilisation sûre et responsable des données et des techniques de la science des données au profit des entreprises et des particuliers.

## 9.8 Bibliographie

Anderson, C. (2008). The End of Theory: The data Deluge Makes the Scientific Method Obsolete. *Wired Magazine*, 16(7):16–07.

Anderson, N. (2006). AOL Releases Search Data on 500,000 Users (Updated). *Ars TECHNICA*.

Becker, M. (2019). Privacy in the Digital Age: Comparing and Contrasting Individual versus Social Approaches Towards Privacy. *Ethics and Information Technology*, 21(4):307–317.

Bhave, D. P., Teo, L. H., and Dalal, R. S. (2020). Privacy at Work: A Review and a Research Agenda for a Contested Terrain. *Journal of Management*, 46(1):127–164.

Bonta, A. G. R. (2023). California Consumer Privacy Act (CCPA).

Breiman, L. (2001). Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*, 16(3):199–231.

Brodkin, J. (2020). Zoom security issues: Here’s everything that’s gone wrong (so far). *Ars TECHNICA*.

Cadwalladr, C. and Graham-Harrison, E. (2018a). Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach. *The Guardian*, 17(1):22.

Chavez, T., O’Hara, C., and Vaidya, V. (2019). *Data Driven: Harnessing Data and AI to Reinvent Customer Engagement*. McGraw-Hill Education, New York.



- Chen, R., Stewart, W. F., Sun, J., Ng, K., and Yan, X. (2019). Recurrent Neural Networks for Early Detection of Heart Failure from Longitudinal Electronic Health Record Data. *Circulation: Cardiovascular Quality and Outcomes*, 12(10):e005114.
- Consulting, I. (2018). General Data Protection Regulation (GDPR).
- Croughan, K. (2021). The Difference between ‘Security’ and ‘Privacy’ Jobs – And Why IT Matters.
- Davenport, T. H. and Harris, J. G. (2007). *Competing on Analytics: The New Science of Winning*. Harvard Business Scholl Press, 15 (217p).
- Dhar, V. (2013). Data Science and Prediction. *Communications of the ACM*, 56(12):64-73.
- Dierks, T. and Rescorla, E. (2008). The Transport Layer Security (TLS) Protocol Version 1.2. IETF.
- Dwork, C. (2008). Differential Privacy: A Survey of Results. In Agrawal, M., Du, D., Duan, Z., and Li, A., editors, *Theory and Applications of Models of Computation*, pages 1–19, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Gligorijević, V., Malod-Dognin, N., and Pržulj, N. (2016). Integrative Methods for Analyzing Big Data in Precision Medicine. *Proteomics*, 16(5):741–758.
- Puig, A. (2019). Equifax Data Breach Settlement: What You Should Know. Federal Trade Commission.
- Henke, N., Bughin, J., Chui, M., Manyika, J., Saleh, T., Wiseman, B., and Sethupathy, G. (2016). *The Age of Analytics: Competing in a Data-driven World*. McKinsey Global Institute Research.
- Hinds, J., Williams, E. J., and Joinson, A. N. (2020). “It wouldn’t happen to me”: Privacy Concerns and Perspectives Following

the Cambridge Analytica Scandal. *International Journal of Human-Computer Studies*, 143:102498.

Institute, P. (2021). Cost of a Data Breach Report 2021. IBM Security.

Mertens, Donna M. (2018). Ethics of qualitative data collection. *The SAGE handbook of qualitative data collection*. 33–48

Klosek, J. (2018). Data Privacy in the Information Age. *American Journal of Information Science and Technology*, 4(1):25–30.

Mohammady, M., Oqaily, M., Wang, L., Hong, Y., Louafi, H., Pourzandi, M., and Debbabi, M. (2021). A Multi-view Approach to Preserve Privacy and Utility in Network Trace Anonymization. *ACM Trans. Priv. Secur.*, 24(3):14:1–14:36.

Mohammady, M., Wang, L., Hong, Y., Louafi, H., Pourzandi, M., and Debbabi, M. (2018). Preserving Both Privacy and Utility in Network Trace Anonymization. pages 459–474.

Obermeyer, Z. and Emanuel, E. J. (2016). Predicting the Future—Big Data, Machine Learning, and Clinical Medicine. *The New England journal of medicine*, 375(13):1216.

Ravinder, M. and Kulkarni, V. (2023). A Review on Cyber Security and Anomaly Detection Perspectives of Smart Grid. In *2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pages 692–697. IEEE.

Reinsel, D., Gantz, J., and Rydning, J. (2018). *Data Age 2025: The Digitization of the World from Edge to Core*. Seagate.

Rijmen, V. and Daemen, J. (2001). Advanced Encryption Standard. *Proceedings of federal information processing standards publications, national institute of standards and technology*, 19:22.

Rivest, R. L., Shamir, A., and Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, 21(2):120–126.

Roth, K., Pemula, L., Zepeda, J., Scholkopf, B., Brox, T., and Gehler, P. (2022). Towards Total Recall in Industrial Anomaly Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 14318–14328.

CISA. (2020). Insider Threat Mitigation Guide. Cybersecurity and Infrastructure Security Agency.

Solove, D. J. (2006). A Taxonomy of Privacy. University of Pennsylvania Law Review, 154(3):477–560.

Stallings, W. and Brown, L. (2017). Computer Security: Principles and Practice, Global Edition. Pearson.

Sweeney, L. (2002). k-Anonymity: A Model for Protecting Privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, 10(5):557–570.

## 9.9 Liste des sigles, abréviations et acronymes

AA	Apprentissage Automatique
AES	Advanced Encryption Standard
AMF	Authentification Multifactorielle
CCPA	California Consumer Privacy Act
DDoS	Distributed Denial of Services (Déni de services distribué)
DoS	Denial of Services (Déni de services)
IdO	Internet des Objets
IA	Intelligence Artificielle
ID	Identifiant
RBAC	Role-based Access Control
RGPD	Règlement Général sur la Protection des Données
RSA	Rivest-Shamir-Adleman

SSL	Secure Sockets Layer
TLS	Transport Layer Security