DATA SCIENCE CONTINUUM

FROM FOUNDATIONS TO PRACTICES

F.A.N. VERRI

DATA SCIENCE CONTINUUM

FROM FOUNDATIONS TO PRACTICES

FILIPE A. N. VERRI

WIP

April 15, 2024

Disclaimer: This version is a work in progress. Many parts of the book have been drafted with the help of GitHub Copilot and may not be revised yet by the author. The author is not responsible for any misinformation contained in this book.

Book cover image was created with the assistance of Gemini and DALL·E 2.

Contents

About this book

I intend to make this book forever free and open-source. You can find (and contribute to) the source code at github.com/verri/dsp-book.

If you like this book, consider *buying me a coffee* at buymeacoffee.com/verri. All donations are used to improve this book, including editing and proofreading.

If you find any typos, grammar errors, incomplete material or have any suggestions, please open an issue at github.com/verri/dsp-book/issues.

Students can found a printable version (A4 paper, double-sided, short-edge spiral binding) of this book at comp.ita.br/~verri/ds-book-print.

This book comprises the lectures notes of the course PO-235 Data Science Project. I hope someday it becomes an actual book. For now, beware many typos, grammar errors, ugly typesetting, disconnected material, etc.

Also, it is important to highlight that:

- This is not a Machine Learning book, and I do not intend to explain how specific ML algorithms work.
- This contains some kind of introductory material on data science. Although I introduce the fundamental concepts, I expect you have strong mathematical and statistics background.
- An artificial constraint I have imposed in the material (for the sake of the course) is that I only consider *predictive methods*, more specifically inductive ones. I not address topics such as clustering, association-rules mining, transductive learning, anomaly detection, time series forecasting, reinforced learning, etc.

I have decided to work on this material because the books I like on data science are either

- too broad and too shallow, in the sense they hide many mathematical foundations and focus on just explaining what data science is and where it is applied;
- too tool-centric, in the sense that they focus only on a specific toolbox or programming language; or
- too machine-learning-y, exposing many machine learning algorithms and missing the foundations of learning.

So..., I expect my approach on the subject provide:

- awareness of all steps in a data science project;
- deeper focus (than most books) on data transformation, describing the semantics of dataset operators instead of restraining ourselves with a specific tool;
- deeper focus (than most books) on why machine learning works, increasing awareness of its pitfalls and limitations;
- deeper focus (than most books) on correct evaluation and validation (pre-deployment) of machine learning models.

This book covers the following material:

- Brief history of data science.
- Background topics.
- Fundamental data concepts.
- Stages in a data science project.
- Data Infrastructure.
- Data integration from multiple sources.
- Data engineering and shaping.
- Inductive learning and principles of statistical learning theory.
- Application of Machine Learning models in real-world problems.
- Experimental planning for data science.
- Model evaluation and Bayesian analysis.
- Documentation and deployment.
- Ethical and legal issues in data science.
- Privacy-preserving computational approaches.

Course plan

In the following, I present the course plan for PO-235 and CMC-16.

Any questions about the classes should be sent via Google Classroom. If your question is of general interest, please use the main stream. If your question is personal and about a specific assignment or grade, please use the private stream.

PO-235 Data science project

Course plan (2024)
Prof. Filipe A. N. Verri

Important: Only graduate students are allowed to take this course.

Number of students: Approx. 20

Course load: 3–0–0–4

Requirements: Advanced programming skills, strong statistics background, and beginner level machine learning skills.

Course program: Brief history of data science. Fundamental data concepts. Stages in a Data Science project. Data Infrastructure. Data integration from multiple sources. Data engineering and shaping. Inductive learning and principles of statistical learning theory. Application of machine learning models in real-world problems. Experimental planning for data science. Model evaluation and Bayesian analysis. Documentation and deployment. Ethical and legal issues in data science. Privacy-preserving computational approaches.

Goals: Providing the theoretical background and the practical concepts to develop an end-to-end data science project for an inductive task.

Teaching methodology: Expository classes in common classroom, using whiteboard, slide presentations, coding examples, books and scientific papers. Supplementary didactic materials will be available in Google Classroom. The development of the case study will happen during home study hours, including programming and scientific paper writing. All classes will be given in English. Students are encouraged to ask questions in English, but Portuguese is also allowed. All written and oral assignments must be in English.

Grading: Two individual written tests in the 1st (T_1 and T_2) and another in the 2nd quarter (T_3). Also, a group activity that includes writting a scientific paper, developing a data science product, and a 30 minutes presentation (L).

Final grades will be calculated as

$$1^{\text{st}}\ \text{Q} = \sqrt{T_1 T_2}, \qquad 2^{\text{nd}}\ \text{Q} = \sqrt{T_3 L}, \qquad \text{Exam} = L.$$

Case study: Exactly 6 groups will be formed. Each group will be responsible for a case study. Students must choose a real-world problem and develop a data science project, including data collection, data transformation, inductive learning, validation, documentation, and deployment. The results must be presented in a scientific paper format and a 30 minutes presentation. The trained models must be incorporated in a data science product, such as a web application, a mobile application, or a web service.

Bibliography:

- N. Zumel and J. Mount (2019). *Practical Data Science with R*. 2nd ed. Manning.
- H. Wickham, M. Çetinkaya-Rundel, and G. Grolemund (2023). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. 2nd ed. O'Reilly Media.
- J. D. Kelleher and B. Tierney (2018). *Data science*. The MIT Press.

The first two books (Zumel and Mount; Wickham, Çetinkaya-Rundel, and Grolemund) are available online for free.

Any required extra material will be made available in Google Classroom.

Calendar: The expected schedule is presented below.

1st Quarter	
Week	Topics
1	Brief history of data science (chapter 1) Preliminaries (chapter 2)
2	**Written test**
3	Fundamental data concepts (chapter 3) Stages in a data science project
4 5	Inductive learning and statistical learning theory
6	Data infrastructure and data integration from multiple sources
7	Data engineering and shaping
8	**Written test**

2nd Quarter	
Week	Topics
1 2	Application of machine learning models in real-world problems
3 4	Experimental planning for data science
5 6	Model evaluation and Bayesian analysis
7	**Written test**
8	Documentation and deployment Ethical and legal issues in data science Privacy-preserving computational approaches

Case studies will be presented during exam weeks. At most 3 case studies will be presented per day, with 30 minutes for each presentation and 20 minutes for questions.

CMC-16 Data science practices

Course plan (2024)
Prof. Filipe A. N. Verri

Important: Only ITA's undergraduate students are allowed to take this course.

Number of students: Approx. 20 (no more than 40 students)

Course load: 2–0–1–5

Requirements: CMC-13 or CMC-15

Course program: Brief history of Data Science. Stages in a Data Science project. Tidy Data. Data integration from multiple sources. Data engineering and shaping. Inductive learning and statistical learning theory. Experimental planning for Data Science. Model evaluation and Bayesian Analysis. Documentation and deployment. Privacy-preserving computational approaches.

Goals: Further studying the practical aspects of Data Science (in relation to CMC-13) and providing the mathematical foundations to ensure the correct usage of Data Science techniques.

The specific goals are:

- Understanding the steps and people involved in Data Science projects;
- Developing an end-to-end case study, including data collection, data transformation, inductive learning, validation, documentation, and deployment; and
- Critically evaluate the results and implications of the case study.

Teaching methodology: Expository classes in common classroom, using whiteboard, slide presentations, coding examples, books and scientific papers. Supplementary didactic materials will be available in Google Classroom. The development of the case study will happen during laboratory classes and home study hours, including programming and writing essays.

Grading: One individual written test in the 1st and another in the 2nd quarter. Essay and oral presentation about the case study (in groups) for the final exam.

Case study: Exactly 6 groups will be formed. Each group will be responsible for a case study. Students must choose a real-world problem and develop a data science project, including data collection, data transformation, inductive learning, validation, documentation, and deployment. The results must be presented in a short essay (max. 3 pages) and a 30 minutes presentation. The trained models must be incorporated in a data science product, such as a web application, a mobile application, or a web service.

Bibliography:

- Nina Zumel and John Mount. Practical Data Science with R. Manning, 2nd Edition, 2019.
- Hadley Wickham and Garret Grolemund, R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. O'Reilly Media, 2017.
- John D. Kelleher and Brendan Tierney. Data Science, MIT Press, 2018.

The first two books (Zumel and Mount, and Wickham and Grolemund) are available online for free.

Recommended reading:

- In-progress textbook at comp.ita.br/~verri/ds-book.
- V. N. Vapnik (1999a). "An overview of statistical learning theory". In: *IEEE Transactions on Neural Networks* 10.5, pp. 988–999. DOI: 10.1109/72.788640.
- A. Benavoli et al. (2017). "Time for a Change: a Tutorial for Comparing Multiple Classifiers Through Bayesian Analysis". In: *Journal of Machine Learning Research* 18.77, pp. 1–36. URL: http://jmlr.org/papers/v18/16-305.html.

Any extra material will be made available in Google Classroom.

Calendar: The expected schedule is presented below.

1st Quarter	
Week	Topics
1	Brief history of Data Science and CMC-13 review
2	Stages in a Data Science project
3	Tidy Data and data integration from multiple sources
4	Data engineering and shaping
5	Inductive learning and statistical learning theory
6	
7	Case study discussion and definitions
8	**Written test**

2nd Quarter	
Week	Topics
1	Experimental planning for Data Science
2	Model evaluation
3	Bayesian Analysis
4	Documentation and deployment
5	Privacy-preserving computational approaches
6	**Written test**
7	**Presentations and discussions**
8	

1 Brief history of data science

"Begin at the beginning," the King said gravely, "and go on till you come to the end: then stop."

— Lewis Carroll, *Alice in Wonderland*

There are many points-of-view about the beginning of data science. For the sake of contextualization, I separate the topic in two approaches: the history of the term itself and a broad timeline of data-driven sciences highlighting the important figures in each age.

I believe that the history of the term is important to understand the context of the discipline. Also, studying the main facts and figures in the history of data-driven sciences is important to understand the evolution of the field and hopefully to guide us to evolve it further.

1.1 The term "data science"

The term data science is recent and has been used to label rather different fields of study. In the following, I emphasize the history of a few notable usage of the term.

Peter Naur (1928 – 2016) The term "data science" itself was coined in the 1960s by Peter Naur (/naʊə/). Naur was a Danish computer scientist and mathematician who made significant contributions to the field of computer science, including his work on the development of program-

Chapter remarks

Context

- The term "data science" is recent and has been used to label rather different fields.
- The history of data-driven sciences is long and rich.

Objectives

- Understand the history of the term "data science."
- Understand the history of data-driven sciences.

Takeways

- There is no consensus on the definition of data science.
- There is enough evidence to support data science as a new science.

ming languages[1]. His ideas and concepts laid the groundwork for the way we think about programming and data processing today.

Naur disliked the term computer science and suggested it be called datalogy or data science. In the 1960s, the subject was practised in Denmark under Peter Naur's term datalogy, which means the science of data and data processes.

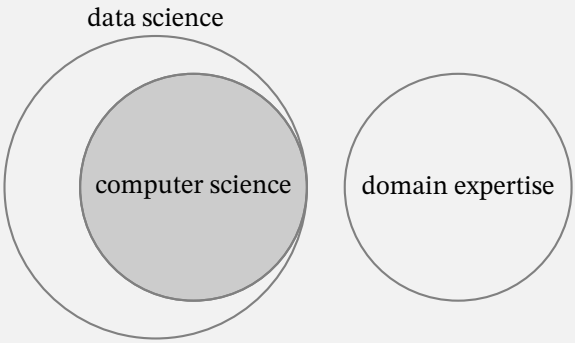
He coined this term to emphasize the importance of data as a fundamental component of computer science and to encourage a broader perspective on the field that included data-related aspects. At that time, the field was primarily centered on programming techniques, but Naur's concept broadened the scope to recognize the intrinsic role of data in computation.

In his book[2], "Concise Survey of Computer Methods", he parts from

[1] He is best remembered as a contributor, with John Backus, to the Backus–Naur form (BNF) notation used in describing the syntax for most programming languages.

[2] Peter Naur: Concise Survey of Computer Methods, 397 p. Studentlitteratur, Lund,

Figure 1.1: Naur's view of data science.



For him, data science studies the techniques to deal with data, but he delegates the meaning of data to other fields.

Slide 1.1: Peter Naur

- Danish computer scientist and mathematician.
- Coined the term "data science" in the 1960s.
- Proposed the term "datalogy" as an alternative to computer science.

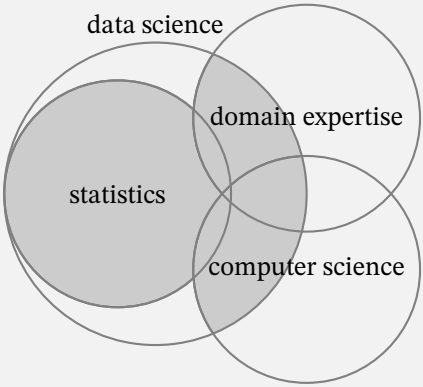
the concept that *data* is "a representation of facts or ideas in a formalised manner capable of being communicated or manipulated by some process."[3] Note however that his view of the science only "deals with data [...] while the relation of data to what they represent is delegated to other fields and sciences."

William Cleveland (born 1943) In 2001, a prominent statistician used the term "data science" in his work to describe a new discipline that comes from his "plan to enlarge the major areas of technical work

Sweden, ISBN 91-44-07881-1, 1974. http://www.naur.com/Conc.Surv.html

[3] I. H. Gould (ed.): 'IFIP guide to concepts and terms in data processing', North-Holland Publ. Co., Amsterdam, 1971.

Figure 1.2: Cleveland's view of data science.



For him, data science is the "modern" statistics, where it is enlarged by computer science and domain expertise.

of the field of statistics[4]." In 2014, that work was republished[5]. He advocates the expansion of statistics beyond theory into technical areas, significantly changing statistics. Thus, it warranted a new name.

Slide 1.2: William Cleveland

- American statistician.
- Proposed the discipline "data science" in 2001.
- Proposed the term "data science" as the new name for expansion of statistics.

As a result, William Swain Cleveland II is credited to define data science as it is most used today. He is a highly influential figure in the fields of statistics, machine learning, data visualization, data analysis

[4]W. S. Cleveland. Data Science: An Action Plan for Expanding the Technical Areas of the Field of Statistics. ISI Review, 69:21–26, 2001.

[5]W. S. Cleveland. Data Science: An Action Plan for the Field of Statistics. Statistical Analysis and Data Mining, 7:414–417, 2014. reprinting of 2001 article in ISI Review, Vol 69.

for multidisciplinary studies, and high performance computing for deep data analysis.

Buzzword or a new science? Be aware that literature has no consensus on the definition of data science, and it is still considered by some to be a buzzword[6].

Most of the usages of the term in literature and in the media are either a rough reference to a set of data-driven techniques or a marketing strategy. Naur (fig. 1.1) and Cleveland (fig. 1.2) are among the few that try to carefully define the term. However, both of them do not see data science as an independent field of study, but an enlarged scope of an existing science. I disagree; the social and economical demand for data-driven solutions led to an evolution in our understanding of the challenges we are facing. As a result, we see many "data scientist" being hired and many "data science degrees" programs emerging.

In chapter 3, I dare to provide a (yet another) definition for the term. I argue that its object of study can be precisely established to support it as a new science.

Slide 1.3: A new science

- Both Naur and Cleveland do not see data science as an independent field of study.
- I argue that data science is not a buzzword.
- Our social and economical reality demands a new science.

1.2 Timeline and historical markers

Kelleher and Tierney (2018) provides an interesting time line of data-driven methods and influential figures in the field. I reproduce it here with some changes, including some omissions and additions. On the subject of data analysis, I include some remarks from V. N. Vapnik (1999b).

I first address data handling — which I include data sources, collection, organization, storage, and transformation—, and then data analysis and knowledge extraction.

[6] Press, Gil. "Data Science: What's The Half-Life of a Buzzword?". Forbes. Available at https://www.forbes.com/sites/gilpress/2013/08/19/data-science-whats-the-half-life-of-a-buzzword/

1.2.1 Timeline of data handling

The importance of collecting and organizing data goes without saying. Data fuels analysis and decision making. In the following, I present some of the most important milestones in the history of data handling.

Figure 1.3: Timeline of the ages of data handling.

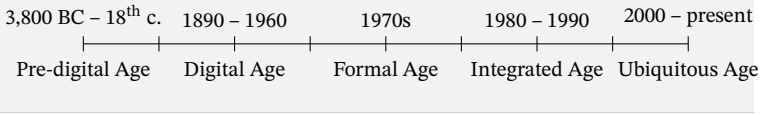


Figure 1.3 illutrates the timeline.

Pre-digital age

We can consider the earliest records of data collection to be the notches on sticks and bones to keep tracking of passing of time. The Lebombo bone, a baboon fibula with notches, is probably the earliest known mathematical object. It was found in the Lebombo Mountains located between South Africa and Eswatini. They estimate it is more than 40,000 years old. It is conjectured to be a tally stick, but its exact purpose is unknown. Its 29 notches suggests that may have been used as a lunar phase counter. However, since it is broken at one end, the 29 notches may or may not be the total number[7].

Another important milestone in the history of data collection is the record of demographic data. One of first known census was conducted in 3,800 BC in the Babylonian Empire. It was ordered to assess the population and resources of his empire. Records were stored on clay tiles[8].

Since the early forms of writing, humanity abilities to record events and information increased significantly. The first known written records date back to around 3,500 BC, the Sumerian archaic (pre-cuneiform) writing. This writing system was used to represent commodities using clay tokens and to record transactions[9].

[7] P. B. Beaumont and R. G. Bednarik (2013). In: *Rock Art Research* 30.1, pp. 33–54. URL: https://search.informit.org/doi/10.3316/informit.488018706238392.

[8] C. G. Grajalez et al. (2013). "Great moments in statistics". In: *Significance* 10.6, pp. 21–28. DOI: 10.1111/j.1740-9713.2013.00706.x.

[9] G. Ifrah (1998). *The Universal History of Numbers, from Prehistory to the Invention of the Computer*. First published in French, 1994. London: Harvill. ISBN: 1 86046 324 x.

"Data storage" was also a challenge. Some important devices that increased our capacity to register textual information are the Sumerian clay tablets (3,500 BC), the Egyptian papyrus (3,000 BC), the Greek alphabet (800 BC), the Roman wax tablets (100 BC), the codex (100 AD), the Chinese paper (200 AD), the printing press (1440), the typewriter (1868).

Besides those improvements in unstructured data storage, at least in the Western and Middle Eastern world, there are no significant advances in structured data collection until the 19th century. (A Eastern timeline research is pending.)

I consider a major influential figure in the history of data collection to be Florence Nightingale (1820 – 1910). She was a passionate statistician and probably the first person to use statistics to influence public and official opinion. The meticulous records she kept during the Crimean War (1853 – 1856) were the evidence that saved lives. She was also the first to use statistical graphics to present data in a way that was easy to understand. She is credited with developing a form of the pie chart now known as the polar area diagram. She also reformed healthcare in the United Kingdom and is considered the founder of modern nursing[10].

Digital age

In the modern period, several devices were developed to store digital[11] information. One particular device that is important for data collection is the punched card. It is a piece of stiff paper that contains digital information represented by the presence or absence of holes in predefined positions. The information can be read by a mechanical or electrical device called a card reader. The earliest famous usage of punched cards was by Basile Bouchon (1725) to control looms. Most of the advances until the 1880s were about the automation of machines (data processing) using hand-punched cards, and not particularly specialized for data collection.

However, the 1890 census in the United States was the first to use machine-readable punched cards to tabulate data. Processing 1880 census data took eight years, so the Census Bureau contracted Herman

[10] C. G. Grajalez et al. (2013). "Great moments in statistics". In: *Significance* 10.6, pp. 21–28. DOI: 10.1111/j.1740-9713.2013.00706.x.

[11] Digital means the representation of information in (finite) discrete form. The term comes from the Latin digitus, meaning finger, because it is the natural way to count using fingers.

Slide 1.4: Pre-digital age

- Babylonian census (3,800 BC)
- Sumerian archaic (pre-cuneiform) writing (3,500 BC)
- Egyptian papyrus (3,000 BC)
- Phoenician alphabet (1,000 BC)
- Greek alphabet (800 BC)
- Roman wax tablets (100 BC)
- Codex (100 AD)
- Chinese paper (200 AD)
- Printing press (1440)
- Typewriter (1868)
- Florence Nightingale (1820 – 1910)

Hollerith (1860 – 1929) to design and build a tabulating machine. He founded the Tabulating Machine Company in 1896, which later merged with other companies to become International Business Machines Corporation (IBM) in 1924. Later models of the tabulating machine were widely used for business applications such as accounting and inventory control. Punched card technology remained a prevalent method of data processing for several decades until more advanced electronic computers were developed in the mid-20th century.

The invention of the digital computer is responsible for a revolution in data handling. The amount of information we can capture and store increased exponentially. ENIAC (1945) was the first electronic general-purpose computer. It was Turing-complete, digital, and capable of being reprogrammed to solve a full range of computing problems. It had 20 words of internal memory, each capable of storing a 10-digit decimal integer number. Programs and data were entered by setting switches and inserting punched cards.

For the 1950 census, the United States Census Bureau used the UNIVAC I (Universal Automatic Computer I), the first commercially pro-

Slide 1.5: Florence Nightingale

- Passionate statistician.
- First person to use statistics to influence public and official opinion.
- Organized data from garden fruits and vegetables into numerical tables at the age of 9.
- At 20 she was receiving two-hour lessons from a Cambridge-trained mathematician.
- She found the sight of a long column of figures "perfectly reviving."
- She went out to the Crimean War, to Scutari in Turkey, in 1854.
- She found that not even the numbers of soldiers entering the hospitals, or leaving them – alive or dead – was known.
- From the first she kept meticulous records.
- The data she collected was the evidence that saved lives.
- She was the first to use statistical graphics to present data in a way that was easy to understand.
- She is credited with developing a form of the pie chart now known as the polar area diagram.
- She reformed healthcare in the United Kingdom and is considered the founder of modern nursing.

duced computer in the United States[12].

It goes without saying that digital computers have become much more powerful and sophisticated since then. The data collection process has been easily automated and scaled to a level that was unimaginable before. However, "where" storing data is not the only challenge. "How" to store data is also a challenge. The next period of history addresses this

[12]Read more in https://www.census.gov/history/www/innovations/.

issue.

Slide 1.6: Digital age

- Punched card (1725)
- 1890 census and Hollerith's tabulating machine (1890)
- ENIAC (1945)
- UNIVAC I used by the United Census Bureau (1950)

Formal age

In 1970, Edgar Frank Codd (1923 – 2003) published a paper entitled "A Relational Model of Data for Large Shared Data Banks"[13]. In this paper, he introduced the concept of a relational model for database management provide more details.

His work was a breakthrough in the field of data management. The standardization of relational databases led to the development of Structured Query Language (SQL) in 1974. SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS).

As a result, a new challenge rapidly emerged: how to aggregate data from different sources. Once data is stored in a relational database, it is easy to query and manage it. However, data is usually stored in different databases, and it is not always possible to combine them.

Integrated age

The solution to this problem was the development of the Extract, Transform, Load (ETL) process. ETL is a process in data warehousing responsible for extracting data from several sources, transforming it into a format that can be analyzed, and loading it into a data warehouse.

The concept of data warehousing dates back to the late 1980s when IBM researchers Barry Devlin and Paul Murphy developed the "business data warehouse"

[13] E. F. Codd (1970). "A Relational Model of Data for Large Shared Data Banks". In: *Commun. ACM* 13.6, pp. 377–387. ISSN: 0001-0782. DOI: 10.1145/362384.362685.

Slide 1.7: Edgar Frank Codd

- British computer scientist
- Introduced the concept of a relational model for database management
- Standardized relational databases
- Led the development of Structured Query Language (SQL)

Two major figures in the history of ETL are Ralph Kimball and Bill Inmon. Although they differ in their approaches, they both agree that data warehousing is the foundation for business intelligence (BI) and analytics, and that data warehouses should be designed to be easy to understand and fast to query for business users.

A famous debate between Kimball and Inmon is the top-down versus bottom-up approach to data warehousing. Inmon's approach is top-down, where the data warehouse is designed first and then the data marts[14] are created from the data warehouse. Kimball's approach is bottom-up, where the data marts are created first and then the data warehouse is created from the data marts.

Slide 1.8: Ralph Kimball

- American computer scientist.
- Developed the bottom-up approach to data warehousing.
- From available data marts, the data warehouse is created.

One of the earliest and most famous case studies of the implementation of a data warehouse is that of Walmart. In the early 1990s, Walmart faced the challenge of managing and analyzing vast amounts of data from its stores across the United States. The company needed a solution that would enable comprehensive reporting and analysis to support decision-making processes.

[14] A data mart is a specialized subset of a data warehouse that is designed to serve the needs of a specific business unit, department, or functional area within an organization.

Slide 1.9: Bill Inmon

- American computer scientist.
- Developed the top-down approach to data warehousing.
- Design the data warehouse first and then the data marts are created from the data warehouse.

Ubiquitous age

The last and current period of history is the ubiquitous age. It is characterized by the proliferation of data sources.

The ubiquity of data generation and the evolution of data-centric technologies have been made possible by a multitude of figures across various domains.

- Tim Berners-Lee, credited with inventing the World Wide Web, laid the foundation for the massive data flow on the internet.
- Vinton Cerf and Robert Kahn, often referred to as the "Fathers of the Internet," developed the TCP/IP protocols, which are fundamental to internet communication.
- Steve Jobs and Steve Wozniak (Apple Inc.) and Bill Gates (Microsoft Corporation), the introduction of personal computers, leading to the democratization of data generation.
- Mark Zuckerberg, the co-founder of Facebook, played a crucial role in the rise of social media and the generation of vast amounts of user-generated content.
- Larry Page and Sergey Brin, the founders of Google, transformed how we access and search for information.
- Elon Musk and Tesla, the rise of the Internet of Things (IoT) and connected devices.

In terms of data handling, this change in the data landscape has brought about the development of new technologies and techniques for data storage and processing. Especially the development of NoSQL databases and distributed computing frameworks.

NoSQL databases are non-relational databases that can store and process large volumes of unstructured, semi-structured, and structured data. They are highly scalable and flexible, making them ideal for big data applications.

Slide 1.10: The V's of big data

- Volume: The amount of data generated is massive
- Velocity: The speed at which data is generated is high
- Variety: The types of data generated are diverse
- Veracity: The quality of data generated is questionable
- Value: The value of data generated is high

Once massive amounts of unstructured data became available, the need for new data processing techniques arose. The development of distributed computing frameworks such as Apache Hadoop and Apache Spark enabled the processing of massive amounts of data in a distributed manner.

Doug Cutting and Mike Cafarella, the developers of Apache Hadoop, revolutionized big data proposed the Hadoop Distributed File System (HDFS) and MapReduce, which are the cornerstones of the Hadoop framework, in 2006. Hadoop's distributed storage and processing capabilities enabled organizations to handle and analyze massive volumes of data.

Currently, Google holds a patent for MapReduce[15]. However, their framework inherits from the architeture proposed in Hillis (1985) thesis.

MapReduce is not particularly novel, but its simplicity and scalability made it popular.

Nowadays, another important topic is Internet of Things (IoT). IoT is a system of interrelated computing devices that communicate with each other over the internet. The devices can be anything from cellphones, coffee makers, washing machines, headphones, lamps, wearable devices, and almost anything else you can think of. IoT increased the challenges of data handling, especially in terms of data storage and processing.

[15] http://static.googleusercontent.com/media/research.google.com/es/us/archive/mapreduce-osdi04.pdf

1.2.2 Timeline of data analysis

The way we think about data and knowledge extraction has evolved significantly over the years. In the following, I present some of the most important milestones in the history of data analysis.

Summary statistics

The earliest known records of systematic data analysis date back to the first censuses. The term *statistics* itself refer to the analysis of data *about the state*, including demographics and economics. That early (and simplest) form of statistical analysis is called *summary statistics*, which consists of describing data in terms of its central tendencies (e.g. arithmetic mean) e variability (e.g. range).

Probability advent

However, after the seventeenth century, the foundations of modern probability theory were laid out. Important figures for developing the probability theory are Blaise Pascal (1623 – 1662), Pierre de Fermat (1607 – 1665), Christiaan Huygens (1629 – 1695), and Jacob Bernoulli (1655 – 1705).

The foundation methods brought to life the field of statistical inference. In the following years, important results were achieved.

Bayes' rule Reverend Thomas Bayes (1701 – 1761) was an English statistician, philosopher, and presbyterian minister. He is known for formulating a specific case of the theorem that bears his name: Bayes' theorem. The theorem is used to calculate conditional probabilities using an algorithm (his Proposition 9, published in 1763) that uses evidence to calculate limits on an unknown parameter.

The Bayes' rule is the foundation of learning from evidence, once it allows us to calculate the probability of an event based on prior knowledge of conditions that might be related to the event. Classifiers based on Naïve Bayes — the application of Bayes' theorem with strong independence assumptions between known variables — is likely to have been used since the second half of the eighteenth century.

Gauss' method of least squares Johann Carl Friedrich Gauss (1777 – 1855) was a German mathematician and physicist who made significant contributions to many fields in mathematics and sciences. Circa

1794, he developed the method of least squares for calculating the orbit of Ceres to minimize the impact of measurement error[16].

Playfair's data visualization William Playfair (1759 – 1823) was a secret agent on behalf of Great Britain during its war with France in the 1790s. He invented several types of diagram between 1780s and 1800s, such as the line, area and bar chart of economic data, and the pie chart and circle graph to show proportions.

Slide 1.11: Probability advent

- Foundations by Blaise Pascal, Pierre de Fermat, Christiaan Huygens, Jacob Bernoulli, and others (17^{th} century)
- Bayes' rule by Thomas Bayes (1763)
- Gauss' method of least squares by Johann Carl Friedrich Gauss (1794)
- Playfair's data visualization by William Playfair (1780s – 1800s)

Learning from data

In the twentieth century and beyond, new advances were made in the field of statistics. The development of learning methods[17] enabled the development of new techniques for data analysis.

Fisher's discriminant analysis In the 1930s, Ronald Fisher (1890 – 1962) developed discriminant analysis, which was considered a problem of constructing a decision rule to assign a vector to one of two categories using given probability distribution functions[18].

See **??** for more details about the technique.

[16] The method was first published by Adrien-Marie Legendre (1752 – 1833) in 1805, but Gauss claimed in 1809 that he had been using it since circa 1794.

[17] Vapnik uses the terminology *learning machines*.

[18] After, Rosenblatt's work, however, it was used to solve inductive inference as well.

Shannon's information theory The field, that studies quantification, storage and communication of information, was originally established by the works of Harry Nyquist (1889 – 1976) and Ralph Hartley (1888 – 1970) in the 1920s, and Claude Shannon (1916 – 2001) in the 1940s. Information theory brought many important concepts to the field of data analysis, such as entropy, mutual information, and information gain.

K-Nearest Neighbors In 1951, Evelyn Fix (1904 – 1965) and Joseph Lawson Hodges Jr. (1922 – 2000) wrote a technical report entitled "Discriminatory Analysis, Nonparametric Discrimination: Consistency Properties." In this paper, they proposed the k-nearest neighbors algorithm, which is a non-parametric method used for classification and regression.

See **??** for more details about the technique.

Rosenblatt's perceptron In the 1960s, Frank Rosenblatt (1928 – 1971) developed the perceptron, the first model of a learning machine. While the idea of a mathematical neuron was not new, he was the first to describe the model as a program, showing the ability of the perceptron to learn simple tasks such as the logical operations AND and OR.

See **??** for more details about the technique.

Hunt inducing trees In 1966, Hunt, Marin, and Stone's book[19] described a way to induce decision trees from data. The algorithm is based on the concept of information entropy and is a precursor of the Quinlan's ID3 algorithm[20] and its variations.

See **??** for more details about the technique.

Empirical risk minimization principle Although many learning machines where developed until the 1960s, they did not advanced significantly the understanding of the general problem of learning from data. Between 1960s and 1986 — before the backpropagation algorithm was proposed —, the field of practical data analysis was basically stagnant. The main reason for that was the lack of a theoretical framework to support the development of new learning machines.

[19] E. B. Hunt, J. Marin, and P. J. Stone (1966). *Experiments in Induction*. New York, NY, USA: Academic Press.

[20] J. R. Quinlan (1986). "Induction of Decision Trees". In: *Machine Learning* 1, pp. 81–106. URL: https://api.semanticscholar.org/CorpusID:13252401.

Slide 1.12: Learning from data (part I)

- Fisher's discriminant analysis by Ronald Fisher (1930s)
- Shannon's information theory by Claude Shannon (1940s)
- K-Nearest Neighbors by Evelyn Fix and Joseph Lawson Hodges Jr. (1951)
- Rosenblatt's perceptron by Frank Rosenblatt (1960s)
- Hunt inducing trees by John Ross Quinlan (1966)

However, these years were not completely unfruitful. As early as 1968, Vladimir Vapnik (1936 –) and Alexey Chervonenkis (1938 – 2014) developed the foundamental concepts of VC entropy and VC dimension for the data classification problems. As a result, a novel inductive principle was proposed: the Empirical Risk Minimization (ERM) principle. This principle is the foundation of statistical learning theory.

Resurgence of neural networks In 1986, researchers developed independently a method to optimize coefficients of a neural network[21]. The method is called backpropagation and is the foundation of the resurgence of neural networks.

This rebirth of neural networks happened in a scenario very different from the 1960s. The availability of data and computational power fueled a new approach to the problem of learning from data. The new approach preferred the use of simple algorithms and intuitive models over theoretical models, fueling areas such as bioinspired computing and evolutionary computation.

[21] Y. Le Cun (1986). "Learning Process in an Asymmetric Threshold Network". In: *Disordered Systems and Biological Organization*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 233–240. ISBN: 978-3-642-82657-3; D. E. Rumelhart, G. E. Hinton, and R. J. Williams (1986). "Learning representations by back-propagating errors". In: *Nature* 323.6088, pp. 533–536. DOI: 10.1038/323533a0.

Ensembles Following the new approach, ensemble methods were developed. Based on ideas of boosting[22] and bagging[23], ensemble methods combine multiple learning machines to improve the performance of the individual machines. The most famous ensemble methods are random forests[24].

Support vector machines In 1995, Cortes and V. Vapnik proposed the support vector machine (SVM) algorithm, a learning machine based on the VC theory and the ERM principle. Based on Cover's theorem[25], they developed a method that finds the optimal hyperplane that separates two classes of data with the maximum margin in a high-dimensional space. The resulting method led to practical and efficient learning machines.

Deep learning revolution Although the ideia of neural networks with multiple layers were around since the 1960s, only in the late 2000s the field of deep learning caught the attention of the scientific community by achieving state-of-the-art results in computer vision and natural language processing. Yoshua Bengio, Geoffrey Hinton and Yann LeCun are recognized for their for conceptual and engineering breakthroughs in the field, winning the 2018 Turing Award.

Generative deep models Nowadays, generative deep models are a hot topic in machine learning. They are a class of statistical models that can generate new data instances. They are used in unsupervised learning to discover hidden structures in unlabeled data (e.g. clustering), and in supervised learning to generate new synthetic data instances. The most famous generative models are the generative transformers and generative adversarial networks.

---

[22] R. E. Schapire (1990). "The strength of weak learnability". In: *Machine Learning* 5.2, pp. 197–227. DOI: 10.1007/BF00116037.

[23] L. Breiman (1996). "Bagging predictors". In: *Machine Learning* 24.2, pp. 123–140. DOI: 10.1007/BF00058655.

[24] T. K. Ho (1995). "Random decision forests". In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. Vol. 1, 278–282 vol.1. DOI: 10.1109/ICDAR.1995.598994.

[25] T. M. Cover (1965). "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition". In: *IEEE Transactions on Electronic Computers* EC-14.3, pp. 326–334. DOI: 10.1109/PGEC.1965.264137.

LUSI learning theory In 2010s, Vapnik and Rauf Izmailov proposed the Learning Using Statistical Invariants (LUSI) principle, which is an extension of the statistical learning theory. The LUSI theory is based on the concept of statistical invariants, which are properties of the data that are preserved under transformations. The theory is the foundation of the learning from intelligent teachers paradigm. They regard the LUSI theory as the next step in the evolution of learning theory, calling it the "complete statistical theory of learning".

Slide 1.13: Learning from data (part II)

- Empirical risk minimization principle by Vladimir Vapnik and Alexey Chervonenkis (1968)
- Resurgence of neural networks (1986)
- Ensembles (1990s)
- Support vector machines by Vladimir Vapnik and Corinna Cortes (1995)
- Deep learning revolution (2000s)
- Generative models (2010s)
- LUSI learning theory by Vladimir Vapnik and Rauf Izmailov (2010s)

2

Preliminaries

Maar ik maak steeds wat ik nog niet kan om het te leeren kunnen.

— Vincent van Gogh, *The Complete Letters of Vincent Van Gogh, Volume Three*

Foundamental concepts in data science come from a variety of fields, including mathematics, statistics, computer science, optimization theory, and information theory. This chapter provides a brief overview of the computational, mathematical and statistical concepts that are used in the rest of the book.

2.1 Algorithms and data structures

Algorithms are step-by-step procedures for solving a problem. They are used to manipulate data structures, which are ways of organizing data to solve problems. They are realized in programming languages, which are formal languages that can be used to express algorithms.

2.1.1 Algoritmic paradigms

Some techniques are used to solve a wide variety of problems. They are called algorithmic paradigms. The most common ones are listed below.

Divide and conquer The problem is divided into smaller subproblems that are solved recursively. The solutions to the subproblems are then combined to give a solution to the original problem. Some example algorithms are merge sort, quick sort, and binary search.

Greedy algorithms The problem is solved with incremental steps, each of which is locally optimal. The overall solution is not guaranteed to be (but might be) optimal. Some example algorithms are Dijkstra's algorithm and Prim's algorithm

Backtracking The problem is solved incrementally, one piece at a time. If a piece does not fit, it is removed and replaced by another piece. Some example algorithms are the naïve solutions for N-queens problem and the Sudoku problem.

2.1.2 Computational complexity

The computational complexity of an algorithm is the amount of resources it uses to run as a function of the size of the input. The most common resources are time and space.

Usually, we are interested in the asymptotic complexity of an algorithm, i.e. how the complexity grows as the size of the input grows. The most common notation for asymptotic complexity is the Big-O notation.

Big-O notation Let f and g be functions from the set of natural numbers to the set of real numbers, i.e. $f, g : \mathbb{N} \to \mathbb{R}$. We say that f is $O(g)$ if there exists a constant $c > 0$ such that $f(n) \leq cg(n)$ for all $n \geq n_0$, where n_0 is a natural number.

We can order functions by their asymptotic complexity. For example, $O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(2^n) < O(n!)$. The Big-O notation is used to describe the asymptotic complexity of algorithms.

2.1.3 Data structures

Data structures are ways of organizing data to solve problems. The most common ones are listed below.

Arrays An array is a homogeneous collection of elements that are accessed by an integer index. The elements are usually stored in contiguous memory locations.

Linked lists A linked list is a collection of elements called nodes. Each node contains a value and a pointer to the next node in the list. The first node is called the head, and the last node is called the tail. The tail points to a null reference.

Stacks A stack is a collection of elements that are accessed in a last-in-first-out (LIFO) order. Elements are added to the top of the stack and removed from the top of the stack.

Queues A queue is a collection of elements that are accessed in a first-in-first-out (FIFO) order. Elements are added to the back of the queue and removed from the front of the queue.

Trees A tree is a collection of nodes. Each node contains a value and a list of references to its children. The first node is called the root. A node with no children is called a leaf.

Graphs A graph is a collection of nodes. Each node contains a value and a list of references to its neighbors; the references are called edges. A graph can be directed or undirected. A graph is directed if the edges have a direction.

Map A map is a collection of key-value pairs. The keys are unique, and each key is associated with a value. The keys are used to access the values.

2.2 Set theory

A set is a collection of elements. The elements of a set can be anything, including other sets. The elements of a set are unordered, and each element is unique. The most common notation for sets is the curly braces notation, e.g. $\{1, 2, 3\}$.

2.2.1 Set operations

Union The union of two sets A and B is the set of elements that are in A or B. It is denoted by $A \cup B$.

Intersection The intersection of two sets A and B is the set of elements that are in both A and B. It is denoted by $A \cap B$.

Difference The difference of two sets A and B is the set of elements that are in A but not in B. It is denoted by $A \setminus B$.

Union and intersection are commutative, associative and distributive.

Universe set The universe set is the set of all elements. It is denoted by Ω.

Empty set The empty set is the set with no elements. It is denoted by $\varnothing$.

Complement The complement of a set A is the set of elements that are not in A. It is denoted by $A^c = \Omega \setminus A$.

Inclusion Given sets A, B, and C, the following statements hold:

- *Reflexity:* $A \subseteq A$;
- *Antisymmetry:* $A \subseteq B$ and $B \subseteq A$ if and only if $A = B$;
- *Transitivity:* $A \subseteq B$ and $B \subseteq C$ implies $A \subseteq C$.

2.3 Linear algebra

Vector A vector is an ordered collection of numbers. It is denoted by a bold lowercase letter, e.g. $\mathbf{v} = [v_i]_{i=1,\ldots,n}$ is a vector of length n.

Matrix A matrix is a rectangular collection of numbers. It is denoted by an uppercase letter, e.g. $A = (a_{ij})_{i=1,\ldots,n;\ j=1,\ldots,m}$ is the matrix with n rows and m columns.

2.3.1 Operations

Addition The sum of two vectors $\mathbf{v}$ and $\mathbf{w}$ is the vector $\mathbf{v} + \mathbf{w}$ whose i-th entry is $v_i + w_i$. The sum of two matrices A and B is the matrix $A + B$ whose i, j-th entry is $a_{ij} + b_{ij}$. (The same rules apply to subtraction.)

Scalar multiplication The product of a scalar α and a vector $\mathbf{v}$ is the vector $\alpha\mathbf{v}$ whose i-th entry is αv_i. The product of a scalar α and a matrix A is the matrix αA whose i, j-th entry is αa_{ij}.

Dot product The dot product of two vectors $\mathbf{v}$ and $\mathbf{w}$ is the scalar

$$\mathbf{v} \cdot \mathbf{w} = \sum_{i=1}^{n} v_i w_i.$$

The dot product is also called the inner product.

Matrix multiplication The product of two matrices A and B is the matrix $C = AB$ whose i, j-th entry is

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}.$$

Unless otherwise stated, we consider the vector $\mathbf{v}$ as a column matrix.

Transpose The transpose of a matrix A is the matrix A^T whose i, j-th entry is the j, i-th entry of A. If A is a square matrix, then A^T is the matrix obtained by reflecting A along its main diagonal.

Determinant The determinant of a square matrix A is a scalar that is a measure of the (signed) volume of the parallelepiped spanned by the columns of A. It is denoted by $\det(A)$ or $|A|$.

The determinant is nonzero if and only if the matrix is invertible and the linear map represented by the matrix is an isomorphism – i.e. it preserves the dimension of the vector space. The determinant of a product of matrices is the product of their determinants.

Particularly, the determinant of a 2×2 matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc.$$

Inverse matrix An $n \times n$ matrix A has an inverse $n \times n$ matrix A^{-1} if

$$AA^{-1} = A^{-1}A = I_n,$$

where I_n is the $n \times n$ identity matrix. If such a matrix exists, A is said *invertible*. A square matrix that is not invertible is called singular. A square matrix with entries in a field is singular if and only if its determinant is zero.

To calculate the inverse of a matrix, we can use the formula

$$A^{-1} = \frac{1}{\det(A)} \operatorname{adj}(A),$$

where $\operatorname{adj}(A)$ is the adjugate (or adjoint) of A, i.e. the transpose of the cofactor matrix of A.

The cofactor of the i, j-th entry of a matrix A is the determinant of the matrix obtained by removing the i-th row and the j-th column of A, multiplied by $(-1)^{i+j}$.

In the case of a 2×2 matrix, the inverse is

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}.$$

2.3.2 Systems of linear equations

A system of linear equations is a collection of linear equations that share their unknowns. It is usually written in matrix form as $A\mathbf{x} = \mathbf{b}$, where A is a matrix of constants, $\mathbf{x}$ is a vector of unknowns, and $\mathbf{b}$ is a vector of constants.

The system has a unique solution if and only if the matrix A is invertible. The solution is $\mathbf{x} = A^{-1}\mathbf{b}$.

2.3.3 Eigenvalues and eigenvectors

An eigenvalue of a square matrix A is a scalar λ such that there exists a non-zero vector $\mathbf{v}$ satisfying

$$A\mathbf{v} = \lambda\mathbf{v}. \tag{2.1}$$

The vector $\mathbf{v}$ is called an eigenvector of A corresponding to λ.

The eigenvalues of a matrix are the roots of its characteristic polynomial, i.e. the roots of the polynomial $\det(A - \lambda I) = 0$, where I is the identity matrix.

2.4 Probability

The Kolmogorov axioms of probability are the foundation of probability theory. They are

1. The probability of an event A is a non-negative real number, i.e. $P(A) \geq 0$;
2. The probability of the sample space Ω is one, i.e. $P(\Omega) = 1$; and
3. The probability of the union of disjoint events, $A \cap B = \emptyset$, is the sum of the probabilities of the events, i.e. $P(A \cup B) = P(A) + P(B)$.

If A and B are not disjoint, then

$$P(A \cup B) = P(A) + P(B) - P(A \cap B).$$

Joint probability The joint probability of two events A and B is the probability that both events occur. It is denoted by $P(A, B) = P(A \cap B)$.

Law of total probability The law of total probability states that if $B_1, \dots, B_n$ are disjoint events such that $\cup_{i=1}^{n} B_i = \Omega$, then for any event A, we have that

$$P(A) = \sum_{i=1}^{n} P(A, B_i).$$

Conditional probability The conditional probability of an event A given an event B is the probability that A occurs given that B occurs. It is denoted by $P(A \mid B)$.

Independence Two events A and B are independent if the probability of A given B is the same as the probability of A, i.e. $P(A \mid B) = P(A)$. It is equivalent to $P(A, B) = P(A) \cdot P(B)$.

Bayes' rule Bayes' rule is a formula that relates the conditional probability of an event A given an event B to the conditional probability of B given A. It is

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}. \tag{2.2}$$

2.4.1 Random variables

A random variable is a function that maps the sample space Ω to the real numbers. It is denoted by a capital letter, e.g. X.

Formally, let $X : \Omega \to E$ be a random variable. The probability that X takes on a value in a set $A \subseteq E$ is

$$P(X \in A) = P(\{\omega \in \Omega : X(\omega) \in A\}). \tag{2.3}$$

If $E = \mathbb{R}$, then X is a continuous random variable. If $E = \mathbb{Z}$, then X is a discrete random variable.

Probability mass function The probability mass function (PMF) of a discrete random variable X is the function $p_X : \mathbb{R} \to [0, 1]$ defined by

$$p_X(x) = P(X = x). \tag{2.4}$$

Probability density function The probability density function (PDF) of a continuous random variable X is the function $f_X : \mathbb{R} \to [0, \infty)$ defined by

$$P(a \leq X \leq b) = \int_a^b f_X(x)dx. \tag{2.5}$$

Cumulative distribution function The cumulative distribution function (CDF) of a random variable X is the function $F_X : \mathbb{R} \to [0, 1]$ defined by

$$F_X(x) = P(X \leq x). \tag{2.6}$$

2.4.2 Expectation and moments

Expectation is a measure of the average value of a random variable. Moments are measures of the shape of a probability distribution.

Expectation The expectation of a random variable X is the average value of X. It is denoted by $E[X]$. By definition, it is

$$E[X] = \begin{cases} \sum_x x \cdot p_X(x) & \text{if } X \text{ is discrete;} \\ \int_{-\infty}^{\infty} x \cdot f_X(x)dx & \text{if } X \text{ is continuous.} \end{cases} \tag{2.7}$$

Given α a real number, the expectation of αX is $\alpha E[X]$.

Given c a real number, the expectation of $X + c$ is $E[X] + c$.

For any two random variables X and Y, the expectation of $X + Y$ is $E[X] + E[Y]$.

For a function $g : \mathbb{R} \to \mathbb{R}$, the expectation of $g(X)$ is

$$E[g(X)] = \begin{cases} \sum_x g(x) \cdot p_X(x) & \text{if } X \text{ is discrete;} \\ \int_{-\infty}^{\infty} g(x) \cdot f_X(x)dx & \text{if } X \text{ is continuous.} \end{cases} \tag{2.8}$$

Variance The variance of a random variable X is a measure of how spread out the values of X are. It is denoted by $V(X)$. By definition, it is

$$V(X) = E\Big[(X - E[X])^2\Big]. \tag{2.9}$$

Note that, as a consequence, the expectation of X^2 is

$$E[X^2] = V(X) + E[X]^2,$$

since

$$\begin{aligned} V(X) &= E\Big[\big(X - E[X]\big)^2\Big] \\ &= E\big[X^2 - 2XE[X] + E[X]^2\big] \\ &= E[X^2] - 2E[X]E[X] + E[X]^2 \\ &= E[X^2] - E[X]^2. \end{aligned}$$

Higher moments are defined similarly, look for skewness and kurtosis.

Law of large numbers The law of large numbers states that the average of a large number of independent and identically distributed random variables converges to the expectation of the random variable. Mathematically,

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} X_i = E[X],$$

given $X_i \sim X$ for all i.

2.4.3 Probability distributions

Several phenomena in nature and society can be modeled as random variables. Some distributions are frequently used to model these phenomena. The main ones are listed below.

Bernoulli distribution The Bernoulli distribution is a discrete distribution with two possible outcomes, usually called success and failure. It is parametrized by a single parameter $p \in [0, 1]$, which is the probability of success. It is denoted by $\text{Bern}(p)$.

The expected value of $X \sim \text{Bern}(p)$ is $E[X] = p$, and the variance is $V(X) = p(1 - p)$.

Normal distribution The normal distribution is a continuous distribution with a bell-shaped density. It is parametrized by two parameters, the mean $\mu \in \mathbb{R}$ and the standard deviation $\sigma > 0$. It is denoted by $\mathcal{N}(\mu, \sigma^2)$.

The special case where $\mu = 0$ and $\sigma = 1$ is called the standard normal distribution. It is denoted by $\mathcal{N}(0, 1)$.

The probability density function of $X \sim \mathcal{N}(\mu, \sigma^2)$ is

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{2.10}$$

The expected value of $X \sim \mathcal{N}(\mu, \sigma^2)$ is $E[X] = \mu$, and the variance is $V(X) = \sigma^2$.

T distribution The T distribution is a continuous distribution with a bell-shaped density. It is parametrized by a single parameter $\nu > 0$, called the degrees of freedom. It is denoted by $\mathcal{T}(\nu)$.

The T distribution generalizes to the three parameter location-scale t distribution $\mathcal{T}(\mu, \sigma^2, \nu)$, where μ is the location parameter and σ is the scale parameter. Thus, given $X \sim \mathcal{T}(\nu)$, we have that $\mu + \sigma X \sim \mathcal{T}(\mu, \sigma^2, \nu)$.

Note that

$$\lim_{\nu \to \infty} \mathcal{T}(\nu) = \mathcal{N}(0, 1).$$

2.4.4 Permutations and combinations

Permutation A permutation is an arrangement of a set of elements. The number of permutations of n elements is $n!$.

Combination A combination is a selection of a subset of elements from a set. The number of combinations of k elements from a set of n elements is

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}.$$

3

Fundamental data concepts

The simple believes everything,

but the prudent gives thought to his steps.

— Proverbs 14:15 *(ESV)*

A useful start point for someone studying data science is the definition of the term itself.

For Zumel and Mount (2019), *"data science is a cross-disciplinary practice that draws on methods from data engineering, descriptive statistics, data mining, machine learning, and predictive analytics."* They compare the area with the operations research, stating that data science focuses on implementing data-driven decisions and managing their consequences.

Slide 3.1: Zumel and Mount's definition

- Cross-disciplinary practice that draws on methods from data engineering, descriptive statistics, data mining, machine learning, and predictive analytics.
- Focuses on implementing data-driven decisions and managing their consequences.

Wickham, Çetinkaya-Rundel, and Grolemund (2023) state that *"data science is an exciting discipline that allows you to transform raw data into understanding, insight, and knowledge."*

Slide 3.2: Wickham's definition

- Transform raw data into understanding, insight, and knowledge.
- Not necessarily a definition; describes the purpose of data science.

I find the first definition too restrictive once new methods and techniques are always under development. We never know when new "data-related" methods will become obsolete or a trend. Also, Zumel and Mount's view gives the impression that data science is a operations research subfield. Although I do not try to prove otherwise, I think it is much more useful to see it as an independent field of study. Obviously, there are many intersections between both areas (and many other areas as well). Because of such intersections, I try my best to keep definitions and terms standardized throughout chapters, sometimes avoiding popular terms that may generate ambiguities or confusion.

The second one is not really a definition. However, it states clearly *what* data science enables us to do. From these thoughts, let's define the term.

Definition 3.1

Data science is the study of computational methods to extract knowledge from measurable phenomena.

I want to highlight the meaning of some terms in this definition. *Computational methods* means that data science methods use computers to handle data and perform the calculations. *Knowledge* means information that humans can easily understand or apply to solve problems. *Measurable phenomena* are events or processes where raw data can be quantified in some way[1]. *Raw data* are data collected directly from some source and that have not been subject to any other transformation by a software program or a human expert. *Data* is any piece of information that can be digitally stored.

Kelleher and Tierney (2018) summarize very well the challenges data science takes up: "extracting non-obvious and useful patterns from

[1] TODO: talk about non-measurable phenomena

Slide 3.3: My definition

- Data science is the study of computational methods to extract knowledge from measurable phenomena.
- Computational methods use computers to handle data and perform the calculations.
- Knowledge is information that humans can easily understand or apply to solve problems.
- Measurable phenomena are events or processes where raw data can be quantified in some way.
- Raw data are data collected directly from some source and that have not been subject to any other transformation by a software program or a human expert.
- Data is any piece of information that can be digitally stored.

large data sets [...]; capturing, cleaning, and transforming [...] data; [storing and processing] big [...] data sets; and questions related to data ethics and regulation."

Slide 3.4: Kelleher and Tierney's challenges

- Extracting non-obvious and useful patterns from large data sets.
- Capturing, cleaning, and transforming data.
- Storing and processing big data sets.
- Questions related to data ethics and regulation.

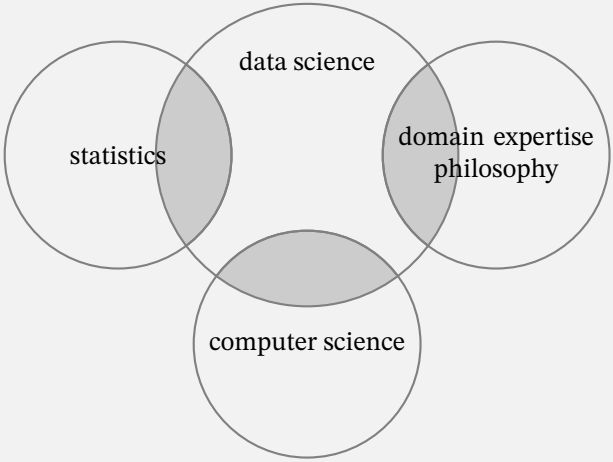
Data science contrasts with conventional sciences. Usually, a "science" is named after its object of study. Biology is the study of the life, Earth science studies the planet Earth, and so on. I argue that data science does not study data itself, but how we can use them to understand a phenomenon.

Besides, the conventional scientific paradigm is essentially model-

driven: we observe a phenomenon related to the object of study, we reason the possible explanation (the model or hypothesis), and we validate our hypothesis (most of the time using data, though). In data science, however, we extract the knowledge directly and primarily from the data. The expert knowledge and reasoning may be taken into account, but we give data the opportunity to surprise us.

Thus, the objects of the study in data science are the computational methods and processes that can extract reliable and ethical knowledge from huge amounts of data.

Figure 3.1: My view of data science.



Data science is an entire new science. Being a new science does not mean that its basis is built from the ground up. Most of the subjects in data science come from other sciences, but its object of study (computational methods to extract knowledge from measurable phenomena) is particular enough to unfold new scientific questions – such as data ethics, data collection, etc.

3.1 Fundamental data theory

As expected, data science is not a isolated science. It incorporates several concepts from other fields and sciences. In this section, I explain

Slide 3.5: Data science vs conventional sciences

- Conventional sciences are model-driven: observation, hypothesis, and validation.
- In data science, we extract the knowledge directly and primarily from the data.
- Data science studies the computational methods and processes that can extract reliable and ethical knowledge from huge amounts of data.

the basis of each component of the provided definition.

3.1.1 Phenomena

Phenomenon is a term used to describe any observable event or process. They are the source we use to understand the world around us. In general, we use our senses to perceive phenomena. To make sense of them, we use our knowledge and reasoning.

Philosophy is the study of knowledge and reasoning. It is a very broad field of study that has been divided into many subfields. One of them is epistemology, which is the study of knowledge. Epistemology is the field of philosophy that studies how we can acquire knowledge and how we can distinguish between knowledge and opinion. In particular, epistemology studies the nature of knowledge, justification, and the rationality of belief.

Another important subfield in philosophy is ontology, which is the study of being. It studies the nature of being, existence, or reality. Ontology is the field of philosophy that studies what exists and how we can classify it. In particular, ontology studies the nature of categories, properties, and relations.

Finally, logic is the study of reasoning. It studies the nature of reasoning and argumentation. In particular, logic studies the nature of inference, validity, and fallacies.

In the context of data science, we usually focus on phenomena from particular domain of expertise. For example, we may be interested in the phenomena related to the stock market, the phenomena related to the weather, or the phenomena related to the human health. Thus, we need to understand the nature of the phenomena we are studying.

Slide 3.6: Philosophy

- Epistemology: the study of knowledge.
- Ontology: the study of being.
- Logic: the study of reasoning.

Thus, fully understading the phenomena we are tackling requires both a general knowledge of epistemology, ontology, and logic, and a particular knowledge of the domain of expertise.

Observe as well that we do not restrict ourselves to the "qualitative" understanding of philosophy. There are several computational methods that implements the concepts of epistemology, ontology, and logic. For example, we can use a computer to perform deductive reasoning, to classify objects, or to validate an argument. Also, we have strong mathematical foundations and computational tools to organize categories, relations, and properties.

The reason we need to understand the nature of the phenomena we are studying is that we need to guarantee that the data we are collecting are relevant to the problem we are trying to solve. Incorrectly perception of the phenomena may lead to incorrect data collection, which may lead to incorrect conclusions.

3.1.2 Measuments

In data science, we are interested in measurable phenomena. Measurable phenomena are those that we can quantify in some way. For example, the temperature of a room is a measurable phenomenon because we can measure it using a thermometer. The number of people in a room is also a measurable phenomenon because we can count them.

When we quantify a phenomenon, we perform data collection. Data collection is the process of gathering data on targeted phenomenon in an established systematic way. Systematic means that we have a plan to collect the data and we understand the consequences of the plan, including the sampling bias. Sampling bias is the influence that the method of collecting the data has on the conclusions we can draw from them. Once we have collected the data, we need to store them. Data storage is the process of storing data in a computer.

Slide 3.7: Phenomena

- Phenomena are the source we use to understand the world around us.
- We use our senses to perceive phenomena.
- We use our knowledge and reasoning to make sense of them.
- Computational methods can be used to implement knowledge and reasoning.
- Phenomena are the source of data.
- We need to understand the nature of the phenomena we are studying.
- Incorrectly perception of the phenomena may lead to incorrect data collection, which may lead to incorrect conclusions.

To perform those tasks, we need to understand the nature of data. Data are any piece of information that can be digitally stored. Data can be stored in many different formats. For example, we can store data in a spreadsheet, in a database, or in a text file. We can also store data in many different types. For example, we can store data as numbers, strings, or dates.

In data science, studying data types is important because they need to correctly reflect the nature of the source phenomenon and be compatible with the computational methods we are using. Data types also restrict the operations we can perform on the data.

The foundation and tools to understand data types come from computer science. Among the subfields, I highlight:

- Algorithms and data structures: the study of data types and the computational methods to manipulate them.
- Databases: the study of storing and retrieving data.

Slide 3.8: Measurable phenomena

- Measurable phenomena are those that we can quantify in some way.
- Data collection is the process of gathering data on targeted phenomenon in an established systematic way.
- The collection/sampling bias influence our conclusions.
- Data storage is the process of storing data in a computer.
- Data are any piece of information that can be digitally stored.
- Data can be stored in many different formats.
- Data can be stored in many different types.
- Data types need to correctly reflect the nature of the source phenomenon and be compatible with the computational methods we are using.
- Algorithms, data structures, and databases are important subfields of computer science when studying data collection, data storage, and data types.

3.1.3 Knowledge extraction

Once we have collected and stored the data, we need to extract knowledge from them. In data science, we use computational methods to extract knowledge from data. These computational methods may come from many different fields. In particular, I highlight:

- Statistics: the study of data collection, organization, analysis, interpretation, and presentation.
- Machine learning: the study of computational methods that can automatically learn from data.
- Artificial intelligence: the study of computational methods that can mimic human intelligence.

Also, many other fields contribute to the development of domain-specific computational methods to extract knowledge from data. For example, in the field of biology, we have bioinformatics, which is the study of computational methods to analyze biological data. Earth sciences have geoinformatics, which is the study of computational methods to analyze geographical data. And so on.

Each method has its own assumptions and limitations. Thus, we need to understand the nature of the methods we are using. In particular, we need to understand the expected input and output of them. Whenever the available data do not match the requirements of the method, we may perform data handling[2].

Data handling mainly includes data cleaning, data transformation, and data integration. Data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate pieces of data. Data transformation is the process of converting data from one format or type to another. Data integration is the process of combining data from different sources into a single, unified view.

3.2 Structured data

As one expects, when we measure a phenomenon, the resulting data come in many different formats. For example, we can measure the temperature of a room using a thermometer. The resulting data is a number. We can assess English proficiency using an essay test. The resulting data is a text. We can register relationships between proteins and their functions. The resulting data is a graph.

Thus, it is important to understand the nature of the data we are working with.

The most common data format is the *structured data*. Structured data are data that are organized in a tabular format. Each row in the table represents a single observation and each column represents a variable that describes the observation.

We restrict the kind of information we store in each cell, i.e. the data type of each measurement. Each column has a data type. The data type restrict the operations we can perform on the data. For example, we can perform arithmetic operations on numbers, but not on text.

[2]It is important to highlight that it is expected that some of the methods assumptions are not fully met. These methods are usually robust enough to extract valuable knowledge even when data contain imperfections, errors and noise. However, it is still useful to perform data handling to adjust data as much as possible.

Slide 3.9: Knowledge extraction

- We use computational methods to extract knowledge from data.
- Statistics, machine learning, and artificial intelligence are important sciences when studying knowledge extraction.
- Computational methods always have their own assumptions and limitations.
- Data handling is the process of adjusting data to the requirements of the computational methods, which includes:
  - Data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate pieces of data.
  - Data transformation is the process of converting data from one format or type to another.
  - Data integration is the process of combining data from different sources into a single, unified view.

The most common classification of data types is Stevens's types: nominal, ordinal, interval, and ratio. Nominal data are data that can be classified into categories. Ordinal data are data that can be classified into categories and ordered. Interval data are data that can be classified into categories, ordered, and measured in fixed units. Ratio data are data that can be classified into categories, ordered, measured in fixed units, and have a true zero. In practice, they differ on the logical and arithmetic operations we can perform on them.

Table 3.1: Stevens's types.

Data type	**Operations**
Nominal	$=$
Ordinal	$=, <$
Interval	$=, <, +, -$
Ratio	$=, <, +, -, \times, \div$

However, Stevens's types do not exhaust all possibilities for data types. For example, probabilities are bounded at both ends, and thus do not tolerate arbitrary scale shifts. Velleman and Wilkinson (1993) provide interesting insights about data types. Although I do not agree with all his points, I think it is a good reading. In particular, I agree with his criticism of statements that data types are evident from the data independent of the questions asked. The same data can be interpreted in different ways depending on the context and the goals of the analysis.

However, I do not agree with the idea that good data analysis does not assume data types. I think that data scientists should be aware of the data types they are working with and how they affect the analysis. With no bias, there is no learning. There is no such a thing as a "bias-free" analysis, the amount of possible combinations of assumptions easily grows out of control. The data scientist must take responsibility for the consequences of their assumptions. Good assumptions and hypothesis are a key part of the data science methodology.

Slide 3.10: Structured data

- Structured data are data that are organized in a tabular format.
- Each row in the table represents a single observation.
- Each column represents a variable that describes the observation.
- Each column has a data type.
- The data type restrict the operations we can perform on the data.
- The most common classification of data types is Stevens's types: nominal, ordinal, interval, and ratio.
- Stevens's types do not exhaust all possibilities for data types.
- Data scientists should be aware of the data types they are working with and how they affect the analysis.
- Inevitably, data scientists make assumptions about the data types.

When we work with structured data, two concepts are very important: database normalization and tidy data. Database normalization is mainly focused on the data storage. Tidy data is mainly focused on the requirements of data for analysis. Both concepts have their mathematical foundations and tools for data handling.

3.2.1 Database normalization

Database normalization is the process of organizing the columns and tables of a relational database to minimize data redundancy and improve data integrity.

Normal form is a state of a database that is free of certain types of data redundancy. Before studying normal forms, we need to understand basic concepts in the database theory and the basic operations in relational algebra.

Relational database theory

Projection The projection of a relation is the operation that returns a relation with only the columns specified in the projection. For example, if we have a relation $X[A, B, C]$ and we perform the projection $\pi_{A,C}(X)$, we get a relation with only the columns A and C, i.e. $X[A, C]$.

Join The (natural) join of two relations is the operation that returns a relation with the columns of both relations. For example, if we have two relations $S[U \cup V]$ and $T[U \cup W]$, where U is the common set of attributes, join $S \bowtie T$ of S and T is the relation with tuples (u, v, w) such that $(u, v) \in S$ and $(u, w) \in T$. The generalized join is built up out of binary joins: $\bowtie \{R_1, R_2, \ldots, R_n\} = R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n$. Since the join operation is associative and commutative, we can parenthesize however we want.

Functional dependency A functional dependency is a constraint between two sets of attributes in a relation. It is a statement that if two tuples agree on certain attributes, then they must agree on another attribute. Specifically, the *functional dependency* $U \to V$ holds in R if and only if for every pair of tuples t_1 and t_2 in R such that $t_1[U] = t_2[U]$, it is also true that $t_1[V] = t_2[V]$.

Multi-valued dependency A multi-valued dependency is a constraint between two sets of attributes in a relation. It is a statement that if two

tuples agree on certain attributes, then they must agree on another set of attributes. Specifically, the *multi-valued dependency* $U \twoheadrightarrow V$ holds in R if and only if $R = R[UV] \bowtie R[UW]$, where W are the remaining attributes.

Join dependency A join dependency is a constraint between subsets of attributes (not necessarily disjoint) in a relation. R obeys the join dependency $*\{X_1, X_2, \ldots, X_n\}$ if $R = \bowtie \{R[X_1], R[X_2], \ldots, R[X_n]\}$.

Normal forms

First normal form (1NF) A relation is in 1NF if and only if all attributes are atomic. An attribute is atomic if it is not a set of attributes. For example, the relation $R[A, B, C]$ is in 1NF if and only if A, B, and C are atomic.

Second normal form (2NF) A relation is in 2NF if and only if it is in 1NF and every non-prime attribute is fully functionally dependent on the primary key. A non-prime attribute is an attribute that is not part of the primary key. A primary key is a set of attributes that uniquely identifies a tuple. A non-prime attribute is fully functionally dependent on the primary key if it is functionally dependent on the primary key and not on any subset of the primary key. For example, the relation $R[U \cup V]$ is in 2NF if and only if $U \to X$, $\forall X \in V$ and there is no $W \subset U$ such that $W \to X$, $\forall X \in V$.

Third normal form (3NF) A relation is in 3NF if and only if it is in 2NF and every non-prime attribute is non-transitively dependent on the primary key. A non-prime attribute is non-transitively dependent on the primary key if it is not functionally dependent on another non-prime attribute. For example, the relation $R[U \cup V]$ is in 3NF if and only if U is the primary key and there is no $X \in V$ such that $X \to Y$, $\forall Y \in V$.

Boyce-Codd normal form (BCNF) A relation R with attributes X is in BCNF if and only if it is in 2NF and for each nontrivial functional dependency $U \to V$ in R, the functional dependency $U \to X$ is in R. In other words, a relation is in BCNF if and only if every functional dependency is the result of keys.

Fourth normal form (4NF) A relation R with attributes X is in 4NF if and only if it is in 2NF and for each nontrivial multi-valued dependency $U \twoheadrightarrow V$ in R, the functional dependency $U \rightarrow X$ is in R. In other words, a relation is in 4NF if and only if every multi-valued dependency is the result of keys.

Projection join normal form (PJNF) A relation R with attributes X is in PJNF[3] if and only if it is in 2NF and the set of key dependencies[4] of R impllies each join dependency of R. The PJNF guarantees that the table cannot be decomposed without losing information (except by decompositions based on keys).

Note that the ideia behind the definition of BCNF and 4NF are slightly different from the PJNF. In fact, if we consider that for each key dependency implies a join dependency, the relation is in the so-called overstrong projection-join normal form[5]. Such a level of normalization does not improve data storage or eliminate inconsistencies. In practice, it means that if a relation is in PJNF, careless joins — i.e. those that violate a join dependency — produce inconsistent results.

Example 1 Consider the 2NF relation $R[A, B, C, D]$ with the functional dependencies $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$. The relation is not in 3NF because C is transitively dependent on A. To normalize it, we can decompose it into the relations $R_1[A, B, C]$ and $R_2[C, D]$. Now, R_2 is in 3NF and R_1 is in 2NF, but not in 3NF. We can decompose R_1 into the relations $R_3[A, B]$ and $R_4[B, C]$. The original relation can be reconstructed by $\bowtie \{R_2, R_3, R_4\}$.

Example 2 Consider the 2NF relation $R[ABC]$[6] such that the primary key is the composite of A, B, and C. The relation is thus in the 4NF, as no column is a determinant of another column. Suppose, however, the following constraint: if (a, b, c'), (a, b', c), and (a', b, c) are in R, then (a, b, c) is also in R. This can be illustrated if we consider A as a agent, B as a product, and C as a company. If an agent a represents companies c

[3] Also known as fifth normal form (5NF).

[4] Key dependency is a functional dependency in the form $K \rightarrow X$.

[5] R. Fagin (1979). "Normal forms and relational database operators". In: *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*. SIGMOD '79. Boston, Massachusetts: Association for Computing Machinery, pp. 153–160. ISBN: 089791001X. DOI: 10.1145/582095.582120. URL: https://doi.org/10.1145/582095.582120.

[6] Here we abreviate A, B, C as ABC.

and c', and product b is in his portfolio, then assuming both companies make b, a must offer b from both companies.

The relation is not in PJNF, as the join dependency $*\{AB, AC, BC\}$ is not implied by the primary key. (In fact, the only functional dependency is the trivial $ABC \rightarrow ABC$.) In this case, to avoid redundancies and inconsistencies, we must split the relation into the relations $R_1[AB]$, $R_2[AC]$, and $R_3[BC]$.

It is iteresting to notice that in this case, the relation $R_1 \bowtie R_2$ might contain tuples that do not make sense in the context of the original relation. For example, if R_1 contains (a, b) and R_2 contains (a, c'), the join contains (a, b, c'), which might not be a valid tuple in the original relation if (b, c') is not in R_3. *This is very important to notice, as it is a common mistake to assume that the join of the decomposed relations always contains valid tuples.*

Example 3 Consider the 2NF relation $R[A, B, C, D, E]$ with the functional dependencies $A \rightarrow D$, $AB \rightarrow C$, and $B \rightarrow E$. To make it PJNF, we can decompose it into the relations $R_1[A, D]$, $R_2[A, B, C]$, and $R_3[B, E]$. The original relation can be reconstructed by $\bowtie \{R_1, R_2, R_3\}$. However, unlike the previous example, the join of the decomposed relations always contains valid tuples — excluding degenerate joins, where there are no common attributes. The reason is that all join dependencies implied by the key dependencies are trivial when reduced[7].

Slide 3.11: Database normalization

- Minimizes data redundancy.
- Improves data integrity.
- Semantics is expressed in terms of dependencies (functional, multivalued, join), which is usually not clear.
- Appropriate to store data.

[7] A proof in under development based on M. W. Vincent (1997). "A corrected 5NF definition for relational database design". In: *Theoretical Computer Science* 185.2. Theoretical Computer Science in Australia and New Zealand, pp. 379–391. ISSN: 0304-3975. DOI: https://doi.org/10.1016/S0304-3975(97)00050-9. URL: https://www.sciencedirect.com/science/article/pii/S0304397597000509.

3.2.2 Tidy data

It is estimated that 80% of the time spent on data analysis is spent on data preparation. Usually, the same process is repeated many times in different datasets. The ideia is that organized data carries the meaning of the data, reducing the time spent on handling the data to get it into the right format for analysis.

Tidy data is a data format that provides a standardized way to organize data values within a dataset. The main advantage of tidy data is that it provides clear semantics with focus on only one view of the data.

Many data formats might be ideal for particular tasks, such as raw data, dense tensors, or normalized databases. However, most of the statitiscal and machine learning methods require a particular data format. Tidy data is a data format that is appropriate to those tasks.

Wickham's thoughts on tidy data

Like families, tidy datasets are all alike but every messy dataset is messy in its own way.

In an unrestricted table, the meaning of rows and columns are not fixed. In a tidy table, the meaning of rows and columns are fixed.

It is based on the idea that a dataset is a collection of values, where:

- Each *value* belongs to a variable and an observation.
- Each *variable*, represented by a column, contains all values that measure the same attribute across (observational) units.
- Each *observation*, represented by a row, contains all values measured on the same unit across attributes.
- *Attributes* are the characteristics of the units, e.g. height, temperature, duration.
- *Observational units* are the individual entities being measured, e.g. a person, a day, an experiment.

Table 3.2 summarizes the main concepts.

If we follow this structure, the meaning of data is implicit in the table itself. However, it is not always trivial to organize data in a tidy format. Usually, we have more than one level of observational units, each one

Table 3.2: Tidy data concepts.

Concept	Structure	Contains	Across
Variable	Column	Same attribute	Units
Observation	Row	Same unit	Attributes

represented by a table. Moreover, there might exist more than one way to define what are the observational units in a dataset.

To organize data in a tidy format, one can consider that:

- Attributes are functionally related among themselves — e.g. Z is a linear combination of X and Y, or X and Y are correlated, or $P(X, Y)$ follows some joint distribution.
- Units can be grouped or compared — e.g. person A is taller than person B, or the temperature in day 1 is higher than in day 2.

A particular point that tidy data do not address is that values in a column might not be in the same scale or unit of measurement[8]. For example, a column might contain the temperature in an experiment, and another column might contain the unit of measurement that was used to measure the temperature. This is a common problem in databases, and it must be addressed for machine learning and statistical methods to work properly.

Note that the order of the rows and columns is not important. However, it might be convenient to sort data in a particular way to facilitate the understanding. For instance, one usually expects that the first columns are *fixed variables*[9], i.e. variables that are not the result of a measurement, and the last columns are *measured variables*. Also, arranging rows by some variable might highlight some pattern in the data.

Usually, columns are named — the collection of all column names is called the header, while rows are numerically indexed.

Common messy datasets

Wickham (2014) lists some common problems with messy datasets and how to tidy them[10]. The problems are summarized below.

[8] Attention: observational unit is not unit of measurement.

[9] Closely related (and potentially the same as) key in database theory.

[10] Operations are presented in chapter 6.

Headers are values, not variable names For example, consider table 3.3. This table is not tidy because the column headers are values, not variable names. This format is frequently used in presentations since it is more compact. It is also useful to perform matrix operations. However, it is not appropriate for general analysis.

Table 3.3: Messy table, from Pew Forum dataset, where headers are values, not variable names.

Religion	<$10k	$10-20k	$20-30k	...
Agnostic	27	34	60	...
Atheist	12	27	37	...
Buddhist	27	21	30	...
...	...	...	...	...

To make it tidy, we can transform it into the table 3.4 by explicitly introducing variables *Income* and *Frequency*. Note that the table is now longer, but it is also narrower. This is a common pattern when fixing this kind of issue. The table is now tidy because the column headers are variable names, not values.

Table 3.4: Tidy version of table 3.3 where values are correctly moved.

Religion	Income	Frequency
Agnostic	<$10k	27
Agnostic	$10-20k	34
Agnostic	$20-30k	60
...	...	...
Atheist	<$10k	12
Atheist	$10-20k	27
Atheist	$20-30k	37
...	...	...

Multiple variables are stored in one column For example, consider the table 3.5. This table is not tidy because the column — interestly

called *column* —, contains multiple variables. This format is frequent, and sometimes the column name contains the names of the variables. Sometimes it is very hard to separate the variables.

Table 3.5: Messy table, from TB dataset, where multiple variables are stored in one column.

country	year	column	cases	...
AD	2000	m014	0	...
AD	2000	m1524	0	...
AD	2000	m2534	1	...
AD	2000	m3544	0	...
...	...	...	...	

To make it tidy, we can transform it into the table 3.6. Two columns are created to contain the variables *Sex* and *Age*, and the old column is removed. The table keeps the same number of rows, but it is now wider. This is a common pattern when fixing this kind of issue. The new version usually fixes the issue of correctly calculating ratios and frequency.

Table 3.6: Tidy version of table 3.5 where values are correctly moved.

country	year	sex	age	cases	...
AD	2000	m	0–14	0	...
AD	2000	m	15–24	0	...
AD	2000	m	25–34	1	...
AD	2000	m	35–44	0	...
...	...	...	...	...	

Variables are stored in both rows and columns For example, consider the table 3.7. This is the most complicated case of messy data. Usually, one of the columns contains the names of the variables, in this case the column *element*.

To fix this issue, we must first decide which column contains the names of the variables. Then, we must lengthen the table in function

Table 3.7: Messy table, adapted from airquality dataset, where variables are stored in both rows and columns.

id	year	month	element	d1	d2	...	d31
MX17004	2010	1	tmax		24	...	27
MX17004	2010	1	tmin	14		...	
MX17004	2010	2	tmax	27	24	...	27
MX17004	2010	2	tmin	14		...	13
...	...	...	...	...	...	...	...

of the variables (and potentially their names), as seen in table 3.8. Aftwards, we widen the table in function of their names. Finally, we remove implicit information, as seen in table 3.9.

Table 3.8: Partial solution to tidy table 3.7. Note that the table is now longer.

id	date	element	value
MX17004	2010-01-01	tmax	
MX17004	2010-01-01	tmin	14
MX17004	2010-01-02	tmax	24
MX17004	2010-01-02	tmin	
...	...	...	...

Table 3.9: Tidy version of table 3.7 where values are correctly moved.

id	date	tmin	tmax
MX17004	2010-01-01	14	
MX17004	2010-01-02		24
...	...	...	...

Multiple types of observational units are stored in the same table For example, consider the table 3.10. It is very common during data collection that many observational units are registered in the same table.

Table 3.10: Messy table, adapted from billboard dataset, where multiple types of observational units are stored in the same table.

year	artist	track	date	rank
2000	2 Pac	Baby Don't Cry	2000-02-26	87
2000	2 Pac	Baby Don't Cry	2000-03-04	82
2000	2 Pac	Baby Don't Cry	2000-03-11	72
2000	2 Pac	Baby Don't Cry	2000-03-18	77
...	...	...	...	...
2000	2Ge+her	The Hardest...	2000-09-02	91
2000	2Ge+her	The Hardest...	2000-09-09	87
2000	2Ge+her	The Hardest...	2000-09-16	92
...	...	...	...	...

To fix this issue, we must each observation unit must be moved to a different table. Sometimes, it is useful to create unique identifiers for each observation. The separation avoids several types of potential inconsistencies. However, take into account that during data analysis, it is possible that we have to denormalize them. The two resulting tables are shown in table 3.11 and table 3.12.

Table 3.11: Tidy version of table 3.10 containing the observational unit *track*.

track id	artist	track
1	2 Pac	Baby Don't Cry
2	2Ge+her	The Hardest Part Of Breaking Up
...	...	...

A single observational unit is stored in multiple tables For example, consider tables 3.13 and 3.14. It is very common during data collection that a single observational unit is stored in multiple tables.

Table 3.12: Tidy version of table 3.10 containing the observational unit *rank of the track in certain week*.

track id	date	rank
1	2000-02-26	87
1	2000-03-04	82
1	2000-03-11	72
1	2000-03-18	77
...	...	...
2	2000-09-02	91
2	2000-09-09	87
2	2000-09-16	92
...	...	...

Usually, the table (or file) itself represents the value of a variable. When columns are compatible, it is straightforward to combine the tables.

Table 3.13: Messy tables, adapted from nycflights13 dataset, where a single observational unit is stored in multiple tables. Assume that the origin filename is called `2013.csv`.

month	day	time	...
1	1	517	...
1	1	533	...
1	1	542	...
1	1	544	...
...	...	...	...

To fix this issue, we must first make the columns compatible. Then, we can combine the tables adding a new column that identifies the origin of the data. The resulting table is shown in table 3.15.

3.2.3 Bridging normalization, tidyness, and data theory

First and foremost, both concepts, normalization and tidy data, are not in conflict.

Table 3.14: Messy tables, adapted from nycflights13 dataset, where a single observational unit is stored in multiple tables. Assume that the origin filename is called `2014.csv`.

month	day	time	...
1	1	830	...
1	1	850	...
1	1	923	...
1	1	1004	...
...	...	...	...

Table 3.15: Tidy data where tables 3.13 and 3.14 are combined.

year	month	day	time	...
2013	1	1	517	...
2013	1	1	533	...
2013	1	1	542	...
2013	1	1	544	...
...	...	...	...	...
2014	1	1	830	...
2014	1	1	850	...
2014	1	1	923	...
2014	1	1	1004	...
...	...	...	...	...

In data normalization, given a set of functional, multivalued and join dependencies, there exists a normal form that is free of redundancy. In tidy data, Wickham, Çetinkaya-Rundel, and Grolemund also state that there is only one way to organize the given data.

Wickham (2014) states that tidy data is 3NF. However, he does not provide a formal proof. Since tidy data focuses on data analysis and not on data storage, I argue that there is more than one way to organize the data in a tidy format. It actually depends on what you define as the observational unit.

Consider the following example. We want to study the *phenomenon* temperature in a certain city. We fix three sensors in different locations

to measure the temperature. We collect data three times a day. If we consider as the observational unit the event of measuring the temperature, we can organize the data in a tidy format as shown in table 3.16.

Table 3.16: Tidy data where the observational unit is the event of measuring the temperature.

date	time	sensor	temperature
2023-01-01	00:00	1	20
2023-01-01	00:00	2	21
2023-01-01	00:00	3	22
2023-01-01	08:00	1	21
2023-01-01	08:00	2	22
2023-01-01	08:00	3	23
...	...	...	...

However, since the sensors are fixed, we can consider the observational unit as the *temperature at some time*. In this case, we can organize the data in a tidy format as shown in table 3.17.

Table 3.17: Tidy data where the observational unit is the temperature at some time.

date	time	temp. 1	temp. 2	temp. 3
2023-01-01	00:00	20	21	22
2023-01-01	08:00	21	22	23
...	...	...	...	...

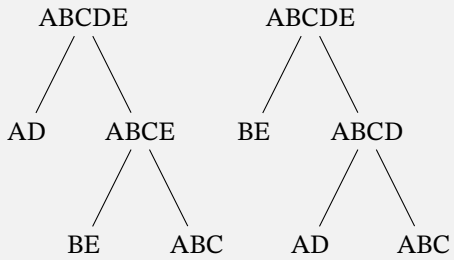
In both cases, one can argue that the data is also normalized. In the first case, the primary key is the composite of the columns *date*, *time*, and *sensor*. In the second case, the primary key is the composite of the columns *date* and *time*.

One can state that the first form is more appropriate, since it is flexible to add more sensors. However, the second form is very natural for machine learning and statistical methods. Given the definition of tidy data, I believe both forms are correct.

Another very interesting conjecture is whether we can formalize the eventual *change of observational unit* in terms of the order that joins and grouping operations are performed.

Example Consider the following example: the relation $R[A, B, C, D, E]$ and the functional dependencies $A \to D$, $B \to E$, and $AB \to C$. The relation can be normalized up to 3NF by following one of the decomposition trees shown in fig. 3.2. Every decomposition tree must take into account that the join of the projections are lossless and dependency preserving.

Figure 3.2: Decomposition trees for the relation $R[ABCDE]$ and the functional dependencies $A \to D$, $B \to E$, and $AB \to C$ to reach 3NF.



Note that the decomposition that splits first $R[ABC]$ is not valid, since the resulting relation $R[AB]$ is not a consequences of a functional dependency, see fig. 3.3.

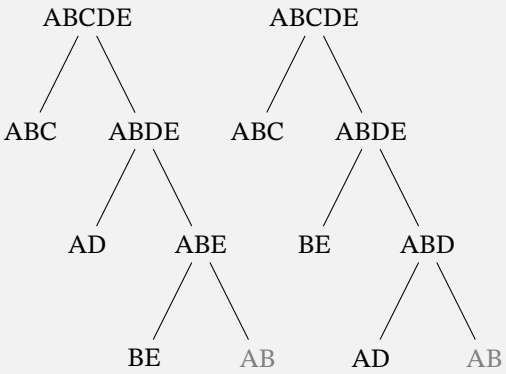
In this kind of relation schema, we have a set of key attributes, here $\mathcal{K} = AB$, and a set of non-prime attributes, here $\mathcal{N} = CDE$. Note that the case $\mathcal{K} \cap \mathcal{N} = \varnothing$ is the simplest we can have.

Observe, however, that transitive dependencies[11] and complex join dependencies restrict even further the joins we are allowed to perform. Further formalization and study is under progress.

Now, consider a very common case: in our dataset, keys are unknown. Let A be a student id, B be the course id, D be the student age, E be the course load, and C be the student grade at the course. If only CDE is known, the table $R[CDE]$ is already tidy — and the observational unit

[11] Actually, when an attribute is both key and non-prime, some joins may generate invalid tables.

Figure 3.3: Invalid decomposition trees for the relation $R[ABCDE]$.



We consider the functional dependencies $A \rightarrow D$, $B \rightarrow E$, and $AB \rightarrow C$. Note that $R[AB]$ is not a consequence of a functional dependency.

is the enrollment — once there is no key to perform any kind of normalization. This happens in many cases where privacy is a concern.

But we can also consider that the observational unit is the student. In this case, we must perform joins traversing the leftmost decomposition tree in fig. 3.2 from bottom to top. After each join, a summarization operation is performed on the relation considering the student as the observational unit, i.e. over attribute A. The first join results in relation $R[ABCE]$ and the summarization operation results in a new relation $R[AFG]$ where F is the average grade and G is the total course load taken by the student. They all calculated based on the rows that are grouped in function of A. It is important to notice that, after the summarization operation, all observations must contain a different value of A. The second join results in relation $R[ADFG] = R[AD] \bowtie R[AFG]$. This relation has functional dependency $A \rightarrow DFG$, and it is in 3NF (which is also tidy).

Unfortunately, it is not trivial to calculate all possible decomposition trees for a given dataset. Further formalization and study is under progress.

3.2.4 Data semantics and interpretation

In the rest of the book, we focus on a statistical view of the data. Besides the functional dependencies, we also consider the statistical dependencies of data. For instance, attributes A and B might not be functionally dependent, but they might exits unknown $P(A, B)$ that we can estimate from the data. Each observed value of a key can represent a instance of a random variable, and the other attributes can represent measured attributes or calculated properties.

For data analysis, it is very important to understand the relationships between the observations. For example, we might want to know if the observations are independent, if they are identically distributed, or if there is a known selection bias. We might also want to know if the observations are dependent on time, and if there are hidden variables that affect the observations.

Following wrong assumptions can lead to wrong conclusions. For example, if we assume that the observations are independent, but they are not, we might underestimate the variance of the estimators.

Although we not focus on time series, we must consider the temporal dependence of the observations. For example, we might want to know how the observation x_t is affected by x_{t-1}, x_{t-2}, and so on. We might also want to know if Markov property holds, and if there is periodicity and seasonality in the data.

For the sake of the scope of this book, we suggest that any prediction on temporal data should be done in the state space, where it is safer to assume that observations are independent and identically distributed. This is a common practice in reinforcement learning and deep learning. Takens' theorem[12] allows you to reconstruct the state space of a dynamical system using time-delay embedding. Given a single observed time series, you can create a multidimensional representation of the underlying dynamical system by embedding the time series in a higher-dimensional space. This embedding can reveal the underlying dynamics and structure of the system.

3.3 Unstructured data

Unstructured data are data that do not have a predefined data model or are not organized in a predefined manner. For example, text, images,

[12] F. Takens (2006). "Detecting strange attractors in turbulence". In: *Dynamical Systems and Turbulence, Warwick 1980: proceedings of a symposium held at the University of Warwick 1979/80*. Springer, pp. 366–381.

and videos are unstructured data.

Every unstructured dataset can be converted into a structured dataset. However, the conversion process is not always straightforward nor lossless. For example, we can convert a text into a structured dataset by counting the number of occurrences of each word. However, we lose the order of the words in the text.

The study of unstructured data is, for the moment, out of the scope of this book.

4

Data science project

Figured I could throw myself a pity party or go back to school and learn the computers.

— Don Carlton, *Monsters University (2013)*

First of all, a data science project is a software project. The difference between a data science software and a traditional software is that some components of the former is constructed from data. This means that part of the solution cannot be designed from the knowledge of the domain expert, but must be learned from the data. (Alternatively, the cost of designing the solution is too high, and it is more efficient to learn it from the data.)

One good example of a data science project is a spam filter. The spam filter is a software that classifies emails into two categories: spam and non-spam. The software is trained using a set of emails that are already classified as spam or non-spam. The software is then used to classify new emails. The software is a data science software because the classification algorithm is learned from the data, i.e. the filters are not designed "by hand".

4.1 CRISP-DM

CRISP-DM[1] is a methodology for data mining projects. It is an acronym for Cross Industry Standard Process for Data Mining. It is a methodol-

[1] Official guide available at https://www.ibm.com/docs/it/SS3RA7_18.3.0/pdf/ModelerCRISPDM.pdf.

ogy that was developed in the 1990s by IBM, and it is still widely used today.

CRISP-DM is a cyclic process. The process is composed of six phases:

1. Business understanding: this is the phase where the project objectives are defined. The objectives must be defined in a way that is measurable. The phase also includes the definition of the project plan.

2. Data understanding: this is the phase where the data is collected and explored. The data is collected from the data sources, and it is explored to understand its characteristics. The phase also includes the definition of the data quality requirements.

3. Data preparation: this is the phase where the data is prepared for the modeling phase. The data is cleaned, transformed, and aggregated. The phase also includes the definition of the modeling requirements.

4. Modeling: this is the phase where the model is trained and validated. The model is trained using the prepared data, and it is validated using the validation data. The phase also includes the definition of the evaluation requirements.

5. Evaluation: this is the phase where the model is evaluated. The model is evaluated using the evaluation data. The phase also includes the definition of the deployment requirements.

6. Deployment: this is the phase where the model is deployed. The model is deployed using the deployment requirements. The phase also includes the definition of the monitoring requirements.

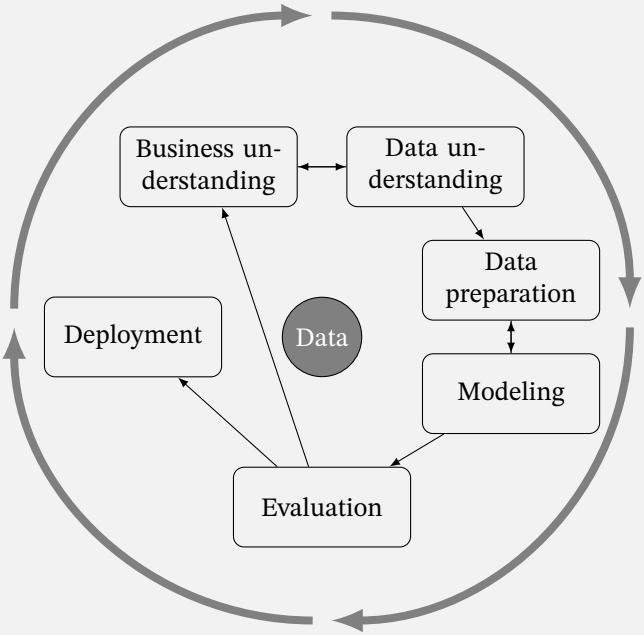
Figure 4.1 shows a diagram of the CRISP-DM process. Note that the process is cyclic and completly focused on the data. The process do not address the software development aspects of the project.

The CRISP-DM methodology is a good starting point for data science projects. However, it does not mean that should be followed strictly. The process is cyclic and flexible, and adaptations are possible at any stage of the process.

4.2 ZN approach

Zumel and Mount (2019) also propose a methodology for data science projects — which we call the ZN approach. Besides describing each step

Figure 4.1: Diagram of the CRISP-DM process.



Each block represents a phase of the CRISP-DM process. Data is the central element of the process. Arrows represent the transitions between the phases.

in a data science project, they further address the roles of each individual involved in the project. They state that data science projects are always collaborative, as they require domain expertise, data expertise, and software expertise. The requirements are dynamic, and the project has many exploratory phases. Usually, projects based on data are urgent, and they must be completed in a short time — not only due to the business requirements, but also because the data changes over time. The authors state that agile methodologies are suitable (and necessary) for data science projects.

4.2.1 Roles of the ZN approach

In their approach, five roles are defined.

Project sponsor It is the main stakeholder of the project, the one that needs the results of the project. He represents the business interests and champions the project. The project is considered successful if the sponsor is satisfied. Note that, ideally, the sponsor can not be the data scientist, but someone that is not involved in the development of the project. However, he needs to be able to express *quantitatively* the business goals and participate actively in the project.

Client The client is the domain expert. He represents the end users' interests. In a small project, he is usually the sponsor. He translates the daily activities of the business into the technical requirements of the software.

Data scientist The data scientist is the one that sets and executes the analytic strategy. He is the one that communicates with the sponsor and the client, effectively connecting all the roles. In small projects, he can also act as the developer of the software. However, in large projects, he is usually the project manager. Although it is not required to be a domain expert, the data scientist must be able to understand the domain of the problem. He must be able to understand the business goals and the client's requirements. Most importantly, he must be able to define and to solve the right tasks.

Data architect The data architect is the one that manages data and data storage. He usually is involved in more than one project, so it is not an active participant. He that receives instructions to adapt the data storage and means to collect data.

Operations The operations role is the one that manages infrastructure and deploys final project results. He is responsible to define requirements such as response time, programming language, and the infrastructure to run the software.

4.2.2 Processes of the ZN approach

Zumel and Mount's model is similar to CRISP-DM, but emphasizes that back-and-forth is possible at any stage of the process. Figure 4.2 shows a diagram of the process. The phases are:

- Define the goal: what problem are we trying to solve?
- Collect and manage data: what information do we need?
- Build the model: find patterns in the data that may solve the problem.
- Evaluate the model: is the model good enough to solve the problem?
- Present results and document: establish that we can solve the problem and how we did it. (This step is a differentiator from CRISP-DM. In ZN approach, result presentation is essential; data scientists must be able to communicate their results effectively to the client/sponsor.)
- Deploy the model: make the model available to the end users.

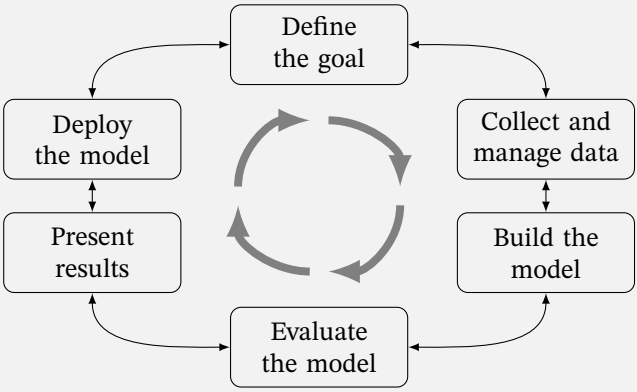
4.3 Agile methodology

Agile is a methodology for software development. It is an alternative to the waterfall methodology. The waterfall methodology is a sequential design where each phase must be completed before the next phase can begin.

The four values of agile manifesto are:

- Individuals and interactions over processes and tools;
- Working software over comprehensive documentation;
- Customer collaboration over contract negotiation;
- Responding to change over following a plan.

Figure 4.2: Diagram of the data science process proposed by Zumel and Mount (2019).



Each block represents a phase of the data science process. The emphasis is on the cyclic nature of the process. Arrows represent the transitions between the phases, that can be back-and-forth.

4.4 SCRUM framework

SCRUM is an agile framework for software development. It is a process framework for managing complex projects. It is a lightweight, which means that it provides just enough guidance to be effective.

Many consider that SCRUM is not adequate for data science projects. The main reason is that SCRUM is designed for projects where the requirements are known in advance. Also, that data science projects have exploratory phases, which are not well supported by SCRUM.

I argue that this view is wrong. SCRUM is a framework, and it is designed to be adapted to the needs of the project. SCRUM is not a rigid process. In the following, I propose an extension to SCRUM that makes it suitable for data science projects.

(In real-world, most developers do not have hacking-level skills. They are not autonomous enough to work without a plan. This is especially true for "data scientists," who are often not even developers. SCRUM is a good compromise between the need for autonomy and the need for a detailed plan. Project methodology is needed to ensure that the project is completed in time and within budget.)

4.5 Our approach

The previously mentioned methodologies lack the focus on the software development aspects of the data science project. For instance, CRISP-DM defines the stages only of the data mining process, i.e. it does not explicitly address user interface or data collection. Zumel and Mount's approach addresses data collection and presentation of results, but delegates the software development to the operations role, barely mentioning it. SCRUM is a good framework for software development, but it is not designed for data science projects. It lacks the exploratory phases of data science projects.

Thus, we propose an extension to SCRUM that makes it suitable for data science projects. The extension is based on the following observations:

- Data science projects have exploratory phases;
- Data itself is a component of the solution;
- The solution is usually modularized, parts of it are constructed from data while the other parts are constructed like traditional software;
- The solution is usually deployed as a service, that must be maintained and monitored.

Moreover, we add two other values besides the agile manifesto values. They are:

- Model confidence/understanding over model performance;
- Code version control over interactive environments.

The first value is based on the observation that the model performance is not the most important aspect of the model. The most important aspect is the being sure that the model behaves as expected (and sometimes why it behaves as expected). It is not uncommon to find models that seems to perform well during evaluation steps[2], but that are not suitable for production.

The second value is based on the observation that interactive environments are not suitable for the development of the model search code, for instance. Interactive environments auxiliate the exploratory phases,

[2]Of course, when evaluation is not performed correctly.

but the final version of the code must be version controlled. Often, we hear stories that models cannot be reproduced because the code that generated them are not runnable anymore. This is a serious problem, and it is not acceptable for maintaining a software solution.

These observations and values are the basis of our approach. The roles and principles of our approach are described in the following sections.

4.5.1 The roles of our approach

Combine SCRUM roles with the roles defined by Zumel and Mount (2019).

Table 4.1: Roles of our approach.

Our approach	SCRUM	ZM
Sponsor	Product owner	Project sponsor
Client	Stakeholder	Client
Data scientist	Scrum master	Data scientist
Dev Team		Data architect/operations

The roles of SCRUM are associated with the roles defined by Zumel and Mount (2019). In our approach, the data scientist leads the development team and interacts with the sponsor and the client. The development team includes people with both database and software engineering expertise.

4.5.2 The principles of our approach

1. Modularize the solution. Usually, in four main modules: frontend, backend, dataset, and model search. The frontend is the user interface. The backend is the server-side code. The dataset is the data that is used to train the model. The model search is the code that searches for the best model.

2. Version control everything. This includes the code, the data, and the documentation. The most used tool for code version control

is Git. For datasets, extensions to Git exist, such as DVC[3]. One important aspect is to version control the model search code. Interactive environments such as Jupyter notebooks are not suitable for this purpose. They can be used to draft the code, but the final version must be version controlled.

3. Continuous integration and continuous deployment. This means that the code is automatically (or at least semi-automatically) tested and deployed. The backend and frontend code is tested using unit tests. The model search code is tested using validation methods such as cross-validation and Bayesian analysis on the discovered models. Usually the model search code is very computationally intensive, and it is not feasible to run it on every commit. Instead, it is run periodically, for example once a day. If the clould infrastructure required to run the model search code is not available to automate validation and deploymen, at least make sure that the code is easily runnable. This means that the code must be well documented, and that the required infrastructure must be well documented. Also aggregate commands using a Makefile or a similar tool. Pay attention on the dependences between dataset and the model training. If the dataset changes significantly, not only the deployed model must be retrained, but the model search algorithm may need to be rethought.

4. Reports as deliverables. During sprints, the deliverables of data exploration are reports. The reports must be version controlled and must be reproducible. The reports must be generated in a way that is understandable by the client and the sponsor.

5. Setup quantitative goals. Do not fall on the trap of forever improving the model. Instead, setup quantitative goals for the model performance. For example, the model must have a precision of at least 90%. Once you reach the goal, prioritize other tasks.

6. Measure *exactly* what you want. During model validation, use your own metrics based on the project goals. Usually, more than one metric is needed, and they might be conflicting. Use strategies to balance the metrics, such as Pareto optimization. Beware of the metrics that are most used in the literature. They might not be suitable for your project. Notice that during model training, some methods are limited to the loss functions that they can

[3] https://dvc.org/

optimize. If possible, choose a method that can optimize the loss function that you want. Even if you are not explicitly optimizing the wanted metric, you might find a model that performs well on that metric. That is a reason validation is important.

7. Report model stability and performance variance. Understanding the limitations and characteristics of the model is more important than the model performance. For example, if the model performance is high, but the model is unstable, it is not suitable for production. Also, in some scenarios, interpretability is more important than performance.

8. In user interface, mask data-science-specific terminology. Usually, data science software gives the user the option to choose the model. In order to avoid confusion, the user interface must mask the data-science-specific terminology. This helps non experts to use the software consciously.

9. Monitor model performance in production. If possible setup feedback from the user interface. Avoid automation of model releases because concept drift usually requires exploratory analysis.

10. Use the appropriate backend. REST API vs websocket. The choice depends on the requirements of the project. REST API is more suitable for stateless requests, while websocket is more suitable for stateful requests. For example, if the user interface must be updated in real-time, websocket is more suitable. If the user interface is used to submit batch requests, REST API is more suitable.

5

Statistical learning theory

To understand God's thoughts we must study statistics, for these are the measure of His purpose.

— Florence Nightingale*, her diary*

We can address several kinds of problems using algorithms that learn from data. However, we focus on the problem of *inductive learning*. Before we go further, let us define some terms.

Definition 5.1: Artificial intelligence

The field that studies algorithms that exhibit intelligent behavior.

Artificial intelligence is a very broad field, including not only the study of algorithms that exhibit intelligent behavior, but also the study of the behavior of intelligent systems. For instance, it encompasses the study of optimization methods, bioinspired algorithms, robotics, philosophy of mind, and many other topics. We are interested in the subfield of artificial intelligence that studies algorithms that exhibit some form of intelligent behavior.

Definition 5.2: Machine learning

The subfield of artificial intelligence that studies algorithms that enable computers to automatically learn from data.

Machine learning is the subfield of artificial intelligence that studies algorithms that enable computers to automatically learn and improve their performance on a task from experience, without being explicitly programmed by a human being.

Definition 5.3: Predictive learning

The machine learning paradigm that studies the problem of making predictions given known input data.

The machine learning paradigm that focuses on making predictions about outcomes (sometimes about the future) based on historical data. Depending on the reasoning behind the learning algorithms, the main predictive algorithms are classified in either inductive or transductive.

Definition 5.4: Inductive learning

The machine learning approach that involves deriving general rules from specific observations.

Induction a type of reasoning that goes from specific instances to more general principles. Inductive learning is the machine learning approach that studies algorithms that, given data representing the set of specific instances, derive general rules that can make predictions about *any* new instances.

Figure 5.1 give us a hierarchical view of the learning field. Alternatives — such as descriptive learning in opposition to predictive learning, or transductive learning in opposition to inductive learning — are out of the scope of this course.

Maybe the most general (and useful) framework for predictive learning is Statistical Learning Theory. In this chapter, we will introduce the basic concepts of this theory.

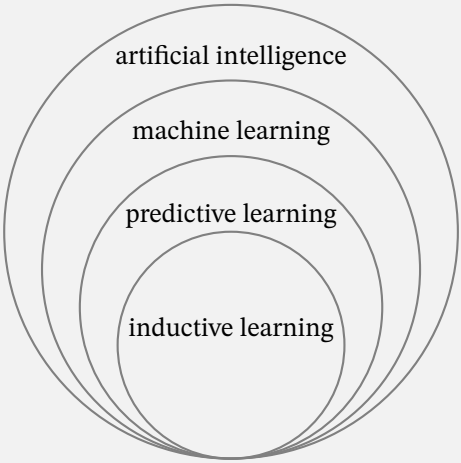
5.1 Hypothesis and setup

Consider the set

$$\{(\mathbf{x}_i, y_i) : i = 1, \dots, n\} \tag{5.1}$$

where each sample i is associated with a feature vector $\mathbf{x}_i \in \mathcal{X}$ and a target variable $y_i \in \mathcal{Y}$. We assume that samples are random independent

Figure 5.1: Organizational chart of the learning field.



identically distributed (i.i.d.) observations drawn according to

$$P(x, y) = P(x)P(y|x).$$

Both $P(x)$ and $P(y|x)$ are fixed but unknown.

This is equivalent to the original learning problem stated by V. N. Vapnik (1999b), where a generator produce random vectors $\mathbf{x}$ according to a fixed but unknown probability distribution $P(x)$ and a supervisor returns an output value y for every input vector x according to a conditional distribution function $P(y|x)$, also fixed but unknown.

Moreover, note that this setup is compatible with the idea of tidy data and 3NF (see section 3.2.3). Of course, we assume X, Y are only the measured variables (or non-prime attributes). In practice, it means that we left aside the keys in the learning process.

5.2 The learning problem

Consider a *learning machine* capable of generating a set of functions $f(x; \theta) \equiv f_\theta(x)$, $\theta \in \Theta$ and $f_\theta : \mathcal{X} \to \mathcal{Y}$. The problem of learning is that of choosing, among all possible f_θ, the one that predicts the target variable the best possible way.

In order to learn, we must first define the *loss* (or discrepancy) $\mathcal{L}$ between the response y to a given input x, drawn from $P(x, y)$, and the response provided by the learning machine.

Then, given the *risk function*

$$R(\theta) = \int \mathcal{L}(y, f_\theta(x))\, dP(x, y), \tag{5.2}$$

the goal is to find the function f_θ that minimizes $R(\theta)$ where the only available information is the *training set* (5.1). This is the *empirical risk minimization* (ERM) problem.

This formulation encompasses many specific problems. I focus on the two of them which I believe are the most fundamental ones: *binary data classification*[1] and *regresssion estimation*[2]. I left aside the density estimation problem, once it is not addressed in the remaining of the book.

Binary data classification task. In this task, the output y take on only two possible values, zero or one, and the functions f_θ are indicator functions. For the loss

$$\mathcal{L}(y, f_\theta(x)) = \begin{cases} 0 & \text{if } y = f_\theta(x) \\ 1 & \text{if } y \neq f_\theta(x), \end{cases}$$

we aim at minimizing the risk (5.2) which becomes the probability of classification error.

Regression estimation task. Let the outcome y be a real value and the *regression* r be

$$r(x) = \int y\, dP(y|x).$$

The regression function is the function $r = f_\theta$ that minimizes the risk function (5.2) with the loss

$$\mathcal{L}(y, f_\theta(x)) = \big(y - f_\theta(x)\big)^2.$$

If $r \notin \{f_\theta : \theta \in \Theta\}$, the function $f_{\theta'}$ that minimizes the risk function is the closest to the regression function in the metric l_2, i.e. we look for θ' such that

$$\theta' = \arg\min_{\theta \in \Theta} \sqrt{\int \big(r(x) - f_\theta(x)\big)^2\, dP(x)}.$$

[1] Vapnik calls it *pattern recognition*.

[2] We are not talking about *regression analysis*.

5.3 ERM inductive principle

In the following sections, z describes the pair (x, y) and $L(z, \theta)$ a generic loss function. The training dataset is thus a set of n i.i.d. samples $z_1, \ldots, z_n$.

Since the distribution $P(z)$ is unknown, the risk functional $R(\theta)$ is replaced by the *empirical risk functional*

$$R_n(\theta) = \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta). \tag{5.3}$$

Approximating $R(\theta)$ by the empirical risk functional $R_n(\theta)$ is the so called ERM inductive principle. The ERM principle is the basis of the statistical learning theory.

Classical methods, such as least-squares, maximum likelihood, and maximum a posteriori are all realizations of the ERM principle for specific loss functions and hypothesis spaces.

In the following sections, we address the four main questions of learning theory. We summarize them in table 5.1.

Table 5.1: The four main questions of learning theory.

Part	Question
Consistency	What are the necessary and sufficient conditions for consistency of a learning process?
Rate of convergence	How fast is the rate of convergence of the learning process?
Generalization	How can one controle the generalization ability of the learning process?
Construction	How can one construct a learning machine that satisfies the conditions of consistency and generalization?

5.4 Consistency of learning processes

Addressing consistency of a learning process means that we are interested in the convergence of the empirical risk functional $R_n(\theta)$ to the

risk functional $R(\theta)$ as $n \to \infty$. In other words, it is an asymptotic theory about the behavior of the empirical risk functional as the sample size n goes to infinity.

The necessary and sufficient conditions for consistency give us guarantees that the learning process is general and cannot be improved given our premises. The most important topic in this section is the Vapnik-Chervonenkis (VC) entropy.

5.4.1 Definition of consistency

An ERM method is consistent if it produces a sequence of functions f_{θ_n}, for $n = 1, 2, \dots$, for which both the expected risk and the empirical risk converge to the their minimum values.

Definition 5.5: Consistency of a learning process

Let θ_n be the solution of

$$\theta_n = \arg\min_{\theta \in \Theta} R_n(\theta).$$

An ERM method is consistent for the set of functions $\{L(z, \theta) : \theta \in \Theta\}$ and the probability distribution $P(z)$ if

$$\lim_{n \to \infty} R(\theta_n) = \inf_{\theta \in \Theta} R(\theta),$$
$$\lim_{n \to \infty} R_n(\theta_n) = \inf_{\theta \in \Theta} R(\theta).$$

This definition means that one can estimate the risk functional $R(\theta)$ by the empirical risk functional $R_n(\theta)$, while the values of achieved risks converge to the minimum value of the risk functional. See fig. 5.2.

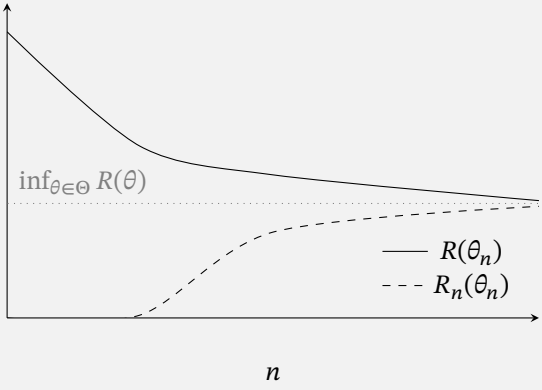
However, since this definition of consistency includes cases of trivial consistency, there is no way to obtain such conditions.

Consider the following example. Suppose we have found a set of functions $\{f_\theta : \theta \in \Theta\}$ such that the ERM method is not consistent. Let's add one more function $\phi(z)$ to the set, such that

$$\inf_{\theta \in \Theta} L(z, \theta) > \phi(z), \ \forall z.$$

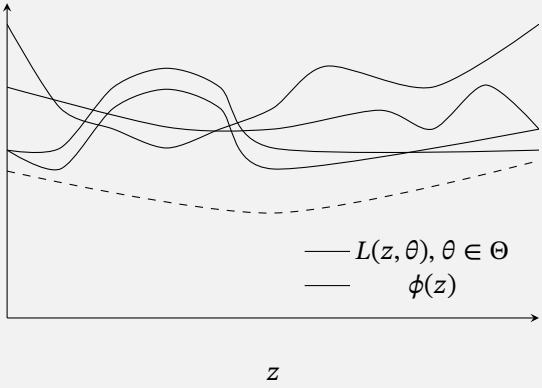
It is straightforward to see that the ERM method is consistent for the new set of functions $\{L(z, \theta) : \theta \in \Theta\} \cup \{\phi\}$ and the probability distribution $P(z)$. In this case, the function $\phi(z)$ gives both the minimum

Figure 5.2: Convergence of the empirical and expected risk functionals.



value of the risk functional and the empirical risk functional. This is illustrated in fig. 5.3.

Figure 5.3: An illustrative case of trivial consistency.



6 Data handling

> *Tidy datasets are all alike, but every messy dataset is messy in its own way.*
>
> — Hadley Wickham, *Tidy Data*

Data handling is the process of adjusting data to make it suitable for analysis. It involves three main tasks: data transformation, data cleaning, and data integration.

In this chapter, we consider that tables are rectangular data structures in which values of the same column share the same properties (i.e. the same type, same restrictions, etc.) and each column has a name. Moreover, we assume that any value is possibly *missing*.

6.1 Data handling operators

In the literature and in software documentation, you will find a variety of terms used to describe data handling operations[1]. They often refer to the same or similar operations, but the terminology can be confusing. In this section, I present a summary of these operations mostly based on Wickham, Çetinkaya-Rundel, and Grolemund (2023) definitions[2].

These operations are the building blocks of the data handling tasks we will discuss in the next sections. They can also be extensively parametrized

[1] The terminology "data handling" itself is not universal. Some authors and libraries call it "data manipulation", "data wrangling", "data shaping", or "data engineering". I use the term "data handling" to avoid confusion with the term "data manipulation" which has a negative connotation in some contexts.

[2] Which are called *verbs*.

Slide 6.1: Data handling operators

- Filtering rows;
- Selecting columns;
- Mutating columns;
- Aggregating rows;
- Binding datasets;
- Joining datasets;
- Pivoting (spreading) and unpivoting (gathering) datasets.

and combined to create more elaborate data handling pipelines. For instance, most of them can use predicates to define the groups, arrangements, or conditions under which they should be applied.

We use the following terminology to refer to the data handling parameters:

- **Predicate**: a function that returns a logical value, used to filter rows/columns or to define the groups of rows/columns to be processed;
- **Aggregation function**: a function that returns a single value given a vector of values (in which, the order of the values may be important);
- **Window function**: a function that returns a vector of values given a vector of values in which, the order of the values is important;
- **Expression**: a function that returns a vector of values elementwise, used to create new columns or to modify existing ones.

Operators are also vectorized, meaning that they can be applied to multiple columns or rows at once. This is a key feature of data handling operations, as it allows for expressive and efficient data manipulation.

Many of them are also reversible, meaning that they can be undone. This is important because it allows for reproducibility and traceability of the data handling process.

Slide 6.2: Data handling pipelines

- Data handling operations can be combined to create complex pipelines;
- Operators may be reversible;
- Operators are vectorized;
- They can be parametrized with predicates, aggregation functions, and expressions;
- They operate on datasets and return new datasets as output.
- They are declarative.

They operate on a dataset (or more than one) given as input and return a new dataset as output. This is important because it allows for the creation of data handling pipelines, where the output of one operation is the input of the next one. Parameters like column names, predicates, aggregation functions, and expressions can be passed to these operations to customize their behavior.

Unlike traditional procedural programming, where conditional statements and loops are used to manipulate data, data handling operations are declarative. This means that they are expressed in terms of what should be done, not how it should be done. This is a powerful abstraction that allows for the creation of complex pipelines with a few lines of code.

6.1.1 Filtering rows

Filtering is the process of selecting a subset of rows from a dataset based on a predicate. If more than a single predicate is used, they are combined using a logical operator, such as `AND` or `OR`.

After filtering, the dataset will contain only the rows that satisfy the predicate. Columns remain unchanged. This operation is potentially irreversible, as the removed rows are lost.

In the basic form, each row is treated independently. For instance, the predicate `age > 18` will select all rows where the value in the `age` column is greater than 18.

However, if the predicate depends on an aggregation or window function, one must specify the groups and/or the order of the rows. For instance, the predicate `age > mean(age) group by country` will select the rows where the value in the `age` column is greater than the mean of the `age` for each `country`. Another example is the predicate `cumsum(price) < 100 sort by date`, which selects the rows that satisfy the condition that the cumulative sum of the `price` column is less than 100 given the order of the rows defined by the `date` column.

The trivial group is the entire dataset, so it is usually not necessary to specify it explicitly. However, it is usually not sensible to not specify the order of the rows.

When dealing with real values, be aware of floating-point precision issues. In other words, do not use the equality operator to compare real numbers. Most of libraries provide operators to compare real numbers within a given tolerance.

Practical tips

- Use filtering to remove rows that are not relevant to your analysis;
- Use predicates to define the conditions under which rows should be removed;
- When aggregation functions are needed to define the predicate, specify the groups and the order of the rows;
- Be aware of floating-point precision issues when comparing real numbers.

6.1.2 Selecting columns

Selecting is the process of choosing a subset of columns from a dataset. The remaining columns are discarded. This operation is not reversible, as the discarded columns are lost. Rows remain unchanged.

There are two main ways to select columns: by name or by predicate. The former is the most common and is used to select a fixed set of columns. The latter is used to select columns that satisfy a given condition, i.e., the values in the columns are used to determine which columns should be selected.

When selecting columns by name, one can use a list of column names or a regular expression[3]. The latter is useful when the column names follow a pattern that reflects the semantics of the columns. For instance, one can use the regular expression `col[0-9]+` to select all columns whose names start with `col` followed by one or more digits.

When selecting columns by predicate, one can use a function that returns a logical value to define the condition under which a column should be selected. For instance, one can use the predicate `isnumeric` to select all columns that contain numeric values. Notice, however, that the predicate is applied to each column independently and returns a single logical value for each column.

Like filtering, selecting predicates might contain aggregation functions. Although it is theorically possible to consider the order of the values in the columns, it is not common to do so. (Especially because one would need to assume that the rows are previously sorted by some criterion.) Groups, however, never make sense in this context, once the predicate is applied to each column independently.

Depending on the context, it may be useful to "drop" columns instead of selecting them. This is the same as selecting all columns except the ones specified. This is useful when the number of columns to be dropped is small compared to the total number of columns. Strictly speaking, we just need to negate the predicate or the regular expression used to select the columns.

Finally, it is very common to find libraries and framework in which the order of the columns is important. As a result, columns can be selected by position as well. I find this practice error-prone and I recommend avoiding it whenever possible.

6.1.3 Mutating columns

Mutating is the process of creating new columns. The operation is reversible, as the original columns are kept. The new columns are added to the dataset.

The values in the new column are determined by an expression. The expression is a function that returns a vector of values given the values in the other columns. The expression can be a simple function, such as `y = x + 1`, or a more complex function, such as `y = ifelse(x > 0,`

[3]Regular expressions are very general and powerful, but they are also complex and error-prone. An alternative is to use some form of hierarchical naming, such as `type.column` to express groups of columns.

Practical tips

- Use selecting to remove columns that are not relevant to your analysis;
- Use column names or regular expressions (or hierarchical names) to select columns;
- Use predicates (many to one, with no aggregation functions) to define the conditions under which columns should be selected;
- Avoid depending on the order of the columns.

`1, 0)`. Here, `x` and `y` are the names of an existing and the new column, respectively.

One may also use an aggregation and window function in the expression. This is particularly useful when performing mutation considering a group. In this case, the returned value is repeated (aggregation function) for each row of the same group. Like in filtering, the more explicit you can be about order and groups, the better.

For example, the expression `y = cumsum(x) group by category sort by date` will create a new column `y` with the cumulative sum of the `x` column for each `category` given the order of the rows defined by the `date` column.

Sometimes, the same expression can be used to create multiple columns. This is useful when the new columns are related. To do so, one first specifies the columns in the same way as when selecting columns. Then, one needs to specify a rule to name the new columns. For instance, `x_new = x + 1 across x matches ^col[0-9]+$`.

Practically speaking, mutation can overwrite existing columns. This is useful when the new column is a replacement for the old one. Formally, overwriting is just a sequence of mutation and selection operations.

6.1.4 Aggregating rows

We can aggregate the rows of a dataset to create a new dataset with fewer rows. The operation is not reversible, as the discarded rows are lost. The columns are also lost, only the new aggregate columns remain.

Practical tips

- Use mutating to create new columns that are relevant to your analysis;
- Use expressions to define the values of the new columns;
- Use aggregation and window functions in the expression to create new columns based on groups and order;
- Use the same expression to create multiple columns when the new columns are related.

The values in the new columns are determined by an aggregation function. Like filtering and mutation, the aggregation function can be parametrized by specifying a group and/or an order.

The resulting dataset will contain one row for each group. The values in the new columns are determined by the aggregation function applied to the values in the other columns. All columns that define the groups are usually kept in the resulting dataset. In this case, as expected, values of such columns are equal for all rows in the same group.

For instance, the aggregation function `mean(x) group by category` will create a new dataset with one row for each different value of `category` and a new column with the mean of the `x` column for each group.

Practical tips

- Use aggregation to summarize the data in a dataset;
- Use aggregation functions to define the values of the new columns;
- Other columns are lost;
- Use the group and order parameters to define the groups and the behavior of the aggregation function.

6.1.5 Binding datasets

One trivial, yet important, operation is to bind datasets. This is the process of combining two or more datasets into a single dataset. The operation is reversible, as the original datasets are kept. The new dataset contains all the rows and columns of the original datasets.

There are two ways to bind datasets: by rows or by columns. The former is used to combine datasets that have exactly the same columns but represent different parts of the same dataset. The latter is used to combine datasets that comprise the same observations (rows) but captures different aspects of the same dataset.

When binding datasets by rows, the datasets must have the same columns[4]. The resulting dataset will contain all the rows of the original datasets. The columns remain unchanged. It is a good practice to create a new column that represents the source of each row. For instance, if each table represents data collected in a different year, one can create a new column `year` that contains the year of the data.

When binding datasets by columns, the datasets must have the same number of rows. Each matching row represent the same observation[5]. The resulting dataset will contain all the columns of the original datasets. The rows remain unchanged.

Practical tips

- Use binding to combine datasets that represent different parts of the same dataset;
- Use binding by rows to combine datasets that have the same columns — in this case, create a new column that represents the source of each row;
- Use binding by columns to combine datasets that have the same number of rows.

Talk about splitting as the reverse function, and the reason why missing columns may be a problem. Example of the unit of measurement.

[4] In practice, it is usually required that they share the same order of the columns as well. This is not a theoretical requirement, but a common limitation of most libraries.

[5] Practically speaking, either the order of the rows or a key column is used to match the rows of the datasets. In both situations, this is equivalent to a join operation by the row number or the key column; assuming that both datasets contains the same observations.

6.1.6 Joining datasets

Joining is the process of combining two datasets into a single dataset based on common columns. The operation may not be reversible, consult section 3.2.1 for more details.

The join of two tables is the operation that returns a new table with the columns of both tables. Let `U` be the common set of columns. For each occurring value of `U` in the first table, the operation will look for the same value in the second table. If it finds it, it will create a new row with the columns of both tables. If it does not find it, no row will be created. This operation assumes that values in `U` are unique in each table.

The variation described above is usually called natural or inner join. Three other variations are possible.

- Left join: for each occurring value of `U` in the first table, the operation will look for the same value in the second table. If it finds it, it will create a new row with the columns of both tables. If it does not find it, it will create a new row with the columns of the first table and missing values for the columns of the second table.
- Right join: the same as the left join, but the roles of the tables are reversed.
- Outer join: for each different value of `U` in both tables, the operation will create a new row with the columns of both tables. If a value is missing in one table, it will be filled with a missing value.

Practical tips

- Use joining to integrate datasets;
- Be aware of the risks of joining datasets (section 3.2.1), for example, that some joins may create invalid rows;
- Use the appropriate variation of the join operation in applications.

6.1.7 Pivoting and unpivoting

Another important operation is to pivot and unpivot datasets. These are the processes of transforming a dataset from a long format to a wide

format and vice versa. The operations are reversible and they are the inverse of each other.

Pivoting requires to specify a name column — whose discrete and finite possible values will become the names of the new columns — and a value column — whose values will be spread across the rows. All remaining columns are considered to be keys, uniquely identifying each row of new the dataset.

Unpivoting[6] is the reverse operation. One must specify all the columns whose names are the values of the before called name column. The values of these columns will be gathered into a new column. As before, all remaining columns are considered to be keys.

In practical applications, where not all remaining columns are keys, one must aggregate rows beforehand.

Table 6.1: Pivoting example.

name	year	value
A	2019	1
A	2020	2
A	2021	3
B	2019	4
B	2020	5
B	2021	6

name	2019	2020	2021
A	1	2	3
B	4	5	6

The left table is in the long format and the right table is in the wide format. The name column is `year` and the value column is `value`.

Table 6.1 shows an example of pivoting. The left table is in the long format and the right table is in the wide format. The name column is `year`, the value column is `value`, and the remaining column is `name` which is an unique identifier of the rows in the wide format.

6.1.8 An algebra for statistical transformations

In recent years, some researchers made an effort to create a formal algebra for statistical transformations. The idea is to create a set of op-

[6] Which Wickham, Çetinkaya-Rundel, and Grolemund call pivot longer.

Practical tips

- Use pivoting to transform datasets from a long format to a wide format;
- Use unpivoting to transform datasets from a wide format to a long format;
- Be aware of the need to aggregate rows before unpivoting.

erations that can be combined to create complex statistical transformations. This is similar to the idea of relational algebra, which is a set of operations that can be combined to create complex queries.

The difference between relational algebra and a formal algebra for statistical transformations is that the latter is more complex. This is because statistical transformations are more complex than queries. For instance, the concept of missing data is not present in relational algebra, but it is in statistical transformations.

Song, Jagadish, and Alter (2021), for example, propose a formal paradigm for statistical data transformation. They present a data model, an algebra, and a formal language. Their goal is to create a standard for statistical data transformation that can be used by different statistical software.

However, in my opinion, the major deficiency of their work is that they mostly try to "reverse engineer" the operations that are commonly used in statistical software. This is useful for the translation of code between different software, but it is not productive to advance in the theoretical understanding of statistical transformations.

If one ought to tackle the challenge of formally expressing statistical transformations, I think one should start from the basic operations. Basic operations mean that they are irreducible, i.e., they cannot be expressed as a sequence of other operations.

Some thoughts about it:

- Binding columns can be expressed as a join operation, thus it is not a basic operation.
- Some software provide features that can be better expressed in other (often simpler) ways. Row naming is an example. It is useful to keep track of the origin of each row, but names can be just

another column. I argue for excluding row naming in a formal algebra.

- Some operations are very useful and recurring, even if they are not basic. Such operations must be omitted from the formal algebra for the sake of simplicity. However, any software that implements a language for the formal algebra can provide syntax sugar for these operations.

- Not defining your algebra in terms of a specific programming language is a good practice. This is because the algebra is a theoretical concept and should be independent of any implementation. It also gives opportunities to rethink the things that commonly done in a specific way. This can lead to new insights and correct error-prone practices.

- Pivoting seems to be "different" enough to the other operations to be considered in the set of basic operations. However, it is not hard to see that they can be rewritten as combinations with the meta tables containing the possible values of the attributes (or some sort of aggregation function).

6.2 Data handling pipeline

Before we study the data handling tasks, we need to understand that a data handling pipeline is a sequence of operations that *does* depend on the input data. This might seem obvious, but the implications are not.

A common error in data handling is to perform operations ad hoc, usually leading to data leakage. For instance, one might impute missing values before splitting the data into training and testing sets. This is a mistake because the imputation is based on the entire dataset, including the testing set.

To avoid this kind of error, one must declare[7] the operations that will be performed on the data before applying them. This is usually done by creating the full data handling pipeline beforehand.

The pipeline, like a model, must be "fitted" to the data. This means that parameters of the operations are not fixed until the first data is given as input. Subsequent data fed to the pipeline will be handled keeping the first "learned" parameters.

[7] This is the declarative nature of data handling operations.

Consider the following example. Suppose we have a dataset with missing values for variable `A`. We want to impute the missing values and then standardize `A`. The pipeline is created as follows: `D -> impute_zero(A) -> standardize(A)`.

The operation `impute_zero(A)` is parametrized by the value 0, which, in this case, is fixed. However, the operation `standardize(A)` is parametrized by the mean and the standard deviation of the values in `A`. These values are not fixed until the first data is given as input.

A note about fixed parameters

Even if your data handling pipeline contains operations that have fixed parameters and can be safely applied to data before the model search, *I strongly recommend* that you declare the pipeline as a whole. This is because it is easier to maintain and reproduce the data handling process, especially in deployment. Performing ad hoc handling in your data is a source of errors and important transformations can be forgotten when receiving new data.

In a practical scenario, the source code of the *model search* method includes not only strategies for the model, but also the data handling pipeline. Moreover, the deployment of the model includes the data handling pipeline as well. In other words, it does not matter which model is used, in the example above, the mean and the standard deviation of the values in `A` should be stored and used in deployed models.

In terms of reproducibility and validation, having a single consolidated pipeline is crucial.

A note about "filtering" operations

Some operations may conditionally remove rows from the dataset. For instance, after observing that there exists few missing values in an important column, one might decide to remove rows with missing values in it. In production, this means that some new observations might be discarded before reaching the model itself. However, the user still expects an answer from the model. In this case, one must define either a default value for the answer or a default behavior to handle discarded examples.

Slide 6.3: Data handling pipeline

- A data handling pipeline is a sequence of operations that depend on the input data;
- The pipeline must be declared before applying the operations;
- The pipeline is fitted to the data;
- The selected pipeline is part of the model search and deployment.
- Even operations that have fixed parameters or that can be safely applied to data before the model search should be declared in the pipeline.

XXX: maybe state that before reaching the pipeline data is already tidy, this way simple integration (not enhancement), pivoting and aggregating are kept outside the pipeline. These operations must depend only on variable names and not variable values.

6.3 Data transformation

The first task in data handling is data transformation. This is the process of adjusting the format and the types of the data to make it suitable for analysis.

Usually, the starting point of data transformation is to make the data tidy, i.e., to have each variable in a column and each observation in a row. Remember that, depending on the problem definition, we target a particular observational unit. Having a clear picture of the observational unit is important to define the columns and the rows of the dataset.

Then, when the data format is acceptable, we can perform a series of operations to make the column's types and values suitable for modeling. The reason for this is that most machine learning methods require the input variables to follow some restrictions. For instance, some methods require the input variables to be real numbers, others require the input variables are in a specific range, etc.

6.3.1 Reshaping

TODO: pipeline exceptions: like pivoting and aggregating are kept outside the pipeline.

Reshaping is the process of changing the format of the data. The most common reshaping operations are pivoting and unpivoting, which we have already discussed. However, there are other reshaping operations that are useful in practice.

For instance, one can reshape a dataset by splitting a column into multiple columns. This is useful when a column contains multiple values that should be separated. This can be done with mutation with appropriate expressions. Some frameworks might provide special functions to do this, usually called splitting functions.

We can also consider reshaping the operations of filtering, selecting, and aggregating. Filtering is usually done to reduce the scope of the data, given some conditions on the variables. Selecting is usually done to remove irrelevant variables or highly correlated ones. Aggregating in a reshaping task is usually applied together with pivoting to change the observational unit of the dataset.

Slide 6.4: Reshaping

- Reshaping is the process of changing the format of the data;
- The most common reshaping operations are pivoting and unpivoting;
- Other common operation include:
  - Splitting a column into multiple columns;
  - Filtering to reduce the scope of the data;
  - Selecting to remove irrelevant variables or highly correlated ones;
  - Aggregating to change the observational unit of the dataset.

6.3.2 Type conversion

Type conversion is the process of changing the type of the values in the columns. This is usually done to make the data suitable for modeling.

For instance, some machine learning methods require the input variables to be real numbers.

The most common type conversions are from categorical to numerical and from numerical to categorical. The former is usually done by creating dummy variables, i.e., a new column for each possible value of the categorical variable. This transformation is also known as one-hot encoding. The latter is usually done by binning (discretizing) the numerical variable, either by frequency or by range.

Slide 6.5: Type conversion

- Type conversion is the process of changing the type of the values in the columns;
- Use one-hot encoding to convert categorical variables to numerical;
- Use binning to convert numerical variables to categorical.

6.3.3 Normalization

Normalization is the process of scaling the values in the columns. This is usually done to keep data in a specific range or to make the data comparable. For instance, some machine learning methods require the input variables to be in the range $[0, 1]$.

The most common normalization methods are standardization and rescaling. The former is done by subtracting the mean and dividing by the standard deviation of the values in the column. The latter is performed so the values are in a specific range, usually $[0, 1]$ or $[-1, 1]$.

Clamping after rescaling

In production, it is common to clamp the values after rescaling. This is done to avoid the model to make predictions that are out of the range of the training data.

Related to normalization is the log transformation. This is usually done to make the data more symmetric or to reduce the effect of outliers. The log transformation is the process of taking the logarithm of the values in the column.

Slide 6.6: Normalization

- Normalization is the process of scaling the values in the columns;
- Use standardization to make the values have mean 0 and standard deviation 1;
- Use rescaling to make the values be in a specific range;
- Use the log transformation to make the data more symmetric or to reduce the effect of outliers.

6.3.4 Sampling

Sampling is the process of selecting a random subset of the data. This is usually done to reduce the size of the data or to create a balanced dataset. For instance, some machine learning methods are heavily affected by the number of observations in each class. Also, some methods are computationally expensive and a smaller dataset might be enough to solve the problem.

The most common sampling methods are random sampling and resampling[8]. The former is done by selecting a random subset of the data. The latter is done by selecting a random subset of the data with replacement.

While random sampling is useful to reduce the size of the data, resampling can be used to increase the size of the data. (Although this has some caveats.) Moreover, resampling can also create variations of the original dataset with the same distribution of the values.

6.3.5 Dimensionality reduction

Dimensionality reduction is the process of reducing the number of variables in the data. This is usually done to reduce the complexity of the model or to identify irrelevant variables. The so-called *curse of dimensionality* is a common problem in machine learning, where the number of variables is much larger than the number of observations.

There are two main types of dimensionality reduction algorithms: feature selection and feature extraction. The former is done by selecting

[8]Resampling is the process of sampling with replacement, sometimes called bootstrapping.

a subset of the variables that leads to the best models. The latter is done by creating new variables that are combinations of the original ones.

Feature selection can be performed before modeling (filter), together with the model search (wrapper), or as a part of the model itself (embedded).

Feature extraction is usually done by linear methods, such as principal component analysis (PCA), or by non-linear methods, such as autoencoders. These methods are able to compress the information in the data into a smaller number of variables.

Slide 6.7: Dimensionality reduction

- Dimensionality reduction is the process of reducing the number of variables in the data;
- Use feature selection to select a subset of the variables that leads to the best models;
- Use feature extraction to create new variables that are combinations of the original ones.

Practice!

Can you identify which data transformation operations are used to make datasets presented in chapter 3 tidy?

6.4 Data cleaning

Data cleaning is the process of removing errors and inconsistencies from the data. This is usually done to make the data more reliable and to avoid bias in the analysis.

6.4.1 Dealing with missing data

Since most models do not cope with missing data, it is crucial to deal with it in the data handling pipelines.

There are four main strategies to deal with missing data:

- Remove the rows with missing data;

- Remove the columns with missing data;
- Impute the missing data;
- Use an indicator variable to mark the missing data.

Removing rows and columns are commonly used when the number of missing data is small compared to the total number of rows or columns. However, be aware that removing rows can artificially change data distribution, especially when the missing data is not missing at random.

Imputing the missing data is usually done by replacing the missing values with some statistic of the available values in the column, such as the mean, the median, or the mode. This is a simple and effective strategy, but it can introduce bias in the data. Also, it is not suitable when one is not sure whether the missing data is missing because of a systematic error or phenomenon.

For this case, creating an indicator variable is a good strategy. This is done by creating a new column that contains a logical value indicating whether the data is missing or not[9]. By doing so, the model can learn the importance of the missing data[10].

6.4.2 Dealing with invalid and inconsistent information

Sometimes, during data collection, information is recorded using special codes. For instance, the value 9999 might be used to indicate that the data is missing. Such codes must be replaced with more appropriate values before modeling.

Another common problem is inconsistent information. For instance, the same category might be represented by different names. This is usually done by creating a dictionary that maps the different names to a single one.

It is also useful to check whether all columns that store physical quantities have the same unit of measurement. If not, one must convert the values to the same unit.

If one knows that a variable has a specific range of values, it is useful to check whether the values are within this range. If not, one must replace the values wit missing data or with the closest valid value.

[9] Some kind of imputation is still needed, but we expect the model to deal better with it

[10] Sometimes the indicator variable is already present: pregnancy and sex example.

6.4.3 Outliers

Outliers are observations that are significantly different from the other observations. They can be caused by errors in the data collection process or by the presence of a different phenomenon. In both cases, it is important to deal with outliers before modeling.

There are many outliers detection methods, consult TODO.

Slide 6.8: Data cleaning

- Data cleaning is the process of removing errors and inconsistencies from the data;
- Use the following strategies to deal with missing data:
  - Remove the rows with missing data;
  - Remove the columns with missing data;
  - Impute the missing data;
  - Use an indicator variable to mark the missing data.
- Replace special codes with more appropriate values;
- Create a dictionary to map different names to a single one;
- Check whether all columns that store physical quantities have the same unit of measurement;
- Check whether the values are within the expected range;
- Use outlier detection methods to deal with outliers.

6.5 Data integration

Data integration is the process of combining data from different sources into a single dataset. This is usually done to create a more complete dataset or to create a dataset with a different observational unit.

To perform integration, consider the discussions in sections 3.2.1 and 3.2.3.

Additionally, one must consider the following points:

- Sometimes the same column may have different names in different datasets. Redundant columns must be removed.
- Separate datasets that share the same variables usually happen because there is a hidden variable that is not present in the datasets. During integration, the new variable must be created.

Slide 6.9: Data integration

- Data integration is the process of combining data from different sources into a single dataset;
- Not every join is possible, consider the discussions in sections 3.2.1 and 3.2.3;
- Remove redundant columns;
- Create new variables to represent the hidden variables.

Hard to incorporate in the pipeline when joins only, but data enhancement works better inside the pipeline.

7 Machine learning tasks

They say "Na prática, a teoria é outra," I say "Se sua teoria não funciona na prática, ela está errada demais."

In the previous chapter, we define two fundamental inductive learning tasks: *classification* and *regression*. In real-world applications, however, we may require different tasks to solve our data science problem. Descriptive learning tasks are out of the scope of this book, I suggest reading .... Even restricting ourselves to discuss only inductive learning, some machine learning tasks comprise a combination of fundamental tasks.

Also, we show examples of different inductive biases and how the main learning algorithms work – symbolic (decision trees), spatial (nearest neighbors), statistical (naïve Bayes and Bayesian networks), gradient optimization (neural networks).

7.1 Multiclass

7.2 Manifold learning

7.3 Recommender systems

7.4 Reinforcement learning

8

Model evaluation

It's dangerous to go alone! Take this.

— Unnamed Old Man, *The Legend of Zelda*

9

Ethical and legal issues

It's a trap!

— Admiral Ackbar, *Star Wars Episode VI: Return of the Jedi*

Bibliography

Beaumont, P. B. and R. G. Bednarik (2013). In: *Rock Art Research* 30.1, pp. 33–54. URL: https://search.informit.org/doi/10.3316/informit.488018706238392.

Benavoli, A., G. Corani, J. Demšar, and M. Zaffalon (2017). "Time for a Change: a Tutorial for Comparing Multiple Classifiers Through Bayesian Analysis". In: *Journal of Machine Learning Research* 18.77, pp. 1–36. URL: http://jmlr.org/papers/v18/16-305.html.

Breiman, L. (1996). "Bagging predictors". In: *Machine Learning* 24.2, pp. 123–140. DOI: 10.1007/BF00058655.

Codd, E. F. (1970). "A Relational Model of Data for Large Shared Data Banks". In: *Commun. ACM* 13.6, pp. 377–387. ISSN: 0001-0782. DOI: 10.1145/362384.362685.

Cortes, C. and V. Vapnik (1995). "Support-vector networks". In: *Machine Learning* 20.3, pp. 273–297. DOI: 10.1007/BF00994018.

Cover, T. M. (1965). "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition". In: *IEEE Transactions on Electronic Computers* EC-14.3, pp. 326–334. DOI: 10.1109/PGEC.1965.264137.

Fagin, R. (1979). "Normal forms and relational database operators". In: *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*. SIGMOD '79. Boston, Massachusetts: Association for Computing Machinery, pp. 153–160. ISBN: 089791001X. DOI:

10.1145/582095.582120. URL: https://doi.org/10.1145/582095.582120.

Grajalez, C. G., E. Magnello, R. Woods, and J. Champkin (2013). "Great moments in statistics". In: *Significance* 10.6, pp. 21–28. DOI: 10.1111/j.1740-9713.2013.00706.x.

Hillis, W. D. (1985). "The Connection Machine". Hillis, W.D.: The Connection Machine. PhD thesis, MIT (1985). Cambridge, MA, USA: Massachusetts Institute of Technology. URL: http://hdl.handle.net/1721.1/14719.

Ho, T. K. (1995). "Random decision forests". In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. Vol. 1, 278–282 vol.1. DOI: 10.1109/ICDAR.1995.598994.

Hunt, E. B., J. Marin, and P. J. Stone (1966). *Experiments in Induction*. New York, NY, USA: Academic Press.

Ifrah, G. (1998). *The Universal History of Numbers, from Prehistory to the Invention of the Computer*. First published in French, 1994. London: Harvill. ISBN: 1 86046 324 x.

Kelleher, J. D. and B. Tierney (2018). *Data science*. The MIT Press.

Le Cun, Y. (1986). "Learning Process in an Asymmetric Threshold Network". In: *Disordered Systems and Biological Organization*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 233–240. ISBN: 978-3-642-82657-3.

Quinlan, J. R. (1986). "Induction of Decision Trees". In: *Machine Learning* 1, pp. 81–106. URL: https://api.semanticscholar.org/CorpusID:13252401.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). "Learning representations by back-propagating errors". In: *Nature* 323.6088, pp. 533–536. DOI: 10.1038/323533a0.

Schapire, R. E. (1990). "The strength of weak learnability". In: *Machine Learning* 5.2, pp. 197–227. DOI: 10.1007/BF00116037.

Song, J., H. V. Jagadish, and G. Alter (2021). "SDTA: An Algebra for Statistical Data Transformation". In: *Proc. of 33rd International Conference on Scientific and Statistical Database Management (SSDBM 2021)*. Tampa, FL, USA: Association for Computing Machinery, p. 12. DOI: 10.1145/3468791.3468811.

Takens, F. (2006). "Detecting strange attractors in turbulence". In: *Dynamical Systems and Turbulence, Warwick 1980: proceedings of a symposium held at the University of Warwick 1979/80*. Springer, pp. 366–381.

Vapnik, V. N. (1999a). "An overview of statistical learning theory". In: *IEEE Transactions on Neural Networks* 10.5, pp. 988–999. DOI: 10.1109/72.788640.

— (1999b). *The nature of statistical learning theory*. 2nd ed. Springer-Verlag New York, Inc. ISBN: 978-1-4419-3160-3.

Velleman, P. F. and L. Wilkinson (1993). "Nominal, Ordinal, Interval, and Ratio Typologies are Misleading". In: *The American Statistician* 47.1, pp. 65–72. DOI: 10.1080/00031305.1993.10475938.

Vincent, M. W. (1997). "A corrected 5NF definition for relational database design". In: *Theoretical Computer Science* 185.2. Theoretical Computer Science in Australia and New Zealand, pp. 379–391. ISSN: 0304-3975. DOI: https://doi.org/10.1016/S0304-3975(97)00050-9. URL: https://www.sciencedirect.com/science/article/pii/S0304397597000509.

Wickham, H. (2014). "Tidy Data". In: *Journal of Statistical Software* 59.10, pp. 1–23. DOI: 10.18637/jss.v059.i10. URL: https://www.jstatsoft.org/index.php/jss/article/view/v059i10.

Wickham, H., M. Çetinkaya-Rundel, and G. Grolemund (2023). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. 2nd ed. O'Reilly Media.

Zumel, N. and J. Mount (2019). *Practical Data Science with R*. 2nd ed. Manning.