ERVIN WILNEDER

Data Analytics

Concepts, tools and techniques, from statistics to the age of AI

SECOND EDITION · ENGLISH · 2026

Contents

Preface 4
Part I. Foundations 7
1. Concepts and definitions 8
2. Working environment 10
3. Programming languages 16
4. Data, units and variables of analysis 18
Part II. Exploratory analysis 20
5. Frequency tables 21
6. Describing variables 23
7. Data visualization 25
Part III. Inferential analysis 27
8. Data sources and collection 28
9. Data wrangling 30
10. Applied probability 33
11. Point and interval estimation 35
12. Hypothesis tests 37
13. Nonparametric statistics 40
14. Sampling techniques 42
15. ANOVA: analysis of variance 44
Part IV. Models 46
16. Linear regression 47
17. Logistic regression 49
18. KNN and CART 52
Part V. Data analysis in the age of AI 56
19. From copilot to colleague 57
20. Asking the data in natural language 60
21. Verifying what the machine says 63

22. Agents that analyze 67
23. Every number with its source 72
24. Validate before you believe 76
25. Text, images and audio as variables 80
26. Synthetic data and simulation 84
27. Privacy, bias and responsibility 88
28. From insight to action 92
Appendices. Appendices 96
A. Programming fundamentals 97
B. Copilots and chatbots 99

Preface

How this book began

This book began to take shape in early 2023 as course material for the subjects *Data Analysis I* and *II* and *Python* in the **University Diploma in Programming at UTN Haedo (National Technological University, Haedo)**, first taught that same year. The idea was simple: to gather, for each topic considered fundamental to data analysis, a summary of the ideas and conclusions worth highlighting, together with readings for going further.

That became the **first edition**: a path that runs from basic concepts and the working environment to inferential statistics and the first models, written for people starting out in the field who need, above all, a clear map of the terrain.

What changes in the second edition

Since then, the way of working with data has changed considerably. What in early 2023 was described as a novelty (copilots and chatbots) is now part of the daily routine of anyone who analyzes data. Yet the fundamentals have not lost their relevance: on the contrary, **they have become more important**, because they are what makes it possible to understand, direct and verify what those tools do.

For that reason, this **second edition** keeps the original content and adds to it:

- The chapters of the first edition remain, with a light revision of the writing. Each one opens with its summary and closes with a selection of readings that are still available; those that no longer exist were removed and those that moved were updated.
- The activities and assignments specific to the course were left out of this edition.
- The three chapters devoted to models (linear regression, logistic regression, KNN and CART) were brought together in a part of their own, **Models**.

- A new part is added, **Data analysis in the age of AI**, with chapters written in the same tone as the rest of the book and connected to the topics of the first edition.
- The chapter on copilots and chatbots is kept exactly as it was written in February 2023, as a record of that moment.

How to read this book

The chapters can be read in order or separately. Those of the first edition follow a progression: fundamentals, exploratory analysis, inferential analysis and models. Those of the new part return to these topics from the perspective of current tools, so it is best to reach them having gone through the fundamentals.

The readings favor open-access materials. When a reading refers to a printed book, the work and the pages are given, without a link.

Contact and license

For recommendations, corrections or other proposals, reach out through LinkedIn.

The text of this book is published under the Creative Commons Attribution 4.0 International (CC BY 4.0) license: it may be copied, adapted and shared, even for commercial purposes, provided the source is credited. Third-party works cited in the readings keep their own licenses.

Reference works

Some readings from the first edition refer to printed books. These are the full references:

- Aguirre, C., Niño, M. F. and Simonetti, E. F. *Estadística aplicada en las ciencias sociales y humanas. Estadística I* (in Spanish). Editorial Universitaria de la Universidad Nacional de Misiones, Cuadernos de Cátedra series, 2005.
- Beaulieu, A. *Learning SQL*. O'Reilly.
- Canavos, G. C. *Probabilidad y estadística. Aplicaciones y métodos* (in Spanish). McGraw-Hill.
- Kelmansky, D. M. *Estadística para todos: estrategias de pensamiento y herramientas para la solución de problemas* (in Spanish). Ministerio de Educación de la Nación, Instituto Nacional de Educación Tecnológica, 2009. Freely distributed: download from the INET website.

- Lind, D. A., Marchal, W. G. and Wathen, S. A. *Estadística aplicada a los negocios y la economía* (in Spanish), 15th edition. McGraw-Hill, 2012.

Part I. Foundations

1. Concepts and definitions

Summary

Data analysis is an activity integrated with others such as **data engineering and data science**, sharing concepts and tools with them and often overlapping with them in the daily work of many organizations. For this very reason, a professional in the field needs to recognize what falls **within the scope of data analysis** and how it relates to the rest of the **data lifecycle**.

This creates the need to define and understand fundamental concepts in order to draw that distinction. As often happens in other fields, several of these concepts are borrowed from English and used that way every day, which in some cases adds to the confusion. Moreover, definitions are not always unanimous: they depend on each author and evolve over time. This calls for **constant training, updating and reflection**.

As an activity, data analysis mainly requires **knowledge of statistics and of the domain** (or the business), so as to be able to work with the data, **draw conclusions** and translate those conclusions into **recommendations or actions** (or simply **produce information / knowledge** for later use).

The role does not necessarily involve **programming languages**, although (at the field's current stage) learning to use them is usually highly advisable. Without them, experience with at least one **visualization tool** (Tableau, Power BI, Looker) and advanced use of **spreadsheets** (Excel, Google Sheets) are essential.

Returning to what was said, some degree of **data transformation** is also commonly needed. In that respect, **Python, R and SQL** are very helpful tools, which of course can also be used later in the analysis itself. In fact, their use has become increasingly mandatory for working with **large volumes of data** and building **complex models**, which would be unfeasible to implement with tools such as those mentioned above.

After the analysis, presenting the results is extremely important, and can even set an analyst apart. In that sense, a well-rounded data analyst has

the **ability to produce reports and visualizations**, setting the technical side aside and showing the results of the analysis effectively and simply.

To sum up, data analysis has become a body of knowledge and techniques that, with the right tools and properly framed within a **data analytics process**, enables **decision-making** within organizations through the **exploration of and inference from data**.

Further reading

- ¿Qué es el análisis de datos? (in Spanish). Amazon Web Services.
- What is data analytics?. Oracle.

2. Working environment

Summary

One of the most common points of friction for anyone starting out in data analysis is the **overwhelming number of tools and working environments** available. Even once a choice is made, setting them up is rarely simple and the learning curve is slow, particularly for people without a background in programming. However, together with a clear understanding of theory and practice, **using the tools well is a key factor** in professional success.

So before starting on the tasks proper to a data analyst, it is necessary to survey the landscape of leading platforms, for each of the different concerns involved in sound development and work.

Starting with **spreadsheets**, these are the tools with the longest track record in the field. Although today there are countless complex tasks and volumes of data they cannot handle, they tend to appear in the final stretch of a project (making results available to end users), both because of their simplicity and because those same users are accustomed to them.

Then there are *notebooks*, generally associated with Jupyter, which are essentially **documents that can hold text and code** (Python, R, etc.), with a **cell-by-cell execution model**, which makes them the tool of choice for data scientists. Their use has since spread, and practically **any role involved with data understands how they work and what they offer**, to the point that they are considered something of a standard. Through different *kernels*, notebooks support several languages, although they are most often associated with Python. For those who program in R, Posit (formerly RStudio) is more comfortable, although it is now becoming a platform that supports more languages and can be an excellent choice as a working environment.

There are also other tools that copy the notebook model and/or extend it with new features. The range of options is quite wide, with *Colab* standing out for being free and offering generous processing capacity, and *JupyterLab*, an *IDE* that handles notebook management, plugin installation and

more. JupyterLab, in fact, has been adopted by several companies as the base for **much more powerful cloud services** such as GCP, AWS and Azure, forming data analytics *suites* that today's companies and organizations keep adopting.

To all of this must be added the tools designed exclusively for **data visualization**, which are now beginning to integrate querying capabilities (once attributable only to a **database client**) and data transformation (beyond **ETL tools** as such) within the same environment.

As may already be clear, this first look has brought quite a few new things to investigate and understand. What matters, in the end, is having enough **flexibility** to adopt, along the way, whatever tools are available at work or in particular projects. It should not be overlooked that there will be certain features common to all of them, as well as certain desirable properties (not only for the tools, but for any data analysis itself): **transparency, replicability, interoperability and scalability**.

With that in mind, another set of important tools comes into play. The main ones are *Git* for transparency (versioning and experimentation) and *Docker* / *Anaconda* for replicability (defining and managing dependencies). A piece of work that cannot be replicated or properly used by other collaborators, or even by end users, is worth nothing: in the era of **collaborative work** these tools are practically mandatory.

• UPDATE 2026

Reproducible environments

Since the first edition, the usual way of guaranteeing replicability has shifted. The first resource is no longer usually a container or a full package distribution, but a **lockfile**: an automatically generated record of the exact version of every library the project uses (including the dependencies of its dependencies), which makes it possible to rebuild the same environment on another machine with a single command. In Python, tools such as *uv* (which works from the `pyproject.toml` file) or *pixi* (built on the community packages of *conda-forge*) fill that role; in R, *renv* does.

In practice, reproducibility is best thought of as a ladder, where each rung is added only if needed:

- **Project file and lockfile**, versioned with Git alongside the code. This is the minimum rung and is enough for most analyses.
- **Container** (Docker, or *dev containers*, which usually rely on it under the hood) when the operating system and its libraries also need to be pinned, or when the analysis is deployed as a service. Docker is still valuable, but as a second step rather than a condition of entry.
- **Identified input data**: the cutoff date, version or *hash* of each file or table. The same code run on different data produces different numbers, and no locked environment prevents that. Tools such as DVC, mentioned above, handle this part.

There is also an issue that is not technical. Since 2020, Anaconda's terms of service have required a paid license to use its package repositories (the *defaults* channel) in for-profit organizations with more than 200 people. *conda* as a tool and the community channel *conda-forge* remain free to use. Recommending "Anaconda" without qualification can therefore create a licensing problem in a company; it is worth knowing which channel is in use.

The criterion for knowing whether an analysis is reproducible is simple and verifiable: **another person clones the repository, runs one command and gets the same results**. The cheapest test is to do it oneself in a clean environment, or to automate it so that it runs with every change.

Finally, AI assistants (see From copilot to colleague) readily write code that imports libraries, and may assume versions or functions different from those installed. A declared, locked environment makes those discrepancies surface explicitly at run time, rather than as a silently different result; it also gives the assistant precise context about which versions the project uses.

• UPDATE 2026

From the notebook to the team's data flow

Notebooks remain the tool of choice for **exploring**: trying out ideas, looking at data, iterating on a chart. What has changed is the place they hold in a team. As the definitive record of an analysis they have

well-known limitations: **hidden state** (cells can be run out of order, so the visible result may not match the visible code), **outputs embedded** in the file, which make diffs between versions in Git nearly unreadable, and the difficulty of testing and reusing what they contain.

In many organizations, the source of truth for analysis has moved to a **data warehouse** or a cloud *lakehouse*, with **transformations written in SQL and versioned** in Git. The usual pattern is to load the raw data and transform it inside the warehouse (known as ELT), through SQL models with tests and documentation; *dbt* is the most widespread tool for that task. Data is increasingly stored in **open columnar formats**, such as *Parquet*, and in table formats such as *Apache Iceberg*, which different engines can read without copying it. For work on a personal computer, *DuckDB* runs SQL directly on Parquet or CSV files, without installing a server.

The path data takes within a team usually looks like this:

- **Ingestion** from the sources (transactional systems, events, files, APIs).
- **Storage** in the warehouse or lakehouse.
- **Transformation** in versioned, tested SQL (see the update in Data wrangling).
- **Semantic layer and BI tools**, where metrics are defined only once (see Asking the data in natural language).
- **Notebook or report**, to explore, model and communicate on data that can already be trusted.

Within that pattern, a few practices greatly improve the use of notebooks: **restarting and running end to end** before sharing; **moving reused logic into modules or SQL models**; and considering text-based formats that version well, such as notebooks paired with scripts through *Jupytext*, *marimo* notebooks (saved as Python files) or *Quarto* documents.

The properties listed in this chapter (transparency, replicability, interoperability and scalability) are precisely what this pattern pursues. And AI assistants, which today write a good share of SQL queries (see From copilot to colleague), perform better when tables and metrics are defined, versioned and documented in a single place.

Further reading

Spreadsheets

- Tareas básicas en Excel (in Spanish). Microsoft Support.
- Google Sheets training and help. Google Workspace.

Jupyter Notebooks and similar platforms

- Python Language Basics, IPython, and Jupyter Notebooks. *Python for Data Analysis*, sections 2.1 and 2.2.
- Using Jupyter. Jupyter documentation.
- Google Colab
- Posit
- Visual Studio Code
- Deepnote
- DataLab (DataCamp)
- Observable

Data visualization and querying tools

- Power BI
- Tableau
- Metabase

Data analytics suites

- Introduction to machine learning on Gemini Enterprise Agent Platform. Google Cloud documentation.
- Amazon SageMaker
- What is Azure Machine Learning?. Microsoft documentation.

Versioning

- Git
- Hello World. GitHub documentation.
- DVC

Environment managers

- Anaconda

Containerization

- Docker
- El ecosistema de Docker: una visión general de la contenerización (in Spanish). DigitalOcean.
- ¿Qué es Docker? (in Spanish). Amazon Web Services.

● UPDATE 2026

- uv. Astral documentation.
- pixi. prefix.dev documentation.
- renv. Posit documentation.
- Writing your pyproject.toml. *Python Packaging User Guide*.
- conda-forge. conda-forge documentation.
- Terms of Service. Anaconda.
- Development Containers. The dev containers specification.
- Guide for Reproducible Research. *The Turing Way*.
- What is dbt?. dbt documentation.
- DuckDB. DuckDB documentation.
- Apache Parquet and Apache Iceberg. Apache Software Foundation.
- Jupytext, marimo and Quarto.

3. Programming languages

Summary

Among the skills required of a data analyst, the use of programming languages stands out today. As mentioned earlier, up to a point this role can do without them. In fact, many of the activities covered from here on can be carried out without them. However, for the sake of professional growth, and given the context of data volume and complexity, programming languages are essential in practical terms. This is especially so when returning to what was said about the desired qualities of a *well-done* data analysis (recall: transparency, replicability, interoperability and scalability), which programming languages can address without difficulty.

This book gives priority to Python, although any language can be used. In practice, some languages are preferred over others for certain tasks. **Python is quite general-purpose**: it serves data analysis as well as data engineering and data science (the leading *machine learning* libraries are written in Python), and is also used as a backend in web development, for *web scraping* and for other tasks.

On the other hand, **R is more closely tied to research and science**, with its origins in computational statistics. **Julia is an emerging language**, with a very interesting range of developments and the aim of combining the best of Python and R.

And finally, **JavaScript**, which like Python is used in different fields, even more of them, but in data analysis can be considered **a very good tool, especially for data visualization**.

There are, of course, several other languages, more *niche* and/or proprietary (not *open source*), such as Stata, Scala, SPSS, SAS and Matlab, among others. There are even lower-level languages such as Java, Rust and C/C++ that may be used in data analysis processes requiring exceptional *performance* (directly or through compilers).

SQL deserves a separate mention because, when it comes to using databases (for storing and querying data), it is the norm. That is, its use is unavoidable, and any job posting for data analysis roles is likely to list it as a hard requirement. Different implementations of this language can be found depending on the company or organization promoting its own database. They usually all share a common set of *commands* and then another set of particular commands and functions / syntax, which is quickly picked up with practice.

Further reading

Python

- Python for Data Analysis. Wes McKinney.
- CS50′s Introduction to Programming with Python. Harvard University.
- Intro a Python (in Spanish). Instituto Humai's repository of Python courses.
- Programación en Python (in Spanish). Repository of the UNSAM Python course.
- Real Python
- Aprendiendo Python: curso de Python 2020 desde cero (in Spanish). YouTube playlist.

Other languages

- R for Data Science (2nd edition). Hadley Wickham, Mine Çetinkaya-Rundel and Garrett Grolemund.
- *Learning SQL*. Alan Beaulieu, O'Reilly.
- Query syntax. BigQuery, Google Cloud documentation.
- Observable documentation (JavaScript).

4. Data, units and variables of analysis

Summary

Data is the basic input for data analysis. A data point is the **smallest unit of information**, the raw material from which **knowledge** is produced and evidence-based action can be taken with a certain **measurable degree of uncertainty**.

It is useful to think of a data point as whatever can be placed inside a cell (of a spreadsheet): a number, a letter, a word, a true or false. Whether *something* counts as a data point depends on the analysis itself, because that data point arises from the intersection of 2 very important concepts: **the unit of analysis and the variable**.

The unit of analysis is the **object or subject** on which the analysis is carried out. The variable is the **characteristic of interest** of that object or subject, on which a data point will be observed or measured. The former is usually identified with each of the rows (of a spreadsheet), the latter with the columns (see table), although this need not always be the case. The opening sequence of any data analysis is therefore to **identify the unit of analysis and the variable or variables of study, and to obtain one or more data points from those variables**.

	variable 1	variable 2	variable 3
unit 1	data point	data point	data point
unit 2	data point	data point	data point
unit 3	data point	data point	data point

Variables can be of 2 types: **quantitative or qualitative**. The former can in turn be continuous or discrete, and the latter ordinal or nominal. The importance of recognizing which type of variable is at hand lies in **antici-**

pating the techniques and limitations that apply in each case. The most flexible and powerful variables are continuous quantitative ones, because they can be discretized, ordered or turned into qualitative ones, whereas the reverse is not possible.

In addition, classifying variables can involve not only their type but also **whether or not they result from a random process**, that is, whether chance plays a part at some point in obtaining them. This brings a complete change in the way the variable is analyzed, as well as in the conclusions that can be drawn.

To sum up, it is very important for a data analyst to take part in, or be fully aware of, the thorough definition of the unit of analysis, the identification of the variables and the collection and organization of the data, since these 3 points are the foundation on which any analysis is entirely built.

Further reading

- Los datos. Aguirre, Niño and Simonetti, *Estadística aplicada en las ciencias sociales y humanas* (in Spanish), pp. 13–15.
- Las variables. Aguirre, Niño and Simonetti, *Estadística aplicada en las ciencias sociales y humanas* (in Spanish), pp. 15–17.
- La primera organización de los datos: la matriz de datos. Aguirre, Niño and Simonetti, *Estadística aplicada en las ciencias sociales y humanas* (in Spanish), pp. 25–27.
- Tidy Data. Hadley Wickham, *Journal of Statistical Software*, sections 1, 2 and 3.
- Tipos de variables. Lind, Marchal and Wathen, *Estadística aplicada a los negocios y la economía* (in Spanish), 15th ed., pp. 8–9.
- Variables aleatorias. Lind, Marchal and Wathen, *Estadística aplicada a los negocios y la economía* (in Spanish), 15th ed., pp. 189–190.
- Types of Data. *STAT 414: Introduction to Probability Theory*, Penn State, lesson 1.4.
- El concepto de variable aleatoria. Canavos, *Probabilidad y estadística. Aplicaciones y métodos* (in Spanish), pp. 52–53.

Part II. Exploratory analysis

5. Frequency tables

Summary

The most basic way to analyze / summarize a dataset is a frequency table. It involves the **operation of counting units of analysis or their variables, grouped by some criterion into mutually exclusive (non-overlapping) categories or classes**. For qualitative variables this is usually quite direct, since the categories are naturally identified; for quantitative ones it depends on the analysis, which may **group by intervals of equal or different size**, as needed.

The specific result of the count for each class or interval is called the *frequency*, and the set of these is known as the *frequency distribution*. The frequency can be *absolute* or *relative* (if it is divided by the total number of units or the total for the variable), the latter giving a better idea of how *relevant* a certain class is with respect to the total and also allowing comparison with other variables. The sum of the column of absolute frequencies must always equal the total count of units, while the relative frequencies must always add up to 1.

The frequency table can thus be built initially as follows: one column with the classes / categories / intervals, one column with the absolute frequencies and another column with the relative ones. These last 2 are usually accompanied by a row with the total, so as to verify what was said above. On top of this basic layout, a column can be added with the class midpoint (usually the sum of the maximum and minimum values of an interval, divided by 2) or the upper limit, which helps identify an interval quickly without resorting to its definition by values (both maximum and minimum), and also supports a later visualization. In addition, another column can be used for the class width or range (the difference between the maximum and minimum values).

Answering some **analysis questions** may also require calculating **cumulative frequencies** (only for variables that can be ordered). Essentially, once

the table has been sorted by some criterion, the absolute and/or relative frequencies are added up row by row. Summing these new columns makes no sense, but it must be checked that the last row of the running sum equals the total absolute or relative frequency, depending on which was used.

Although this can be done by hand, with large datasets it is possible to use *pivot tables* or some implementation specific to frequency tables (for example, the FREQUENCY function in Google Sheets).

To close on the construction of the frequency table, the most important point is its flexibility and reach for any dataset. **Building one is recommended in every analysis**, for every variable, since it may reveal problems in them or even yield some first conclusions. Frequency tables will also be needed for some specific types of analysis covered later.

Further reading

- Construcción de una tabla de frecuencias. Lind, Marchal and Wathen, *Estadística aplicada a los negocios y la economía* (in Spanish), 15th ed., pp. 23–24.
- Construcción de distribuciones de frecuencias: datos cuantitativos. Lind, Marchal and Wathen, *Estadística aplicada a los negocios y la economía* (in Spanish), 15th ed., pp. 29–34.
- Problema de trabajo e investigación estadística. Aguirre, Niño and Simonetti, *Estadística aplicada en las ciencias sociales y humanas* (in Spanish), pp. 12–13.
- El análisis de la matriz de datos. Aguirre, Niño and Simonetti, *Estadística aplicada en las ciencias sociales y humanas* (in Spanish), p. 28.
- Las distribuciones de frecuencias en el análisis univariado. Aguirre, Niño and Simonetti, *Estadística aplicada en las ciencias sociales y humanas* (in Spanish), pp. 29–45.
- Frecuencia, tablas de frecuencia y niveles de medición (in Spanish). *Introducción a la estadística*, OpenStax, section 1.3.

6. Describing variables

Summary

In the process of characterizing / summarizing variables, **exploratory data analysis**, measures of 3 fundamental aspects come into play: **location, dispersion and association**.

Location essentially shows **where most of the data lies**, and dispersion shows whether the data making up that majority is *close* together. When dispersion increases, measures of location do not really say much (even if they seem to); for this very reason, both must always be present. Precisely because dispersion affects location, there is a set of possible measures that are more *robust* to a situation of high dispersion or to the particular dispersion of some data points. The most widely used is the median, which, unlike the average (mean), holds firm against **extreme values**.

Dispersion can also be misleading, because it depends on the units in which it is measured. For this reason, the coefficient of variation is recommended: it expresses the standard deviation relative to the value of the mean. As its name indicates, it is a coefficient and has no units. A value below 0.3 (30%) indicates low dispersion and, therefore, the measures of location will be *reliable*.

Even so, the next chapter will show that, even in this last situation, **relying only on measures of location is not 100% safe**. **Visualization** must always be used to understand the behavior of a variable or dataset, together with a frequency table.

Beyond analyzing a single variable on its own, there may also be a need to compare it with another, or even with itself under other circumstances (or at another time). For that there are measures of association, which are common in all kinds of reports. This chapter does not cover the correlation coefficient, which deserves a fuller treatment of its use and implications, given later.

Further reading

Measures of location: mean, median, quantiles and mode

- Medidas resumen (in Spanish). Kelmansky, *Estadística para todos*, pp. 120–125.
- Los valores que caracterizan al conjunto de datos. Aguirre, Niño and Simonetti, *Estadística aplicada en las ciencias sociales y humanas* (in Spanish), pp. 59–79.
- Descripción de datos. Lind, Marchal and Wathen, *Estadística aplicada a los negocios y la economía* (in Spanish), 15th ed., pp. 57–75.
- Medidas numéricas descriptivas. Canavos, *Probabilidad y estadística. Aplicaciones y métodos* (in Spanish), pp. 11–14.
- Medidas del centro de los datos (in Spanish). *Introducción a la estadística*, OpenStax, section 2.5.

Measures of dispersion: standard deviation, coefficient of variation and range

- Medidas resumen (in Spanish). Kelmansky, *Estadística para todos*, pp. 125–135.
- Análisis de la variación y asimetría. Aguirre, Niño and Simonetti, *Estadística aplicada en las ciencias sociales y humanas* (in Spanish), pp. 85–97.
- Descripción de datos. Lind, Marchal and Wathen, *Estadística aplicada a los negocios y la economía* (in Spanish), 15th ed., pp. 75–84.
- Medidas numéricas descriptivas. Canavos, *Probabilidad y estadística. Aplicaciones y métodos* (in Spanish), pp. 15–22.
- Medidas de la dispersión de los datos (in Spanish). *Introducción a la estadística*, OpenStax, section 2.7.

Measures of association: ratios, rates and proportions

- Razón, tasas y porcentajes (in Spanish). Kelmansky, *Estadística para todos*, pp. 15–19.

7. Data visualization

Summary

Within data analysis, the person who builds visualizations may hold a separate role. It is an area that **involves technical knowledge not only of data but of design**. Without this specific training, it is often acquired through experience, and even then, in a presentation, content and form **will depend on who receives the information**. The advisable approach is to **start from a minimal, replicable design and take cues from other presentations**, and then to complement and improve it as the basic principles of design, color and even trends are absorbed.

As for the data, the visualizations will be determined by the **number of variables and their type**. It is usually a good habit to consult a *visualization gallery* to check which chart fits what is available or what needs to be shown (for example, Python Graph Gallery). The visualization should always be adapted to the **question it must answer**, avoiding overload / repetition of data, and accompanied by the corresponding title, labels and legend and, where needed, a brief, plain-language explanation.

That said, visualizations can be built with tools designed for that purpose (Tableau, Power BI, Looker), with more generic presentation tools (PowerPoint, Google Slides) or with Python / R / JavaScript (through the appropriate libraries). In that respect, the advisable approach is to **adopt one in particular and go deep into its use**.

Further reading

Visualizing amounts, proportions and distributions

- Fundamentals of Data Visualization. Claus O. Wilke, part I: chapters 2, 5, 6, 7 and 10 (supplemented with the rest of the usual bibliography if needed).
- Exploratory Visualizations. *Feature Engineering and Selection: A Practical Approach for Predictive Models*, chapter 4.

- Data Visualization. *Python for Everybody*, chapter 17.

Basics of storytelling and effective presentations

- Fundamentals of Data Visualization. Claus O. Wilke, part III: chapters 28 and 29.
- Look at Data. *Data Visualization: A Practical Introduction*, Kieran Healy, chapter 1.

Part III. Inferential analysis

8. Data sources and collection

Summary

Where data come from and how they are collected determines the conclusions that can be drawn from them. Having a **census** (a complete survey of the units of analysis) is not the same as having a **sample** (a partial survey); the latter carries the **uncertainty inherent in not having all the data**. Yet that is precisely the setting where data analysis plays its largest part.

A sample is usually the most feasible way to obtain data. Even in the era of *big data*, however much data there is, it tends to be a partial representation of reality, of the processes that generate it. The sample is the rule, not the exception.

Sampling is a procedure by which **a set of units of analysis is drawn, at random, from a larger (finite or infinite) set of units**. In its simplest version it consists of drawing them directly, although this may not be the best way to do it. Later chapters present other methods that, in essence, seek to improve the **representativeness of the data obtained** with respect to the whole set, or equivalently, to reduce **errors due to sampling itself**. Randomness plays a fundamental role here: it is the channel through which every other kind of **error due to subjectivity** is meant to be eliminated, while **errors from operational issues** must also be avoided. It is also worth noting that sampling is not only a technique applied to *real* data; samples can also be obtained from a simulation or from *synthetic data*, which is useful for experimentation.

A sampling procedure (from here on always understood as random) necessarily yields random variables; the goal is to characterize them while taking their random nature into account. Because **each time a sample is taken, the results of a mean or a standard deviation (or any other measure) for a given variable will differ**. Every single time, it will be different. These results

can themselves be treated as new variables and characterized as well. This is precisely one of the great pillars of data analysis.

To recap, what matters from here on is the ability to deal with a sample and draw conclusions about the variables from it. The following chapters introduce the theory and concepts of probability, which will be of great help.

Further reading

- Origen de los datos (in Spanish). Kelmansky, *Estadística para todos*, pp. 51–52.
- Muestreo (in Spanish). Kelmansky, *Estadística para todos*, pp. 29–35.
- Fuentes de datos. Aguirre, Niño and Simonetti, *Estadística aplicada en las ciencias sociales y humanas* (in Spanish), pp. 18–20.
- Métodos de muestreo. Lind, Marchal and Wathen, *Estadística aplicada a los negocios y la economía* (in Spanish), 15th ed., pp. 266–268.
- Muestras aleatorias. Canavos, *Probabilidad y estadística. Aplicaciones y métodos* (in Spanish), pp. 214–217.
- Métodos de recolección de datos para una investigación (in Spanish). Mariela Torres, Karim Paz and Federico G. Salazar, Boletín Electrónico No. 3, Faculty of Engineering, Universidad Rafael Landívar.

9. Data wrangling

Summary

Data wrangling involves **joining, transforming, aggregating and/or filtering data**, not necessarily in that order. A common saying holds that *80% of a data analyst's time goes to tasks related to this*, and it may well not be a bad estimate.

Detecting **missing values** (which will not always mean the same thing), **outliers** (which will not necessarily always be outliers) and **duplicate records** (which will not necessarily always be wrong to have duplicated) is only the first step in a series of **tasks that are complex both technically and conceptually**, and that do take a great deal of time. Added to this is the need to **integrate different data sources**, performing join and concatenation operations between datasets (or tables in a database), with filtering and/or aggregation of the data before or after, not only to summarize the information or to work with different units of analysis, but perhaps also for the **creation of new variables** and/or the transformation of existing ones (known as *feature engineering*, which deserves a whole chapter of its own).

The key point is that **the process itself (every operation performed) must be known and understood by the data analyst**, because the results of the analysis, especially when working with inference, depend heavily on what was done, including of course the origin of the data itself.

In the best case, within companies and organizations with well-established analytics processes, the role of the **data engineer** tackles these aspects and makes the information available as needed. However, even when this holds, some degree of data wrangling will inevitably be required.

● UPDATE 2026

Data contracts and data tests

The 80% saying has no verifiable source, and what matters is not the exact proportion but that this work determines the validity of everything that follows. What has changed since the first edition is **where** it is done and **how it is controlled**. In teams of some maturity, much of the cleaning and joining of tables has moved to versioned SQL transformations, owned by a role between engineering and analysis (*analytics engineering*). And AI assistants now write a significant share of transformation code (see From copilot to colleague), so the bottleneck has shifted from writing the transformation to **verifying that it is correct**.

Data tests

A **data test** is an assertion about a table that is checked automatically every time the data are updated. The most common are:

- **Uniqueness** of the key (each order appears only once).
- **Absence of nulls** where there should be none.
- **Accepted values** for categorical variables (the possible statuses of an order).
- **Referential integrity**: every value of a foreign key exists in the reference table.
- **Ranges and freshness**: non-negative amounts, plausible dates, data updated as of the expected date.

dbt ships the first four as built-in generic tests; in Python, libraries such as *pandera* or *Great Expectations* allow declaring similar expectations on a DataFrame. An example tied directly to this chapter: a join between tables with duplicate keys multiplies rows without raising any error, and a uniqueness test after the join catches it before total sales show up inflated in a report.

Data contracts

A **data contract** is an explicit agreement between whoever produces a dataset and whoever consumes it, covering its schema (columns and types), the meaning of each field and the quality expectations. Its value is that a change that breaks the agreement (a renamed column,

a changed type) fails on the producer's side, instead of surfacing weeks later as a strange number in an analysis. There are concrete implementations, such as *dbt* model contracts, and open specifications, such as the *Open Data Contract Standard*.

The tools

pandas remains the reference and the basis of this chapter's readings; *Polars* and *DuckDB* are alternatives when the volume grows. The checks described apply to any of them. It should be added that an AI assistant can propose tests from a profile of the data, but **what counts as a valid value is a domain decision**, not the tool's. Tests are, in the end, the way to check generated code against explicit expectations instead of reviewing it line by line (see Verifying what the machine says).

Further reading

- Python for Data Analysis. Wes McKinney, ch. 7, 8 and 10.
- Encoding Categorical Predictors. *Feature Engineering and Selection: A Practical Approach for Predictive Models*, ch. 5.
- Engineering Numeric Predictors. *Feature Engineering and Selection: A Practical Approach for Predictive Models*, ch. 6.
- Handling Missing Data. *Feature Engineering and Selection: A Practical Approach for Predictive Models*, ch. 8.
- A Beginner's Guide to Clean Data.

● UPDATE 2026

- Add data tests to your DAG. dbt documentation.
- Model contracts. dbt documentation.
- Open Data Contract Standard. Bitol project.
- pandera. pandera documentation.
- Great Expectations. Great Expectations documentation.

10. Applied probability

Summary

Probability is a fundamental tool for data analysis, not so much for its direct use but because **it is the mathematical foundation of many proofs of the procedures and formulas commonly applied**. Knowing its basic concepts and definitions is therefore a door to a better understanding of both the analysis and the conclusions drawn.

It is particularly useful to absorb the most relevant distributions that arise from the analysis of random variables: **the binomial distribution, the normal distribution and the chi-square distribution**. Each is associated with the study of the measures of location, dispersion and association seen earlier. That is, these measures, under the right conditions, have a **theoretically defined and proven behavior**, which makes their analysis far easier. Knowing how to compute **joint and conditional probabilities**, common in multivariate analysis, is also useful.

Further reading

- Epílogo: estadística y probabilidad (in Spanish). Kelmansky, *Estadística para todos*, pp. 245–247.
- Estudio de los conceptos de la probabilidad. Lind, Marchal and Wathen, *Estadística aplicada a los negocios y la economía* (in Spanish), 15th ed., pp. 144–151.
- Distribuciones de probabilidad discreta. Lind, Marchal and Wathen, *Estadística aplicada a los negocios y la economía* (in Spanish), 15th ed., pp. 186–202.
- Distribuciones de probabilidad continua. Lind, Marchal and Wathen, *Estadística aplicada a los negocios y la economía* (in Spanish), 15th ed., pp. 222–242.
- Probability. *STAT 500: Applied Statistics*, Penn State, lesson 2.
- Probability Distributions. *STAT 500: Applied Statistics*, Penn State, lesson 3.

- Sampling Distributions. *STAT 500: Applied Statistics*, Penn State, lesson 4.
- Introducción a conceptos de probabilidad y estadística descriptiva (in Spanish). *Laboratorio de Datos*, first term 2021, UBA: class 4.
- Temas de probabilidad (in Spanish). *Introducción a la estadística*, OpenStax, ch. 3.

11. Point and interval estimation

Summary

To estimate is to approximate. Since, for operational or conceptual reasons, not all the information is available, some characteristics of the variables will inevitably be unknown; these can be any of the measures of location, dispersion or association seen earlier. Their true value is what is called a **parameter**.

The goal is precisely to approximate these parameters through different formulas, called **estimators**. For each case there is no single possible estimator, although some meet **certain properties that make them optimal**, or at least better than others.

Data are needed to reach an approximate value of the parameters through the estimators. This is where the sampling process comes in: **obtaining data, necessarily at random,** so that the estimation yields valid results. The larger and more representative a sample is with respect to the population, the closer the values of the estimators get to the values of the parameters, always differing from one sample to the next but with less variation as the sample size approaches the full number of units of analysis (or infinity).

The **variation in the results of the estimators** plays a fundamental role in managing uncertainty. It is common not to state a single value for the estimator but rather an **interval**. It can then be said that this interval, with a certain degree of confidence, will contain the true value of the parameter a given number of times (number of samples or experiments).

All the results obtained are subject to the **assumption that the population (of units of analysis) follows a given distribution**. When trying to estimate the mean, the variance or a proportion, it must be kept in mind that these are precisely parameters used in particular distributions, some of those

seen earlier, such as the normal or the binomial. This, in a way, makes the calculations easier and gives greater certainty about the conclusions; but in many cases the distributions are not known, or some of these parameters are not the target of estimation. In that case, **nonparametric estimation methods** are needed, which are covered later.

Further reading

- Estimación y parámetros (in Spanish). Kelmansky, *Estadística para todos*, p. 55.
- Variabilidad entre muestra y muestra (in Spanish). Kelmansky, *Estadística para todos*, pp. 58–63.
- Teorema central del límite (in Spanish). Kelmansky, *Estadística para todos*, pp. 200–207.
- Distribuciones de muestreo de estadísticas. Canavos, *Probabilidad y estadística. Aplicaciones y métodos* (in Spanish), pp. 218–238.
- Estimación por intervalos (in Spanish). Kelmansky, *Estadística para todos*, pp. 214–228.
- Estimación e intervalos de confianza. Lind, Marchal and Wathen, *Estadística aplicada a los negocios y la economía* (in Spanish), 15th ed., pp. 297–315.
- Propiedades deseables de los estimadores puntuales. Canavos, *Probabilidad y estadística. Aplicaciones y métodos* (in Spanish), pp. 251–261.
- Confidence Intervals. *STAT 500: Applied Statistics*, Penn State, lesson 5.

12. Hypothesis tests

Summary

Starting from an estimator that has already been computed (that is, an approximation of some characteristic of a variable has been obtained), the aim is often to **check the result against some assumption about the true value**, or perhaps against a value taken as given. This **putting of what is known to the test** is called a **hypothesis test**. A hypothesis test, **using the evidence (data) collected**, seeks to reject (or not) what is held to be true so far. It is worth noting that this is precisely how science works in conceptual terms; **hypothesis tests are therefore the tool par excellence of scientific research**.

As with the computation of estimators, the result of a hypothesis test depends on the quality of the data obtained and on the representativeness of the sample. The data must be obtained at random, at least at some point in the overall procedure, and are also subject to whether or not the variable being tested is assumed to follow a given distribution. If it is not, then, just as with estimation, nonparametric methods are used.

In addition, statistical tests vary depending on whether there are one or more samples, one or more variables, independent or paired variables (for example, the same variable at two different points in time), known or unknown dispersion, among other things. It is a truly extensive battery that can hardly be memorized, so it is advisable instead to learn to **recognize the scenarios in which a test can be applied**, and then look into which one would be most appropriate. In that sense, the approach is similar to that of data visualization, but it calls for much more caution because, being **a critical tool for drawing conclusions**, it can lead to poorly grounded actions.

• EXTENSION

Statistical significance and practical significance

Hypothesis tests are a central tool of scientific research, though not the only one, and their use in isolation has drawn growing criticism. In 2016, the American Statistical Association published a statement of principles on the use of the p-value, among them: that it **does not measure the probability that the hypothesis is true**; that it **does not measure the size of an effect or the importance of a result**; and that conclusions and decisions **should not be based only on whether the p-value passes a threshold**. Several practices now considered basic follow from this.

Report the effect and its interval

The main result of a comparison is **the effect size**: the estimated difference in the units of the problem (pesos, percentage points, minutes) or a standardized measure (such as Cohen's *d*), accompanied by its **confidence interval** (see Point and interval estimation). The p-value complements it; it does not replace it. This is especially important with today's volumes: with very large samples, differences that are negligible in practice turn out statistically significant; with small samples, relevant effects may not. That is why it pays to define, **before looking at the data**, the smallest difference that would have practical consequences.

Many tests, many false positives

At a 5% significance level, roughly one in every twenty comparisons with no real effect will come out significant by chance. When testing many metrics, segments or variants, finding "something significant" is almost certain. There are corrections for multiple comparisons (Bonferroni, Holm) and procedures that control the expected proportion of false discoveries (Benjamini-Hochberg). An equivalent problem arises from **looking at the results repeatedly** while data are being collected and stopping when it is convenient.

Plan before analyzing

- Set the hypothesis, the primary metric and the analysis before seeing the data (in research, through preregistration).

- Compute the **power** and the sample size needed to detect the smallest relevant difference.
- Report **all** the tests performed, not only the significant ones.

These precautions carry more weight with AI assistants, which can run dozens of tests in seconds and return only the ones that "worked". Knowing how many analyses were run, and why, remains the analyst's responsibility (see Verifying what the machine says). How all this applies to controlled experiments in digital products is taken up again in From insight to action.

Further reading

- Decisiones en el campo de la estadística (in Spanish). Kelmansky, *Estadística para todos*, pp. 231–241.
- Pruebas de hipótesis de una muestra. Lind, Marchal and Wathen, *Estadística aplicada a los negocios y la economía* (in Spanish), 15th ed., pp. 333–346.
- Pruebas de hipótesis estadísticas. Canavos, *Probabilidad y estadística. Aplicaciones y métodos* (in Spanish), pp. 303–346.
- Hypothesis Testing for One-Sample Proportion. *STAT 500: Applied Statistics*, Penn State, lesson 6, part A.
- Hypothesis Testing for One-Sample Mean. *STAT 500: Applied Statistics*, Penn State, lesson 6, part B.
- Pruebas de hipótesis con una muestra (in Spanish). *Introducción a la estadística*, OpenStax, ch. 9.

● EXTENSION

- The ASA's Statement on p-Values: Context, Process, and Purpose. Wasserstein and Lazar, *The American Statistician*, 70(2), 2016.
- Improving Your Statistical Inferences. Daniël Lakens. Open book on p-values, effect sizes, power and multiple comparisons.

13. Nonparametric statistics

Summary

First and foremost, many hypotheses go beyond the simple question of whether one (estimated) value equals another, as seen in the previous chapter. For example, one may want to check (reject or not) whether a variable follows a certain distribution, without needing to know or estimate its parameters. Along similar lines, one may want to confirm (reject or not) whether a variable is independent of another, or even whether it is random, which are assumptions in a large share of analyses. Since the aim is neither to obtain nor to use or define a parameter of interest (at least not one associated with a known distribution, such as the median), a battery of methods / tests called nonparametric arises.

As with parametric tests, the situations in which they can be applied are countless, and it is preferable to have the ability to recognize when (or when not) their use is appropriate and under what conditions. It should be noted that these tests, very much at home in the field of statistics, **tend to be implemented more completely or in more detail in the R language**, and using it is advisable since more documentation and academic information is available.

In recent years, alongside data science and the growth in computing power, **one nonparametric method in particular has become quite popular**: the *bootstrap*. In essence, it involves resampling (that is, sampling from a sample) with replacement, so as to generate a set of samples larger than the one initially available (possibly a single one). For each sample an estimator is obtained (in the usual way already seen), and the resulting distribution of estimated values is precisely the added value of this method, something that would otherwise be obtained through real samples (operationally impossible in most cases). It is important to note that, with this extremely simple procedure, **any parameter or characteristic of interest of a variable can be estimated**. All it takes is one sample, whose quality and representativeness become crucial, and a support for the resampling

(a software simulation). Bootstrapping is implemented in Python / R, very well documented and with many examples, given how often it is used today.

Further reading

Test on the median

- Prueba de los signos. Lind, Marchal and Wathen, *Estadística aplicada a los negocios y la economía* (in Spanish), 15th ed., pp. 681–685.
- Inference for the Population Median. *STAT 500: Applied Statistics*, Penn State, lesson 11.1.

Tests on distributions (goodness of fit / independence / randomness)

- Métodos no paramétricos: pruebas de bondad de ajuste. Lind, Marchal and Wathen, *Estadística aplicada a los negocios y la economía* (in Spanish), 15th ed., pp. 648–658.
- Prueba de hipótesis de que la distribución de datos proviene de una población normal. Lind, Marchal and Wathen, *Estadística aplicada a los negocios y la economía* (in Spanish), 15th ed., pp. 659–664.
- Análisis de tablas de contingencia. Lind, Marchal and Wathen, *Estadística aplicada a los negocios y la economía* (in Spanish), 15th ed., pp. 667–670.
- La prueba de bondad de ajuste chi-cuadrada. Canavos, *Probabilidad y estadística. Aplicaciones y métodos* (in Spanish), pp. 363–368.
- La estadística de Kolmogorov-Smirnov. Canavos, *Probabilidad y estadística. Aplicaciones y métodos* (in Spanish), pp. 368–370.
- Chi-Square Test for Independence. *STAT 500: Applied Statistics*, Penn State, lesson 8.
- La distribución chi-cuadrado (in Spanish). *Introducción a la estadística*, OpenStax, ch. 11.

Bootstrapping

- Introduction to Bootstrapping. *STAT 500: Applied Statistics*, Penn State, lesson 11.2.
- Bootstrapping Main Ideas!!!. StatQuest, YouTube.

14. Sampling techniques

Summary

Real-world data collection rarely allows a simple random sample to be carried out. For example, issues such as the **operational costs** of covering a large territory are common in any social statistics study. This is why different sampling techniques have emerged, along with estimators associated with them, so that the approximation to the true population parameters is as good as possible (which is the goal of any estimator).

That said, there is a very important initial consideration when sampling begins: the **sample size**. If it is too small, representativeness may not be adequate (with everything that implies, as noted earlier); if it is too large, it is essentially an unnecessary expense of resources. Determining an adequate sample size is therefore the starting point of a good data analysis, which is then tied to the technique used for sampling.

In that regard, 2 resources stand out: the use of **strata** and the use of **clusters**.

The former (strata) are meant for studying *regions* or groupings of interest to the research, obtaining statistics not only at the overall level but within each one. It is a division that can be made on the population (the whole set of units of analysis) before or after sampling (post-stratification), on the condition that the complete set of strata is studied, since each stratum is likely a partial representation of the population. A random sample is drawn from each stratum and then studied together with the rest. An example of strata could be the neighborhoods of a city.

The latter (clusters) are groupings of units of analysis that tend to occur naturally together, with the particularity that each is a representation of the population in itself, and that they also make the study / sampling easier. Clusters are not sampled internally; instead, all the units within them are surveyed. The sampling unit is therefore the cluster, and it is the clusters

that are selected, usually systematically. An example of clusters could be the passengers on a flight.

Further reading

- Métodos de muestreo. Lind, Marchal and Wathen, *Estadística aplicada a los negocios y la economía* (in Spanish), 15th ed., pp. 266–271.
- Elección del tamaño adecuado de una muestra. Lind, Marchal and Wathen, *Estadística aplicada a los negocios y la economía* (in Spanish), 15th ed., pp. 316–321.
- Estimating Population Mean and Total under SRS. *STAT 506: Sampling Theory and Methods*, Penn State, lesson 1.
- Confidence Intervals and Sample Size. *STAT 506: Sampling Theory and Methods*, Penn State, lesson 2.
- Stratified Sampling. *STAT 506: Sampling Theory and Methods*, Penn State, lesson 6.
- Cluster and Systematic Sampling. *STAT 506: Sampling Theory and Methods*, Penn State, lessons 7 and 8.
- Multi-Stage Designs. *STAT 506: Sampling Theory and Methods*, Penn State, lesson 9.

15. ANOVA: analysis of variance

Summary

In practice, comparing means (through statistical tests) usually involves **more than two categories, groups and/or variables**. Consider, for example, an experiment with 6 different treatments: the outcome of each will vary to a greater or lesser extent, and the researcher will try to understand that **variation**, concluding whether one or more of them really differ from the rest in their effects. The analysis of that variation is precisely what is known as ANOVA (analysis of variance).

The technique is very simple in essence. It compares the **variation within each group** with the **variation between groups**. If the latter is *considerably* large (and the former considerably small), it can be said that the groups are different. That said, depending on whether different variables (factors) that affect the outcome are to be tested, ANOVA as such can become quite complex. Whatever the scenario, it is very important to keep in mind what introducing more components into the analysis implies, a key consideration for drawing valid conclusions. An important case in this respect is the use of **blocking factors**, which are deliberate arrangements in the experiment made to control the variation of the response.

After the conclusions of the ANOVA, other **procedures are usually carried out to determine specifically which treatment / group differs from the rest**. This is what is known as *post hoc* analysis.

ANOVA, being a statistical test, is a procedure widely used in scientific research; but it is worth stressing that it should not be seen simply as a test, but as a way of **estimating a continuous variable as a function of one or more categorical variables**, which is broadly called a *regression* (a topic covered in the next chapter). It is also important to note that analysis of variance, as presented here, is a **parametric method**, with a nonparametric

counterpart for cases where the assumptions are not met (or where a parameter other than the mean is investigated).

Further reading

- Análisis de la varianza. Lind, Marchal and Wathen, *Estadística aplicada a los negocios y la economía* (in Spanish), 15th ed., pp. 410–440.
- Introduction to ANOVA. *STAT 500: Applied Statistics*, Penn State, lesson 10.
- Distribución F y análisis de varianza (ANOVA) de una vía (in Spanish). *Introducción a la estadística*, OpenStax, ch. 13.

Part IV. Models

16. Linear regression

Summary

Linear regression is a model in very wide use, applied to the **estimation of a continuous variable from continuous and/or qualitative (binary) variables**. It first involves estimating certain **parameters associated with each independent variable**, which make up a mathematical expression from which, once those values are estimated, an approximation of the dependent variable is obtained.

For this approximation, it will be relevant to evaluate how *close* it is to the values actually observed and whether the **residuals** resulting from that evaluation show any particular behavior, which is a sign of how *good* the model is, as it seeks to be as complete a representation as possible of the true model underlying the data.

It should be noted that, like every model, it is a **mathematical simplification of reality**, so it will not necessarily reach the expected results. The estimates may be far from the observed values, possibly because other variables with an impact are being left out, or because the nature of the phenomenon itself is not **linear in its parameters** (an incorrect specification of the model), among several other reasons. Returning to what was said at the beginning, it is a widely used model, but that does not mean it is used correctly. **The data may present a set of problems** such as multicollinearity, heteroscedasticity or autocorrelation, which must be addressed properly; doing so saves time in concluding that the linear regression model will not, in fact, be adequate.

Very complete implementations exist in both Python and R, in multiple libraries. In data science / machine learning, *scikit-learn* (sklearn) is commonly used, and it is even considered a standard for the way it structures model building. This is precisely the library used from here on for the rest of the chapters.

Further reading

General concepts

- Relación entre variables (in Spanish). Kelmansky, *Estadística para todos*, pp. 168–196.
- El estudio de la relación entre variables. Aguirre, Niño and Simonetti, *Estadística aplicada en las ciencias sociales y humanas* (in Spanish), pp. 105–130.

Linear regression

- Regresión lineal y correlación. Lind, Marchal and Wathen, *Estadística aplicada a los negocios y la economía* (in Spanish), 15th ed., pp. 461–494.
- Análisis de regresión múltiple. Lind, Marchal and Wathen, *Estadística aplicada a los negocios y la economía* (in Spanish), 15th ed., pp. 513–542.
- Análisis de regresión: el modelo lineal simple. Canavos, *Probabilidad y estadística. Aplicaciones y métodos* (in Spanish), pp. 443–478.
- Fundamentals of Data Visualization. Claus O. Wilke, part I: ch. 12.
- Linear Regression Foundations. *STAT 500: Applied Statistics*, Penn State, lesson 9.
- Linear Regression. *Interpretable Machine Learning*, Christoph Molnar, ch. 6.
- Regresión lineal (in Spanish). *Laboratorio de Datos*, first term 2021, UBA: class 6.
- Regresión lineal y correlación. *Introducción a la estadística* (in Spanish), OpenStax, ch. 12.

Implementation in Python

- Introduction to scikit-learn. *Python for Data Analysis*, ch. 12.
- Linear Regression in Python. Real Python.
- In Depth: Linear Regression. *Python Data Science Handbook*, Jake VanderPlas.

17. Logistic regression

Summary

Logistic regression is, in essence, a **binary classification model** (at least in its basic form), widely used in the credit industry for many years, and it has the particular feature of being, like linear regression, **interpretable for decision-making**. It is an intrinsically linear model and the parameter estimates it produces are accessible, so individual conclusions can be drawn about the importance each variable carries in the final estimate of the dependent variable, and statistical tests can also be applied to those parameters.

Unlike linear regression, the output of the model is not a quantity in measurable units but rather a **probability**, which lies between 0 and 1. Different criteria can therefore be set to finally choose one value or the other, usually with the midpoint (0.5) as the cutoff. A common scenario is also one in which the aim is to **classify correctly in favor of one category or the other** (for example, in medicine, where it is preferable to tell a patient that they are ill rather than that they are not), and for that the model is tuned to produce results that lean in one direction or the other. The usual measures for this are **precision, recall**, the false and true positive rates, and several other measures available through the construction of the **confusion matrix**, the standard tool for evaluating not only logistic regression but classification models (or classifiers) in general.

With the rise of data science / machine learning, logistic regression has gradually given way to other models that are simpler to compute / interpret, such as decision trees or random forests, and that also need no assumptions about the distribution of the data (the model itself) to be used and to draw conclusions from them. The next chapter covers some of them.

• EXTENSION

Beyond the confusion matrix

The confusion matrix remains the starting point for evaluating a classifier, but it summarizes performance **at a single cutoff point**, and a complete evaluation requires looking further.

Choosing the threshold

The midpoint (0.5) is a convention, not an optimum. The right threshold depends on the **cost of each type of error** and on the **frequency of the class of interest**: in fraud detection, letting a case through can cost much more than reviewing a legitimate one. The recommended approach is to choose the threshold that minimizes the expected cost on validation data (not training data); scikit-learn includes tools to tune it with cross-validation.

Threshold-independent metrics

- The **ROC curve** and the area under it (AUC) summarize performance across all possible thresholds.
- When the class of interest is rare, the **precision-recall curve** is more informative than the ROC.
- **Accuracy** is misleading with imbalanced classes: if 1% of transactions are fraudulent, a model that never detects fraud is right 99% of the time.

Calibration

If the model returns a probability and that probability is used as such (to set a price, prioritize cases or compute an expected value), ranking well is not enough: it has to be **calibrated**. That is, among the cases to which it assigns around 0.2, approximately 20% should belong to the positive class. This is checked with a **calibration curve** and summarized with measures such as the **Brier score** or the **log loss**. Logistic regression tends to produce well-calibrated probabilities, because it is fitted by optimizing precisely that loss; many other models need to be checked and, if necessary, recalibrated (for example, with Platt scaling or isotonic regression).

A clarification on the last paragraph

What the summary says about trees and random forests deserves a nuance: these models are more **flexible** (they do not require assuming a functional form), but a random forest is not simpler to interpret than a logistic regression; the opposite is true. Logistic regression remains standard in fields such as credit risk precisely because of its interpretability and its calibration, and it is a mandatory baseline against which to compare more complex models (see KNN and CART).

These same tools serve to evaluate a language model used as a classifier, a task developed in Verifying what the machine says.

Further reading

- Logistic Regression. *An Introduction to Statistical Learning* (2nd ed.), James, Witten, Hastie and Tibshirani, pp. 133–139.
- Logistic Regression. *Interpretable Machine Learning*, Christoph Molnar, ch. 7.
- Logistic Regression. *Introduction to Modern Statistics*, OpenIntro, ch. 9.
- Logistic Regression in Python. Real Python.
- Regresión logística (in Spanish). *Laboratorio de Datos*, first term 2021, UBA: class 8.

• EXTENSION

- Tuning the decision threshold for class prediction. scikit-learn User Guide.
- Probability calibration. scikit-learn User Guide.
- Metrics and scoring: quantifying the quality of predictions. scikit-learn User Guide.
- The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. Saito and Rehmsmeier, *PLOS ONE*, 2015.

18. KNN and CART

Summary

These models, **applicable to both regression and classification**, are excellent approximations to the data, without necessarily requiring prior knowledge of how the data behave or making assumptions about the underlying model. They are useful both in the **data exploration** stage and for building **baseline models** (because in practice they do not perform well enough to be deployed in production; they therefore serve as a point of comparison for building models that outperform them).

Like linear and logistic regression, their results **are interpretable**, although they do not have estimated parameters that can be analyzed afterward. Decision trees also have the advantage of being able to **ingest any type of data**, since in essence they are "yes or no" rules based on the discriminating power of the available data. Because they depend exclusively on the data, they are very sensitive to its quality, and also to the randomness of the starting point set by the analyst. Care must therefore be taken **not to draw hasty conclusions** from these models alone.

Later it will be shown that these are building blocks from which more complex models can be built (widely used in industry, such as *random forests*), with the same characteristics but overcoming the disadvantages mentioned.

• EXTENSION

From the baseline to the model that really competes

The "later" that this chapter announces was never developed in the first edition, and it is worth completing, because **tree ensembles**, in particular *gradient boosting*, are today the practical reference for tabular data (rows and columns, like most of the data in this book).

A ladder of baseline models

A new model must beat **every step** of a ladder of simpler alternatives, evaluated with the same cross-validation split and the same metric:

- A **trivial predictor**: the mean (regression) or the majority class (classification).
- A **linear or logistic** model (see Linear regression and Logistic regression).
- **KNN and a decision tree**, like those in this chapter.
- A **tree ensemble**: random forest or gradient boosting.

If a complex model barely improves on a linear one, the additional complexity (of maintenance, of explanation) is rarely justified.

Ensembles: bagging and boosting

A **random forest** combines many deep trees, each trained on a bootstrap sample of the data (see Nonparametric statistics) and with a random subset of variables at each split; averaging their predictions reduces the variability that makes a single tree unstable. **Gradient boosting**, by contrast, builds shallow trees **sequentially**, each focused on correcting the errors of the ensemble before it. Its widespread implementations (XGBoost, LightGBM, CatBoost and the histogram-based one in scikit-learn) accept missing values and, in several cases, categorical variables without prior encoding. The main hyperparameters are the learning rate, the depth of the trees and the number of trees, which are best set with **early stopping** on validation data.

The evidence in its favor is solid: a 2022 comparative study (Grinsztajn, Oyallon and Varoquaux) of 45 datasets found that tree-based models still outperformed neural networks on medium-sized tabular data (on the order of 10,000 observations), even without counting their greater speed.

The cost: interpretability

An ensemble of hundreds of trees can no longer be read as a set of rules. To understand what it learned, methods that explain the model from the outside are used, such as **permutation importance** or **SHAP values**. It is worth remembering that they explain the model,

not the phenomenon: a variable that is important for prediction is not necessarily a cause.

With AI assistants, putting together a boosting pipeline takes minutes (see From copilot to colleague). The risk shifts to evaluation: **data leakage** from the test set into the training set, poorly designed validation, unfair comparisons. The baseline ladder and a fixed validation split are the main defense (see Verifying what the machine says).

Further reading

General concepts

- Model-Agnostic Methods. *Interpretable Machine Learning*, Christoph Molnar, "Methods Overview".
- Example-Based Explanations. *Interpretable Machine Learning*, Christoph Molnar, "Methods Overview".

KNN

- K-Nearest Neighbors. *An Introduction to Statistical Learning* (2nd ed.), James, Witten, Hastie and Tibshirani, pp. 39–42.
- Nearest Neighbors. scikit-learn User Guide.
- The k-Nearest Neighbors (kNN) Algorithm in Python. Real Python.
- Comparison of Linear Regression with K-Nearest Neighbors. *An Introduction to Statistical Learning* (2nd ed.), James, Witten, Hastie and Tibshirani, pp. 105–110.

CART

- Decision Tree. *Interpretable Machine Learning*, Christoph Molnar, ch. 9.
- Tree-Based Methods. *An Introduction to Statistical Learning* (2nd ed.), James, Witten, Hastie and Tibshirani, pp. 327–339.
- Decision Trees. scikit-learn User Guide.

• EXTENSION

- Tree-Based Methods. *An Introduction to Statistical Learning* (2nd ed.), James, Witten, Hastie and Tibshirani, ch. 8.2: bagging, random forests and boosting.
- Ensembles: Gradient boosting, random forests, bagging, voting, stacking. scikit-learn User Guide.

- XGBoost: A Scalable Tree Boosting System. Chen and Guestrin, 2016.
- Why do tree-based models still outperform deep learning on tabular data?. Grinsztajn, Oyallon and Varoquaux, 2022.
- Permutation Feature Importance and SHAP. *Interpretable Machine Learning*, Christoph Molnar.

Part V. Data analysis in the age of AI

19. From copilot to colleague

From copilot to colleague

In February 2023, the first edition of this book described AI tools with a flight analogy: we were the pilots, and the assistant acted as copilot. It suggested the next line of code, answered questions, proposed corrections. The chapter closed with a warning: the responsibility for defining, thinking and anticipating errors *still falls on human skill (at least for now)*. That text is preserved intact in the appendix Copilots and chatbots.

Since then, the copilot has stopped limiting itself to suggestions. Current tools can **receive a goal, break it into steps, write and run code, read the results and try again** when something fails, until they deliver a complete analysis with tables, charts and written conclusions. In the jargon they are often called *agents*. The flight analogy, then, falls short: rather than a copilot assisting in the cockpit, the tool resembles a **colleague** to whom an entire task can be assigned.

This chapter opens the new part of the book with a simple question: if a tool can cover a good part of the analysis process on its own, what changes in the work of the person who analyzes data? The short answer is that the form changes a great deal, but not the substance.

What an AI tool can do in an analysis today

It is worth reviewing the stages of the data analytics process mentioned in Concepts and definitions and identifying where these tools come in:

- **Framing the question.** They can help turn a vague concern into concrete questions, suggest variables to consider or point out missing information.
- **Obtaining and wrangling the data.** They write SQL queries and code to join, clean and transform tables, and explain what each step does. That is, much of what was covered in Data wrangling.
- **Exploratory analysis.** They compute measures of location and dispersion, build frequency tables and propose visualizations, as in Describing variables and Data visualization.

- **Inferential analysis and models.** They suggest which statistical test or which model to apply, implement it and report the results.
- **Communication.** They write reports, executive summaries and presentations tailored to different audiences.

What is notable is not that each task is possible on its own (many already were, with libraries and templates), but that **a single tool can chain them together** from an instruction in natural language.

What still belongs to the analyst

However, several of the issues that the first edition pointed to as the foundation of all analysis are not solved with more computing power.

The **definition of the unit of analysis and of the variables**, for example, depends on the problem to be solved and on knowledge of the domain. A tool can propose a definition, but it does not know, unless told, what that organization means by "active customer" or which period makes sense to compare. The same happens with **assumptions**: knowing whether a sample is representative, whether a variable can be considered random or whether the groups being compared are really comparable requires knowing how the data were generated, something rarely written in the table.

The **critical reading of the results** also still belongs to the analyst. A conclusion can be correctly computed and still be irrelevant or misleading for the decision to be made. And, above all, **responsibility** still belongs to the analyst: whoever signs the report or recommends an action answers for it, regardless of which tool did the calculations. In that sense, the "at least for now" of the 2023 chapter can be read differently today: it is not a question of technical capability, but of who is accountable.

Uneven performance

One characteristic of these tools worth keeping firmly in mind is that **their performance is uneven**. They can handle with ease a task that would take a person hours and, in the same conversation, get something apparently trivial wrong, such as a count, a date or the name of a column. Moreover, they are wrong in the same confident tone in which they are right.

A practical consequence follows: **it is not enough to know what a tool can do in general; one has to know how much to trust it for a particular task**. That trust is not presumed; it is measured. The chapter Verifying what the machine says returns to this question with the tools of inferential statistics.

How to work with an artificial colleague

Some practices, closer to common sense than to a technique, help make good use of these tools without losing control of the analysis:

- **Giving context.** The clearer the description of the problem, the data and the definitions, the better the results. A vague request produces a generic analysis.
- **Asking for the path, not only the answer.** Requesting the code, the query or the reasoning that led to each result makes it possible to review it and, if needed, correct it.
- **Reviewing in parts.** An error is easier to detect in an intermediate step (a filter, a join between tables) than in the final report.
- **Keeping the trail.** The instructions, the generated code and the versions of the data are part of the analysis. The desirable properties mentioned in Working environment (transparency, replicability, interoperability and scalability) apply just as much, or more, when part of the work was done by a tool.
- **Distrusting the convenient.** A result that confirms exactly what was expected deserves a second look, whether it was computed by a person or by a machine.

Recap

The arrival of tools capable of carrying out a complete analysis does not make the fundamentals obsolete: **it makes them more necessary**. To assign a task well, one has to know what it consists of; to review a result, one has to understand how it was obtained; to decide how much to trust it, one has to know how to measure uncertainty. The chapters that follow go, one by one, through the points where the practice of data analysis changes with these tools, always connected to the topics of the first edition.

Further reading

- Building effective agents. Anthropic. A clear description of what is meant by an agent and of the common patterns for building agents.
- AI Index. Stanford Institute for Human-Centered Artificial Intelligence. An annual report on the state of artificial intelligence.
- People + AI Guidebook. Google PAIR. A guide to designing products in which people and AI systems work together.
- Attention Is All You Need. Vaswani et al., 2017. The paper that introduced the Transformer architecture, already cited in the first edition.

20. Asking the data in natural language

From SQL to the question in words

Programming languages pointed out that SQL is the standard for querying databases and that any job posting in the field will probably list it as a requirement. That statement remains true, but today it coexists with another possibility: **asking the question in natural language** ("how much did we sell by region last quarter?") and letting a tool translate it into a query, run it and return the result, sometimes accompanied by a chart.

This task is known as *text-to-SQL* and, when it is built into a conversation, as *conversational analytics*. It promises something very appealing: that anyone in the organization can query the data without depending on someone who knows how to program. That promise is partly real. To understand which part, it is worth seeing how it works and where it tends to fail.

How it works

In general terms, the tool receives three things: the **question**, a **description of the available data** (which tables exist, which columns they have, how they relate to each other) and, ideally, the organization's own **definitions and examples**. With that, it generates a query, runs it against the database and presents the answer.

Each of those inputs shapes the result. If the description of the tables is poor (columns with cryptic names, no documentation), the tool will have to guess what each one means. If the question is ambiguous, it will choose an interpretation, not necessarily the one that was intended. And if it does not know the organization's definitions, it will use the usual ones, which may not match.

The underlying problem: ambiguity

Consider the question "how many active customers do we have?" Answering it requires deciding, at a minimum:

- what a **customer** is (a person, an account, a company with several branches?);
- what **active** means (bought in the last 30 days, in the last year, has a current subscription?);
- which **point in time** it refers to (today, the close of last month?).

Each combination produces a different number, and all of them are technically correct. This is not a limitation of AI tools: it is exactly the problem raised in Data, units and variables of analysis. **The definition of the unit of analysis and of the variables comes before any calculation**, and what changes now is that this definition, previously implicit in the head of whoever wrote the query, must be made explicit for a tool to use it correctly.

Put another way: querying in natural language does not remove the need to define; it **moves** it. Whoever used to write SQL now writes definitions.

Definitions as infrastructure

For this reason, many organizations build what is called a **semantic layer**: a place where the metrics (sales, margin, active customers), the dimensions by which they can be broken down (region, product, month) and the correct way to compute them from the tables are defined once, in a shared way. Query tools, whether traditional dashboards or conversational assistants, take those definitions instead of reinventing them with each question.

Beyond the specific technology used, the underlying idea long predates AI: a well-maintained **data dictionary**. What is new is that its absence shows much faster, because a tool without definitions answers with the same confidence when it is right as when it invents an interpretation.

Verifying the query, not only the answer

When the answer is a number, it is tempting to take it at face value. Some simple practices reduce the risk:

- **Reading the generated query.** Writing SQL fluently is not necessary to check which tables were used, which filters were applied and how the data were grouped. Many errors (a misplaced date filter, a join that duplicates records) can be spotted at a glance.

- **Checking against known totals.** If total sales for the year are known, the sum by region should match. Frequency tables are, once again, a control tool: the absolute frequencies must add up to the total number of units.
- **Rephrasing the question.** Asking the same thing in another way and comparing the results is a quick way to detect different interpretations.
- **Distrusting what is too clean.** A result with no missing values or odd cases, in data known to have them, suggests that something was left out.

When it pays off and when it does not

Natural language querying works especially well for **exploratory and recurring questions** about well-documented data: getting a first overview, answering specific questions, quickly putting together a table for a meeting. It works less well when the question is new, depends on subtle definitions or the result will support an important decision. In those cases, the tool is still useful for writing a first draft of the query, but **review by someone who understands the data is not optional**.

Recap

Asking the data questions in words is a highly visible transformation of recent years, and it greatly broadens who can query information. But it does not solve the central problem of analysis, which is defining precisely what is to be measured. On the contrary, it brings that problem to the foreground: **the easier it is to ask, the more important it is to have defined well what is being talked about**.

Further reading

- A Survey on Text-to-SQL Parsing: Concepts, Methods, and Future Directions. Qin et al., 2022. An overview of the task of translating natural language into SQL.
- Spider. Yale University. A widely used benchmark for measuring text-to-SQL systems.
- BIRD. A text-to-SQL benchmark on large databases with real values, closer to the problems found in practice.
- dbt Semantic Layer. dbt documentation. A concrete example of a semantic layer.

21. Verifying what the machine says

An answer is not a data point

When an AI tool answers a question, summarizes a document or classifies a set of records, it delivers a result that **looks a lot like data**: it is neatly formatted, written with confidence and often correct. But it is not observed data. It is the output of a model, and like any model output, it can be wrong.

This chapter proposes looking at those answers the same way the first edition looked at an estimate. Point and interval estimation stated that estimating is approximating, and that the variation in the results of estimators plays a fundamental role in managing uncertainty. Something similar happens with a machine's answers: **the point is not to trust or distrust them wholesale, but to measure how often they are right, on which tasks and with what variability**.

Why they get things wrong

Language models generate text by choosing, word by word, plausible continuations based on what they learned during training. That way of working explains both their fluency and their main flaw: they can produce statements that are **plausible but false**, such as a citation that does not exist, a made-up figure or a library function with a name that sounds right. This phenomenon is usually called *hallucination*.

The causes are varied and are still being researched. A 2025 paper, for example, argues that the training and evaluation procedures themselves tend to reward a model for venturing an answer rather than admitting it does not know, in the same way that on a multiple-choice exam with no penalty for wrong answers it pays to guess rather than leave a question blank. Whatever the explanation, the practical consequence is clear: **the tone of an answer says nothing about its accuracy**.

The answer as a sample

One peculiarity of these tools is that they can give different answers to the same question. This, puzzling at first sight, admits a familiar statistical reading: **each answer can be thought of as an observation of a random variable**, with its own distribution.

From that perspective, asking once amounts to drawing a sample of size one. If the question is repeated several times and the answers agree, there is some evidence of stability (though not necessarily of accuracy: a model can be consistently wrong). If the answers vary widely, that **dispersion** is in itself a sign that the tool has no solid basis for answering. The notions of variability covered in Describing variables and of sampling distribution covered in Applied probability find a direct application here.

Building an evaluation set

The most powerful verification tool is also the simplest: **a set of cases whose correct answer is known in advance**. For example, to use a model to classify customer complaints into categories, one can take a sample of complaints, classify them carefully by hand and compare that classification with the model's.

That comparison is nothing other than the **confusion matrix** presented in Logistic regression, with its associated measures: accuracy, sensitivity, specificity and precision. And since the evaluation sample is exactly that, a sample, the results are subject to sampling error: an accuracy of 90% measured on 20 cases says much less than the same accuracy measured on 500. What was covered in Sampling techniques about sample size and representativeness applies unchanged: if the evaluation cases are only the easy ones, the measurement will be optimistic.

Some recommendations for building these sets:

- **Include hard cases and edge cases**, not just the typical ones.
- **Keep evaluation separate from tuning.** If the tool's instructions are adjusted while looking at the same cases used to evaluate it, there is a risk of overfitting, just as with any model.
- **Repeat the evaluation** every time the tool, the model version or the instructions change.

Comparing against a baseline

Knowing that a tool is right 85% of the time is not enough to decide whether it is worth using: a **point of comparison** is needed. How often is a simpler

method, such as a keyword-based rule, right? How often is a trained person right? What is the cost of each type of error?

The logic is the same as that of a hypothesis test: start from a null hypothesis ("the tool is no better than the simple alternative") and assess whether the evidence allows rejecting it. When several alternatives are compared at once, the reasoning behind ANOVA or, if its assumptions do not hold, non-parametric statistics techniques such as the bootstrap, makes it possible to do so with sound judgment.

The machine as evaluator

When the answers are long texts (summaries, explanations, reports), evaluating them by hand is costly. An increasingly common practice is to use **another model as an evaluator**, asking it to grade the answers according to defined criteria. This practice can be useful, but it introduces a new source of error: the evaluator also has biases (for example, a preference for longer answers or for a certain style). That is why **the evaluator must also be evaluated**, by comparing its grades with those given by people on a sample of cases.

Demanding traceability

Finally, a very effective way to verify is to require the tool to **show where each statement comes from**: the code that produced a number, the query that generated a table, the passage of the document that supports a summary. A traceable result can be reviewed; one that is not can only be believed or discarded.

Recap

Verifying what a machine says does not require new tools, but applying the ones this book has been presenting since the first edition: thinking in terms of samples and variability, measuring errors with a confusion matrix, comparing against a baseline with the logic of a hypothesis test. **Statistics, which was born to reason under uncertainty, turns out to be the best ally for working with tools that produce it.**

Further reading

- Why Language Models Hallucinate. Kalai, Nachum, Vempala and Zhang, 2025. The paper mentioned above on training and evaluation incentives.

- Survey of Hallucination in Natural Language Generation. Ji et al., 2022. An overview of the types of hallucination and the approaches to measuring them.
- Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. Zheng et al., 2023. An analysis of the scope and the biases of using a model as an evaluator.
- Your AI Product Needs Evals. Hamel Husain. A practical guide to building evaluations for products based on language models.
- AI Risk Management Framework. NIST. A reference framework for identifying, measuring and managing the risks of AI systems.

22. Agents that analyze

Many agents, many copies

An analysis agent is an AI tool that **carries out a stretch of the analysis on its own**: it queries data, writes and runs code, loads results and drafts conclusions (see From copilot to colleague). When several of them work at once on the same sources, a problem appears that teams of people know well and that agents make far more frequent: **each one has its own copy of the data, and the copies drift apart**.

This chapter is about how to organize the data of a team of agents. The proposal has two pieces: **a local warehouse per agent**, to work fast without stepping on anyone, and **a central team warehouse**, which records who loaded what, keeps the history of each table and defines how conflicts are resolved. What follows is the technical side; the organizational side (what role each agent plays, who approves what and how they are directed) is developed in the book *Business factories*, by the same author.

A local warehouse for each agent

Working environment presented *DuckDB* as a way to run SQL on files without installing a server. That same property makes it suitable for an agent: it is an **embedded** analytical database (it runs inside the same process as the query), columnar and with no network in between.

On top of DuckDB, *DuckLake* adds what a folder of loose files lacks. It is an open table format (MIT-licensed) that stores the data in *Parquet* files and the metadata (which tables exist, which files make them up, what changed in each version) in a separate database called the **catalog**. Each load is recorded as a version or *snapshot*, which can carry an **author and a message**, just like a commit, and any earlier version can be queried with `AT (VERSION ⇒ n)`. If, in addition, each table and each column carries a comment, a new agent can discover the tables by reading the catalog.

The limit of a local warehouse is obvious: **what is in one agent's folder is invisible to every other agent**, and in many environments it disappears when the session that created it ends.

A central warehouse for the team

The second piece is a shared warehouse, which in the DuckDB ecosystem can be built with DuckLake on a catalog in *Postgres* and the data in object storage. Its job is to answer three questions:

- **Who loaded what.** Each change that reaches the central warehouse is a snapshot with its author and its message. It is worth noting that in DuckLake the author is a text string written by whoever makes the change, and nothing authenticates it; that is why it is better for the service that receives the changes to stamp it, based on the agent's identity, rather than for the agent to declare it.
- **What the history is.** Snapshots make it possible to see a table as it was at any point in time.
- **How conflicts are resolved.** When two agents change the same thing, the central warehouse decides what happens, according to rules set in advance.

The key point is that **agents do not write directly to the central warehouse**. They work on their local copy, and a synchronization process carries changes back and forth. The analogy with *git* is direct: it is the same problem that software development solved for code. However, git does not work for the data itself. The catalog is a binary file, and when two agents load rows at the same time and an attempt is made to merge their versions, git responds `Cannot merge binary files` and leaves one of the two out. **Tables need to be merged row by row, not file by file.**

Clone, push and pull

The three git operations have their equivalent for tables:

- **Clone.** The agent receives the schema and the rows of the central warehouse as they are at a given snapshot, with their columns and comments, and records that number as its **base**.
- **Push.** The agent sends only what changed since its base. For this, DuckLake offers a change feed between snapshots (the `ducklake_table_changes` function), which returns the inserted rows, the deleted rows and the before and after image of each modified row. The central warehouse

applies everything in a single transaction, so that **each push is a central snapshot** with the agent as its author.
- **Pull.** The agent receives what others changed since its base and moves its base forward.

It is better for synchronization to be done by the environment at fixed moments (when a session opens, when each task ends) and not by the agent whenever it sees fit: that way nothing is left unpushed by oversight, and the agent does not need the central warehouse's credentials.

How conflicts are resolved

A conflict occurs when an agent tries to push a change to data that another agent has already modified. Four rules are enough to handle it in an orderly way.

One key per table. Data, units and variables of analysis defined a data point as the intersection of a unit of analysis and a variable. The key is what identifies the unit: the id of a customer, of an order, of a campaign. Without a declared key there is no way to know whether two rows coming from different agents refer to the same thing, and therefore **no conflict rule is possible**. It is the same requirement as for a join between tables (see Data wrangling).

One owner per table. Each table has a single agent, or a single group of agents with the same function, that can write to it; the others read it. If two functions need to record data about the same entity, those are two tables with the same key, joined at analysis time. The central warehouse rejects any write from a non-owner, and conflicts between different functions disappear by construction.

Reject by key and rebase. Within a single owner, several agents can work at the same time. A push is rejected when another agent changed any of its keys in the central warehouse after the base; keys that do not overlap go through without trouble. For example, an agent marks a lead as *lost* based on its copy, unaware that another agent has just marked it as *qualified*. The central warehouse rejects the change and reports which key conflicted and in which snapshot it changed; the first agent pulls the new version, sees the current state, decides and pushes again.

Set aside, never overwrite. The rejected change is not lost: it is stored separately, with an identifier, so that the agent (or a person) can review it. The usual alternative in synchronization tools, **last write wins**, would have

silently erased the other agent's *qualified*, and nobody would have found out. Append-only tables (events, measurements, costs) do not run into conflicts, so it is advisable to model everything that allows it that way.

The catalog defines how many can write

In DuckLake, **the type of catalog determines how many processes can write at the same time**:

- **Catalog in a DuckDB file.** Only one process can attach it at a time (unless all of them open it in read-only mode). If two agents try to load data into the same local warehouse at once, the second one fails immediately with a lock error (`Could not set lock on file … Conflicting lock`), and not even a reader can attach it while the other one is writing.
- **Catalog in SQLite.** SQLite supports a single writer at a time, but DuckLake retries lock conflicts, which in practice lets several local processes write interleaved. In a test with two processes that simultaneously loaded 50 batches of 20 rows each, the table ended up with all 2,000 rows, with no losses.
- **Catalog in Postgres.** This one is designed for many users, including remote ones, and that is why it is the one for the central warehouse.

Hence a simple rule: a DuckDB file for a single process, SQLite when several processes share a local folder, and Postgres for the central warehouse. It is worth noting that DuckLake resolves conflicts between simultaneous writes to the same catalog (it retries if the changes are compatible and aborts if they are not), but that is not the same as reconciling out-of-date local copies: as of this writing it offers no clone, push or pull, no branches and no *merge*, and that thin layer has to be built. *Dolt* does offer the full git model, with cell-by-cell *merge*, although it is a row-oriented database that its own authors do not recommend for large-scale analytics.

Recap

A team of agents needs the same things as a team of people sharing data, only more often. **Each agent works in its local warehouse, the team maintains a central warehouse**, and changes are cloned, pushed and pulled between them with an author and a message. Conflicts are handled with one key per table, one owner per table and a rule that rejects, sets aside and asks for a decision, instead of overwriting. DuckDB and DuckLake provide the engine and the history; synchronization, in the end, is still a piece the team must own.

Further reading

- Choosing a Catalog Database. DuckLake documentation. What each catalog (DuckDB, SQLite, Postgres) supports in terms of concurrent processes and writers.
- Conflict Resolution. DuckLake documentation. How transactions that conflict on the same catalog are resolved.
- DuckLake 1.0: The Lakehouse Format Built on SQL Reaches Production-Readiness. DuckLake team, 2026. What the first stable release includes and what is left for later.
- DuckLake Extension. DuckDB documentation. Snapshots, commit messages and queries on earlier versions.
- Conflicts. Dolt documentation. An example of cell-by-cell conflict detection in a database versioned like git.

23. Every number with its source

A number without a source cannot be checked

A report is **traceable** when every figure it contains can be reproduced: it is known which version of the data it comes from, which table and which rows, or which query. Verifying what the machine says closed with an idea that is developed here: a traceable result can be reviewed, while one that is not can only be believed or discarded.

The requirement is not new. A good analyst has always been able to say where each number in a presentation came from. What changes with agents is the scale and the style: an agent can produce a ten-page report in minutes, well written and with dozens of figures, and **the neat format says nothing about whether those figures come from the data**. This chapter proposes a simple rule for reports made by agents (every number with its source) and a concrete format for following it.

The usual failure: a confident claim with no support

When thinking about the errors of an agent that works with data, the first thing that comes to mind is a miscalculation: a sum that leaves out a filter, a join that duplicates rows. Those errors exist, but they have an advantage: they leave a trail in the code, and someone can find them. The harder failure is a different one: **an agent that confidently states something the data do not support**. A count that was never queried, a "grew 12%" computed on an old copy, a "500 records were loaded" when the table holds 480. The text is correct in form and has no connection whatsoever to the database.

Recent research measures it. A 2026 paper on agents that operate databases found that, before any control was applied, **21 of 220 responses contained an unsupported claim** of the type being controlled for. When every claim of that type was required to be tied to a trace of what was

actually executed, none of the 199 responses issued had one, at the cost of 21 fewer responses. The result offers two lessons. The first is that it is better to **verify the state of the data and not the agent's description of it**. The second is that abstaining is a valid outcome: the rule trades coverage for reliability, a tradeoff similar to the one made when setting a cutoff point in Logistic regression.

Stopping that failure is what makes an analysis done by agents reliable. And the most direct way to stop it is for every figure to carry its source.

First, the version of the data

Data change. An orders table at 9 a.m. is not the same as at 6 p.m., and a query repeated the next day can give a different result without anyone having made a mistake. That is why **a report states the version of the data it comes from**, once, at the beginning or at the end.

In a versioned warehouse, like the one described in Agents that analyze, that version is a *snapshot* number. With DuckLake, a query written with `AT (VERSION ⇒ n)` returns exactly the same thing today as it will a year from now, even if thousands of new rows have been loaded. Without that reference, when a number does not match what someone sees later, there is no way to tell an error from a change in the data. It is, in essence, the property of **replicability** that this book has been pointing out since Working environment.

Then, the table and the row, or the query

The figures in a report usually come in two kinds:

- **A value from a row.** A customer's status, an order's date, a campaign's budget. It is cited with the table and the row's key.
- **A computed value.** A total, an average, a count, a change. It is cited with the query that produces it, complete and runnable, and with the result it returned.

A concrete format, which an agent can produce effortlessly and a person can read without training, consists of short markers in the text and a block of sources at the end:

> In October there were **1,284 paid orders**, for a total of **$18,420,500** [F1]. The average ticket was **$14,346** [F2], and the customer with the highest billing for the month was C-0193 [F3].

```
Sources
Data: central warehouse, snapshot 412 (2026-11-02 09:00)

[F1] sales.orders
     SELECT count(*) AS orders, sum(amount) AS total
     FROM sales.orders AT (VERSION ⇒ 412)
     WHERE status = 'paid'
       AND fecha ⩾ DATE '2026-10-01' AND fecha < DATE '2026-11-01';
     → orders 1284 | total 18420500

[F2] Computed from [F1]: total / orders = 18420500 / 1284
     → 14346.18

[F3] sales.customers, key customer_id = 'C-0193'
     (the first one when the query in [F1] is grouped by customer_id
      and ordered by sum(amount) descending)
```

A few rules go with the format:

- **One version for the whole report.** If for some reason versions are combined, each source states its own.
- **The query exactly as it was run**, with its filters and its unrounded result; the rounding goes in the text.
- **A row is cited by its key, never by its position.** "Row 15" stops meaning anything as soon as the table changes order.
- **A derived figure cites the figures it comes from and the operation**, like [F2], so that an error in the base can be traced to everything that depends on it.
- **A figure without a source does not go out**, or it goes out explicitly marked as an estimate or as unverified data (as Data wrangling does with the 80% saying, which has no verifiable source).

Who checks the citations

A citation is only useful if someone checks it, and this is where the format pays off: **checking is mechanical**. It is enough to run each query on the cited snapshot and compare the result with what the text says, something a program can do without a person's intervention. The same goes for rows cited by key. People are left for what cannot be automated.

And some things cannot be automated. A citation verifies that the number comes from the query, **not that the query answers the right question**. A sales total can be computed correctly and still have excluded returns when the question included them; the ambiguity discussed in Asking the data in

natural language is still there. To reduce it, it is better for the agent to query metrics defined once, in a semantic layer, rather than write free-form SQL for each report: that way "sales" means the same thing in every report the team produces. It is also worth noting that who signs a report and who decides when a figure without a source may go out are questions of organization more than of data, and they are developed in the book *Business factories*, by the same author.

Finally, traceability has a cost: the report is a little longer and the agent must keep every query it runs. It is a small cost compared with the alternative, which is reviewing an entire report because a single figure turned out to be false and there is no telling which others are too.

Recap

A report made by agents is reliable to the extent that **every number can be reproduced**. To that end, it states the version of the data it comes from and cites, for each figure, the table and the row's key or the query that produces it, along with its result. The failure this rule stops is not the miscalculation, which leaves a trail, but the confident claim the data do not support. A citation does not replace the reader's judgment, but it points that judgment where it is needed: at the question, not at the arithmetic.

Further reading

- No Trace, No Claim: Two Contracts for Database Agents. Zhang, 2026. The paper that measures unsupported claims in database agents and the effect of requiring a trace for each one.
- Enabling Large Language Models to Generate Text with Citations. Gao, Yen, Yu and Chen, 2023. How to measure whether the citations a language model produces actually support what it claims.
- Time Travel. DuckLake documentation. How to query a table as it was at a given snapshot.
- dbt Semantic Layer. dbt documentation. A way to define metrics once and query them by name.

24. Validate before you believe

Validating is putting to the test

Validating an agent's work means **testing what it delivers against checks defined in advance, before using it**. A loaded table, a fitted model, a report with figures: none of these becomes an input for a decision until it passes a set of tests that someone set before seeing the result.

Hypothesis tests described putting what is known to the test, using evidence, as the way science works. Validating an agent's work is the same operation applied to a smaller case: one starts from a claim (the result is correct) and looks for evidence that allows rejecting it. What changes is the starting point. **An agent's work is not believed until it is validated**, and not the other way around. This chapter presents what research measures about the effect of validating, how to read those figures with the tools of this book and where it makes sense to place the tests.

What research measures

Two recent papers provide concrete figures, and they are worth knowing in context.

The first is **AutoKaggle** (Li et al., 2024), a multi-agent system that solves Kaggle competitions with tabular data: cleaning, feature engineering, modeling and submission of predictions. Among its tools it includes **unit tests** that run on the code and the data of each stage (missing values, outliers, duplicates and other consistency problems). When those tests were removed and everything else was left the same, **the average completion rate fell from 0.85 to 0.10**, that is, from 85% to 10% of runs. The authors sum it up by saying that, without unit tests, completing the tasks becomes almost impossible.

The second is a scaling study by Google Research, Google DeepMind and MIT (Kim et al., 2025), which compared 260 configurations of agent systems across six task sets, with five architectures and three model families. One of its measures is an **error amplification factor**: how much errors multiply when going from a single agent (factor 1) to several. With **independent agents**, which work in parallel and whose outputs are combined without any cross-validation, the factor was **17.2**. With a **centralized** architecture, where an orchestrator receives the others' work and verifies it before combining it, it was **4.4**. It is worth noting that this measure is taken on execution traces; the paper itself reports another one, on the error in the final result of the task, where the differences between architectures are much smaller (between 1.1 and 1.3 times). The direction is the same; the magnitude depends on what is measured.

Reading those figures with a statistician's eye

The figures in a research paper are estimates too, and they deserve the same treatment as any other (see Point and interval estimation).

The AutoKaggle comparison was made on 4 tasks. If, as in the rest of the paper's experiments, each task was repeated 5 times, that makes **20 runs per condition**: 17 complete with tests and 2 without them. With so few observations, a 95% confidence interval (computed with the Wilson method, which is better suited than the normal approximation when the sample is small or the proportion is close to 0 or 1) runs roughly **from 0.64 to 0.95 with tests and from 0.03 to 0.30 without them**. The intervals are wide, but they do not overlap: the effect is large enough to show even with a small sample. What the sample does not guarantee is generality. The 4 tasks were chosen by the authors and are not a random sample of all possible data tasks, so what was covered in Sampling techniques about representativeness applies unchanged.

The same reasoning holds for one's own measurements. Saying that an agent "got it right 9 out of 10 times" is not the same as saying it is right 90% of the time: with 10 cases, the Wilson interval runs roughly from 0.60 to 0.98. **A success rate without its interval is a small sample presented as a parameter.**

A validation is a test

The analogy with a hypothesis test is not just a figure of speech. A check on an agent's work has the same two possible errors:

- **Rejecting correct work** (the equivalent of a Type I error). Its cost is usually rework: the agent redoes something that was fine.
- **Accepting wrong work** (the equivalent of a Type II error). Its cost is a decision made on false data, and it is usually much higher.

A test that only checks that the file exists almost never rejects anything, that is, it has very little **power**. Tests with power are the ones that could fail if the work were wrong. Some worth always having:

- **Structure.** Expected columns and types, a unique key, no missing values where they do not belong.
- **Control counts and totals.** The rows read must equal the rows written plus the rows discarded, and a total in the table must match the one in the source.
- **Business rules.** Non-negative amounts, dates within the period, statuses within a closed list.
- **Independent recomputation.** Rerunning the queries cited in the report on the same version of the data, using the format proposed in Every number with its source.
- **Cases with a known answer.** An evaluation set like the one described in Verifying what the machine says.

There is also the problem of **multiple comparisons**. An agent can run dozens of statistical tests in seconds and return only the ones that came out significant; at a 5% level, some will be significant by chance. Validation must require the agent to report **all** the tests it ran, not just the ones that support the conclusion.

Where to put the verifier

The scaling study suggests something about how tests are organized, and not just about their content: **errors are better contained with a central point that verifies before combining** than with agents that check one another, or that do not check at all. In the terms of Agents that analyze, that point is a natural one: the central warehouse receives each change from each agent and can validate it at the door (types, declared key, the table's rules, a cap on deletions per push) before accepting it, and send it back with the reason when it does not pass.

Two caveats. The first is that **the verifier must also be evaluated**: if it is another model, it has its own biases, and if it is a set of rules, it can miss precisely the error that matters. The second is that not everything can be validated by a program. Which tests to require for each table, what to do

when a piece of work fails several times and who authorizes an exception are decisions of organization, which are developed in the book *Business factories*, by the same author.

Estimating the error, not just approving or rejecting

When the volume is large (thousands of records classified by an agent, for example), reviewing everything by hand is not possible, and approving without reviewing is not acceptable. The way out is the usual one: **review a random sample and estimate the error rate with its interval**. If the upper end of the interval exceeds the error one is willing to tolerate, the batch does not pass, even if the point estimate looks good. It is the logic of the acceptance sampling that industrial quality control has used for decades, applied to a process now carried out by a machine.

Recap

The measured evidence is consistent: without tests, a system of agents working with data completed its tasks in 10% of runs instead of 85%, and independent agents without a central verifier multiplied their errors 17.2 times, compared with 4.4 with one. Read with the tools of this book, those figures are estimates with their uncertainty, and the same view applies to the success rate of any agent. **Validating is running a test**: with a hypothesis set before looking, with tests that have power, with every comparison reported and with an interval instead of a single number. Statistics is, once again and in the end, the tool for deciding how much to believe.

Further reading

- AutoKaggle: A Multi-Agent Framework for Autonomous Data Science Competitions. Li, Zang, Ma et al., 2024. The paper whose comparison with and without unit tests is cited in this chapter (table 3).
- Towards a Science of Scaling Agent Systems. Kim, Gu, Park et al. (Google Research, Google DeepMind and MIT), 2025. The scaling study and its error amplification factor by architecture.
- Why Do Multi-Agent LLM Systems Fail?. Cemri, Pan, Yang et al., 2025. A classification of the failures of multi-agent systems, including weak task verification.
- Add data tests to your DAG. dbt documentation. How to declare structure and rule tests on the tables of a warehouse.
- Improving Your Statistical Inferences. Daniël Lakens. An open book on power, effect sizes and multiple comparisons.

25. Text, images and audio as variables

Beyond the cell

Data, units and variables of analysis proposed thinking of a data point as anything that can be placed inside a cell: a number, a letter, a word, a true or false. That definition is still useful, but it leaves out a vast amount of information that organizations generate every day and that does not fit comfortably in a spreadsheet: the comments in a survey, complaint emails, the photos from a field survey, the recordings of a call center.

For a long time, analyzing that kind of information meant reading it, looking at it or listening to it, and coding it by hand, which limited the analysis to small samples. AI tools changed that situation radically: today it is possible to **turn text, images and audio into variables** at a scale that used to be impractical. This chapter is about how to do it with judgment.

The model as a measuring instrument

The central idea is simple: an AI model can work as a **measuring instrument**. Just as a scale turns an object into a number of kilograms, a model can turn a comment into a category ("complaint about a delay", "price inquiry"), a rating (positive, neutral, negative), a list of the topics mentioned or an amount extracted from the text.

The result of that process is, once again, a **data matrix**: each row is a unit of analysis (a comment, a photo, a call) and each column is a variable obtained from it. From there, everything covered in the first edition applies unchanged: frequency tables, descriptive measures, visualizations, tests and models.

But like every measuring instrument, the model has **error**. And the quality of the analysis that follows depends on knowing that error.

Coding with judgment

Before asking a tool to classify thousands of texts, it pays to do the work that the social sciences call building a **codebook**: defining precisely which categories exist, what each one means and what to do with doubtful cases. The condition required of the classes of a frequency table holds here too: the categories must be **mutually exclusive** (each case falls into only one, or it is stated explicitly that a case can fall into several). And it pays to add another: that they be **exhaustive**, that is, that every case has a place, even if it is "other".

A reasonable procedure is the following:

1. **Define the categories** and write down their criteria, with examples.
2. **Code a sample** of cases by hand, ideally by more than one person, to detect ambiguities in the definitions.
3. **Apply the model** to that same sample and compare its results with the manual coding, using a confusion matrix.
4. **Adjust the definitions or the instructions** and repeat until the level of agreement is acceptable for the intended use.
5. Only then, **apply the model to the full set**.

This procedure is not a formality. There is evidence that, for certain text annotation tasks, conversational models can match or outperform people hired on crowdwork platforms; but that depends on the task, the definitions and the data, and **it can only be known by measuring** in the specific case. What was developed in Verifying what the machine says applies here directly.

Representing meaning with numbers

Besides classifying, there is another way to turn texts (and images) into variables: **embeddings**, or vector representations. An embedding assigns each text a list of numbers, in such a way that texts with similar meanings are represented by similar lists. There is no need to define categories in advance: similarity emerges from the representation.

With that representation it is possible, for example, to find the comments most similar to a given one, to group automatically the texts that talk about the same thing, or to use those numeric columns as input variables for a model. The idea of "closeness" between observations already appeared in KNN and CART: the k nearest neighbors algorithm classifies a case

according to the cases closest to it. Embeddings make it possible to apply that same logic to objects that previously had no natural notion of distance.

One limitation is worth pointing out: the dimensions of an embedding do not, in general, have a direct interpretation. They are useful for measuring similarity and feeding models, but they do not replace an explicitly defined variable when what is needed is to explain a result.

Images and audio

The same principles extend to other formats. In the case of **audio**, automatic transcription models turn a recording into text, which can then be coded as shown above. In the case of **images**, there are models that can describe them, classify them or relate them to descriptions in natural language, which makes it possible, for example, to label the photos from a field survey according to what they show.

In both cases it is worth remembering that **each conversion step adds a layer of error**: a transcription with errors produces a coding with errors, even if the classification model works perfectly. Reviewing a sample of each intermediate stage is the way to know where quality is lost.

Measurement error does not go away

Suppose a model classifies comments as "positive" and "negative" and that, on a hand-coded sample, it is observed to tend to confuse ironic comments with positive ones. If the proportion of positive comments over the total is then calculated, **that proportion will be biased upward**. Measurement error is not noise that cancels out; it can have a direction.

That is why, when reporting results obtained this way, it is good practice to **also report the quality of the measurement**: how it was validated, on how many cases and with what level of agreement. Just as an estimate without a confidence interval says little, a proportion calculated on automatic classifications with no information about their accuracy should be read with caution.

Recap

The possibility of turning text, images and audio into variables greatly expands what can be analyzed. But the principle is the same as always: **a data point is the result of a measurement, and every measurement has an error worth knowing**. Defining the categories well, validating against a

carefully coded sample and reporting the quality of the measurement are the conditions for that expansion to also be an improvement.

Further reading

- The Illustrated Word2vec. Jay Alammar. A visual introduction to the idea of representing words as vectors.
- ChatGPT Outperforms Crowd-Workers for Text-Annotation Tasks. Gilardi, Alizadeh and Kubli, 2023. The study on text annotation mentioned in this chapter.
- Sentence Transformers. Documentation for a widely used Python library for obtaining text embeddings.
- Robust Speech Recognition via Large-Scale Weak Supervision. Radford et al., 2022. The paper that introduced Whisper, an open audio transcription model.
- Learning Transferable Visual Models From Natural Language Supervision. Radford et al., 2021. The paper that introduced CLIP, a model that relates images and text.
- Text as Data: A New Framework for Machine Learning and the Social Sciences. Grimmer, Roberts and Stewart, Princeton University Press, 2022. A rigorous treatment of text as a data source in the social sciences.

26. Synthetic data and simulation

Data that were never observed

Data sources and collection raised an idea that at the time looked like a detail: that a sample is not obtained only from real data, but can come from a simulation or from **synthetic data**, something useful for experimentation. Since then, that idea has moved to the center. Generative models can produce tables, texts and images that imitate real ones with a degree of plausibility that was unthinkable a few years ago.

This chapter is about what synthetic data are, what they are for and, above all, what they cannot do. Because the ease of generating them brings a dangerous temptation: treating them as if they were observations of the world.

What a synthetic data point is

A synthetic data point is one **generated by a model** rather than observed directly. That definition covers a wide range of techniques, some of which have already appeared in this book:

- **Simulation from known distributions.** Drawing random values from a normal or binomial distribution, like those covered in Applied probability, is the oldest and most transparent way to produce synthetic data. It is known exactly which process they come from.
- **Resampling.** The bootstrap generates new samples from an existing sample by drawing observations with replacement. It does not invent new values, but it does create new combinations.
- **Statistical models fitted to real data.** The joint distribution of the variables (including their relationships) is estimated, and new records are then generated from it.

- **Generative AI models.** Language or image models that produce examples from instructions, such as fictitious surveys, customer conversations or product descriptions.

What all these techniques have in common is that **a synthetic data point inherits the properties of the model that generated it**, with its strengths and its limitations.

What they are for

Synthetic data have legitimate and valuable uses:

- **Testing processes.** Before running an analysis on real data, it is useful to check that the code works on data with the same structure: the same columns, the same types, rare cases included.
- **Sharing without exposing.** When the original data contain personal information, a synthetic version that preserves their statistical properties can let other people work with them. The chapter Privacy, bias and responsibility returns to this point, with its nuances.
- **Exploring scenarios.** Simulation makes it possible to answer "what if...?" questions: how a result would change if demand grew, or what sample size would be needed to detect a given difference with a hypothesis test.
- **Teaching and learning.** A simulated dataset, whose generating process is known, is ideal for checking whether a method recovers the parameters that were used to create it.
- **Balancing rare classes.** In classification problems where one of the categories is very rare, generating synthetic examples of that category can help train a model, taking care not to evaluate it on those same examples.

What they cannot do

The fundamental limitation is simple to state: **a synthetic data point contains no information about the world that was not already in the model that generated it**. If the model was fitted to a sample of a thousand records, generating a million synthetic records does not produce a million observations: it produces a thousand observations restated many times.

This has a direct consequence for inference. Point and interval estimation shows that the larger the sample, the more precise the estimates. But that relationship holds for **random samples from the population**, not for data generated from a sample. Calculating a confidence interval on a million synthetic records would give a completely false sense of precision.

Something similar happens with data generated by language models. Asking a model to "answer as a 40-year-old customer would" can produce plausible answers, but those answers reflect what the model learned about how people tend to write, not what the real customers of a given organization think. They can help design and test a questionnaire; **they do not replace the survey**.

The risk of recursion

One phenomenon illustrates these limits particularly well. Recent research showed that when a generative model is trained on data produced by other models, and this is repeated over several generations, quality degrades: **the tails of the original distribution disappear** and the results become more and more homogeneous. This phenomenon was named *model collapse*.

From the perspective of this book, the explanation is intuitive. Each generation of synthetic data is a sample from a model that is, in turn, an approximation of the previous data. Extreme values, being infrequent, are less likely to appear in each sample, and what does not appear is not learned. Generation after generation, **dispersion** shrinks, in the sense studied in Describing variables. The rare, which is often the most interesting part, is lost first.

How to evaluate a synthetic dataset

Before using synthetic data, it pays to evaluate them on at least three dimensions:

- **Fidelity.** Do they resemble the real data? The distribution of each variable can be compared with frequency tables and descriptive measures, the relationships between variables can be compared (for example, with correlation coefficients) and, more formally, goodness-of-fit tests like those covered in Nonparametric statistics can be applied.
- **Utility.** Do they serve the intended purpose? A common test is to train a model on the synthetic data and evaluate it on real data: if its performance is similar to that of a model trained on real data, the synthetic dataset preserves what matters for that task.
- **Privacy.** Do they reveal information about real people? A generative model can memorize and reproduce records from the original dataset, especially the unusual ones. Checking that no synthetic record is identical or nearly identical to a real record is a minimum control.

It is worth noting that these dimensions are often in tension: the more faithful a synthetic dataset is, the more it resembles the real data and the greater the risk that it reveals information about them.

Recap

Synthetic data are a powerful tool for testing, teaching, sharing and exploring scenarios. But **they are not observations**: they are the reflection of a model, and they cannot contribute information that the model does not have. Used with awareness of that limitation, they expand what can be done; treated as if they were real data, they produce conclusions with an apparent precision they do not have.

Further reading

- Synthetic Data: what, why and how?. Jordon et al., 2022. An accessible report on the uses, techniques and limits of synthetic data.
- AI models collapse when trained on recursively generated data. Shumailov et al., *Nature*, 2024. The work that describes model collapse.
- On LLMs-Driven Synthetic Data Generation, Curation, and Evaluation: A Survey. Long et al., 2024. An overview of synthetic data generation with language models.
- Synthetic Data Vault (SDV). Documentation for a Python library for generating and evaluating tabular synthetic data.

27. Privacy, bias and responsibility

The questions that are not technical

The previous chapters dealt with how to use AI tools to analyze data: how to assign them tasks, how to query in natural language, how to verify their results, how to turn texts and images into variables and how to generate synthetic data. All of them assume a prior question that is not technical: **should it be done?** Can these data be used for this purpose? Can they be sent to this tool? Whom can the result harm?

These questions existed before AI, but they became more urgent for two reasons. The first is scale: what used to take weeks of manual work is now done in minutes, on millions of records. The second is distance: many tools process the data outside the organization, on third-party servers, and their decisions can be hard to explain. This chapter goes through three axes, privacy, bias and responsibility, with the same applied outlook as the rest of the book.

Personal data

A large share of the data analyzed in organizations refers to people: customers, employees, patients, students. In Argentina, its processing is regulated by **Law 25,326 on the Protection of Personal Data**, which sets out, among others, principles of data quality (that the data be adequate and not excessive in relation to the purpose for which they were obtained), of consent, of informing data subjects, of security and of confidentiality, and which gives special treatment to sensitive data. Other jurisdictions have their own rules, and it pays to know the ones that apply in each case.

From the point of view of analysis, some concrete practices follow from these principles:

- **Minimize.** Use only the variables needed to answer the question. If the analysis is by region, the exact address is not needed; if it is by age group, the date of birth is not needed.
- **Respect the purpose.** That a data point is available does not mean it can be used for any purpose. Data collected to manage one service cannot necessarily be reused for another end.
- **Aggregate when possible.** Many questions can be answered with frequency tables or summary measures, without working with individual records. It should be noted, however, that very small groups may make it possible to identify the people in them.
- **Pseudonymize with awareness of its limits.** Replacing names with codes reduces the risk but does not eliminate it: the combination of several apparently harmless variables (age, town, occupation) can be enough to identify someone.

Sending data to an external tool

When an AI tool hosted by a third party is used, the data sent to it leave the organization. Before doing so, it pays to answer a few questions: **who processes the data and where? are they retained, and for how long? can they be used to train other models? what do the contractual terms and the organization's policies say?**

In many cases, the appropriate answer is not to send the original data, but a version that does not expose them: aggregated or pseudonymized data, or a synthetic sample with the same structure (with the precautions noted in Synthetic data and simulation). Another alternative is to ask the tool for the code to perform the analysis and run it locally, so that the data never leave the controlled environment.

Bias starts in the sample

Sampling techniques and Data sources and collection stressed the importance of the sample being **representative** of the population. That requirement takes on a new dimension with models: **a model learns from the data it was trained on, and it reproduces their imbalances**. If certain groups are underrepresented in the data, the model will tend to work worse for them; if the data reflect past decisions that were biased, the model will learn to repeat them.

A frequently cited case is a 2018 study that evaluated commercial gender classification systems based on photos of faces and found that errors were

much more frequent for darker-skinned women than for lighter-skinned men. None of those systems had been designed to discriminate; the problem lay in the data and in how their performance had been evaluated.

Measuring bias

The practical lesson of that case is that **an overall indicator can hide important differences between groups**. An accuracy of 95% over the total can be the average of 99% in one group and 70% in another. That is why, when evaluating a model that affects people, it pays to **calculate the confusion matrix separately for each relevant group** and compare the error rates.

It should be noted that there is no single definition of a "fair model". It can be required that the proportion of positive cases be equal across groups, that the false positive rates be equal, or that the precision be equal, and in general **it is not possible to meet all those conditions at once**. Choosing which to prioritize is a decision that depends on the context and on the consequences of each type of error, and it cannot be delegated to the algorithm.

Documenting

A practice that helps a great deal is to **document data and models** systematically. Specific formats have been proposed for this: sheets that describe how a dataset was collected, what it contains, what it was intended for and what it should not be used for; and cards that describe a model, its intended use, how it was evaluated and how it performs across different groups. Beyond the format, the goal is that anyone who uses those data or that model knows their limitations without having to rediscover them.

Responsibility is not delegated

Finally, the axis that runs through the other two: **decisions based on an analysis have people responsible for them, and those responsible are people**. That a result was produced by an AI tool does not transfer the responsibility to the tool.

Regulatory and reference frameworks point in that direction. In 2024 the European Union approved a regulation on artificial intelligence that classifies systems by their level of risk and sets stricter obligations for high-risk ones. The US NIST published a voluntary framework for managing AI risks, and UNESCO adopted a recommendation on the ethics of artificial intelligence. Whichever applies in each case, they all agree on a few points: **know the risks, document, supervise and be able to explain**.

Recap

Privacy, bias and responsibility are not side topics dealt with at the end of an analysis, but conditions defined at the start: which data are used, how they were obtained, whom they represent, where they are sent and who answers for the conclusions. **The same statistical tools this book presents for analyzing data also serve to detect when an analysis may harm someone**; using them for that purpose is part of the craft.

Further reading

Regulations and reference frameworks

- Ley 25.326 de Protección de los Datos Personales (in Spanish). Argentina.gob.ar.
- Regulation (EU) 2024/1689 (Artificial Intelligence Act). EUR-Lex.
- AI Risk Management Framework. NIST.
- Recomendación sobre la ética de la inteligencia artificial (in Spanish). UNESCO.

Bias and documentation

- Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. Buolamwini and Gebru, 2018. The study mentioned in this chapter.
- Fairness and Machine Learning: Limitations and Opportunities. Barocas, Hardt and Narayanan. An open book on the definitions of fairness and the tensions between them.
- Datasheets for Datasets. Gebru et al., 2018. A proposal of sheets for documenting datasets.
- Model Cards for Model Reporting. Mitchell et al., 2018. A proposal of cards for documenting models.

28. From insight to action

The last stretch

Concepts and definitions described data analysis as an activity that makes it possible to draw conclusions and translate them into **recommendations or actions**, and that, framed within an analytics process, supports **decision making** in organizations. Throughout the book, however, the focus was almost always on the first part: how to obtain, describe, infer and model. The last stretch, the one that goes from a conclusion to an action, lay outside the scope of analysis proper.

AI tools shorten that stretch considerably. A single system can detect a pattern, write a recommendation and, connected to the right tools, **carry it out**: send an alert, adjust a price, prioritize a list of cases. This chapter, which closes the new part of the book, is about what it takes for that step, from conclusion to action, to be well grounded.

Closing the loop

When the result of an analysis automatically becomes an action, the analysis stops being a report that someone reads and becomes **part of a system** that acts on the world. And systems that act have consequences that reports do not: an error in a report can be caught before anyone uses it; an error in an automated system repeats every time the system acts.

Connecting a model to the tools that let it act (a database, a management system, a messaging channel) is technically simple today, and there are even open protocols designed to standardize those connections. Precisely for that reason, the relevant question is no longer "can it be automated?" but **"what evidence is needed to do it, and how will anyone know whether it keeps working?"**.

Deciding under uncertainty

Every conclusion drawn from data carries some degree of uncertainty, as has been seen from Data, units and variables of analysis onward. When

moving to action, that uncertainty does not disappear: **it becomes risk**. Some ideas from the first edition are especially useful for managing it:

- **Intervals, not just points.** A point estimate says what the most likely value is; a confidence interval says how far off it could be. An action that only pays off if the true value lies at the optimistic end of the interval is a bet, not a grounded decision.
- **The cutoff point is a decision.** Logistic regression mentioned that, to go from a probability to a classification, 0.5 is usually used as the cutoff point, although other criteria can be set. When an action is automated, that criterion stops being a technical detail: it defines how many false positives and false negatives are accepted.
- **Errors do not cost the same.** Failing to detect a fraud is not the same as blocking a legitimate customer by mistake. The appropriate cutoff point depends on the relative cost of each type of error, something that can only be defined with knowledge of the business.

Experimenting before automating

An analysis of historical data can show that a certain action is associated with better results, but that does not prove the action causes them. Perhaps the customers who received an offer bought more because they were already better customers, and that is why they were sent the offer. The soundest way to find out whether an action has the expected effect is to **test it in a controlled way**.

A **controlled experiment** (in the digital world it is usually called an *A/B test*) randomly assigns the units of analysis to different groups, applies the action to only one of them and compares the results. Random assignment is what makes it possible to attribute the difference to the action and not to other factors. The analysis of those results is exactly what was covered in Hypothesis tests and, if more than two variants are compared, in ANOVA, whose very formulation starts from experiments with different treatments.

It should be noted that experimenting has its own requirements: defining in advance what will be measured, calculating the sample size needed to detect a relevant difference, and not stopping the experiment as soon as the results look favorable. These are the same precautions as in classical inferential statistics, applied to everyday decisions.

Monitoring what was automated

A model is fitted with data from a given moment, but it is used in a world that changes: customers change their habits, new products appear, rules are modified. Over time, **the data the model acts on may stop resembling the data it was built with**, and its performance degrades with nothing to announce it.

That is why automating an action is not the end of the analysis, but the start of a follow-up task. It pays to monitor at least three things: **the input data** (do they still have distributions similar to the original ones?), **the system's decisions** (has the proportion of cases classified each way changed abruptly?) and **the results** (does the action still have the effect that was measured in the experiment?). The descriptive tools of the first edition, such as frequency tables, summary measures and charts over time, are enough for much of this follow-up.

People in the loop

Not all decisions should be automated to the same degree. A useful criterion is to consider two dimensions: **how much harm an error can cause and how easy it is to reverse**. A wrong content recommendation corrects itself on the next visit; a decision on a loan, a diagnosis or a penalty affects a person in a much more serious and lasting way.

Between full automation and a completely manual decision there are many intermediate points: the system can propose and a person approve, it can act only on the clear cases and refer the doubtful ones, or it can act within limits (of amount, of quantity, of scope) that a person sets. Choosing the right point for each decision is itself part of the work of analysis, and it builds on everything covered in Privacy, bias and responsibility.

Recap

The value of an analysis is realized when it turns into a good decision. AI tools make it possible to cover that last stretch at an unprecedented speed, and for that very reason they demand more rigor, not less: **managing uncertainty instead of ignoring it, experimenting before automating, monitoring what was automated and reserving human judgment for what cannot be undone**. It is, in the end, the same statistics as always, put to work for action.

Further reading

- Trustworthy Online Controlled Experiments. Kohavi, Tang and Xu, Cambridge University Press, 2020. A practical reference on controlled experiments in the digital world.
- Rules of Machine Learning. Martin Zinkevich, Google for Developers. Practical recommendations for taking models to production and maintaining them.
- Hidden Technical Debt in Machine Learning Systems. Sculley et al., 2015. A classic on the hidden costs of maintaining model-based systems.
- Model Context Protocol. Documentation for an open protocol for connecting AI models to tools and data sources.
- Building effective agents. Anthropic. Patterns for building systems that act, and criteria for deciding when it makes sense to do so.

Appendices. Appendices

A. Programming fundamentals

Summary

Some concepts are common to all programming. Therefore, regardless of whether one uses Python, R, Julia, JavaScript, SQL, etc., there will always be a way to build what is needed by recognizing and using different **data/variable types and structures**, **operators and control/flow structures**, and **functions/procedures**. Of course, each language will have variations that have to be learned in due course, but the essence will lie in the functionality of each of the elements just mentioned.

So, first of all, there are data and/or variables. It is worth reviewing the chapter Data, units and variables of analysis, since once again the data point appears as the smallest unit being worked with. **Data, which can be of one type or another, are stored in variables, and those variables can have various structures**, depending on the logic by which the information is stored. For example, a very basic structure is a **table or matrix**, an arrangement of elements in 2 dimensions (horizontal and vertical), which, as noted earlier, can be used as a means of presenting aggregations of data (see Frequency tables). Of course, data can carry relationships as complex as those of the real world, so there are data structures (or combinations of them) able to represent that complexity. Python has **lists, tuples, arrays, dictionaries, dataframes and many more**, commonly used for data analysis. In R and other languages they may or may not have the same name, and the operations that can be performed on them may also vary.

Operations are the way to relate elements. In practice, one can start from the parallel with mathematical operations (addition, subtraction, multiplication, etc.), which is correct, given that they are present in almost any analysis process. One level below that are *Boolean* operations, which are the basis of all programming. These are operations with ones and zeros,

simple in nature, but just as present in everyday work with data, since they allow the **comparison and conditioning of elements and processes**, through structures designed in advance for that purpose.

Any operation on data, variables and/or data structures will likely need to be reused (in whatever language); so what is called (almost universally) functions comes into play. **Functions are blocks of operations** that usually take data as input and output, with some transformation in between. It should be noted that this may not be the case, but they at least make a change (in the code or in the working environment). A set of functions is usually called a *library* (or module), which will be of vital importance, since it is practically impossible to build anything from scratch. Hence the importance of reuse (and of the concepts of transparency, replicability, interoperability and scalability mentioned in Working environment).

To all of the above should be added a matter related to **design patterns in programming**, that is, to the way programs are developed, since they can become extremely complex and there must be a common criterion for creating not only functions but any reusable element. In that sense, **object-oriented programming** has been widely adopted. Its building block is the *class*, which can be an **abstraction of any entity** that can be thought of (cars, houses, feelings). From those classes, concrete elements can be generated, which is called an *instance* of that class (a particular car, house or feeling). Instances will have an associated set of *attributes* (the color of a car, the height of a house, the fervor of a feeling) and operations, called *methods* (driving the car, painting the house, making someone laugh), through which they can relate to the rest of the elements and to the programming environment itself.

Further reading

- Python for Everybody. Charles Severance, chapters 3 to 11 and 15.
- Python Tutorial. W3Schools.

B. Copilots and chatbots

> **Written in February 2023.** This chapter is kept as it was written for the first edition, as a record of a moment when these tools were just starting to be used. What happened afterward is taken up in the part Data analysis in the age of AI, starting with From copilot to colleague.

Summary

As of early 2023, the rise and popularity of these *tools* is the order of the day. Like every contemporary technological novelty, it seems to be settling into **exponential growth that is hard to take in fully and to understand in terms of applications and consequences**. The impact may become substantial in many fields, mainly those related to programming, an activity in which their potential and performance are proving to be remarkable.

From the perspective of the professional role in data analysis, what matters is **keeping up with the advances in this first stage of discovery and development, and progressively acquiring the ability to use the wide range of platforms**, tools, libraries, etc. that will emerge from it. At present, at least two stand out:

- ChatGPT
- GitHub Copilot

The first is a chatbot, an interface on which all kinds of questions, statements and/or **expressions in natural language** (the one humans use to communicate with each other) can be entered, and an appropriate answer obtained (which may include natural language, code and images, among others), with the characteristic of being possibly *indistinguishable* from a well-thought-out answer that another person would give us. That said, there is much talk that it could be **a new way of searching** (the internet), leaving behind the old habit of *googling*, a skill tacitly required of any data analyst.

The second is an implementation designed to be used within an IDE (for example, VS Code), with the aim of speeding up the *coding* process through: requests (in natural language) to create functions or other pieces of code with specific requirements; context-based prediction of what one wants to code next (*predictive text* or autocomplete); and suggestions for error fixes and improvements. To use the analogy of a flight, with us as the pilots of an airplane, this tool gives us considerable assistance as a copilot. **The responsibility for defining, thinking and being creative in developing solutions and anticipating errors outside the realm of code** (or landing an airplane with hundreds of passengers on board), among many other matters, **still rests on human skill** (at least for now).

A feature common to these tools (and to the others now emerging) is the use of natural language. This is possible thanks to the creation of a novel machine learning architecture called **Transformers**. Those who want to go deeper into this topic can start by reading the well-known 2017 paper titled *Attention Is All You Need*.

To recap, what has been mentioned so far can be of great use, **in both professional and academic settings**, providing a **new means of acquiring knowledge**; for inspiration and the search for new solutions; for rethinking communication and making it more effective; for teaching and assessing; for redefining every task that depends, at some point, on the use of natural language. Let us see what ChatGPT *thinks*...

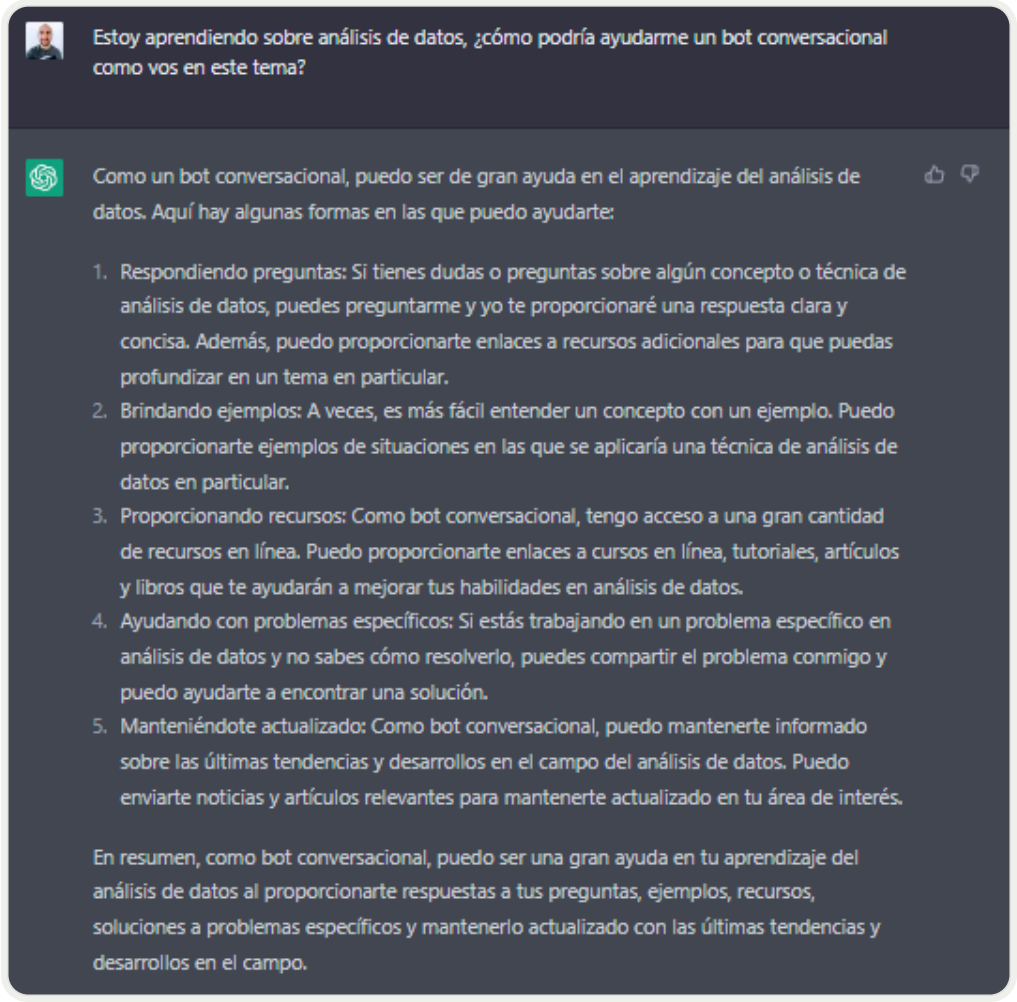


Figure 1: ChatGPT's answer, in February 2023, on how it can help someone who is learning data analysis

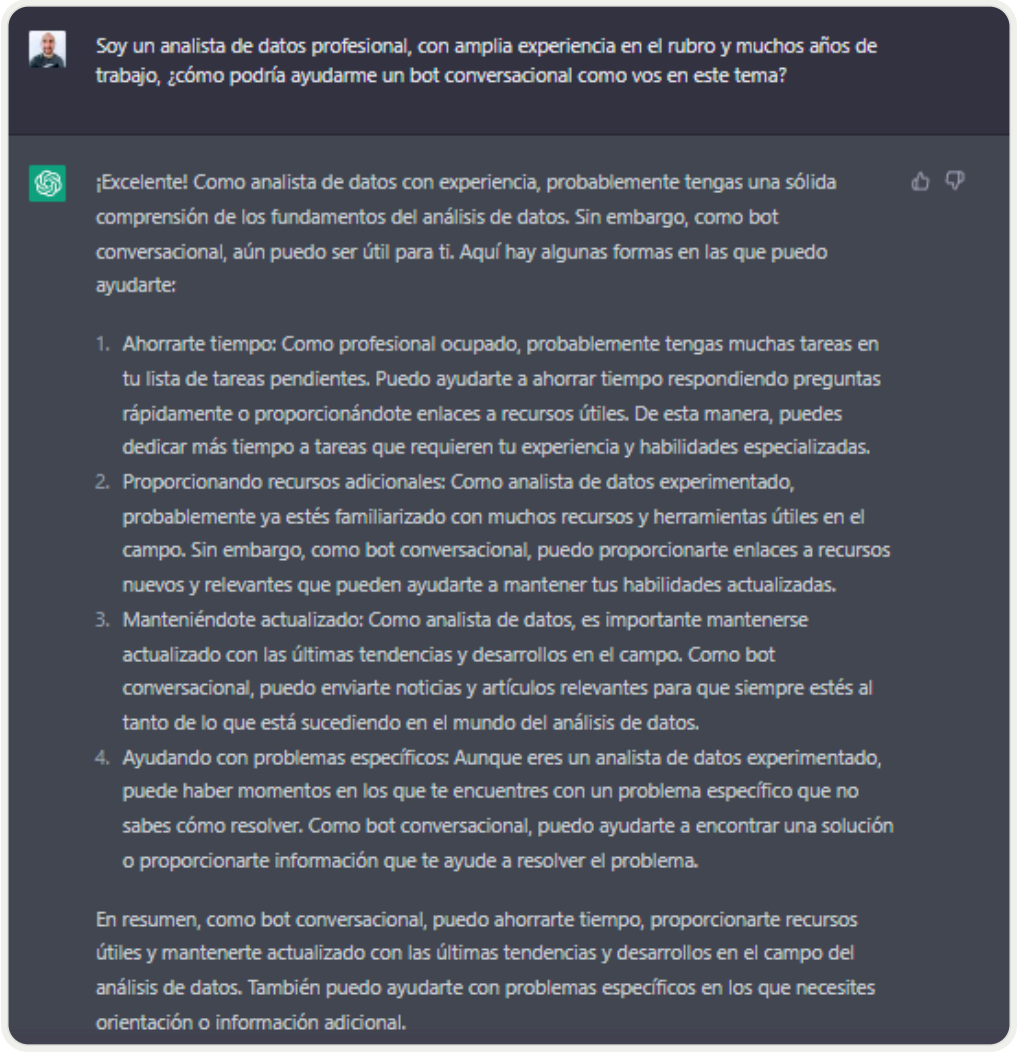


Figure 2: ChatGPT's answer, in February 2023, on how it can help a data analysis professional

Finally, as far as this book and the solving of activities are concerned, those who want to use these tools (or similar ones) may of course do so at their discretion, without any restriction. However, in pedagogical terms, if they do, **it is suggested that their use be accompanied by further reading on the various topics and by written reflections (by the students themselves) on the results obtained**. Having these tools presumably saves time, so it ends up being a matter of using the time left over to strengthen learning. In the mandatory assignments their use is also allowed, but students are

encouraged to state which tool or tools they used and for what, with a brief description of the process.

Further reading

- Harnessing the Power of ChatGPT for Data Science. Analytics Vidhya, February 2023.
- End-2-End Data Analytics using Open-AI ChatGPT. Kaggle discussion thread.
- How useful is ChatGPT for data visualisation work?. The Data Visualisation Catalogue.