Part 1: data wrangling

Lino Galiana
Version of September 29, 2026

`Python` has established itself as a credible alternative to R for data manipulation. The `Pandas` ecosystem has made it possible to democratize the use of DataFrames in `Python` and make it easier to manipulate of structured data thanks to the `SQL` philosophy. `Python` is also the most practical language for retrieving and manipulating unstructured data (*webscraping*, APIs). Python is tending to become, thanks to its dynamic ecosystem, the *"one to rule them all"* language.

⚠️ **Automatically generated PDF.** This document is produced from the course website and may contain formatting issues. For an up-to-date version adapted to your reading, visit the course page: pythonds.linogaliana.fr.

Table of contents

Although data scientists are often associated with the implementation of artificial intelligence models, it is important not to forget that training and using these models do not necessarily represent the daily work of data scientists.

In practice, gathering heterogeneous data sources, structuring and harmonizing them for exploratory analysis prior to modeling or visualization represents a significant part of data scientists' work. In many environments, this is even the essence of a data scientist's role. Developing relevant models indeed requires deep reflection on the data; an essential step that should not be overlooked.

This course, like many introductory resources on data science [1], [2], [3], will therefore offer a lot of content on data manipulation, an essential skill for data scientists.

Programming software based around the database concept has become the main tool for data scientists. Being able to apply a number of standard operations on databases, regardless of their nature, allows programmers to be more efficient than if they had to repeat these operations manually, as in Excel.

All the dominant programming languages in the data science ecosystem are based on the dataframe principle. It is even a central object in some software, notably R. The logic of `SQL`, a language for declaring data operations that has been around for over fifty years, provides a relevant framework for performing standardized operations on columns (creating new columns, selecting subsets of rows, etc.).

However, the dataframe only recently became established in Python, thanks to the `Pandas` package created by Wes McKinney. The rise of the `Pandas` library (downloaded over 5 million times per day in 2023) is largely responsible for Python's success in the data science ecosystem and has led, in just a few years, to a complete renewal of how coding in Python, such a flexible language, is approached for data analysis.

This part of the course is a general introduction to the rich ecosystem of data manipulation with Python. These chapters cover both data retrieval and the restructuring and analysis of that data.

Summary of that section

`Pandas` has become essential in the `Python` ecosystem for *data science*. `Pandas` itself is built on top of the `Numpy` package, which is useful to understand to be comfortable with `Pandas`. `Numpy` is a low-level library for storing and manipulating data. `Numpy` is at the heart of the *data science* ecosystem because most libraries, even those handling unstructured objects, use objects built from `Numpy`[1].

The `Pandas` approach, which provides a unified entry point for manipulating datasets of very different natures, has been extended to geographic objects with `Geopandas`. This allows for the manipulation of geographic data as if it were classic structured data. Geographic data and cartographic representation are becoming increasingly common with the rise of open localized data and geolocated *big-data*.

However, structured data imported from flat files is not the only data source. APIs and *web scraping* allow for flexible downloading or extraction of data from web pages or specialized portals. These data, particularly those obtained through *web scraping*, often require a bit more data cleaning work, especially with character strings.

The `Pandas` ecosystem thus represents a Swiss army knife for data analysis. This is why this course will cover it extensively. Before trying to implement an *ad hoc* solution, it is often useful to ask the following question: *"Could I do this with the basic functionalities of `Pandas`?"* Asking this question can prevent arduous paths and save a lot of time.

However, `Pandas` is not suitable for handling large volumes of data. To process such data, it is recommended to use `Polars` or `Dask`, which adopt the logic of `Pandas` but optimize its functionality, `Spark` if you have suitable infrastructure, generally in big data environments, or `DuckDB` if you are willing to use SQL queries rather than a high-level library.

Exercises

This section provides both detailed tutorials and guided exercises. You can view them on this site or use one of the badges at the beginning of the chapter, for example these to open the Pandas exercises chapter:

Going further

This course does not really address issues of volume or speed of computation. `Pandas` can show its limits in this area with large datasets (several gigabytes).

It is therefore interesting to consider:

- The book Modern Pandas for additional insights on performance with `Pandas`;
- The question of sparse objects;
- The *packages* `Dask` or `Polars` to speed up computations;
- `DuckDB` for very efficient SQL queries;
- `PySpark` for very large datasets.

[1] Some libraries are gradually moving away from `Numpy`, which is not always the most suitable for managing certain types of data. The `Arrow` framework is becoming the lower layer used by more and more data science libraries. This blog post provides a detailed explanation of this topic.

References

Here is a selective bibliography of interesting books complementary to the chapters in the “Manipulation” section of this course:

Informations additionnelles

i Python environment

This site was built automatically through a `Github` action using the `Quarto` reproducible publishing software (version 1.10.18).

The environment used to obtain the results is reproducible via uv. The `pyproject.toml` file used to build this environment is available on the `linogaliana/python-datascientist` repository

```
[project]
name = "python-datascientist"
version = "0.1.0"
description = "Source code for Lino Galiana's Python for data science course"
readme = "README.md"
requires-python = "⩾3.13,<3.14"
dependencies = [
    "altair⩾6.0.0",
    "cartiflette",
    "contextily==1.6.2",
    "duckdb⩾0.10.1",
    "folium⩾0.19.6",
    "gdal==3.11.4",
    "graphviz==0.20.3",
    "great-tables⩾0.12.0",
    "gt-extras⩾0.0.8",
    "ipykernel⩾6.29.5",
    "jupyter⩾1.1.1",
    "jupyter-cache⩾1.0.0",
    "kaleido⩾0.2.1",
    "langchain-community⩾0.3.27",
    "loguru==0.7.3",
    "markdown⩾3.8",
    "nbclient⩾0.10.0",
    "nbformat⩾5.10.4",
    "nltk⩾3.9.1",
    "pandas⩾3.0",
    "pip⩾25.1.1",
    "plotly⩾6.1.2",
    "plotnine⩾0.15",
    "polars⩾1.8.2",
    "pyarrow⩾17.0.0",
    "pynsee⩾0.1.8",
    "python-dotenv⩾1.0.1",
    "python-frontmatter⩾1.1.0",
    "pywaffle⩾1.1.1",
    "requests⩾2.32.3",
```

```
    "scikit-image≥0.24.0",
    "scikit-learn≥1.8.0",
    "scipy≥1.13.0",
    "seaborn≥0.13.2",
    "selenium<4.39.0",
    "spacy≥3.8.4",
    "webdriver-manager≥4.0.2",
    "wordcloud==1.9.3",
]

[tool.uv.sources]
cartiflette = { git = "https://github.com/inseefrlab/cartiflette" }
gdal = [
  { index = "gdal-wheels", marker = "sys_platform == 'linux'" },
  { index = "geospatial_wheels", marker = "sys_platform == 'win32'" },
]

[[tool.uv.index]]
name = "geospatial_wheels"
url = "https://nathanjmcdougall.github.io/geospatial-wheels-index/"
explicit = true

[[tool.uv.index]]
name = "gdal-wheels"
url = "https://gitlab.com/api/v4/projects/61637378/packages/pypi/simple"
explicit = true

[dependency-groups]
dev = [
    "nb-clean≥4.0.1",
]
```

To use exactly the same environment (version of `Python` and *packages*), please refer to the documentation for `uv`.

Bibliography

[1] H. Wickham, M. Çetinkaya-Rundel, and G. Grolemund, *R for data science.* " O'Reilly Media, Inc.", 2023.

[2] J. VanderPlas, *Python data science handbook: Essential tools for working with data.* " O'Reilly Media, Inc.", 2016.

[3] W. McKinney, *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython.* " O'Reilly Media, Inc.", 2012.