Git: data scientist's most essential tool

Lino Galiana
Version of September 29, 2026

A part complementary to the course to discover `Git`, a tool that has become essential for data scientists in order to carry out projects involving `Python` code.

Table of contents

1 Introduction **1**
1.1 The problem 1
1.2 The technical solution: `Git` 2
2 How to use `Git` when doing `Python` ? **2**
3 Content of this part **3**
Informations additionnelles 3

This part of the site presents something that is not specific to `Python` but is nevertheless essential: the practice of `Git`.

A large part of the content of this part comes from a dedicated course I gave with Romain Avouac (in French).

Scroll down to read the *slides* below or click here to display the slides full screen (the slides are in French).

1 Introduction

1.1 The problem

The natural approach when you start working for a long time on a project for which you want to avoid erasing and losing your code because of a human error is to duplicate files and create **multiple versions of the same code that are snapshots at a given point in time**:

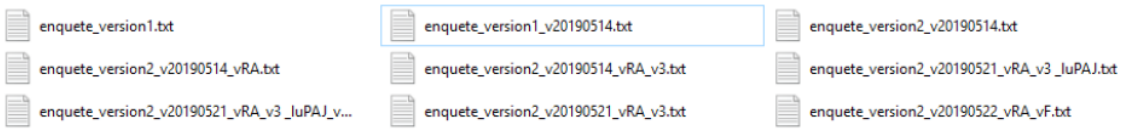

Figure 1: Handmade version control

This is a way of saving the evolution of your code and therefore the life of a project. Nevertheless, it is a very artisanal way of proceeding that does not eliminate human error, since it is always possible to make a mistake when duplicating the file or to forget to save a version that satisfied us.

This practice has many drawbacks. The first is that searching for relevant information, for example about the introduction of a bug, is complicated by this duplicated structure. It is difficult, without

going into the detail of each file, to know its actual evolution between two dates. For information extraction, it would be much more relevant to build files describing the evolution between one version and another, but this requires many manual steps and a considerable amount of time. The second problem is that we do not necessarily know, when we come back to the project a few days later, why we saved a given version: on 28 November, do we remember what distinguishes the versions of 4 May at 12:37 and at 16:02?

If we add the collaborative dimension of working on code, other drawbacks of this artisanal approach quickly appear. First, someone taking over the project will have trouble getting into it. Moreover, it will be even harder for this person to look for relevant information about the version choices that were made: this history is ultimately useless to them. Especially since the question of sharing this code arises: through what channel is this set of files shared? By email? On a shared drive? But what happens if several participants in the project work on it at the same time? How can collaboration be organised and changes reconciled if they happen jointly?

1.2 The technical solution: `Git`

`Git` provides a technical answer to these many questions. This software, specialised in version control, that is, in tracking the evolution of a project, solves many problems related to carrying out *data science* projects in organisations. The purpose of this chapter is to present a few concepts needed to understand `Git` and to show how it helps manage the evolution of a software project. The next chapter will introduce how `Git` smooths collaboration within teams involved in *data science* projects. This has become essential because the era when *data scientists* all worked alone on *notebooks* is over. *Data science* projects have become, in most organisations, more ambitious and often involve several people with varied profiles[1], so discipline is needed for collaboration to be smooth.

2 How to use `Git` when doing `Python` ?

`Git` is version control software, that is, software in charge of recording the evolutions of a file over time (what is called versioning). It is not a `Python` *package*, we will not use it that way.

`Python` users can use `Git` through two intermediaries: through the command line or through graphical extensions in their development environments (`VSCode`, `Jupyter`, etc.). On the `SSPCloud`, the recommended infrastructure for this course, we can do `Git` with both approaches but we will mainly use the `VSCode` graphical interface, which lowers the cost of entry into `Git`.

> Where to find additional information for this course?
>
> `Git` is part of the collaborative practices that have become standard in the *open-source* world but are also increasingly common in data science administrations and companies.
>
> There are many resources on using `Git` on the internet. Unfortunately, many are technical and assume an already advanced knowledge of some computing notions useful for understanding `Git`. This course will not make such assumptions, except for a minimal knowledge of the logic of a *filesystem*, that is, of how files are organised on a computer.
>
> Regarding content close to this one, a series of training resources has been gathered by Insee on this site (in French).

[1] This observation is the starting point of the course "Putting *data science* projects into production", available at ensae-reproductibilite.github.io/website/ (in French), which Romain Avouac and I teach at the end of the ENSAE curriculum.

3 Content of this part

Learning `Git` is split into two chapters:

- The first is devoted to presenting the general logic of `Git` and its important concepts, and to illustrating them through practice in an individual work setting.
- The second chapter is devoted to the challenges of collaborative work

The general objective of this part is to demystify `Git`. While self-taught discovery can be particularly painful, this is not the case when accompanied by a resource that illustrates the important concepts through practice and gradually increases the complexity of the setting in which `Git` is used.

> 👉 A number of terms that are new when you discover `Git`, but which are the concepts useful for understanding it, are defined in the margins of the next two chapters, as shown here.

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`.