A working environment in Python and R

Install Python and R so they still work on a bad connection, lay out a project directory that survives a handover, and run the same first script in both languages.

Pierre Robentz Cassion

Lesson 2 of 8

Data Analysis Foundations for M&E

What this lesson covers

- Why this lesson is not optional
- Python: use uv, not the system interpreter
- R: use Projects and renv
- A project layout that survives a handover
- The same first script, in both languages
- What the two languages disagree about
- What comes next

Why this lesson is not optional

- The most common reason an M&E analysis stops being reproducible is not a statistical error.

Python: use uv, not the system interpreter — Shell

```
# Install uv (macOS and Linux)
curl -LsSf https://astral.sh/uv/install.sh | sh

# Create a project and pin an interpreter
uv init muac-analysis
cd muac-analysis
uv python install 3.13

# Add what this course uses
uv add pandas pyarrow matplotlib
```

Python: use uv, not the system interpreter — Shell

```
uv sync
```

R: use Projects and renv — In R

```r
# Once per machine
install.packages("renv")

# Once per analysis, from inside an RStudio Project
renv::init()

# Add what this course uses
install.packages(c("readr", "dplyr", "tidyr", "ggplot2", "janitor"))

# Record exactly what is installed
renv::snapshot()
```

R: use Projects and renv

- **Work inside a Project.** The working directory is then the project root, for everyone, always. Paths like...
- **Never write** `setwd()`. It encodes your username into the analysis and guarantees it fails for the next person.

A project layout that survives a handover — Example

```
muac-analysis/
  data/
    raw/          # exactly as it arrived. Read-only. Never edited.
    interim/      # intermediate output. Disposable.
    processed/    # analysis-ready. Regenerable from raw + code.
  scripts/
    01-read.py
    02-clean.py
    03-indicators.py
  output/
    figures/
    tables/
  README.md
```

A project layout that survives a handover

A useful test: delete `data/processed/` and `output/` entirely, rerun your scripts, and see whether you get the same results. If you cannot bring yourself to try it, you already know the answer.

The same first script, in both languages — In Python

```python
import pandas as pd

muac = pd.read_csv("data/raw/muac-screening-artibonite-2024.v1.csv")

print(muac.shape)
print(muac.head())
print(muac.dtypes)
```

The same first script, in both languages — In R

```r
library(readr)
library(dplyr)

muac <- read_csv("data/raw/muac-screening-artibonite-2024.v1.csv")

dim(muac)
head(muac)
glimpse(muac)
```

What the two languages disagree about

Concern	pandas	dplyr / readr
Missing value	`NaN`, and `None` for objects	`NA`, typed per column
Reading a CSV	`pd.read_csv` guesses types	`read_csv` guesses and reports
Chaining steps	Method chaining or reassignment	The pipe, `|>`
Grouped summary	`groupby().agg()`	`group_by() |> summarise()`
Categories	`category` dtype, opt-in	Factors, and they bite

What comes next

- The next lesson reads the export properly — which is harder than `read_csv(path)`, because the defaults will quietly damage identifiers, dates and the missing-value code before you have looked at anything.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/data-analysis-foundations/foundations-02-environment