It works on my machine, and that is the bug report

The same script, the same data and a different pandas version produce different numbers. A lock file is what turns "it works on my machine" from a defence into a testable claim, and it costs one command to make.

Pierre Robentz Cassion

Lesson 3 of 8

Reproducible Analysis Workflows — The same answer twice

What this lesson covers

- What a colleague's laptop does differently
- Python: `uv`
- R: `renv`
- Why a lock file and not `requirements.txt`
- What this platform pins, and why one pin is unusual
- Containers, and when they are worth it
- The test that proves it
- Report it whole
- What comes next

What a colleague's laptop does differently

What differs	What it changes
Package version	A default argument, a rounding rule, a sort order
Language version	Dictionary ordering, integer division, string handling
Operating system	Line endings, file ordering, path separators, locale
Locale	`1,234` parsed as 1234 or as 1.234
What is already installed	A library your script imports and never declares

What a colleague's laptop does differently

- None of those is in your repository — which is why the code being identical is not enough

Python: uv — Shell

```
uv init                       # creates pyproject.toml
uv add pandas matplotlib      # records the dependency and resolves it
uv run python run.py          # runs inside the pinned environment
```

Python: `uv`

- Two files, and both are committed — `pyproject.toml` says what you asked for — `pandas>=2.0` — and `uv.lock` says...
- A colleague runs `uv sync` and has your environment — on their operating system, without you knowing what they had...
- Pin the language version too

Python: uv — Example

```
# pyproject.toml
requires-python = ">=3.12,<3.13"
```

Python: `uv`

- A version range, not a single version — Pinning to `3.12.4` exactly means a colleague on `3.12.7` cannot run it at all,...

R: renv — In R

```
renv::init()                # snapshots the project library
renv::snapshot()            # after adding a package
renv::restore()             # on the colleague's machine
```

R: `renv`

- `renv.lock` is the equivalent artefact and is committed — It records every package, its version, and the repository it...
- Record the R version, which `renv` does automatically — and check it in the script where a difference would matter

R: renv — In R

```
if (getRversion() < "4.3.0") stop("This analysis requires R >= 4.3.0")
```

R: `renv` — In Python

```python
# The Python equivalent, at the top of the entry point.
import sys
assert sys.version_info >= (3, 12), "This analysis requires Python 3.12+"
```

Why a lock file and not `requirements.txt` — Example

```
pandas>=2.0
matplotlib
```

Why a lock file and not `requirements.txt`

- That file says almost nothing — It resolves to different versions on different days and does not mention the fifty...

Why a lock file and not `requirements.txt`

	Declares	**Reproduces**
`requirements.txt`	Intent	No
`requirements.txt` with `==` pins	One layer	Partly — transitive dependencies float
`uv.lock` / `renv.lock`	The whole tree	**Yes**

Why a lock file and not `requirements.txt`

- Commit both the intent and the lock — The intent file is what a human edits; the lock is what a machine reproduces...

What this platform pins, and why one pin is unusual — Example

```
{
  "packageManager": "pnpm@11.9.0",
  "engines": { "node": ">=22" },
  "devDependencies": { "typescript": "^6.0.3" }
}
```

What this platform pins, and why one pin is unusual

- The package manager itself is pinned — because a different pnpm resolves the lockfile differently and that is the layer...
- TypeScript is held at 6.x deliberately — TypeScript 7 — the native compiler — does not yet expose the programmatic API...
- That is the shape a pin should have — A version constraint with no comment is a constraint nobody will dare remove and...

What this platform pins, and why one pin is unusual — Example

```
# pandas is pinned below 3.0 because the copy-on-write default changes the
# behaviour of the recode in src/clean.py:88. Revisit when that is rewritten.
pandas = ">=2.1,<3.0"
```

Containers, and when they are worth it

- Worth it when — the analysis has non-Python or non-R dependencies — GDAL, a TeX distribution, a database client — or…
- Not worth it when — a lock file already reproduces the result and the audience is two colleagues with laptops

Containers, and when they are worth it — Example

```
FROM python:3.12-slim
COPY pyproject.toml uv.lock ./
RUN pip install uv && uv sync --frozen
COPY . .
CMD ["uv", "run", "python", "run.py"]
```

Containers, and when they are worth it

- Start with the lock file — Reach for a container when you can name the dependency it is pinning that the lock file...

The test that proves it

- Reproduce your own result on a machine that has never seen the project — A fresh clone in a temporary directory is most...

The test that proves it — Shell

```
git clone <repo> /tmp/check && cd /tmp/check
uv sync
uv run python run.py
diff -r outputs/ ~/project/outputs/
```

The test that proves it — In R

```
# renv::restore() then source("run.R"), and compare.
```

The test that proves it

- If you cannot do that, "it is reproducible" is untested — A colleague's laptop is better and a CI runner is better...

Report it whole — Example

```
Computational environment

  Python 3.12, dependencies pinned in uv.lock (committed). Reproduce with:

      uv sync && uv run python run.py

  pandas is held below 3.0 because the copy-on-write default changes the
  recode in src/clean.py; this is revisited when that function is rewritten.

  The pipeline runs on every push in CI on a clean machine, so the claim
  that a fresh checkout reproduces these outputs is tested rather than
  asserted.

  Analysis run on 2026-03-14 with commit a3f9c21.
```

Report it whole

- The last line is the one that makes the rest usable a year later — A result without the commit that produced it can be...

What comes next

- A pinned environment still produces a different answer on a second run if the code reads the clock, draws a random number, or trusts the order files come back in.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/reproducible-workflows/repro-03-pin-the-environment