Introduction to marimo

Alán F. Muñoz

2024/02/04

Outline

Introduction

The case against Jupyter notebooks

- Only 4% of notebooks are reproducible.
- Hidden state
- Hard to parse diffs with standard tools
- They do not scale upon growth in size and number

Similar existing tools

- Streamlit: General-purpose apps
- Papermill: Parameterize Jupyter notebooks
- gradio: Apps with an ML focus

Enter marimo

Officially released at the start of 2024.

Developed at National Accelerator Lab. In need of a reproducible and reusable notebook for iterative, data-heavy code work.

Similar to:

- Observable (Javascript)
- Pluto.jl (Julia)
- Livebook (Elixir)

Main features

What exactly is marimo?

> *"marimo is fundamentally a reinvention of the Python notebook as a reproducible, interactive, and reusable Python program, instead of an error-prone JSON scratchpad."*

marimo dev team

How is it different from Jupyter Notebooks?

- Consistent state
- Built-in interactivity
- Pure Python programs

Reproducibility:

- Serialize package requirements in-notebook (sandboxed notebooks)
- The notebook's input and output state are the same
- Reuse cells within a notebook

Maintainability

- Small changes -> small diffs
- Test notebooks with pytest or doctests

Interactivity

Interactive sync with the Python kernel:

- no callbacks
- no observers
- no manually re-running cells.

Reusability

- Reuse notebooks from command line
- (WIP): Import symbols (functions/classes) into python programs/notebooks

Shareability

- Share notebooks without paying for compute
- Embed notebooks in websites

How can you use it?

- Fetch images as you explore and process profiles
- Low-effort data annotators
- Explore data and statistics before committing to code an analysis
- Create a chatbot to interact with code

QoL features

General features

- Built-in data exploration tools
- Control cells' reactive execution
- Seamless alternation between notebook and app

First-class SQL support

- Mix and match SQL and Python, always coming back to python DataFrames.
- Out-of-the-box interactive DataFrames

Run it on your browser, not through your browser!

- Run github notebooks with one click
  - Prepend marimo.app to any github notebook
  - The WASM app can be hosted on Github Pages

Bring your own editor

- Edit the file and monitor changes on the app/notebook. `marimo edit --watch`
- Integration of autocompletion and code-checking is undergoing.

AI Assistants

- Enable copilots like GitHub Copilot or Codeium
- Generate entire cells using an AI assistant that knows your dataframe schemas A new framework for computational experiments

The downsides

- New and non-standard
- (Soft) Lock on their editor
- Development loop outside their interface is clunky
- Unlike *Jupyter*, the kernel is not accessible in isolation of the interface

WASM limitations

A specific list of python libraries work under WASM

- Most numeric python, but not polars
- Maximum 2GB of memory

Goals of marimo 1.0

1. *Rapid data manipulation with Python and SQL*
2. *Excellent developer experience in the marimo editor*
3. *Excellent developer experience working with notebook files*
4. *Seamless embedding of notebooks and application on web*
5. *Working with data at any scale*
6. *Enterprise-readiness*

Final notes

Marimo vs Jupyter

- .py files vs JSON
- Widgets are synced with the Python kernel, unlike ipywidgets
- There is no hidden state

Take-home messages. Marimo can:

- Accelerate data exploration through native interactive elements
- Increase reproducibility of results
- Reduce the cost and complexity of deploying infrastructure
- Make scientific computing accessible

Where to start?

Try online on marimo.new

or install

```
uvx/pip install marimo
marimo tutorial ui
```

References and resources

- Slides + Additional notes
  github.com/afermg/2025_02_marimo_tutorials
- marimo gallery
- Blog entries
- Design lessons of building marimo
- Guides to transition: Streamlit, Jupytext, Papermill