

The Software Gardening Almanack

Dave Bunten¹, Will Davidson¹, Faisal Alquaddoomi¹, Vincent Rubinetti¹, Gregory P. Way¹

¹Department of Biomedical Informatics, University of Colorado Anschutz Medical Campus



Anschutz

I. Fragile code & sustainability debt

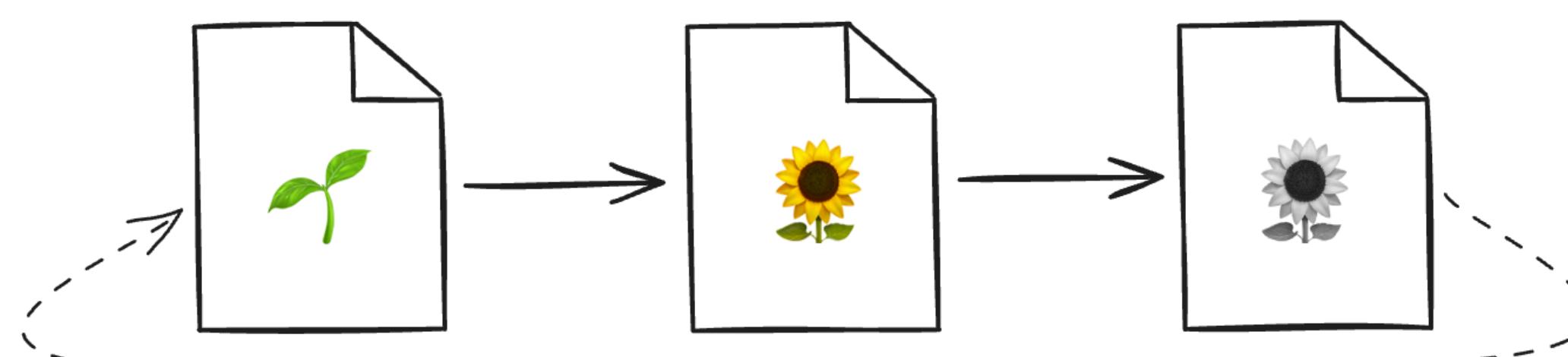


Figure 1: “Software grows and decays.” Repositories accumulate complexity, missing docs, and other issues that erode maintainability over time. The Almanack treats software like a living ecosystem and provides measurements + checks to slow decay and guide care.

Scientific software often becomes brittle: documentation gaps, weak tests, and ad-hoc practices combine with time to create **sustainability debt**. This debt hides defects, slows research, and undermines reproducibility. Existing guidance is scattered and qualitative. Teams need **measurable signals** and **actionable checks** that map to better long-term outcomes.

II. Handbook + Python package

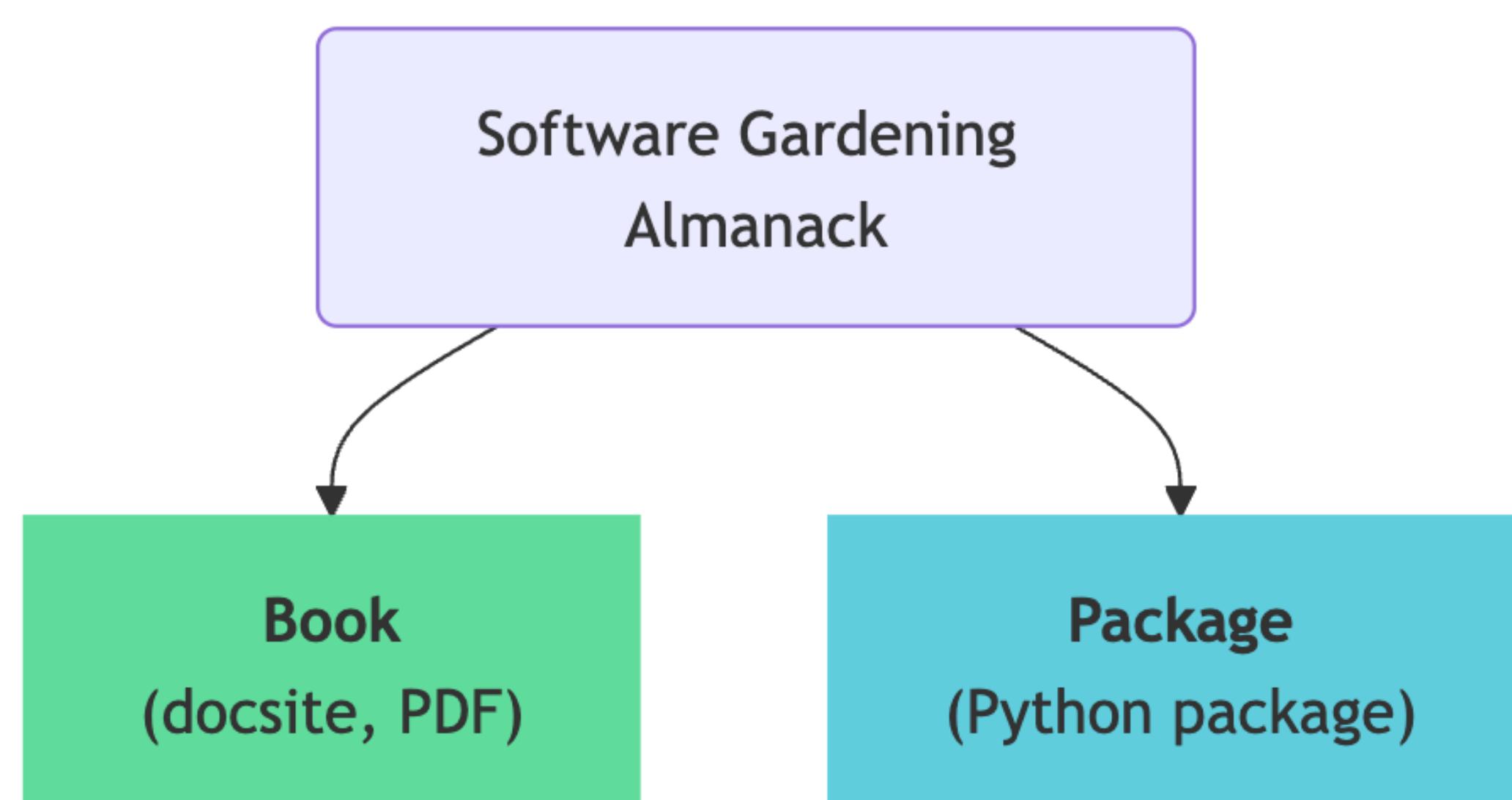


Figure 2: Two parts: a public handbook (education, examples, and rationale) and a Python package that generates metrics and runs checks on any repo. Integrate as CLI, pre-commit hook, or Python API to surface sustainability risks early.

We present The Software Gardening Almanack, an open-source project with two complementary components:

- **Handbook (Jupyter Book)**: teaching the “why” and “how”—concepts, examples, and tutorials. The handbook provides a portable foundation for learning sustainable practices.
- **almanack Python package**: the “do”—compute metrics (including software information entropy), build reports, and run lint-style **sustainability checks** through Python API, CLI, or pre-commit CI.

Together, they help scientific developers quantify maintenance risk, prioritize fixes, and cultivate code that lasts. The Almanack fills an educational technology gap which is endemic in scientific software development today.

III. Getting started with almanack

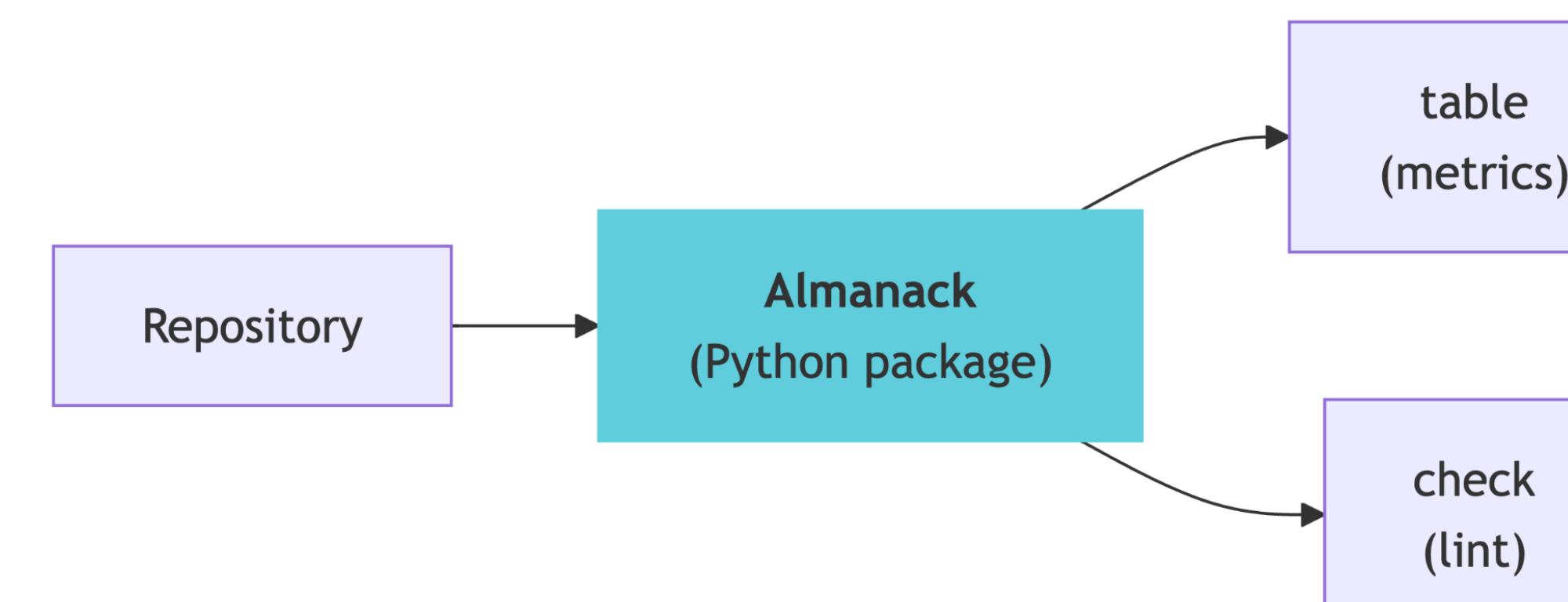


Figure 3: The Almanack package allows you to gather raw metrics for deep analysis or perform a lint-style check on repositories.

Installation

```
# install from PyPI  
pip install almanack
```

The almanack package is pip installable directly from PyPI.

Use the CLI

```
# run lint-style sustainability checks  
almanack check https://github.com/an-org/your-repo  
  
# gather all metrics about a project  
almanack table path/to/repository
```

You can run the almanack using a CLI to gather metrics or perform checks on a repository.

Pre-commit integration

```
# .pre-commit-config.yaml  
- repo: https://github.com/software-gardening/  
almanack  
  rev: v0.1.1  
  hooks:  
    - id: almanack-check
```

The almanack includes a ready-to-use pre-commit hook which can check repositories as part of their continuous integration (CI).

Python API

```
import almanack  
import pandas as pd  
  
# Build a table of repository metrics  
table = almanack.table("path/to/repository")  
pd.DataFrame(table).head()
```

IV. What the metrics capture

- **Repository hygiene:** README, license, citation, contribution guide, code of conduct, docs location.
- **Process health:** CI configuration, build status, test presence, coverage parsing (e.g., Python).

- **Community signals:** contributors, tags/releases, issue tracker availability.
- **Evolution/complexity:** information entropy at file and repo levels to highlight hotspots and decay risk.
- **Scoring & gating:** boolean checks mapped to maintenance direction (positive/negative) plus an overall Almanack score.

V. Visual analytics & examples

The **Seed Bank** demonstrates how to reproduce and explore measures (e.g., normalized Shannon entropy) across thousands of research repositories, relating time, complexity, and engagement.

VI. Typical workflows in a lab

1. **Survey repositories** to establish a baseline metrics table and identify immediate fixes (missing license, broken badge, etc.).
2. **Add guardrails** with pre-commit/CI so regressions are caught early (e.g., missing citation file).
3. **Plan maintenance** using entropy hotspots and failed-check rationales to guide refactoring and documentation.

VII. Real-world value for scientists

- **Reproducibility & trust:** standard checks and reports make expectations visible for collaborators and reviewers.
- **Onboarding faster:** new team members see structure (docs/tests/CI) and can contribute earlier.
- **Longevity:** entropy and process signals help choose where to invest limited maintenance time.
- **Automation-ready:** outputs integrate cleanly with dashboards, CI logs, and PR comments.

VIII. Acknowledgements

We thank contributors and supporters of the Almanack effort, including the **Better Scientific Software (BSSw) Fellowship Program**, U.S. Department of Energy (DOE), National Nuclear Security Administration (NNSA), and the **National Science Foundation (NSF, award 2327079)**; and collaborators across the University of Colorado Anschutz Medical Campus.

- Project: <https://github.com/software-gardening/almanack>
- Handbook: <https://software-gardening.github.io/almanack>
- PyPI package: <https://pypi.org/project/almanack/>