National Laboratory of the Rockies

RdTools

Keeping a 10-year-old open-source project healthy

Lessons from rdtools

Martin Springer · core maintainer, rdtools

2nd Open Source Software Summit · 2026

What rdtools is — and why it matters

An open-source Python library for reproducible analysis of time-series data from solar power plants.

It provides vetted, peer-reviewed analysis routines alongside composable building blocks — with reproducibility as the design priority, so the same inputs yield the same, defensible results.

- **Degradation — is performance declining?**
- **Soiling — how much energy is lost?**
- **Availability — was the system operational?**

THE VETTED ENGINE BEHIND NATIONAL-SCALE RESEARCH

5,272 systems

>29,000 inverters

>29.7 GW of solar analyzed

Mean system 5.9 MW · mean age 5.6 yrs

Powers **PVFleets** — the DOE PV Fleet Performance Data Initiative — with the same code anyone can **pip install**.

A real, sustained project

2016
first release

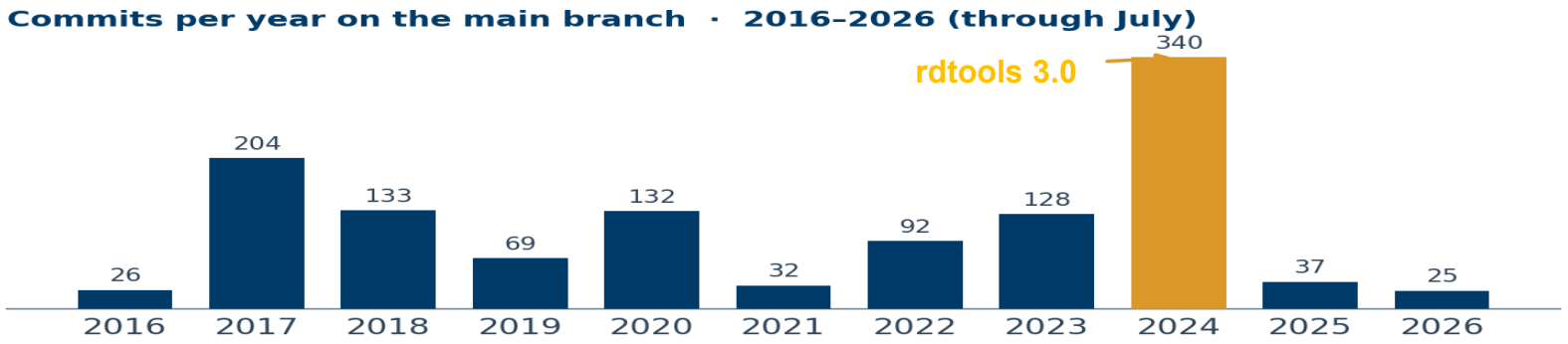
~1,200
commits on main

35
contributors

v3.2.1
latest (2026)

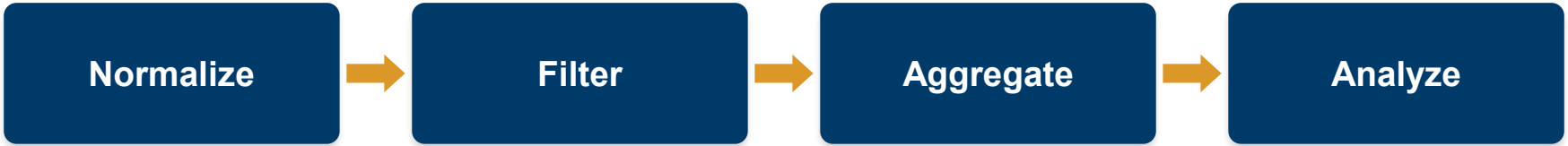
MIT
PyPI + ReadTheDocs

Nearly a decade of sustained, active development — mature enough that the lessons have been tested in practice.



How it's built

The analysis workflow is one simple pipeline — wrapped in a single object, **TrendAnalysis**.



TrendAnalysis — a single guided workflow over the whole chain

MODULES

- degradation
- soiling
- availability
- filtering
- normalization
- plotting

A default workflow — with the primitives still exposed — serves both newcomers and power users.

Engaging contributors & users

THEME 1 · LOWERING THE BARRIER

Reproducible environments (pixi + lockfile)

A single lockfile resolves identically on a laptop and on the HPC — and lets us reproduce a prior analysis with the exact package versions.

Documentation + runnable example notebooks

Users learn by example; contributors gain a starting point. The notebooks serve three roles at once: documentation, tests, and onboarding.

A clear contribution path

A development-branch workflow, issue and PR templates, and PEP8 + NumpyDoc conventions — drawing national-lab, industry, and academic contributors.

Removing as many barriers as possible to turn passive users into active contributor.

Maintaining OSS in the age of AI

THEME 2 · CODING AGENTS AT WORK

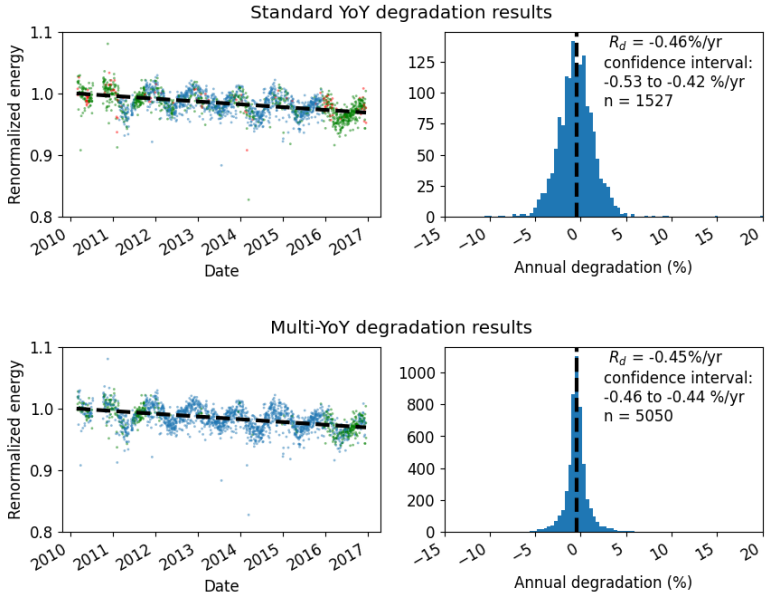
Rather than managing a flood of AI-generated PRs — we use coding agents to sustain and modernize a niche library.

- ✓ **Substantially reduced development time on recent releases (v3.x.x)**
- ✓ **Accelerated first-pass pull-request review**
- ✓ **Made modernization tractable — e.g. the migration to pixi**

Safe because the checks already existed: tests + CI (Codecov), reproducible lockfiles, notebook validation (nbval), human-review PR checklist.

New feature highlight!

multi-yoy of degradation analysis



Shipped in an agent-assisted cycle: multi-year-on-year degradation. On the same dataset, the confidence interval narrows from ±0.05 to ±0.01 %/yr (n = 1,527 → 5,050).

Collaboration among OSS projects

THEME 3 · A SHARED ECOSYSTEM

Doing the foundational methods in the open — and depending on the ecosystem — is what gives a small library real reach.

AI coding agents

A maintainer’s perspective

A small-team effort — not an army.

- Thin bandwidth, and a real bus-factor risk.
- A relentless dependency & version treadmill.

What changed for me: coding agents.

- They speed up the work — faster releases and fixes.
- Code quality has jumped with the latest models significantly.
- They handle the mundane well — like writing tests.

VS Code + Copilot
in-editor completions & quick edits

Claude Code
agentic, multi-file work · CUI-approved models (via LiteLLM)

Takeaways for maintainers

1

Build the guided path

Reproducible environments, a clear default workflow, and runnable documentation lower the barrier for contributors — and let agents operate safely.

2

Let agents carry maintenance

For small teams, the greatest value of AI is sustaining essential maintenance — because tests and reproducibility let you verify the output.

3

Compose, don't conquer

Depend on the ecosystem, implement methods in the open, and collaborate across organizations — so a small library can support large-scale work.

Takeaways: make it trivial to reproduce your project and verify a change — that is what makes both AI tools and collaborators dependable.

RdTools

Get started

install pixi *https://pixi.prefix.dev/latest/installation/*

clone the repo *github.com/NatLabRockies/rdtools*

pixi run lab

www.nlr.gov | martin.springer@nlr.gov

NLR/PR-5K00-101603

This work was authored by the National Laboratory of the Rockies for the U.S. Department of Energy (DOE), operated under Contract No. DE-AC36-08GO28308. Funding provided by U.S. DOE Office of Critical Minerals and Energy Innovation (CMEI) Integrated Energy Systems Office (IESO), Agreement 52789. The views expressed herein do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes"

National Laboratory of the Rockies