Universität Potsdam

Empirical Analysis of Software Quality Assurance Practices in Scientific Computing

Akshay Devkate[1], Mario Frank[2], Anna-Lena Lamprecht[3]

Abstract

As scientific research increasingly relies on software to handle complex data, limited formal training in software development among researchers often leads to issues with documentation, code reliability, and reproducibility. In this study, we conducted an empirical analysis of 2100+ open-source research repositories from the SciCat collection [1], focusing on practices aligned with the FAIR4RS recommendations [2,3], and the 'Best Practices in Scientific Computing' [4] and 'Good Enough Practices in Scientific Computing' [5] as published in PLOS Biology journal. Our preliminary results indicate that not all repositories adhere to the FAIR recommendations, that basic documentation is inconsistent, and that researchers tends to favor dependency configuration files over .lock files or explicitly pinned dependency files for managing project dependencies. Additionally, testing requires more attention, particularly in Python and C++ projects. Continuous Integration (CI) is preferred over Pre-Commit Hooks.

Data: SciCat Collection of Scientific Software Repositories

- SciCat dataset [1]: a curated collection of FLOSS projects from 131 million scientific software repositories, focusing on research software published on JOSS (Journal for Open Source Software).
- 96% from GitHub, with the remaining 4% distributed over Bitbucket, GitLab, and self-hosted GitLab instances. Our analysis focuses on GitHub.
- Python (46%) is the most prevalent language, followed by R (18.5%) and C++ (7%), shaping our study around these three languages.

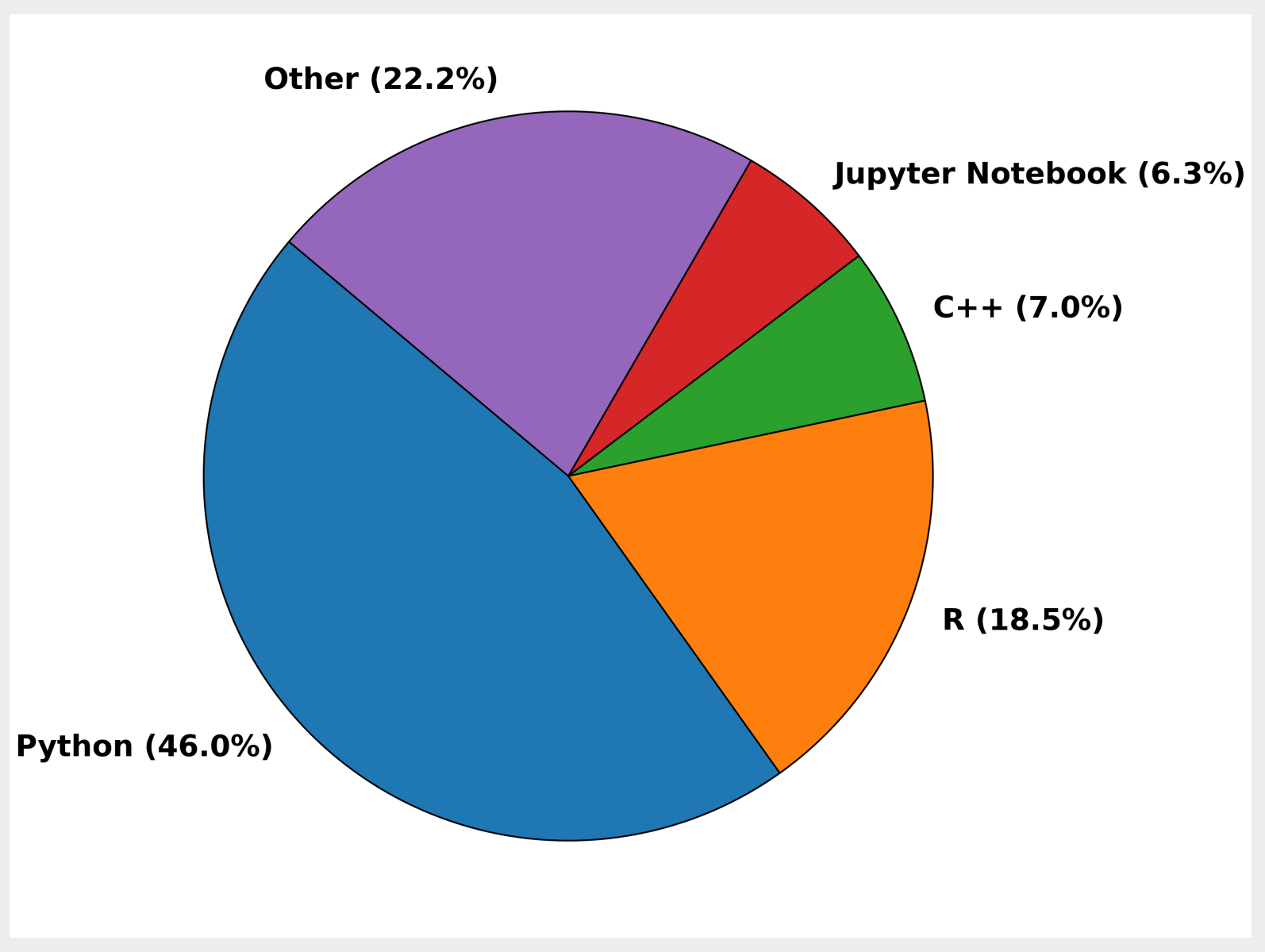


Figure 1: Language distribution

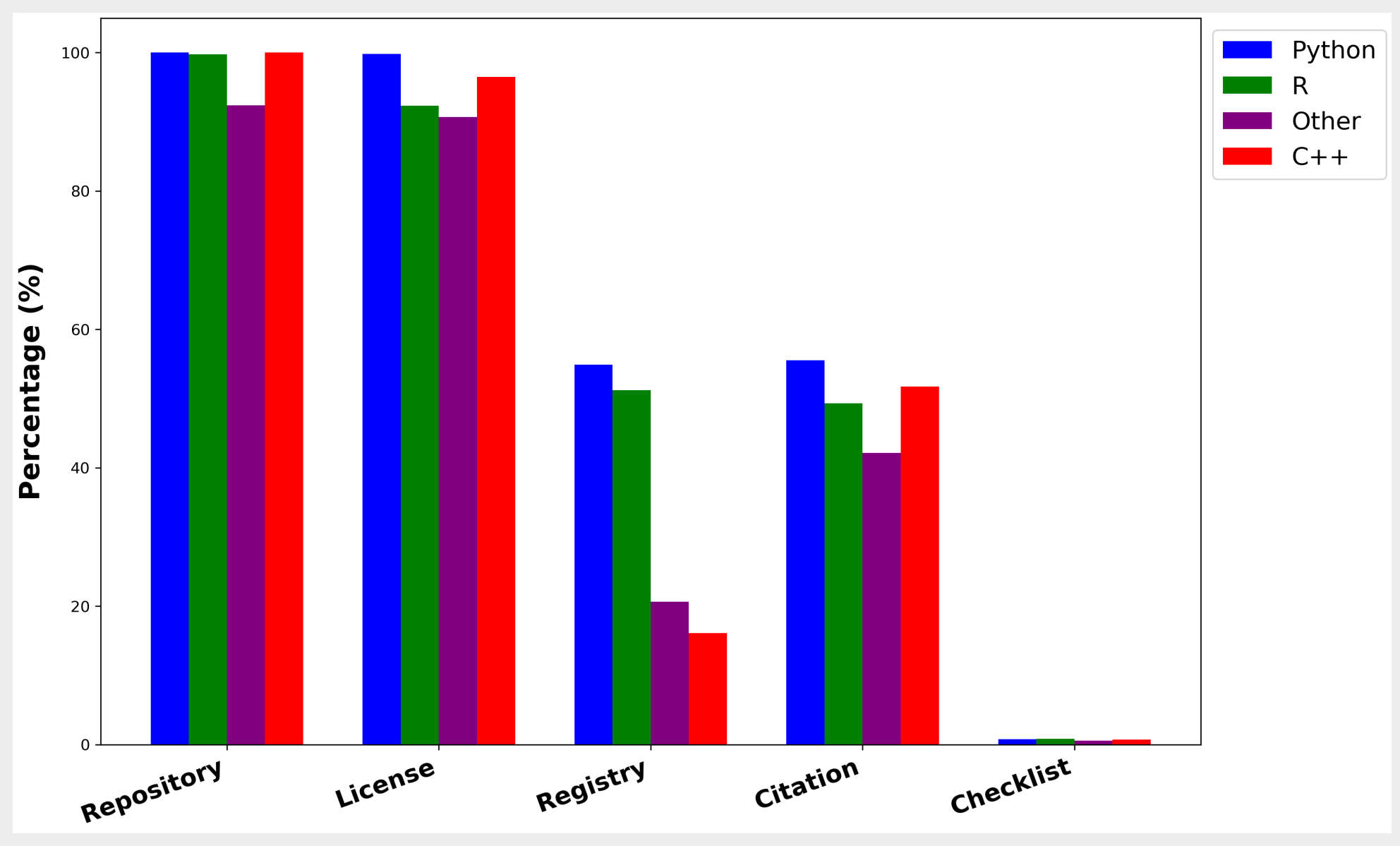


Figure 2: Adherence to FAIR principles

Other Languages: Julia, HTML, JavaScript, MATLAB, C, Java, Fortran, TeX, Rust, Go, C#, Perl, Cython, Ruby, Cuda, TypeScript, Kotlin, Mathematica, Shell, IGOR Pro, OCaml, IDL, Vue, Matlab, PHP, Scala, CMake, GLSL, Nextflow, GAP, Pascal, Tcl, Scheme, Stata, Prolog, JetBrains MPS, Yacc, F#, Gnuplot, HCL, PureBasic, Slash, Modelica, Arc, Verilog, PostScript, Clojure, Makefile, Haskell, Groovy, CSS

FAIR

We assessed the conformance of repositories to the "Five Recommendations for FAIR Software" (https://fair-software.eu/) using the NLeSC's howfairis tool [3] (see Figure 2).

- **Repository:** All the repositories in SciCat dataset were in a public repository (GitHub).
- **Licenses:** The most common were MIT, GPL-3.0, BSD-3-Clause, and Apache-2.0. Permissive and GPL 3.0 licenses were predominant with about 57% and 22 %, respectively.
- **Registry :** The C++ community has low adoption of publishing software on community registries, while Python and R have higher adoption but still require attention.
- **Citation:** Software citation needs further attention across all programming languages.
- **Checklist:** Only 15 (2.5%) repositories have a checklist (OpenSSF best practices) badge.

Table 1: Analysis of documentation coverage

Language	Ratio of code files with brief comment at start				README		Other	
	Less (<25%)	Some (25-50%)	More (50-75%)	Most (>75%)	Installation	Usage	Contributing Guidelines	Code of conduct
Python	31%	16%	20%	30%	60%	54%	54%	2%
R	18%	17%	42%	21%	24%	19%	46%	6%
C++	16%	5%	16%	61%	48%	41%	51%	0.70%
Other	NA	NA	NA	NA	42%	37%	46%	6%

1 2 3

Documentation

1. Brief comments at the start of code files is more common in C++ than in Python and R.
2. Having installation and usage guides in the README file is less common in R projects compared to those in other programming languages.
3. Contribution guidelines were present in roughly half of the repositories across Python, R, C++, and other languages. However, a code of conduct was notably rare, appearing in only a minimal fraction of repositories.

Project Dependencies

- Dependency configuration files (DepConf) are commonly used in Python, R but less frequently defined in C++.
- The use of .lock files or explicitly pinned dependencies to ensure reproducibility is minimal in Python (5%), R (3%) and entirely absent in C++ (see Figure 3 and Table 2).

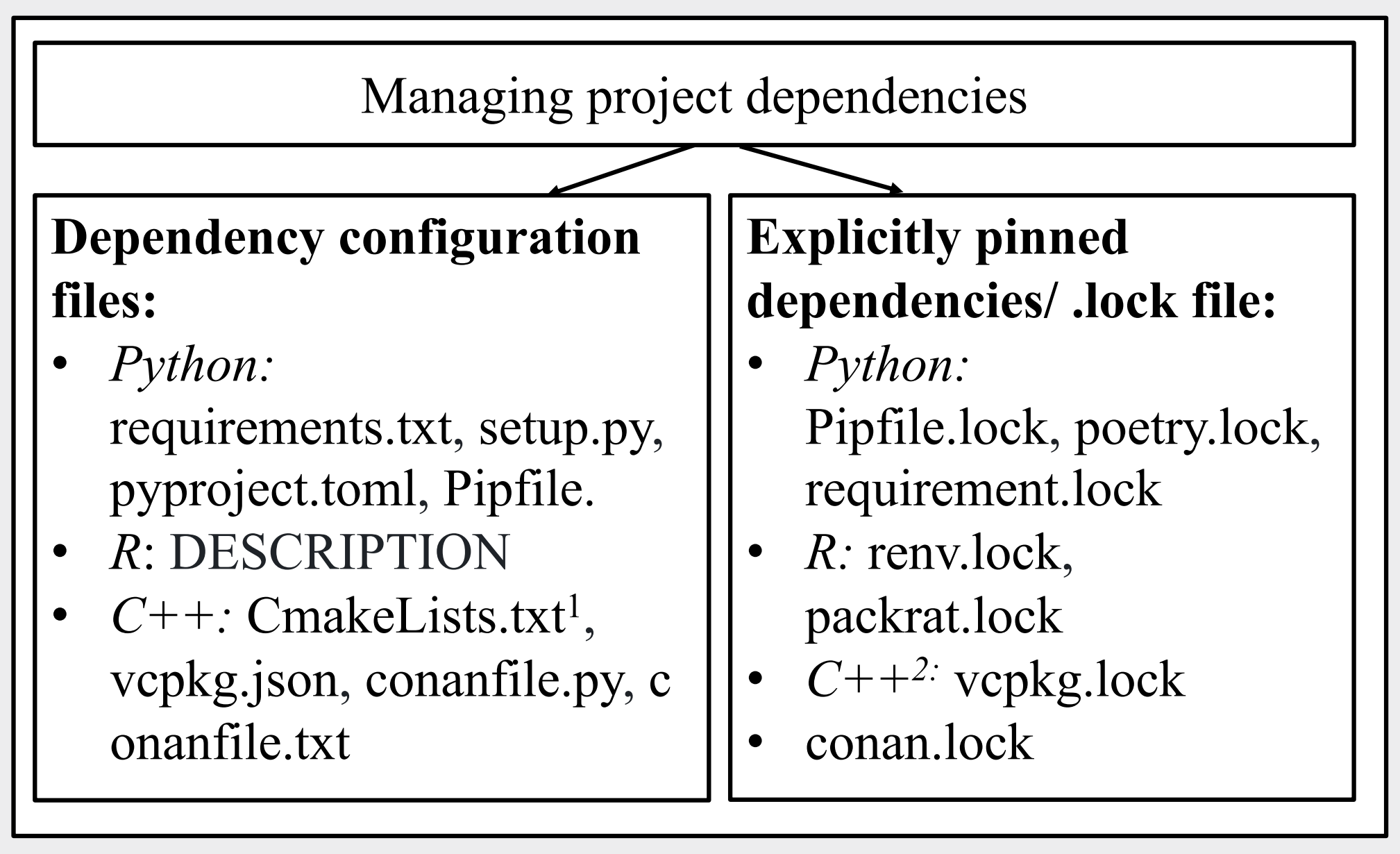

Managing project dependencies
Dependency configuration files:
Python: requirements.txt, setup.py, pyproject.toml, Pipfile.
R: DESCRIPTION
C++: CmakeLists.txt[1], vcpkg.json, conanfile.py, conanfile.txt
Explicitly pinned dependencies/ .lock file:
Python: Pipfile.lock, poetry.lock, requirement.lock
R: renv.lock, packrat.lock
C++[2]: vcpkg.lock
conan.lock

Figure 3: Dependency files

Table 2: Dependencies

Language	Has DepConf files	Has .lock file
Python	96%	4.9%
R	97%	1.6%
C++	53%	0%

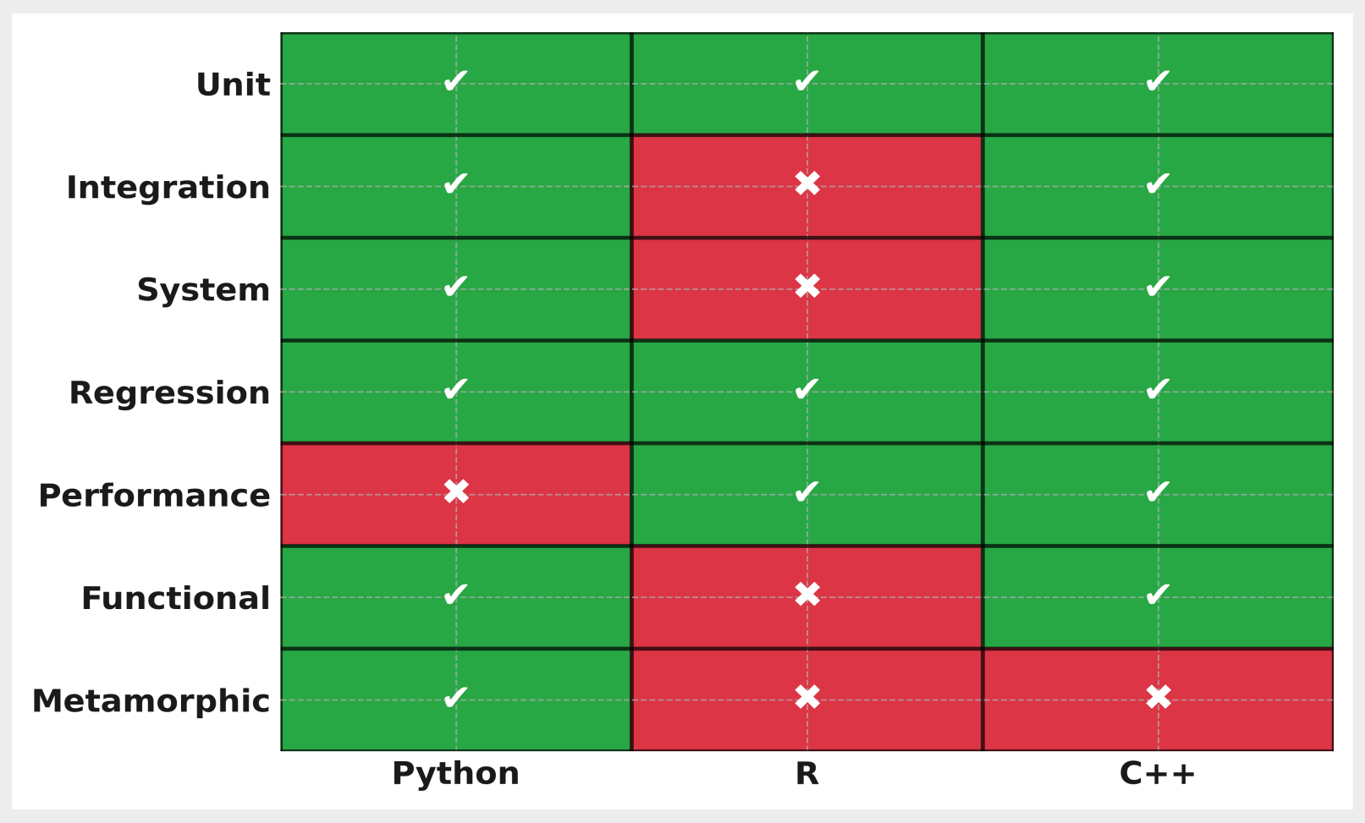
1. From existence of a CMakeList file, we inferred that it includes dependency configuration (e.g., find_package()).
2. CMake does not generate .lock files unlike Conan and vcpkg.

Testing

- Testing appears to be more common in R compared to Python and C++ (see Table 3).
- Recursive analysis of test subfolder names revealed the listed testing types (see Figure 4), with no e2e, acceptance, security, sanity, or mutation testing.
- Few repositories organize test files by testing types, in subfolders; most use module names.

Table 3: Analysis of presence of test folder

Language	Has test folder
Python	57%
R	83%
C++	62%



	Python	R	C++
Unit	✓	✓	✓
Integration	✓	✖	✓
System	✓	✖	✓
Regression	✓	✓	✓
Performance	✖	✓	✓
Functional	✓	✖	✓
Metamorphic	✓	✖	✖

Figure 4: Testing types

Continuous Integration and Pre-Commit Hooks

- Continuous Integration (CI) is more commonly used than Pre-Commit Hooks to enforce testing and code quality across all languages.
- Pre-Commit Hooks are more common in R projects than in other languages.

Table 4: CI and Pre-Commit Hook coverage.

Language	Has Pre-commit hooks	Has CI	Has both
Python	4 %	75%	4%
R	20%	80%	18%
C++	2%	81%	2%

Conclusion & Outlook

- Recommended research software practices (like publishing software on community registries and software citation) are not yet widely adopted.
- Existing checklists like OpenSSF best practices are not tailored for research software, highlighting the need for specialized checklists.
- Clear installation guides and explicitly pinned (.lock) dependency files are crucial for reproducibility, but are often missing.
- Testing remains underdeveloped, lacking clear guidelines on structuring and managing test files and folders.
- Clear need for further empirical investigation of different types of testing using alternative methods, tooling support available across programming languages, and how tools like Continuous Integration and Pre-Commit Hooks can help effectively automate testing tasks.
- This is work in progress, all comments are very welcome! ☺

References

[1] Malviya-Thakur A, Milewicz R, Paganini L, et al. SciCat: A Curated Dataset of Scientific Software Repositories. arXiv preprint arXiv:2312.06382. 2023 Dec 11.
[2] Barker M, Chue Hong NP, Katz DS, et al. Introducing the FAIR Principles for research software. Scientific Data. 2022 Oct 14;9(1):622.
[3] Spaaks JH, Kuzak M, Martinez-Ortiz C, et al. howfairis. Zenodo; 2021.
[4] Wilson G, Aruliah DA, Brown CT, et al. Best Practices for Scientific Computing. PLOS Biology. 2014 Jan 7;12(1):e1001745.
[5] Wilson G, Bryan J, Cranston K, et al. Good enough practices in scientific computing. PLOS Computational Biology. 2017 Jun 22;13(6):e1005510.

1. akshay.devkate@uni-potsdam.de
2. mario.frank@uni-potsdam.de
3. anna-lena.lamprecht@uni-potsdam.de