openstatsguide: Checklist for Good Statistical Software Packages

Developers Value Tests For Software Longevity

Audrey Yeo Te-ying[1], Nils Penard[2], Alessandro Gasparini[3], Daniel Sabanés Bové[4] for openstatsware.org

[1]Finc Research, [2]UCB, [3]Red Door Analytics, [4]RCONIS

Scope

We encourage developers of statistical software packages to follow this minimum set of good practices around:

Documentation, **V**ignettes, **T**ests, **F**unctions, **S**tyle, **L**ifecycle

These keywords can be easily remembered with the mnemonic bridge sentence:

Developers **V**alue **T**ests **F**or **S**oftware **L**ongevity

While the recommendations are rather generic, we focus on functional programming languages and give links to implementations in R, Python and Julia.

This guide primarily addresses developers of statistical packages. Users interested in assessing the quality of existing statistical packages will find complementary "validation"-focused resources on the guide website.

Documentation

... is important for both users and developers to understand all objects in your package, without reading and interpreting the underlying source code.

1. Use in-line comments to generate their corresponding documentation. {**roxygen2**}
2. Do also document internal functions and classes.
3. Add code comments for ambiguous or complex pieces of internal code.

Vignettes

... provide a comprehensive and long-form overview of your package's functionality from a user point of view.

1. Provide an introduction vignette that opens up the package to new users.
2. Include deep-dive vignettes, including describing statistical methodology.
3. Host your vignettes on a dedicated website. {**pkgdown**}

Tests

... are a fundamental safety net and development tool to ensure that your package works as expected, as you develop and use it.

1. Write unit tests for all functions/ classes in your package ("white box" testing).
2. Write functional tests for your user API ("black box" testing). {**testthat**}
3. In addition, ensure good coverage of your code with your tests as a final check. {**covr**}

Functions

... should be short, simple and enforce argument types with assertions.

1. Write short functions for a single and well defined purpose, with few arguments.
2. Use type hints declaring which arguments expect which type of input. {**roxytypes**}
3. Enforce types and other expected properties of function arguments with assertions. {**checkmate**}

Style

... should be language idiomatic and enforced by style checks.

1. Use language idiomatic code and follow the "clean code" rules.
2. Use a formatting tool to automate code formatting. {**styler**}
3. Use a style checking tool to enforce a consistent and readable code style. {**lintr**}

Lifecycle

... management is crucial to build up your user base in a sustainable way. Life cycle management is simplified by reducing dependencies, and should comprise a central code repository.

1. Reduce dependencies to simplify maintenance of your own package and depend on other packages that you trust and deem stable enough.
2. Track dependencies and pin their versions to produce consistent results and behaviours by using more system level that serves as a source of truth for all packages developers.
3. Maintain the change log and deprecate functionality before deleting it.
4. Use a central repository for version control and publication of a contributing guide to enable community contributions.

References

[1] M. Padgham *et al.*, "rOpenSci Statistical Software Peer Review." [Online]. Available: https://doi.org/10.5281/zenodo.5556756

[2] rOpenSci *et al.*, "rOpenSci Packages: Development, Maintenance, and Peer Review." [Online]. Available: https://doi.org/10.5281/zenodo.10797633

[3] J. Manitz *et al.*, "A Risk-based approach for assessing R package accuracy within a validated infrastructure," 2020. [Online]. Available: https://www.pharmar.org/white-paper/

[4] H. Wickham, J. Bryan, M. Barrett, and A. Teucher, "usethis: Automate Package and Project Setup." 2024. [Online]. Available: https://usethis.r-lib.org/

[5] H. Wickham and J. Bryan, *R Packages: Organize, Test, Document, and Share Your Code*, 2nd ed. O'Reilly Media, Inc., 2023. [Online]. Available: https://r-pkgs.org/

[6] H. Wickham, *Tidy design principles*. [Online]. Available: https://design.tidyverse.org/

[7] P. Rodriguez-Sanchez, B. Vreede, and L. de Boer, *A Software Carpentries lesson on R packaging*. 2023. [Online]. Available: https://carpentries-incubator.github.io/lesson-R-packaging/

[8] D. Sabanés Bové *et al.*, "Good Software Engineering Practice for R Packages." [Online]. Available: https://rconis.github.io/workshop-r-swe-zrh/

[10] Wikipedia contributors, "Software verification and validation — Wikipedia, The Free Encyclopedia." [Online]. Available: https://en.wikipedia.org/w/index.php?title=Software_verification_and_validation&oldid=1220844793

[9] D. Sabanés Bové *et al.*, "Improving Software Engineering in Biostatistics: Challenges and Opportunities." [Online]. Available: https://arxiv.org/abs/2301.11791

contact : **audrey@finc-research.com**
visit us : **openstatsware.org**

Scan the QR code to read the online version of **openstatsguide**: