openstatsguide

Checklist for good statistical software packages

Daniel Sabanés Bové (RCONIS) for `openstatsware.org`

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.

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}`

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}`

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}`

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}`

Life cycle

… management is crucial to build up your user base in a sustainable way.

1. **Reduce dependencies** to simplify maintenance of your own package.
2. Maintain the **change log** and **deprecate functionality** before deleting it. `{lifecycle}{fledge}`
3. Use a **central repository** for version control and collecting feedback.

"Developers Value Tests For Software Longevity"

https://www.openstatsware.org/guide.html

Contact: daniel@rconis.com and openstatsware.org

Beta version 0.0-2!
What do you think?
Please give us input!