University of Nebraska - Lincoln
DigitalCommons@University of Nebraska - Lincoln

The R Journal | Statistics, Department of

12-2021

# We Need Trustworthy R Packages

William Michael Landau

Follow this and additional works at: https://digitalcommons.unl.edu/r-journal

 Part of the Numerical Analysis and Scientific Computing Commons, and the Programming Languages and Compilers Commons

This Article is brought to you for free and open access by the Statistics, Department of at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in The R Journal by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.

# We Need Trustworthy R Packages

*by William Michael Landau*

**Abstract** There is a need for rigorous software engineering in R packages, and there is a need for new research to bridge scientific computing with more traditional computing. Automated tools, interdisciplinary graduate courses, code reviews, and a welcoming developer community will continue to democratize best practices. Democratized software engineering will improve the quality, correctness, and integrity of scientific software, and by extension, the disciplines that rely on it.

## Commentary

Most contributors to R (R Core Team, 2021) do not see themselves as software engineers. In a way, this is part of the success of the language. As Dr. Vidoni explains, R developers are usually statisticians, economists, geneticists, ecologists, psychologists, sociologists, archaeologists, and other quantitative scientists who collectively pool a staggering diversity of academic knowledge into a cohesive repository of interoperable software packages. This rich ecosystem attracts a diverse user base and spurs popularity on a global scale.

But quality software still requires software engineering: specification, design, implementation, version control, testing, profiling, benchmarking, and documentation to produce packages worthy of trust. And because of its explosive adoption in recent decades, R needs trustworthy packages now more than ever. In the life sciences, for example, researchers increasingly use R to design, simulate, and analyze clinical trials (The R Foundation for Statistical Computing (2021), Nicholls et al. (2021), Gans et al. (2021), Wassmer and Pahlke (2021)). The resulting claims about safety and efficacy influence the medical treatments of millions of patients.

Fortunately, software engineering has begun to spread among self-described non-engineers. Workflow packages such as **testthat** (Wickham, 2011) and **covr** (Hester, 2020) identify essential but accessible practices and adapt them to an R-focused audience. On top of the popular packages that the article cites, newer specialized ones are under active development. One such example is **autotest** (Padgham, 2021), which automatically generates testing specifications that help developers identify uncommon boundary cases in statistical packages. Another is **valtools** (Hughes et al., 2021), a validation framework in which package developers declare formal requirements and explicitly map each requirement to one or more unit tests. **valtools**, part of the Pharmaceutical Users Software Exchange (PHUSE, Tinazzi et al. (2008)), was created to help R developers in the life sciences meet the requirements of regulatory authorities such as the United States Food and Drug Administration.

Still, key engineering issues remain underexplored for R, many of which fall outside the scope of the article. For example, what are the best ways to translate the logic of an algorithm into a collection of concise pure functions with sufficient encapsulation? Under what circumstances is it beneficial to clone an external function? (Claes et al. (2015) argue that cloning is not always harmful.) When is it appropriate to use ordinary functions, generic function object-oriented programming, e.g. S3 (Chambers, 2014), or more traditional message-passing OOP, e.g. **R6** (Chang, 2020)? Which design patterns are available for OOP and functional programming, how do they apply to R specifically, and which problems can they solve in real-life statistical modeling packages? When an anti-pattern is identified and classified, how can a technical debt taxonomy offer tailored recommendations for refactoring? How exactly does a package author write a specification to communicate the package's architecture and design principles to other developers? How do developers find optimal tradeoffs among automation, coverage, and computation time in unit testing?

As Dr. Vidoni points out, additional research may help translate long-established aspects of traditional software development to the world of R, and graduate courses may help instill this knowledge in new generations of quantitative scientists. Courses could borrow heavily from traditional computer science, especially the long history of object-oriented programming and functional programming. And just a little bit of exposure to a language like Haskell (Marlow, 2010), C++ (Stroustrup, 2013), Java (Gosling et al., 2015), or Python (Rossum, 1995) can help foster a well-rounded perspective. Even if students abandon these languages later on, they will retain pertinent concepts that R programmers seldom consciously utilize: for example, how immutable bindings serve as helpful guardrails in functional programming.

Code review, which the R community underutilizes, also aligns with the article's call to action. Reviews typically happen during a formal gatekeeping process, such as acceptance into CRAN (CRAN Volunteers, 2021), Bioconductor (Huber et al., 2015), or rOpenSci (Ram et al., 2019), or within small teams in order to expedite specific deliverables. There is usually a clear extrinsic need and a clearly identified expected extrinsic outcome. Pedagogical retrospectives are far less common, especially

across different organizations, but they can be eye-opening experiences that substantially improve the awareness and capabilities of the mentees.

New social technologies could increase the frequency and effectiveness of code reviews and raise the collective software engineering skill level. For example, conferences in R, Statistics, and related fields could organize code review workshops, where mentees bring their own projects and experienced mentors provide in-person one-on-one feedback. In fact, entire R conferences could be dedicated to code review. Precedents include the Tidyverse Developer Day (Wickham et al., 2020) and the rOpenSci Unconference (rOpenSci, 2018), in which participants spend the majority of their time collaboratively working on code.

It is also possible to systematize an ongoing community-driven code review process for packages in public repositories. A fit-for-purpose public online forum could carry out ad hoc code reviews, and much like Stack Overflow, support a reward and reputation system for both mentors and mentees. A working group, possibly funded by the R Consortium (R Consortium, 2021) or similar, could kickstart the forum by selecting packages from CRAN, GitHub, etc. and inviting the authors to participate.

In summary, there is a need for rigorous software engineering in R packages, and there is a need for new research to bridge scientific computing with more traditional computing. Automated tools, interdisciplinary graduate courses, code reviews, and a welcoming developer community will continue to democratize best practices. Democratized software engineering will improve the quality, correctness, and integrity of scientific software, and by extension, the disciplines that rely on it.

## Bibliography

J. Chambers. Object-Oriented Programming, Functional Programming and R. *Statistical Science*, 29, 09 2014. URL https://doi.org/10.1214/13-STS452. [p15]

W. Chang. *R6: Encapsulated Classes with Reference Semantics*, 2020. URL https://CRAN.R-project.org/package=R6. R package version 2.5.0. [p15]

M. Claes, T. Mens, N. Tabout, and P. Grosjean. An Empirical Study of Identical Function Clones in CRAN. *2015 IEEE 9th International Workshop on Software Clones, IWSC 2015 - Proceedings*, 03 2015. doi: https://doi.org/10.1109/IWSC.2015.7069885. [p15]

CRAN Volunteers. The Comprehensive R Archive Network, 2021. URL https://cran.r-project.org/. [p15]

M. Gans, A. Clark, R. Krajcik, M. Gotti, and N. Mockler. *tidyCDISC: tidyCDISC: Quick Exploratory Data Analyses on ADaM-ish Datasets*, 2021. URL https://github.com/Biogen-Inc/tidyCDISC. R package version 0.0.0.9000. [p15]

J. Gosling, B. Joy, G. Steele, G. Bracha, and A. Buckley. The Java Language Specification, 2015. URL https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf. [p15]

J. Hester. *covr: Test Coverage for Packages*, 2020. URL https://CRAN.R-project.org/package=covr. R package version 3.5.1. [p15]

W. Huber, V. J. Carey, R. Gentleman, S. Anders, M. Carlson, B. S. Carvalho, H. C. Bravo, S. Davis, L. Gatto, T. Girke, R. Gottardo, F. Hahne, K. D. Hansen, R. A. Irizarry, M. Lawrence, M. I. Love, J. MacDonald, V. Obenchain, A. K. Ole's, H. Pag'es, A. Reyes, P. Shannon, G. K. Smyth, D. Tenenbaum, L. Waldron, and M. Morgan. Orchestrating high-throughput genomic analysis with Bioconductor. *Nature Methods*, 12(2):115–121, 2015. URL https://doi.org/10.1038/nmeth.3252. [p15]

E. Hughes, E. Miller, M. Vendettuoli, and P. Eshghi. *valtools: Automate Validated Package Creation*, 2021. URL https://github.com/phuse-org/valtools. [p15]

S. Marlow. Haskell 2010 Language Report, 2010. URL https://www.haskell.org/onlinereport/haskell2010/. [p15]

A. Nicholls, P. R. Bargo, and J. Sims. A risk-based approach for assessing R package accuracy within a validated infrastructure, 2021. URL https://www.pharmar.org/white-paper/. [p15]

M. Padgham. *autotest: Automatic Package Testing*, 2021. URL https://docs.ropensci.org/autotest/. [p15]

R Consortium. R Consortium, 2021. URL https://www.r-consortium.org. [p16]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL https://www.R-project.org/. [p15]

K. Ram, C. Boettiger, S. Chamberlain, N. Ross, M. Salmon, and S. Butland. A Community of Practice Around Peer Review for Long-Term Research Software Sustainability. *Computing in Science and Engineering*, 21(2):59–65, 2019. URL https://doi.org/10.1109/MCSE.2018.2882753. [p15]

rOpenSci. rOpenSci Unconference 2018, 2018. URL https://unconf18.ropensci.org. [p16]

G. Rossum. Python Tutorial. Technical report, CWI (Centre for Mathematics and Computer Science), 1995. [p15]

B. Stroustrup. *The C++ Programming Language*. Addison-Wesley Professional, 4th edition, 2013. ISBN 0321563840. [p15]

The R Foundation for Statistical Computing. R: Regulatory Compliance and Validation Issues A Guidance Document for the Use of R in Regulated Clinical Trial Environments, 2021. URL https://www.r-project.org/doc/R-FDA.pdf. [p15]

A. Tinazzi, M. Scott, and A. Compagnoni. A 'systematic review' of PhUSE: what PhUSE users made available to the public after three years. *Pharmaceutical Programming*, 1(1):5–8, 2008. URL https://doi.org/10.1179/175709208X334597. [p15]

G. Wassmer and F. Pahlke. *rpact: Confirmatory Adaptive Clinical Trial Design and Analysis*, 2021. URL https://CRAN.R-project.org/package=rpact. R package version 3.1.0. [p15]

H. Wickham. testthat: Get Started with Testing. *The R Journal*, 3(1):5–10, 2011. URL https://doi.org/10.32614/RJ-2011-002. [p15]

H. Wickham, M. Averick, and J. Bryan. Tidyverse Developer Day, 2020. URL https://github.com/tidyverse/tidy-dev-day. [p16]

*William Michael Landau*
*Eli Lilly and Company*
*893 Delaware St, Indianapolis, IN 46225*
*United States of America*
*ORCID: 0000-0003-1878-3253*
will.landau@gmail.com
*URL:* https://wlandau.github.io