Ten simple rules for selecting an R package

Caroline J. Wendt [1,2], G. Brooke Anderson [3*]

1 Department of Statistics, Colorado State University, Fort Collins, Colorado, United States of America
2 Department of Mathematics, Colorado State University, Fort Collins, Colorado, United States of America
3 Department of Environmental & Radiological Health Sciences, Colorado State University, Fort Collins, Colorado, United States of America

* Corresponding author: Brooke.Anderson@colostate.edu

Abstract

Write the abstract here.

Author summary

Write the author summary here.

Text based on plos sample manuscript, see
http://journals.plos.org/ploscompbiol/s/latex

Here are two sample references: [1,2].

Introduction

R is a language and environment for statistical computing and graphics that was developed by statisticians and is collaboratively maintained by an international core group of contributors. Unlike many popular proprietary languages (e.g., MATLAB, SAS, SPSS), R is highly extensible, free and open-source software; the user can access and thus change, extend, and share code for desired applications. Accordingly, a vibrant community of R users has emerged, many of which engage in the development of extensions to the functionality of base R software known as packages. There are many analogies in computing that draw comparisons between programming and culinary arts: recipe structures, coding cookbooks, and the like. To conceptualize packages, imagine you are the chef, R is the kitchen, and packages are the special gadgets which allow you to cook and bake new recipes. R packages are coding delectables that enable the user to perform practical tasks and solve problems with interesting techniques.

Are there R packages for wrangling and cleaning data frames, designing interactive apps for data visualization, or performing dimensionality reduction? Yes! How do you find an R package that will help you train regression and classification models, assess the beta diversity of a population, or analyze gene expression microarray data? The answer is not as simple; there are tens of thousands of R packages. As a natural consequence of the open-source nature of R, there is variation in the quality of different packages among the numerous choices that exist. The advanced R user—having developed an intuition for their workflow—may tend to be relatively confident when searching for and selecting packages. By contrast, a common experience that

characterizes learning R at the outset is the struggle to 1) find a package to accomplish a particular task or solve a problem of interest and 2) choose the best package to perform that task. Even so, there remain obscure and complicated problems that morph selecting an R package into a barrier despite experience.

In coding as in life, we endeavor to make choices that optimize outcomes. Just as one may go about shopping for shoes, deciding which graduate program to pursue, or conducting a literature review, there is a science behind selection. We inform our decisions by assessing, comparing, and filtering options based on indicators of quality such as utility, association, and reputation. Likewise, choosing an R package requires attending to similar details. We outline ten simple rules for finding and selecting R packages so that you will spend less time searching for the right tools and more time coding delicious recipes.

List of 10 rules

(currently in no particular order and not precisely worded)

1. **Consider your purpose**

- What do you want to use the package to accomplish?
- features
- functions
- organization
- package description
- compare similar options

2. **Spend time searching; find and collect options**

- internet searches (keyword "...in R")
- textbooks ("[x] with R" series)
- tutorials
- courses
- social media (#rstats)
- conferences (e.g., RStudio, useR!)
- consult collaborators
- CRAN task views
- Research articles
  - Check which packages have been used in research in your field (provide suggestions for good Google Scholar search queries to identify papers that have used certain packages or that present a package) Alternatively, check the Methods and References sections of papers in your field.
  - Related to that, we could talk about how packages can be cited (the `citation` function produces one in the preferred format for any package). You can look up most packages in Google Scholar to see how many times it's been cited by look-
    ing at the "Cited by" link with the reference. See for example the first listing at https://scholar.google.com/scholar?hl=en&as_sdt=0%2C6&q=dplyr&btnG=
- Blogs
  - posts with overviews of new packages
  - Joe Rickert of RStudio used to regularly highlight interesting new packages (check to see if he still does).

  - Mara Averick of RStudio advertises cool new R things; check if any focus on packages.

3. **Check how it's shared**

- check repository association
  - CRAN
  - Bioconductor
  - GitHub
  - GitLab (alternative to GitHub)
  - ROpenSci (runs its own repository, only includes ones it has peer-reviewed)
  - Self-hosted repositories (can be made with the `drat` package; see paper)
  - purpose of repositories: mechanisms of quality control that regularly check code and manage webs of dependencies
- alternative ways R packages can be shared (not repo)
  - zipped file
  - collaborators

4. **Explore the availability and quality of help**

- help files
- `help()`
- vignettes
- `DOCUMENTATION` file
- "cheatsheets" from RSudio
- RDocumentation (key word search, task views)
- websites (e.g., `packagedown`)
- `bookdown` books
- compare documentation completeness and resource quality
- find ways to get help beyond initial documentation
- listservs
- online communities
- Stack Overflow (frequency of questions and answers on the topic)
- See if GitHub repo for the package seems responsive to Issues
- `Rcpp` is an example of high-quality help
  - associated book
  - maintainer, Dirk, is known to be responsive to user questions (listserv)
  - ample documentation including examples to get started

5. **Verify the credibility of the author(s)**

- team or single author (robust team?)
- associations (e.g., academia, industry, labs)
- expertise
- reputation
- experience (e.g., portfolio of packages, history of R development)
- role in R development (e.g., RStudio, regarded bio labs)
- profiles (e.g., GitHub, Google Scholar, Research Gate, Twitter)

6. **Investigate the package development**

- best practices

- unit testing (manage quality control)
- dependencies
- coverage by tests
- number of versions
- clarity of NEWS (explain updates and changes)
- GitHub Issues (history, resolution)

7. **Read, research literature, seek evidence of peer review**

- publications
- package itself
- papers about the package
- ROpenSci
- associations with books or publications from scientific publishers

8. **Quantify how established the package is**

- dependencies
- versions
- updates
- number of downloads
- popularity
- leaderboard
- ranking systems

9. **Put the package to the test**

- explore code
- interact with trial and error
- get a feel for using it in context of your goal
- open-source framework
- GitHub mirror of CRAN as an alternative to downloading zipped package file
- How interoperable it is with other packages that you want to use?
- some packages do what they do really well, but it is hard to use them with the tidyverse or other outside packages
    - S3 or S4 objects that make it hard to work them into a pipeline where their functions are not the last step
- packages that help with interoperability
    - `broom` and `biobroom`: make it easier to put numerous statistical functions into a larger tidyverse workflow
    - Max Kuhn's `caret` package for machine learning—adds a layer that lets you use the same interface to work with machine learning algorithms from lots of different packages that otherwise all have slightly different interfaces for calling the algorithm and working with the results.

10. **Develop your own package**

- necessity
- innovative idea
- novel approach or method
- unique and specialized purpose

References

1. Feynman RP, Vernon Jr. FL. The theory of a general quantum system interacting with a linear dissipative system. Annals of Physics. 1963;24: 118–173. doi:10.1016/0003-4916(63)90068-X

2. Dirac PAM. The lorentz transformation and absolute time. Physica. 1953;19: 888–896. doi:10.1016/S0031-8914(53)80099-6