Ten Rules Outline and Research

Caroline Wendt

List of 10 rules

(currently in no particular order and not precisely worded)

1. **Consider your purpose**

- features
- functions
- organization
- package description
- compare similar options
- particular task
- general purpose
- improve base R (efficiency, elegance)
- expand base R (do more)
- What do you want to use the package to accomplish?
- What functionalities of base R could be improved?
- What functionalities would you like to add to base R to expand what you can do?
- How do you want to work with inputs and what are the desired outputs?

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

- internet searches (keyword "... in R")
- textbooks ("[x] with R" series)
  - RStudio books
- tutorials
- courses
- social media (#rstats)
- conferences (e.g., RStudio, useR!)
- consult collaborators
- CRAN task views (based on certain disciplines and methodologies)
- 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 looking 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
  - find posts with overviews of new packages
  - RStudio packages developed, maintained, or contributed to by the RStudio team categorized by purpose
  - RStudio webinars

  - ROpenSci packages site includes links to blogs, tutorials, and examples for using specific packages
  - R-bloggers
  - Revolutions daily blog
  - R Weekly has specific sections for new packages and updated packages
- People
  - Joe Rickert of RStudio (Ambassador at Large) used to regularly highlight interesting new packages (check to see if he still does).
    * what makes a great R package
    * active on Twitter (`@RStudioJoe`)
  - Mara Averick of RStudio (`tidyverse` developer advocate) advertises cool new R things; check if any focus on packages.
    * I cannot find any recent activity or posts by Mara beyond Twitter (`@dataandme`), but some of her sites have interactive info about cool packages and how to use their features.
    * There are also several videos and posts from her that relate to conferences and unconfs.

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
- number of citations in Google Scholar

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
- standardize
- automate
- novel approach or method

- unique and specialized purpose
- copying and pasting functions repeatedly
- collection of functions you use often
- desire to easily share code, data, documentation, and tests with others
- “personal R packages”
- suggested package development resources for readers:
  - R Packages: Organize, Test, Document, and Share Your Code by Hadley Wickham
  - Ten simple rules for making research software more robust (Taschuk & Wilson)