Developing an R package: a tutorial

Introduction

Ghislain Durif and Jean-Michel Marin

January 2023

Laboratory of Biology and Modeling of the Cell (LBMC), ENS Lyon, France and CNRS – Institut Montpelliérain Alexander Grothendieck (IMAG), University of Montpellier

Requirements

- **R** (the latest version if possible[1], `4.2.2` since 2022-10-31)
- You can use the R command line combined with any text editor, but we recommend to use an R-oriented IDE[2] like **Rstudio**
- **Note for Windows users:** you can update R from within R with the `installr` package and you will need to install Rtools to enable all R development functionality

[1]Keep your software up-to-date! If you need an older version of R for a specific project, use appropriate tools like containers, it should be an exception not a habit.

[2]Integrated Development Environment

References

- **Official R documentation**: *Writing R Extensions*
- Karl Broman **tutorial**: *R package primer* (web version and sources)
- Hadley Wickham and Jenny Bryan **book**: *R packages* (web version and sources)
- Hilary Parker **tutorial** on writing R packages
- Rstudio **cheatsheets** on package development and Rstudio IDE

What is an R package?

- a library containing a set of R functions (and possibly more) implementing functionality not available in default R functions[3]
- a standardized way to distribute R codes (for other users)

[3]or reimplementing existing functionality in a different way

Where can I find R packages?

- the `CRAN` (Comprehensive R Archive Network): official repository for R packages

```r
install.packages("devtools")
```

- `bioconductor`: bioinformatics-oriented package repository
- any git forge: github, gitlab
- on your colleagues' computers[4]

[4] if they develop in R

Why R packages?

- The **best way** to write and distribute **R code** with **documentation**, **examples**, **tests**, etc.

- A good practice[5] when coding in R:
  - your project is structured (code, data, doc), easier to use and re-use
  - documentation is essential (including for your future self)
  - your code is standardized, you can check it and test your functions
  - easy management of dependencies
  - etc.

[5] even for codes you don't plan to publish/distribute

How to write an R package?

A wide variety of tools to help you:

- Rstudio IDE built-in development features
- R base built-in tools: build (`R CMD build`), check (`R CMD check`)
- Some packages to develop packages:
  - `usethis`: to automate package and project setup
  - `devtools`: complete collection of development tools
  - `roxygen2`: to document your code and generate help pages
  - `testthat`: to implement automatic tests of your functions
  - `remotes`: to install package from anywhere (integrated in `devtools`)
  - `rmarkdown` and `knitr`: to create detailed documentation materials and notebooks (code showcase)

Outline

1. The essentials to write your package (`.md` and `.pdf`)

- Getting started
- R package structure
- Workflow

2. Going further with your R package development (`.md` and `.pdf`)

- Getting started
- Digression: Good practice for software development and programming (not just in R)
- Test your functions
- Sharing (your code) is caring
- Advanced documentation
- Non R code
- Control your R environment