Developing an R package: a tutorial

Ghislain Durif (https://gdurif.perso.math.cnrs.fr/)

July 2022

CNRS – LBMC (Lyon, France)

License

Training materials

https://plmlab.math.cnrs.fr/gdurif/devRpkg

https://github.com/gdurif/devRpkg

Note for the reader

Many hyperlinks are directly embedded in the slide contents.

Requirements (1)

- **R** (the latest version if possible[1], `4.2.1` since 2022-06-23)
- You can use the R command line combined with any text editor, but we recommend to use an R-oriented IDE[2] like **Rstudio** or RKWard
- All content presented here have been tested on a Linux environment but should work on **any OS**
- **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

Requirements (2)

- See the script `install_requirements.R` to install the packages that will be used in the tutorial

- To (re)generate the slides, see the scripts[3] `.setup.R` to install the requirement, `.build.R` to build the `pdf` slides

[3] or the attached `Makefile` if you are confortable with using `make`

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[4]
- a **standardized** way to **distribute** R codes (for other users)

[4] 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

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

- `bioconductor`: bioinformatics-oriented package repository
- any git forge: github, gitlab, etc.
- on your colleagues' computers[5]

[5] 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**[6] 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.

[6] 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** (non-exhaustive):
  - `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