

R packages development training

Mathieu Depetris

Table of contents

Welcome!	3
Introduction	4
I Training framework and requirements	5
1 Operating system and working environment associated	6
1.1 Requirements software	6
1.2 Additional software	8
2 Overview of R packages used	11
References	16
Appendices	17
A Not define yet	17

Welcome!

Welcome to the online support to learn how to organise your R code through the creation of an R package. You will find that this process is easier than you can imagine and with a little bit of pratiques you will be able to create your own package. In addition, you will discover that this way open you the field to develop reproducible code, documented faster and simply your processes at many levels and learn how other people can download and use it.

All this website and material associated if free to use, licenced under the [CC BY-NC-ND 4.0](#) licence. At least you will find below several training sessions available if you want a guided explanation for R package creation.

To conclude, this support was developed according to the book *R Packages (2e)* [1] written by [Hadley Wickham](#) and [Jennifer Bryan](#). An online version is available through this [link](#), don't hesitate to take a look.

Introduction

To do

Part I

Training framework and requirements

1 Operating system and working environment associated

Before going into the fun of R package development, take a moment to setup our working environment and overview some interesting software which can simplify our life.

First, all the training and test that we will do among this book will be executed with Windows Operating System (OS). When you develop your package, one of the aims could be to share or propose your work to other peoples and you have to take into account about the variability of the processes among OS, software for example R dependencies. We will put in place several processes to support us in our task but keep in your mind that processes that work on an OS could not work in the same pattern in another one. Futermore, one of our tasks among the lifecycle of our package will be to maintain a maximum of compatibility, for example through integration of update or evolution in our code.

Developing R packages assume that you have a little experience with the R software and not start from scratch with it. Even if you see that R package development is not so hard that it looks like, you will find a lot of resources on the internet to support you if you want to start with R and develop your skills in it.

1.1 Requirements software



For the training we will use the *R* version 4.5.0 for Windows available through the web page of [The Comprehensive R Archive Network](#). You will find also precompiled binary distributions for Linux and macOS.



In addition, we will use the Integrated Development Environment (IDE) *RStudio*. Today this software is one of the most used in the R communities and includes a lot of things like a console, syntax-highlighting editor that supports direct code execution, and tools for plotting, history, debugging, and workspace management. Furthermore, recent evolution like the creation of [Posit](#) compagny make the software ecosystem more friendly connected with R and Python. For the training we will use the RStudio version 2025.05.0+496 “Mariposa Orchid” for Windows OS. You can download the one for your configuration throught the [Posit download webpage](#).

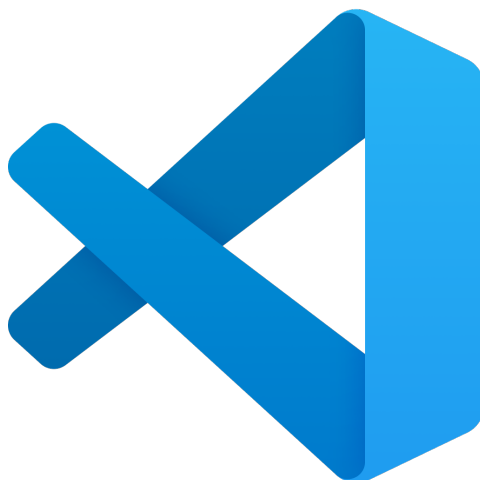
Under Windows OS, it's necessary to install the [RTools](#) software. This is a toolchains for building R and R packages from source on Windows. And don't need it on macOS and Linux because theses OS have usually pre-installed compilation tools. You can download it from the [official link](#). Be aware that you need to take the correct version according to our R version installed (for this training it would be the RTools 4.5).



At least, we will use Git, for version control system, to host our R package and apply several processes of Continuous Integration (CI), most of them already prepared by the R communities. Today, this is a very precious tool that helps developers track changes in their code, collaborate efficiently, and manage different versions of a project. During this training we will use one of them. You can download the software on the official website through this [link](#).

1.2 Additional software

Several software are not necessary to create your own R package but bring you a lot of improvement and at least simplify your life.



The first one is [Visual Studio Code](#) (or VS Code) which is a free, lightweight, extensible code editor developed by Microsoft. He supports a wide variety of programming languages such as R, JavaScript, Python, C++, Java, HTML/CSS. In addition is cross-platform compatibility, support Git and other version control systems and one of his best advantages is to provide extensions to add functionality and customize the environment. If you want you can download it from this [official website](#) and if it's possible according to your configuration prefer a system installer version. For this training I will use the version 1.100.2 on Windows OS.



According to the use of Git, we will use one of them call [GitHub](#). You will see that a lot of processes were developed for work natively with GitHub and facilitate your R package creation. If you want to use another one, like a GitLab hosted by your company for example, you can of

course but you will have to modify and adapted by yourself several processes propose during this training. In addition we will use here a Graphical User Interface (GUI) for managing Git repositories called GitHub Desktop. This is a free, open-source application which simplifies current Git operations, making it easier for developers to clone repositories, commit changes, create branches, and push updates without using the command line. Like its name suggests is working natively with GitHub repositories but you can also use it for connection with other Git like GitLab. Officially is available for Windows and MacOS but you can also find adapted version for Linux on the internet. For download it goes through the following [link](#).

2 Overview of R packages used

This section aim to introduce brively which R packages environment we will use. Since several years, the R communities evolve in many interesting ways and today if you want to write your own package there are a lot of support that you can find.

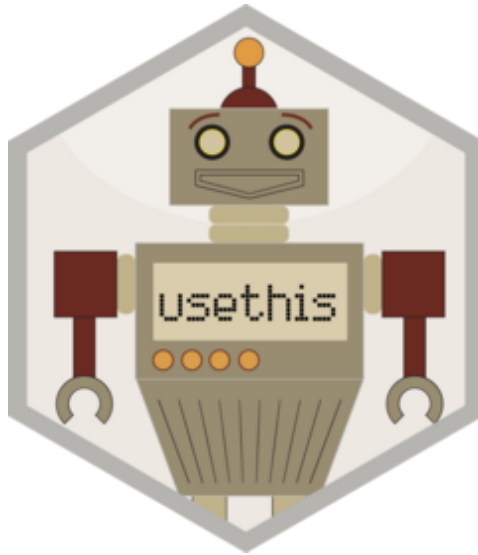


First of all, it's difficult to talk about package without talk about the [tidyverse](#) approach. Tidyverse is a [collection of R packages](#) designed for data science. They share an underlying design philosophy, grammar, and data structures. If you use R, it's almost sure that you have already use this kind of packages, maybe even without knowing them. Even if there are none mandatory for R use, these packages bring a lot of benefices and at the end make data science easier, faster, and more fun (yes, you will see).

So far we don't go deeper into all the useful R packages of the tidyverse collection, but only focusing on these related to the R packages development.



The first one is called [devtools](#) [2]. The aim of this package is to make R package development easier by providing R functions that simplify and expedite common tasks. In practical, if you use the Rstudio IDE you will not interact directly with this package because the majority of common processes will be integrated automatically in the Rstudio environment and become “click button” processes.



The second one is the sub-package you are most likely to interact with directly. It be called [usethis](#) [3] and is aim is to automates repetitive tasks that arise during project setup and development. The idea is to, if it's possible, use functions available by the package, rather that modify directely the code, to reduce error and be sure to have the last version of process strucutre.





There are two more important packages related to design of the documentation: [roxygen2](#) [4] and [pkgdown](#) [5]. With the first one through the use of tags, the code and the documentation are adjacent and dynamically inspects the objects that it's documenting. We will go back to them into the following sections but keep in your mind that theses two tools simplify and automate the documentation creation and evolution.



The last one is called `testthat` [6] and focusing on the field of testing your code, which can be painful and tedious, but it greatly increases the quality of your code.

In addition we will use or talk about several other packages, for example `lintr` [7] or `renv` [8], but it's more a case-by-case utilisation and we will discuss them when the time comes.

References

1. Wickham H, Bryan J (2023) R packages: Organize, test, document, and share your code (2nd edition). O'Reilly Media
2. Wickham H, Hester J, Chang W, Bryan J (2022) [Devtools: Tools to make developing r packages easier](#)
3. Wickham H, Bryan J, Barrett M, Teucher A (2024) [Usethis: Automate package and project setup](#)
4. Wickham H, Danenberg P, Csárdi G, Eugster M (2024) [roxygen2: In-line documentation for r](#)
5. Wickham H, Hesselberth J, Salmon M, Roy O, Brüggemann S (2025) [Pkgdown: Make static HTML documentation for a package](#)
6. Wickham H (2011) [Testthat: Get started with testing](#). The R Journal 3:5–10
7. Hester J, Angly F, Hyde R, et al (2025) [Lintr: A 'linter' for r code](#)
8. Ushey K, Wickham H (2025) [Renv: Project environments](#)

A Not define yet

Appendice a