How to make an R package

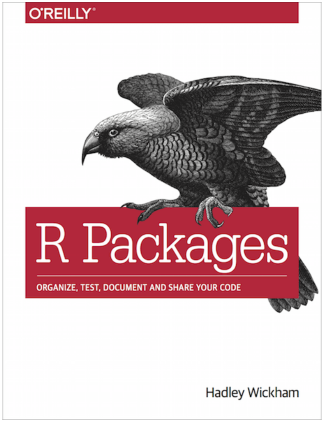
Ben Lambert

May 18, 2023

Why produce an R package?

- As of March 2023, there were over 19,000 packages available on the Comprehensive R Archive Network, or CRAN;
- In R, the fundamental unit of shareable code is the R package;
- If your code is in an R package (and on a Github repository / CRAN), it is straightforward to install;
- But useful even if you don't want to share code, because R packages come with conventions which force you to develop in standardised ways;
- Useful for a publications: R package to house functions that are then called in a *separate* repository that runs analyses.

What we will cover today? Essentially, Chapter 2 in


O'REILLY

R Packages

ORGANIZE, TEST, DOCUMENT AND SHARE YOUR CODE

Hadley Wickham

Plus a few more things, including software licenses and continuous integration testing.

Aims

We aim to create an R package, "regexcite", which contains a few simple functions for processing strings. In doing so, we will:

- create the functions themselves;
- show how to set up version control;
- explain workflows for package development including:
  - function documentation;
  - unit testing;
  - package documentation.
- if time allows, we will discuss setting up continuous integration testing via Github Actions.

Repository

Visit: `https://github.com/ben18785/how_to_make_r_package`

Note, this also contains a finished package example in the "completed R package" folder.

Software licenses

Important because:

- you need to pick a license that declares how you want your code to be used;
- if you include code written by someone else, you need to respect the license that it uses.

Crucial: if you don't include a license, default copyright laws apply, which means that no one is allowed to make a copy of your code without your express permission / similarly for use of others' code.

Types of license

Two broad categories of open-source license:

- *Permissive* licenses are very easy going. The MIT and Apache licenses are the most common modern permissive licenses;
- *Copyleft* licenses are stricter. The most common copyleft license is the GPL which allows you to freely copy and modify the code for personal use, but if you publish modified versions or bundle with other code, the modified version or complete bundle must also be licensed with the GPL.

How to choose a license

- People can use your code with minimal restrictions $\rightarrow$ MIT license;
- All derivatives and bundles of your code are also open source $\rightarrow$ GPL license.

Example: CHASTE software

- Developed at Oxford; general purpose simulation package aimed at multi-scale, computationally demanding problems arising in biology and physiology: https://www.cs.ox.ac.uk/chaste/;
- Formerly had a GPL license, which meant that industry was unlikely to use it;
- Subsequently changed to a more permissive license;
- Downside was that CHASTE cannot use any GPL libraries;
- A friend used CHASTE in his PhD, which used *fftw* which is GPL code;
- His work could not then be incorporated into CHASTE without finding an alternative to *fftw*.

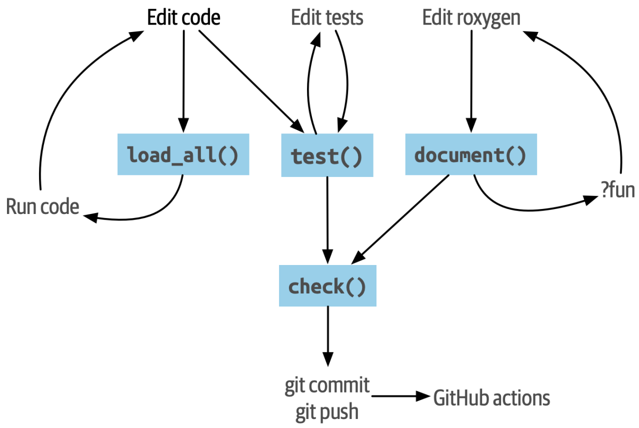
Overall, choosing a software license is not easy!

Code bundling

Before you bundle someone else's code into your package, you need to first check that the bundled license is compatible with your license. Five cases that Wickham and Bryan list:

- If your license and their license are the same: it's OK to bundle;
- If their license is MIT or BSD, it's OK to bundle;
- If their code has a copyleft license and your code has a permissive license, you can't bundle their code. You'll need to consider an alternative approach, either looking for code with a more permissive license, or putting the external code in a separate package;
- If the code comes from Stack Overflow, it's licensed with the Creative Common CC BY-SA license, which is only compatible with GPLv3;
- Other licenses require more thought.

Key workflows



Reproduced from “R Packages“, 2nd Edition, Hadley Wickham and Jennifer Bryan.