Albert Dorador Spring 2017

# Creating an R Package in 7+1 easy steps

As a complement, or if you would like to go deeper, check John Muschelli's excellent 1-hour video on Youtube.

https://www.youtube.com/watch?v=OIirKRgIsdc

This file I am writing is basically a summary of John's video, but you will most likely need to watch the video to fill in the details about the "preamble", type of license, etc.

If you already have templates and know those kind of things (perhaps you have created a package in the past and just want a refresher), then this 2-page file on its own will probably suffice and save you >1 hour.

Finally, for more great tips check http://r-pkgs.had.co.nz/description.html by Hadley Wickham.

---

**Step 0:**

Create the R functions you would like to include in your package.

Important: Functions to be used in package must be delivered without require/library statements. Then you add this information in the preamble of the function script.

I recommend writing an appropriate "preamble" inside each function script before creating the package.

**Step 1:**

Go to File / New Project… / New Directory / R package, and fill in the form, adding the functions you have created and want to include in your new R package.

**Step 2:**

Go to Build / Configure Build Tools…

Check the two boxes (one relating to devtools, and another to Roxygen). When you check the Roxygen box, probably a small window will open with some more configuration options, and I recommend that you check all of those boxes too. In the field titled "Check Package – R CMD check additional options:" type -- as - cran (that is, two consecutive dashes, 'as', dash, 'cran'). This makes sure that the checks performed on your package are as strict as the ones that CRAN will perform before publishing your package.

**Step 3:**

Go to Build / Clean and Rebuild.

Check on the View pane (usually, located on the bottom right) a file called NAMESPACE. Open it and see if it only contains one line of code (written automatically by Roxygen); if that's the case, I suggest that you select this file and delete it (using the appropriate buttons in that very same pane). Clean and Rebuild again. Check the

NAMESPACE file again: now it should contain a list that begins with the functions you have included in your package (exported) and then all the packages you use for your functions to work (imported). In any case, do not manually modify that file. It must be created by Roxygen automatically. At least this is my suggestion.

**Step 4:**

Still in the View pane, click on the file called DESCRIPTION.

You'll have to complete appropriately the basic template that Roxygen automatically provides, and perhaps add some more fields as needed. Note: If your description spans multiple lines, indent them with 4 spaces.

**Step 5:**

In the View pane, click on R to access the functions included in your package.

You will (probably) have to write from scratch an appropriate "preamble" before the function definition. The preamble must be similar in nature to the basic template that Roxygen produced in the DESCRIPTION file.

**Step 6:**

Go to Build / Check Package to make sure all is fine and ready for submission.

The R CMD results should be 0 Errors, 0 Warnings, and 0 Notes. Otherwise, it will not be accepted for publication on CRAN, most likely.

**Step 7:**

Create the .tar file CRAN will ask you to submit as your package.

To create .tar file, simply go to Build / Build Source Package. That's it! You will be informed where it has been created.

Now you can go to the top right and close the current project. You are done.

**Finally…**

Go to https://cran.r-project.org/submit.html (or the equivalent url in case this one no longer works at the time you are reading this) and submit your package.

That's it. If your package is useful and there is no other package that does the same thing (or better) then your package should be approved. In any case, you will be contacted at the email address you have provided as maintainer in the DESCRIPTION file, first to confirm you have indeed submitted a package for publication (and not a robot, for example), and then to be informed whether your package is now on CRAN or some fix is required (or any other objection).