Creating an R Package

An introduction to creating your own R package

Dominique Maucieri

05 April, 2022

Contents

Setup

Welcome to this tutorial on how to create an R package. To begin you will need to ensure you have all the needed packages installed, and then load the packages.

```
install.packages(c("devtools", "roxygen2", "usethis", "testthat", "covr"))
```

```
library(devtools)
library(roxygen2)
library(usethis)
library(testthat)
library(covr)
```

These are the packages needed to build an R package, though if your package functions will depend on other functions, you will need to add those as you go as well.

The Name

Determining the name of your package can be very challenging, as it will be how your package will be known to everyone that uses your functions.

Time and care should be put into your package name choice. The available package has a great function called available() which you can use to ensure that there isn't already a package with the name you want or unintentional meanings to your package name.

Your name must fit 3 criteria:

1. Only letters, numbers and periods (though periods are not recommended)
2. Start with a letter
3. Cannot end with a period

We are going to create an example package called standarderror, so you can practice checking that package name with this code:

```
install.packages("available")
library(available)

available::available("standarderror")
```

This shows that "standarderror" is valid on all platforms and nothing was found on any of the online databases either. So this is a good name to continue with.

Creating the package skeleton

To create the skeleton of your new package, you only need one line of code. You will need to change `/path/to` to the location path were you want this package on your computer

```
usethis::create_package("~/path/to/standarderror")
```

Looking at this directory, there are a few files that were created

- .Rbuildignore
  - Files we need but will not be included when building the package
- .gitignore
  - Ignores some files created by R and RStudio
  - Harmless
- R/
  - Folder containing the .R files
- standarderror.Rproj
  - RStudio project
- .Rproj.user
  - May not have
  - A directory used by RStudio internally
- DESCRIPTION
  - Package metadata
- NAMESPACE
  - Declares your package exports and the imports from other packages
  - DO NOT EDIT

Editing the DESCRIPTION file

At first your description file will look something like this:

Package: standarderror
Title: What the Package Does (One Line, Title Case)
Version: 0.0.0.9000
Authors@R:
*First Last first.last@example.com [aut, cre] (YOUR-ORCID-ID)
Description: What the package does (one paragraph).
License: `use_mit_license()`, `use_gpl3_license()` or friends to pick a license

Encoding: UTF-8
Roxygen: list(markdown = TRUE)
RoxygenNote: 7.1.2

But lets update this so it is more functional. The package name looks great but lets add this title: Calculation of Standard Error. The version number ends in 9000 which signifies this package is in development. Next you can edit the Author section with your name email and ORCID if you have one, but I will include how I have filled it out so you know what to add. If you were creating a package with collaborators you would also add them to this spot too, there are a lot of resources online for adding more authors and people with other roles. We can also add a description of what this package will do and add a license. For this example we will use the MIT license but for your own work you should look into which license is best for you.

```
usethis::use_mit_license()
```

This should leave us with a DESCRIPTION file that looks something like this (but with your name):

Package: standarderror
Title: Calculation of Standard Error
Version: 0.0.0.9000
Authors@R:
person("Dominique", "Maucieri" , "dominiquemaucieri@gmail.com", role = c("aut", "cre"), comment = c(ORCID = "0000-0003-1849-2472"))
Description: This package will aid in calculation of the standard error of means.
License: MIT + file LICENSE
Encoding: UTF-8
Roxygen: list(markdown = TRUE)
RoxygenNote: 7.1.2

Creating a function

We are going to create the function std.err(). To create a new function, you use the use_r() function from the usethis package in your console. It will create the file for you and open it.

```
usethis::use_r("std.err")
```

Now in this blank .R file, we will add the code for our function. If you are unfamiliar with how to code functions, there are many great resources online including Functions in Advanced R and this tutorial from our Coding Club

We want our function to receive a vector of numbers and calculate the standard error of the mean of that vector. To calculate standard error, it will be the standard deviations divided by the square-root of the sample size, which written as a function will look like this:

```
std_err <- function(x){

  x.std.err <- sd(x) / sqrt(length(x))

  return(x.std.err)

}
```