An Introduction to Developing R Packages

Women in Statistics and Data Science, 2017

October 6, 2017

Christina Knudson, Ph.D.
University of St. Thomas

Lindsey Dietz, Ph.D.
Minnesota Federal Reserve

Haema Nilakanta, M.S.
University of Minnesota

We have produced the R package wisdom as our contribution to the Women in Statistics and Data Science conference. A small minority of R packages on CRAN are maintained by women. We hope this document can help guide women through some basic steps of R package production.

1 Before Starting

1.1 Design Doc

Before you even touch your computer, develop a clear vision of your project. Put this vision and all the nitty gritty details into a "design document." The design document is the document you will refer to every time you sit down to program your package. Ideally, your design document will be so thorough that you will only think about programming according to that document and you will not need to think about statistics or data science anymore. The design document is essential for long-term projects so that you can communicate to your future self. A design document is also essential if you are working on a team: it will ensure that one function can call another seamlessly and with no mismatching arguments, even if the functions are written by different people.

A design document should have an overview of the purpose of the package as well as all the details. Decide exactly what you want your package to do. Then, carefully plan out each function that you will include in your package. For each function,

- write down all equations that function relies on.
- decide the inputs and outputs, as well as the properties of each. For example, will the input be a matrix, vector, or scalar?
- decide whether you will provide any defaults for the arguments. If you do, decide what the defaults will be.
- write pseudocode and try to imagine if there is anything tricky you need to consider.

- decide whether it will be available to the user or not. You might write helper functions that will be called by other functions. Users might not see these helper functions, and you will not need to write documentation for them later.

We also highly recommend you brainstorm tests and include these in the design document. You want to test your code every step of the way to ensure every command and function is doing what you think it is doing. Testing both small chunks of code and entire functions will help you isolate bugs or other mistakes. The specific tests you use will depend on the goals and functions of your package. As an example, if you are calculating the value of a function and its first and second derivatives, you can use the method of finite differences to check that the derivatives are in the right neighborhood.

1.2 Version Control

We highly recommend you use version control while you work. This will help you track your changes and revert to previous versions of your project. Some methods of version control also make collaboration smoother. Our preferred method of version control is git. Free public Github accounts are available for your remote repository.

2 Creating the Package

When you inspect the `wisdom` folder, you will find three folders (`tests`, `R`, and `man`) and two files (`DESCRIPTION` and `NAMESPACE`).

2.1 R Files

2.2 Documentation

Each function that you export (make available to users) must be documented. The package skeleton ...

2.3 Description

2.4 Namespace

List all functions in the package that you want to make available to users.

2.5 Tests

Ensuring that your package does what you think it does is essential. You have written your functions generally, but you can test the functions by comparing them against specific examples. In the wisdom package, we wrote `mPower` to calculate x^m for any x and any m. To check that it is calculating these powers correctly, we choose a few specific examples and check the `mPower` results against the results we calculate ourselves.

When you write these tests, try to think creatively. It is easy to imagine testing the square of a positive number. What other values of x and m might someone enter? They can enter any real x, so we test $x < 0$. Additionally, the easiest power to imagine is integer m, but we do not require that m is an integer. The test in the `wisdom` package also checks the function for a few values of m between 0 and 1.

3 Checking the Package

When you are satisfied with your code, documentation, description file, and namespace file, it is time to build and check the R package. The first step is to create a "tarball," a compressed version of your package. To do this, navigate to the parent directory of your R package and type the following:

```
R CMD build wisdom
```

The time needed to build your package will increase as you add functions and tests to your package. Next, you will check your package by checking the tarball you have created. Type

```
R CMD check wisdom_1.0.tar.gz
```

to check your package. The number in the tarball name will depend on the version number of your package. The `wisdom` package is version 1.0, as the description file shows, so the tarball name includes 1.0.

4 Additional Considerations

4.1 Submitting to CRAN

If you intend to submit your package to CRAN, you will need to alter your check slightly to meet CRAN's specifications. Build your tarball as usual and check it with the following:

```
R CMD check wisdom.1.0.tar.gz --as-cran
```

When your package check returns no errors, check your package on other platforms. For example, you can use WinBuilder to check your package on the Windows platform. Once you are confident your package produces no errors, you may upload the tarball to CRAN. Do not upload a package that cannot pass `R CMD check`!

4.2 Computational Stability

Before programming, it is wise to look at each equation in your design document and consider the computational stability. For example, the log likelihood is much more stable than the likelihood itself. As another example, the variance calculation

$$\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

is much more stable than the variance calculation

$$\left[\frac{1}{n}\sum_{i=1}^{n} x_i^2\right] - [\bar{x}]^2 .$$

4.3 Licenses

4.4 Other Stuff?