Tech Guide: Writing an R Package

STAT 5400

Nov 16, 2020.

Introduction

The use of R is highly replied on the development of the huge variety of R packages, which include code, data, documentation, vignettes, etc. As of Nov 16, 2020, there are 16599 available R pacakges on Comprehensive R Archive Network (CRAN, https://cran.r-project.org/).

It is time for us to build our own R packages. It is always a good habit to organize our code in a package. This manner not only makes our code more standardized (saving our time), but also easier to distribute.

In this class, you have installed many packages through the `install.packages` function. We will now create an R package, which has a Linux-type extension `.tar.gz` (called tar files, which is a compressed version of a tarball).

R package devtools

Let us first install the R package `devetools`.

Windows users need to install Rtools (https://cran.r-project.org/bin/windows/Rtools/) and set environment variables. Mac users need Xcode from App Store `xcode-select --install`. See details in https://www.r-project.org/nosvn/pandoc/devtools.html for questions regarding the installation of `devtools`. Also make sure you are using the latest version of R before you install `devtools`.

If you still have trouble with installing `devtools`, you may try to remotely access the departmental server and try to build your package there.

R package structure

There are several components in an R package.

- R Code (`R\`) This directory contains all the R files.
- Package metadata (`DESCRIPTION`) This file stores important meta information about this R package, for example, title, version, description, license.
- Namespaces (`NAMESPACE`) This file is used for the package namespaces.
- Documentation (`man\`) This directory contains the documentations of R functions.
- Complied files (`src\`) This directory is used if you call C or Fortran, etc, in your package.

An R package may also contain other directories like 'data' and 'vignettes'.

Creating an R package

We now build a package based on the function `ridgereg`, which solves ridge regression based on SVD decomposition. We simply use the function name as the package name. When you create a "real" R package, try your best to have a good name, which should be easy to remember and self-explanatory.

1. We first create a directory `ridgereg`, and then create a sub-directories: R. To do so, you may also try `create_package(path)`, where the argument is the path to the directory `ridgereg`.
2. We put two R files into this directory, `ridgereg.R` and `predict.ridgereg.R`.

   Remark. In this example, we have two functions `ridgereg` and `predict.ridgereg`. We also define the class (type) of the `ridgereg` function output to be `ridgereg` (although the names of the function and output class do not have to be same). When we apply `predict.ridgereg`, we directly call `predict(fit)`, where `fit` is the object produced by `ridgereg`. This approach is the `S3` object-oriented system in R. When a generic function `predict` is called, the S3 system dispatches to a specifc function `predict.ridgereg` due to the type of the object.

S3 is a commonly used and also simpliest object oriented system in R. Other systems include S4 and S5, and details about the object oriented system in R can be seen in http://adv-r.had.co.nz/S3.html.

3. We put a file names `COPYING` into the main directory `ridgereg`. You may use the file as it as, and the distribution of your package will be under GPL-3 license (https://tldrlegal.com/license/gnu-general-public-license-v3-(gpl-3)).

4. We put a file named `DESCRIPTION` into the main directory `ridgereg`. You may edit the following code to make this file.

```
Package: ridgereg
Type: Package
Title: Algorithm for fitting ridge regression
Version: 0.0.1
Date: 2020-11-16
Author: Boxiang Wang <boxiang-wang@uiowa.edu>
Maintainer: Boxiang Wang <boxiang-wang@uiowa.edu>
Description: Implements singular value decomposition to efficiently solve ridge regression. This packag
License: GPL-3
```

5. Now we are going to generate the namespace file and R documents. To ease our work, we resort to `Roxygen2`.

   We add roxygen comments to our R files. The roxygen comments start with `#'` and have several components such as `@param`, `@return`, `@examples`, `@export`, and so on. Run the following command to generate `man/ridgereg.Rd` and `man/predict.ridgereg.Rd`, as well as the `NAMESPACE` file.

   Set the working directory as the `ridgereg` folder.

```r
install.packages("devtools")
library(devtools)
document()
```

6. We now build our R package. Set the working directory to the parent directory of `ridgereg`

- On Windows, run

```r
shell("R CMD build ridgereg")
shell("R CMD INSTALL ridgereg_0.0.1.tar.gz")
```

- On LINUX or other UNIX-based platforms, run

```r
system("R CMD build ridgereg")
system("R CMD INSTALL ridgereg_0.0.1.tar.gz")
```

`0.0.1` is the version number you specified in ‘DESCRIPTION’.

- You may also run the code above in terminal or cmd directly without using `shell` or `system`.
- Check your package. Include the option `--as-cran` if you are going to submit your package to CRAN.

```
R CMD check ridgereg_0.0.1.tar.gz
R CMD check --as-cran ridgereg_0.0.1.tar.gz
```

Submit your package to CRAN https://cran.r-project.org/submit.html when you feel it is ready!