Step by Step Tutorial to creating R Packages

Heng Wang

Michigan State University

Introduction

- R is an open source statistical software
- R provides functions to perform statistical operations
  - Classical (regression, logistic regression, ANOVA, etc)
  - Modern (neural networks, bootstrap, genomic selection, etc)
- Can be easily extended by make new packages

- To install an R package, use function

  install.packages()
- For example, to fit a mixed model, we could use function **mixed.solve()**. It requires package “**rrBLUP**”.

Steps to Build an R package

- Step 1. Prepare your functions
- Step 2. Build the structure of the package using **package.skeleton()**
- Step 3. Edit **DESCRIPTION** File
- Step 4. Edit the help File
- Step 5. Preparation for Windows Users (**RTools**)
- Step 6. Build and install the R package
- Step 7. Check the R package
- Step 8. Add functions and data to a package

Build an R Package
-- Step 1. Prepare your functions

- Before you write your functions, clear the working space using **rm(list=ls())**.
- Write your function. Load all the data you want to include in the package.
- Set working directory to the position containing the **.R** file.

Build an R Package
-- Step 2. package.skeleton()

- Run **package.skeleton(name, list)**.
- For example: **package.skeleton(name="cum", list=c("my.cumsumprod", "xvec.example", "output.example")**
- Or, **package.skeleton(name="cum", code_files="cumsumprod.R")**
- A new folder **cum** is built. If just run **package.skeleton()**, then **anRpackage** will be built.

Step 2 (Cont.)

- Inside **cum / anRpackage** you many find several folders:
  - **R**: contains R code files
  - **data**: contains data files
  - **man**: contains documentation files (.Rd)
  - You may also have **src** folder, if your function contains C, C++, or FORTRAN source.
  - Other files: **tests**, **exec**, **inst**, etc.

Step 2 (Cont.)

- … also some files.
  - **Read-and-delete-me** : contain instructions for following steps.

    * Edit the help file skeletons in 'man', possibly combining help files for multiple functions.

    * Edit the exports in 'NAMESPACE', and add necessary imports.

    * Put any C/C++/Fortran code in 'src'.

    * If you have compiled code, add a useDynLib() directive to 'NAMESPACE'.

    * Run R CMD build to build the package tarball.

    * Run R CMD check to check the package tarball.

    Read "Writing R Extensions" for more information.

  - **DESCRIPTION**: manual file for the users.

  - **NAMESPACE**

Build an R Package
-- Step 3. Edit **DESCRIPTION** File

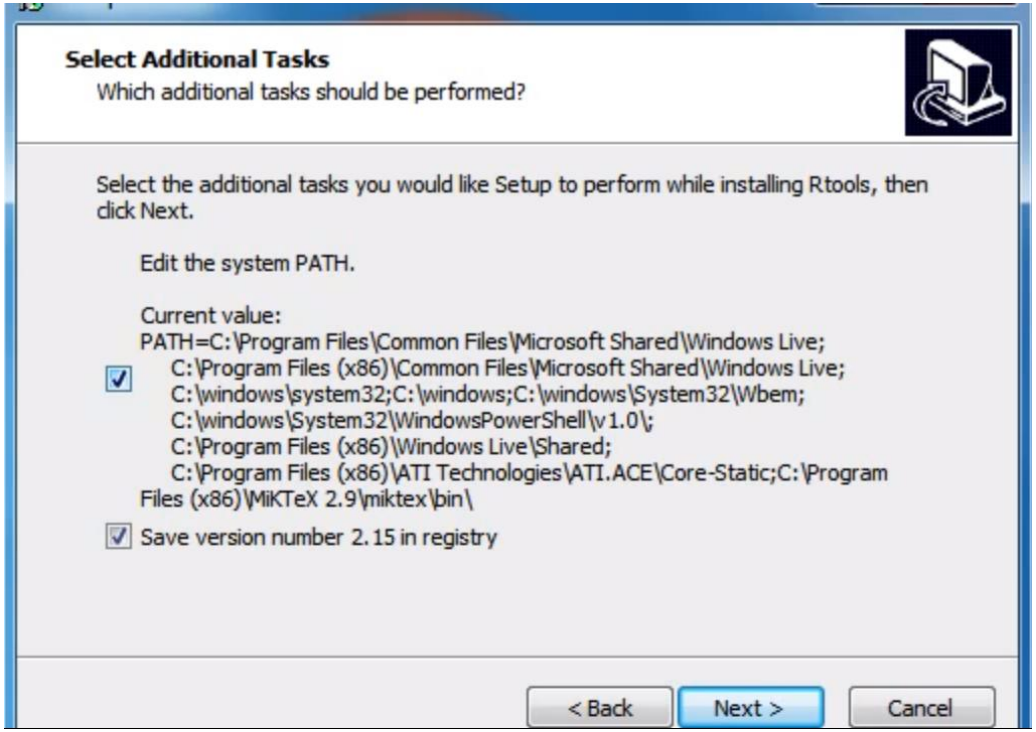
- Package: cum
  -- name of the package
- Type: Package
- Title: What the package does (short line)
  -- contains no more than 65 characters
- Version: 1.0
  -- a sequence of non-negative integers, like: 1.0.2, 1-0-2
- Date: 2013-02-27
  -- Date that the package was created. Today's date by default
- Author: Who wrote it
  -- all the authors, no limit
- Maintainer: Who to complain to yourfault@somewhere.net
  -- one name and an email address
- Description: More about what it does (maybe more than one line)
  -- Description of the package, no length limit
- License: What license is it under?
  -- Usually GPL-2 (GNU General Public License Version 2), which is good for CRAN / Bioconductor. Check "Writing R Extensions" for all license abbreviations.

Build an R Package
-- Step 4. Edit the help File

- Fill the content in each category
- Delete the comments or instructions. Change the default content
- Do this for each **.Rd** file in **man** folder.

Build an R Package
-- Step 5. Preparation for Windows Users

- Download and install **Rtools.** http://cran.r-project.org/bin/windows/Rtools/
- **Attention!** Check the checkbox to update the current **PATH**.



Step 5 (Cont.)

- Change the **PATH** in **Control Panel.**
- Click **System**, then **Advanced system settings.**
- Click the **Advanced** tap in the prompt window. Then click the **Environment Variables**.
- In **PATH**, click **Edit**…
- **C:\Windows\System64\;c:\Rtools\bin;c:\Rtools\gcc-4.6.3\bin;C:\Program Files\R\R-2.15.1\bin\x64;**c:\Rtools\perl\bin;c:\Rtools\MinGW\bin;c:\R\bin;c:\Rtools\MinGW;c:\Perl\bin;c:\Program Files\MiKTex 2.6\miktex\bin;C:\Program Files (x86)\SSH Communications Security\SSH Secure Shell

Build an R Package -- Step 6. Build and install the R package

- In search box, type **command prompt**
- In **command prompt**, change directory to the place that contains the R package
- Build R package using **R CMD build pkgName**. For example I use **R CMD build cum**. A **tar.gz** file is built under the working directory.

Step 6 (Cont.)

- Install the R package using **R CMD INSTALL pkgName**. Here I use **R CMD INSTALL cum_1.0.tar.gz**.
- If any error occurs, check the .**Rd** file. Then re-run **R CMD build, R CMD INSTALL.**

Build an R Package
-- Step 7. Check the R package

- Install Miktex / (Mactex) package **inconsolata** using **mpm --verbose --install inconsolata**.
- Check the R package using **R CMD check pkgName.** I use **R CMD check cum**.
- In **R** environment, type **library(pkgName)**. For example, **library(cum)**.
- You can type

  ?cum

  ?my.cumsumprod

  ?xvec.example

  ?output.example

Build an R Package -- Step 8. Add functions and data to a package

- Change the working directory to the folder that contains your new functions and/or data.
- Copy the functions into working space.
- Run **prompt()** to the new function, i.e., **prompt(cumadd)**. Now you have a help file for **cumadd**.
- Edit the **.Rd** help file.
- Move the **.R** file and the **.Rd** file to the package folder. Put the **.R** file in the **R** folder. Put the **.Rd** file in the **man** folder.

Step 8 (Cont.)

- Read the data file into the working space.
- Save the data as an **.rda** file.
- Create the help file using **prompt()** function.
- Edit the **.Rd** help file.
- Move the **.rda** file and the **.Rd** file to the package folder. The **.rda** file goes to the **data** folder. The **.Rd** file goes to the **man** folder.

Step 8 (Cont.)

- Build and install the package again.
  - **R CMD build cum**
  - **R CMD INSTALL cum_1.0.tar.gz**
  - **R CMD check cum**
- In R console, type

  library(cum)

  ?cum

  ?my.cumsumprod

  ?xvec.example

  ?output.example

  ?yvec

  ?cumadd

Questions?

Thank you!