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()

Steps to Build an R package

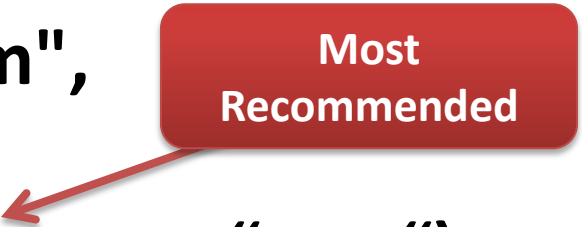
- Step 1. Prepare your functions
- Step 2. Build the structure of the package using **package.skeleton()**
- Step 3. Edit **DESCRIPTION** File
- Step 4. Edit the help File
- Step 5. Preparation for Windows Users
- Step 6. Build and install the R package
- Step 7. Check the R package
- Step 8. Use the R 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")**
- Or, just simply **package.skeleton(name="cum")**
- 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/manual 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 of the package.

- **NAMESPACE**: You can edit it to hide some of the functions.

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: 2014-05-30
  -- 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.

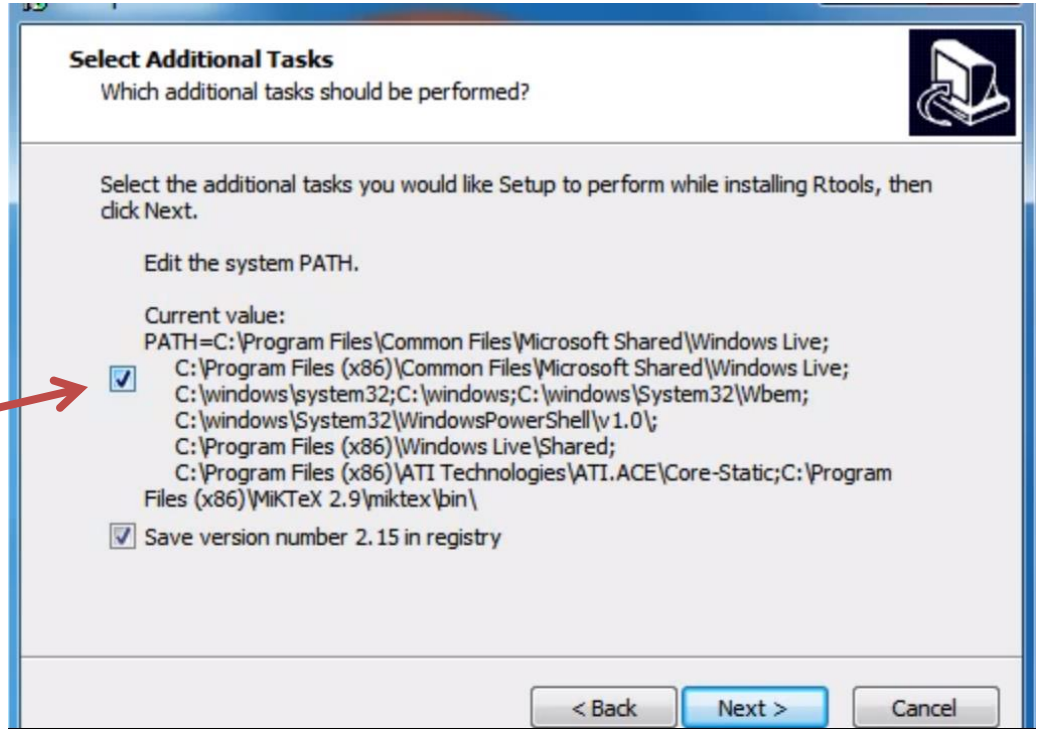
Build an R Package
-- Step 4. Edit the **.Rd** File

- Do the similar thing to all the **.Rd** files in **man** folder.
- Delete the comments or instructions. Change the default content.
- Note 1: All the content in \examples{} should be compliable, or there will be an error later when you check the R package.
- Note 2: All the original comments need to be deleted, or there will be an error.
- Note 3: Do not leave a blank section. You can delete the sections that are not applicable.

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**.



Check this box!

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\SysWOW64\;c:\Rtools\bin;c:\Rtools\gcc-4.6.3\bin;C:\Program Files\R\R-3.0.3\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.
- Install the R package using **R CMD INSTALL pkgName**. For example, **R CMD INSTALL cum_1.0.tar.gz**.
- If any error occurs, check the **.Rd** file. Then delete **cum_1.0.tar.gz**, and 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.**
- If any errors or notes, check and edit the **.Rd** files according to the notes, and then re-run **R CMD build, R CMD INSTALL**.
- You cannot skip this step because the pdf manual file is generated in this step.

Build an R Package
-- Step 8. Use the R package

- In **R** environment, type **library(pkgName)**. For example, **library(cum)**.
- You can type

  ?cum

  ?my.cumsumprod

  ?data1

  , and see the manual you just edited in **.Rd** files.

Note 1: Import and Export in NAMESPACE

- The original NAMESPACE file contains a line **exportPattern("^[[:alpha:]]+")**. If nothing is changed, then every function in the package is visible.
- If you want only part of the functions visible, use **export().** Delete the line **exportPattern("^[[:alpha:]]+")**, and change it to **export(function names)**. Then delete the **.Rd** files that you want to hide.
- If you used functions from other R packages, use **import(package names).** You can also add a line **Depends: package names** in file **DESCRIPTION**.

Note 2: Build an R package for different systems

- Once you get the R package folder in one system, just use it and run **R CMD build, R CMD INSTALL** in different systems. You do not need to run **package.skeleton** in different systems.
- In windows system, if you run **R CMD INSTALL --build cum_1.0.tar.gz** instead of **R CMD INSTALL cum**, you will get a **.zip** R package.

Questions?

Thank you!