Creating R Package in Windows

Fentaw Abegaz

University of Groningen
Johann Bernoulli Institute of Mathematics and Computer Science

March 21, 2012

Outline

Required software components

Download and Install

- Rtools and accompanying components: perl, MinGW
  - Download and run *RtoolsXXX.exe* from CRAN
- LaTex (Optional)
- Microsoft HTML Help Workshop (Optional)

Set PATH for Windows

- Depending on the system (eg. Windows XP)

  $Control\ \ Panel \rightarrow System \rightarrow Advanced \rightarrow Environment\ \ Variables \rightarrow PATH \rightarrow edit$

- Edit your PATH variable (type or copy-paste) to include:

  $C:\backslash Rtools \backslash bin; C:\backslash Rtools \backslash perl \backslash bin; C:\backslash Rtools \backslash MinGW \backslash bin;$

  $C:\backslash Program\ \ Files \backslash R \backslash R-2.12.2 \backslash bin;$

  $C:\backslash Program\ \ Files \backslash HTML\ \ Help\ \ Workshop;$

  $C:\backslash Program\ \ Files \backslash MiKTeX\ \ 2.8 \backslash miktex \backslash bin;$

Checking installed components

- Open Command Prompt

  Start → *Programs* → *Accessories* → *Command Prompt*

- Type the commands into the Command Prompt window to test whether you get correct output.
  - `path` Learn what your path variable looks like
  - `R.exe` Open up R, should boot up from any directory if you've done this correctly. q() to quit.
  - `gcc --help` Test MinGW.
  - `perl --help` Test Perl.
  - `Tex --help` Test Miktex.
  - `grep --help` Test to see that Rtools is connected.

Package Structure

R library package is a set of directories, that, when named correctly, R will be able to load during Run-time as a library under the `library()` command.

- A file named **DESCRIPTION** with descriptions of the package, author, dependence, and license conditions in a structured text format.
- **man** : subdirectory of documentation files.
- **R** : subdirectory of R code.
- **data** : subdirectory of datasets(this is optional).

Package Structure - Less common

- **src** : subdirectory of C, Fortran or C++ source.
- **tests** : for validation tests.
- **exec** : for other executables (eg Perl or Java).
- **inst** : for other stuffs.
- **configure**: script to check for other required software or handle differences between systems.

R code

- Save the R function code as *functionname.R*
- The R-library name is then the same function name *functionname* without the extension *.R*

Data

Save data using R data file extensions: *datafilename.Rdata*, *datafilename.Rda*

Create Package Structure

- Start R and run the command `package.skeleton()`
- Make sure the current directory is set where you want to create the package or use `setwd()`.
- To create a library called "copula", use the command

$$package.skeleton(\text{name} = \text{“}copula\text{”}, \text{code\_files} = \text{“}copula.R\text{”})$$

Create Package Structure

The command *package.skeleton* generate a directory `copula` and several sub-directories and files.

- Sub-directories
  1. **R** includes the function code `copula.R`
  2. **man** includes `copula.Rd` and `copula-package.Rd`
  3. **data** (create manually and paste data file)
- Files
  1. `DESCRIPTION`
  2. `Read-and-delete-me`.
- Edit the sub-directories and files so that they contain the right information.

Compiling the package for Windows

- Compile the package into a zip file: use `DOS prompt` and go to the directory containing your package.

- To process the library package there are three commands of form

  `R CMD command` *packagename*

  - `build:` packs everything into an archive file `packagename_1.0.tar.gz` or

    `R CMD build` *packagename*

  - `check:` runs a battery of tests on the package.

    `R CMD check` *packagename*

  - `INSTALL:` installs the package into a library and makes it available for usage in R.

    `R CMD INSTALL` *packagename_1.0.tar.gz*

  - To compile the package into a zip file

    `R CMD INSTALL --build` *packagename*

Building a package for other operating systems

Use commands

- *package.skeleton*(name = "*packagename*", code_files = "*packagename.R*")
- `R CMD command` *packagename*

Compiling C and FORTRAN codes

- Use DOS command: R CMD SHLIB *ccode.c*
- Use into R function code

  dyn.load(*ccode.dll*)

  .C()

Putting your package on CRAN

- Upload the *tar.gz* file to *ftp* : *//CRAN.R − project.org/incoming/* using anonymous as login name and your e-mail address as password.
- Send a message to *CRAN@R − project.org* about it.

Thank You !