How to make R packages

- use C, C++, and Fortran code in R

Eun-kyung Lee

February 4, 2004

Outline

- What is R?
- What is R package?
- Package structure
- How to build package
- How to call C/Fortran function in R
- Summary

R

- open source statistical analysis software, similar to S-plus
- provides a wide variety of statistical and graphical techniques
- is highly extensible.
- for computationally-intensive tasks, C, C++ and Fortran code can be linked and called at run time.
- can be extended (easily) via packages.

R packages

- provide a mechanism for loading optional code and attached documentation as needed.
- Once a source package is created, it must be installed. (use `R CMD INSTALL`)
- `library(package.name)`

Package Structure

- DESCRIPTION
- INDEX - optional
- R/
- data/
- man/
- src/

* `package.skeleton`

DESCRIPTION file

- Package : the name of the package
- Version : the version of the package
- Title : a short description of the package.
- Description : a comprehensive description.
- Author : describe who wrote the package.
- Maintainer : single name with email address
- License :

INDEX file - optional

- contains a line for each sufficiently interesting object in the package, giving its name and a description.
- it can be automatically generated from the documentation sources.

R/

- contain R code files
- it should be possible to read in the files using `source()`
- if necessary, one file(historically "zzz.R") should use `library.dynam()` inside `.First.lib()` to load compiled code.

eg)
```
.First.lib<-function(lib,pkg){
    library.dynam("ClassPP",pkg,lib) }
```

man/

- contain documention files for the objects in the package in R docoumentation(.Rd) format
- all user-level objects in a package should be documented.
- it is used for writing package vignettes.

src/

- contain C, C++, or FORTRAN source files
- optionally "Makevars" or "Makefile"

data/

- contain additional data files
- load using `data()`
- plain R code, tables, or images from `save()`

check and build package

- `R CMD check pkgname (Rcmd check)`
  - provide subdirectory pkgname.Rcheck/
  - install package
  - provide pkgname-manual.tex
- `R CMD build pkgname(Rcmd build)`
  - provide pkgname_version.tar.gz
  - to install this library, `R CMD INSTALL pkgname_version.tar.gz`

Interface functions .C and .Fortran

- the mapping between the modes of R vectors and the types of arguments to a C and Fortran

R storage mode	C	Fortran
logical	int*	INTEGER
integer	int*	INTEGER
double	double*	DOUBLE PRECISION
character	char**	CHARACTER*255

- the compiled code should not return anything except through its arguments :
  - C function : type `void`
  - Fortran function : should be subroutines.

- For C function,
  - `#include <t.h>`
  - memory allocation
    1. `R_alloc()` : R manages the clean-up
    2. `Calloc()/Free()` : user has full control

Creating shared objects

- `R CMD SHLIB fun1.c fun2.c ···` : create `fun1.so`
- in **Makevars** file
  - `PKG_FLAGS` : for '`-I`' flags
  - `PKG_LIBS` : for '`-l`' or '`-L`' flags

How to use your functions in the shared objects

- `dyn.load(‘‘*.so’’)` : load the shared object
- `is.loaded(sumbol.C(‘‘function.name’’))`
  : check whether your function is loaded properly or not
- `test <- .C(‘‘function.name’’,arg1, x=arg2, y=arg3, ···)`
  : It returns `test$x,test$y, ···`
- `dyn.unload(‘‘*.so’’)` : unload the shared object

Summary

- R can be extended (easily) via packages
- R has its own LaTeX-like documentation format, which is used to supply comprehensive documentation, both on-line in a number of formats and in hardcopy.
- For computationally-intensive tasks, C, C++ and Fortran code can be linked and called at run time.
- If you want to submit your package, upload the `'tar.gz'` file to
  `ftp://ftp.ci.tuwien.ac.at/incoming`
  and send a message to `cran@r-project.org`
- For more information, visit `http://www.r-project.org/`