Distribute your R code with R package

Feng Li
feng.li@cufe.edu.cn

School of Statistics and Mathematics
Central University of Finance and Economics

June 2, 2014

Revision: June 2, 2014

Today we are going to learn...

Why package

- Packages provide a mechanism for loading optional code, data and documentation as needed.
- Code and functions are documented.
- Easy to share.
- Provide easy interface in R for code written in other languages (C, Fortran...)

The R package structure

- The sources of an R package consists of a subdirectory containing a files `DESCRIPTION` and `NAMESPACE`, and the subdirectories `R`, `data`, `demo`, `exec`, `inst`, `man`, `po`, `src`, `tests`, `tools` and `vignettes` (some of which can be missing, but which should not be empty).
- The package subdirectory may also contain files `INDEX`, `configure`, `cleanup`, `LICENSE`, `LICENCE` and `NEWS`.
-

The difference between source and binary packages

- Source package: you can check the source code. You need to compile it to make it binary.
- Binary package: Ready to use.

The DESCRIPTION file

- The `Package`, `Version`, `License`, `Description`, `Title`, `Author`, and `Maintainer` fields are mandatory, all other fields are optional.

The `Licensing` file

- It is very important that you include license information! Otherwise, it may not even be legally correct for others to distribute copies of the package, let alone use it.

Package Dependencies

- The `Depends` field gives a comma-separated list of package names which this package depends on.
- Those packages will be attached before the current package when library or require is called.
- Each package name may be optionally followed by a comment in parentheses specifying a version requirement. The comment should contain a comparison operator, whitespace and a valid version number, e.g. `MASS (>= 3.1-20)`.

The INDEX file

- The optional file INDEX contains a line for each sufficiently interesting object in the package, giving its name and a description.
- Normally this file is missing and the corresponding information is automatically generated from the documentation sources when installing from source.

The `R` subdirectory

- The R subdirectory contains R code files, only.
- The code files to be installed must start with an ASCII (lower or upper case) letter or digit and have one of the extensions `.R`, `.S`, `.q`, `.r`, or `.s`.

The man subdirectory I

- The man subdirectory should contain (only) documentation files for the objects in the package in R documentation (Rd) format
- A typical R function documentation

```
\name{load}
\alias{load}
\title{Reload Saved Datasets}
\description{
Reload the datasets written to a file with the function
\code{save}.
}
\usage{
load(file, envir = parent.frame())
}
\arguments{
\item{file}{a connection or a character string giving the
name of the file to load.}
\item{envir}{the environment where the data should be
loaded.}
```

The man subdirectory II

```
}
\seealso{
\code{\link{save}}.
}
\examples{
## save all data
save(list = ls(), file= "all.RData")
## restore the saved values to the current environment
load("all.RData")
## restore the saved values to the workspace
load("all.RData", .GlobalEnv)
}
\keyword{file}
```

Other files

- The sources and headers for the compiled code are in `src`, plus optionally a file `Makevars` or `Makefile`.
- The `data` subdirectory is for data files.
  - Data files can have one of three types as indicated by their extension: plain R code (.R or .r), tables (.tab, .txt, or .csv, see ?data for the file formats, and note that .csv is not the standard CSV format), or save() images (.RData or .rda).
- The `demo` subdirectory is for R scripts (for running via `demo()`) that demonstrate some of the functionality of the package.

Build a package

- Using `R CMD check`, the R package checker, one can test whether source R packages work correctly.
- Packages may be distributed in source form as âĂIJtarballsâĂİ (.tar.gz files) or in binary form.
  - The source form can be installed on all platforms with suitable tools and is the usual form for Unix-like systems
  - the binary form is platform-specific, and is the more common distribution form for the Windows and OS X platforms.
- Use the following commands to find more help under a termial.
  - `R CMD check --help`
  - `R CMD build --help`