Creating R Packages

Yihui Xie   Sizhe Liu

The 2nd Chinese R Conference @ Shanghai

December 12, 2009

Outline

Skeleton

- a source package contains source code for functions (and data) and documentations
- with functions already defined in the current workspace, just start from `package.skeleton()`

```
package.skeleton(name = "anRpackage", list,
                    environment = .GlobalEnv, path = ".",
                    force = FALSE, namespace = FALSE,
                    code_files = character())
```

- typically a file `DESCRIPTION` and two subdirectories `R` and `man` will be automatically created; modify them as you wish, and...

Skeleton

- a source package contains source code for functions (and data) and documentations
- with functions already defined in the current workspace, just start from `package.skeleton()`

```
package.skeleton(name = "anRpackage", list,
                    environment = .GlobalEnv, path = ".",
                    force = FALSE, namespace = FALSE,
                    code_files = character())
```

- typically a file `DESCRIPTION` and two subdirectories `R` and `man` will be automatically created; modify them as you wish, and...
- we are done! what?! the talk is over!

More subdirectories

- `data`: for datasets (`*.rda` if use `package.skeleton`, other formats are allowed, cf man `R-exts` p9)
- `demo`: for demos (`*.r`, can be run via `demo()`)
- `src`: source code written in C, C++, Fortran
- `inst`: will be copied to the root directory of the package when building and installing the package (useful for non-standard files and directories, e.g. a news file `NEWS`, a vignette directory `doc`)

Outline

Building and checking

- `R CMD build` *`YourPKG`*
  - package your files into a `YourPKG_*.*-*.tar.gz` file (check possible errors, remove unneeded files and compress the files)
  - `R CMD build --binary YouPKG`: build a binary package (compile the code under `src` into `dll`'s and vignettes under `inst/doc/` to PDF's, copy all files under `inst` to the root directory, add `md5sum`, etc)
  - you can install the binary package and use it as an add-on package (in fact, can `R CMD INSTALL` *`YourPKG`*, i.e. install from source)
- `R CMD check` *`YourPKG`*
  - check for possible problems in the code and documentation
  - this is an important step before submitting your package to CRAN! make sure your package can pass `R CMD check`

Tools for building packages

- Linux and Mac users
  - usually no additional tools needed (my vague memory: install `r-core-dev` under Ubuntu)
  - open a terminal and type `R CMD build` there
- Windoze users
  - must install `Rtools`[1] which is a collection of GNU utilities and libraries required for building R packages (e.g. `gcc`, `tar`)
  - need LaTeX if you want to build help pages into a PDF document
  - important step: make sure the directories of these utilities are in the environment variable `PATH`! (so that these commands can be executed without specifying the directories, e.g. if the `bin` directory of R is not in `PATH`, you need type `"C:\Programe Files\R\bin\R.exe"` in the `cmd` window to run R)

[1] http://www.murdoch-sutherland.com/Rtools/

Outline

Facts on documentation

Theorem 1

Nobody bothers to read software documentation, no matter how many times you tell them to RTFM.

Facts on documentation

Theorem 1

Nobody bothers to read software documentation, no matter how many times you tell them to RTFM.

Theorem 2

Authors think there are two types of users: either too smart so that they do not need documentation, or too stupid to understand anything in the documentation.

Facts on documentation

Theorem 1

Nobody bothers to read software documentation, no matter how many times you tell them to RTFM.

Theorem 2

Authors think there are two types of users: either too smart so that they do not need documentation, or too stupid to understand anything in the documentation.

Theorem 3

But users think there is only one type of authors: ***stupid*** *authors.*

R Documentation

- personal feeling: writing R code is much easier than writing documentation!!
- open the `*.Rd` files and begin the "battle" with your users (think carefully what to write there and how)
- a documentation file consists of several sections, e.g. `\title{}`, `\description{}`, `\usage{}`
- you can mark up your texts, e.g. `\emph{}`, `\bold{}`, `\url{}`
- can use lists, tables and math formulae in `\eqn{}` and `\deqn{}` (all like LaTeX syntax)
- since R 2.10.0, can also insert Sweave-like macro `\Sexpr{}` to generate dynamic help pages! (figures still not supported)
- personal experience: the example section is the most important (nobody is patient enough to read your long long description)

Vignettes

- a formal or informal paper describing your package
- you can directly put a PDF document under `inst/doc/` as a vignette
- or put a Sweave (Leisch, 2002) document (`*.Rnw`) there, and R will compile it using `R CMD Sweave`
- you may refer to the `quantreg` package by Koenker (2009) as an example

Outline

Call C functions

- people complaining R is slow may consider C or Fortran
- you can either put them under `src` or use `R CMD SHLIB` to compile them into `dll`'s and call the functions in R with `.C`/`.Fortran` interface
- a simple example here: reverse a numeric vector

Listing 1: Reverse a vector

```
void reverse(double *a, int *na, double *b)
{
    int i;
    for(i = 0; i < *na; i++)
    b[i] = a[*na - i - 1];
}
```

C experts needed!

- ImageMagick and SWF Tools are mainly written in C
- I want to extract the single command `convert` out of ImageMagick
- and `png2swf`, `jpeg2swf`, `pdf2swf` out of SWF Tools
- so that we can use `convert` and `*2swf` directly in R, otherwise users have to install additional software (for Windoze users, they have to set the '`path`' variable, which is often confusing to them)

Outline

CRAN and R-Forge

- CRAN is a place to store R code, packages and documentations with mirrors worldwide (including ISU)
- we can submit our source packages to CRAN via FTP (`ftp://cran.r-project.org/incoming`) and send email to cran@r-project.org to notify CRAN masters
- or a more convenient way for developing R packages – R-Forge: `http://r-forge.r-project.org`
    - register for an account
    - then register a new project for your package(s)
    - you will have a whole bunch of tools there: SVN, mailing list, website
    - R-Forge will check and build your packages on a daily basis
    - need to talk about SVN?

Bibliography

Koenker R (2009). *quantreg: Quantile Regression*. R package version 4.44, URL http://CRAN.R-project.org/package=quantreg.

Leisch F (2002). "Sweave: Dynamic Generation of Statistical Reports Using Literate Data Analysis." In W Härdle, B Rönz (eds.), "Compstat 2002 — Proceedings in Computational Statistics," pp. 575–580. Physica Verlag, Heidelberg. ISBN 3-7908-1517-9, URL http://www.stat.uni-muenchen.de/~leisch/Sweave.

R Development Core Team (2009). *Writing R Extensions*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-11-9, URL http://www.R-project.org.

Thanks!

- We didn't talk about writing functions using S3 or S4; if you are interested, please read this paper: `http://cran.r-project.org/doc/contrib/Leisch-CreatingPackages.pdf`
- Questions and comments?
- Email: `sprintf("%s@%s", c("yihui.xie", "sizhe.liu"), "cos.name")`
- Slides available online: `browseURL("http://yihui.name/en/vitae")`