Sustainable package development using documentation generation

http://inlinedocs.r-forge.r-project.org

Toby Dylan Hocking
toby.hocking AT inria.fr

21 October 2010

Outline

General package structure

Documenting a function in several ways
  - Filling in `package.skeleton` templates by hand
  - Doc generation from headers using roxygen and R.oo::Rdoc
  - Doc generation from inline comments using inlinedocs

Package publication, conclusions, and references

Sharing your code with the R community

- Most likely you have some interesting functions you would like to share.
- You could just email your code.R file to a colleague.
- However, there is a standardized process for documenting, publishing and installing R code.
- If you want your code to be used (and potentially modified) by anyone, then you should consider making a **package**.

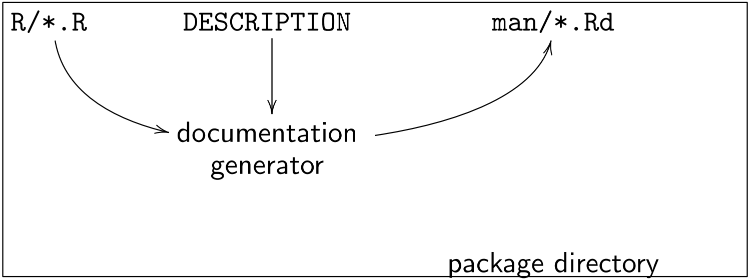
What is an R package?

- It is a collection of code and data for a specific task, in a specific format.
- Give your package a name, make a corresponding directory `pkgdir`
- Required items:
  1. `pkgdir/R/*.R` for R code.
  2. `pkgdir/DESCRIPTION` to describe its purpose, author, etc.
  3. `pkgdir/man/*.Rd` for **documentation**.
- Optional items:
  - `pkgdir/data/*` for data sets.
  - `pkgdir/src/*` for C/FORTRAN/C++ source to be compiled and linked to R.
  - `pkgdir/inst/*` for other files you want to install.
  - `pkgdir/po/*` for international translations.
- All of these need to be in a standard format as described in "Writing R Extensions" in excrutiating detail.

How to write the package?

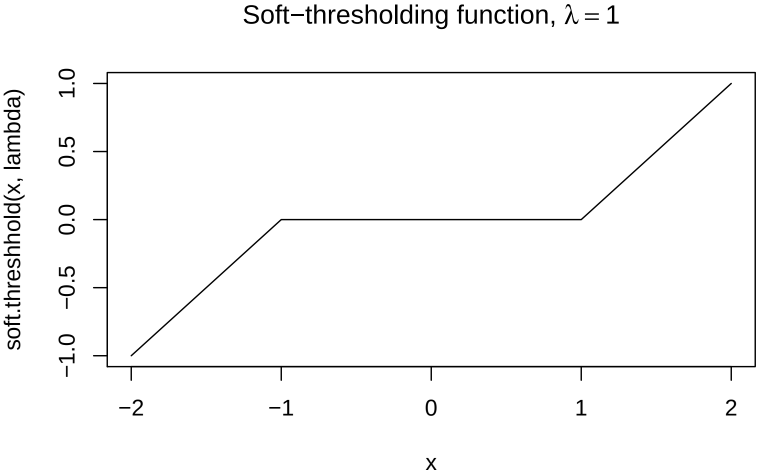
- Do it yourself! Read "Writing R Extensions," only 141 pages in PDF form, as of 22 September 2010.
- Luckily, there are several packages/functions that can simplify the package-writing process.
- `package.skeleton()`
- `roxygen::roxygenize()`
- `R.oo::Rdoc$compile()`
- `inlinedocs::package.skeleton.dx()`

Documentation generation diagram



Outline

Example: soft-thresholding function



$$f(x,\lambda) = \begin{cases} 0 & |x| < \lambda \\ x - \lambda\,\mathrm{sign}(x) & \text{otherwise} \end{cases}$$

R implementation of soft-thresholding function

$$f(x,\lambda) = \begin{cases} 0 & |x| < \lambda \\ x - \lambda\,\mathrm{sign}(x) & \text{otherwise} \end{cases}$$

Make a new directory `softThresh` for the package, and put R code files in the `R` subdirectory:

`softThresh/R/soft.threshold.R`

```
soft.threshold <- function(x,lambda=1){
  stopifnot(lambda>=0)
  ifelse(abs(x)<lambda,0,x-lambda*sign(x))
}
```

Outline

General package structure

Documenting a function in several ways

- Filling in `package.skeleton` templates by hand
- Doc generation from headers using roxygen and R.oo::Rdoc
- Doc generation from inline comments using inlinedocs

Package publication, conclusions, and references

Use package.skeleton to start a new package

```
R> package.skeleton("softThresh",
                    code_files="soft.threshold.R")
```

will create `./softThresh/man|R|DESCRIPTION` with templates:

```
\name{soft.threshold}
\alias{soft.threshold}
%- Also NEED an ’\alias’ for EACH other topic documented here.
\title{
%%  ~~function to do ... ~~
}
\description{
%%  ~~ A concise (1-5 lines) description of what the function does. ~~
}
\usage{
soft.threshold(x, lambda = 1)
}
%- maybe also ’usage’ for other objects documented here.
\arguments{
  \item{x}{
%%     ~~Describe \code{x} here~~
}
  \item{lambda}{
%%     ~~Describe \code{lambda} here~~
}
}
\details{
%%  ~~ If necessary, more details than the description above ~~
}
\value{
%%  ~Describe the value returned
%%  If it is a LIST, use
%%  \item{comp1 }{Description of ’comp1’}
```

Fill in the Rd templates generated by `package.skeleton`

`softThresh/man/soft.threshold.Rd`

```
\name{soft.threshold}
\title{Soft-thresholding}
\description{Apply the soft-threshold function to a vector.}
\usage{
soft.threshold(x, lambda = 1)
}
\arguments{
  \item{x}{A vector of numeric data.}
  \item{lambda}{The largest absolute
    value that will be mapped to zero.}
}
\value{The vector of observations
    after applying the soft-thresholding.}
\author{Toby Dylan Hocking <toby.hocking@inria.fr>}
\examples{
  x <- seq(-5,5,l=50)
  y <- soft.threshold(x)
  plot(x,y)
}
```

Write the metadata in the DESCRIPTION file

`softThresh/DESCRIPTION`

```
Package: softThresh
Maintainer: Toby Dylan Hocking <toby.hocking@inria.fr>
Author: Toby Dylan Hocking
Version: 1.0
License: GPL-3
Title: Soft-thresholding
Description: A package documented by hand.
```

Doing it by hand versus documentation generation

- Doing it by hand is simple but has some disadvantages
  - Easy to do, LaTeX-like syntax
  - Possibility of conflict between code and documentation
  - Every time the function changes, need to copy to docs
- Documentation generation has several advantages
  - Documentation is written in comments, nearer to the source code
  - Can exploit the structure of the source code
  - Simplifies updating documentation (!!)
  - Reduces the probability of mismatch between code and docs

Different approaches to documentation generation

- Put the documentation in a big header comment
    - roxygen::roxygenize()
    - R.oo::Rdoc$compile()
- Put the documentation in comments right next to the relevant code
    - inlinedocs::package.skeleton.dx()

Outline

General package structure

Documenting a function in several ways

- Filling in `package.skeleton` templates by hand
- Doc generation from headers using roxygen and R.oo::Rdoc
- Doc generation from inline comments using inlinedocs

Package publication, conclusions, and references

roxygen reads documentation from comments above

softThresh/R/soft.threshold.R

```
##' Apply the soft-threshold function to a vector.
##'
##' @title Soft-thresholding
##' @param x A vector of numeric data.
##' @param lambda The largest absolute value that
##' will be mapped to zero.
##' @return The vector of observations after applying the
##' soft-thresholding.
##' @author Toby Dylan Hocking <toby.hocking@@inria.fr>
soft.threshhold <- function(x,lambda=1){
  stopifnot(lambda>=0)
  ifelse(abs(x)<lambda,0,x-sign(x)*lambda)
}
```

Note: headers can be automatically generated using the `ess-roxy-update-entry` `C-c C-o` command in Emacs+ESS.

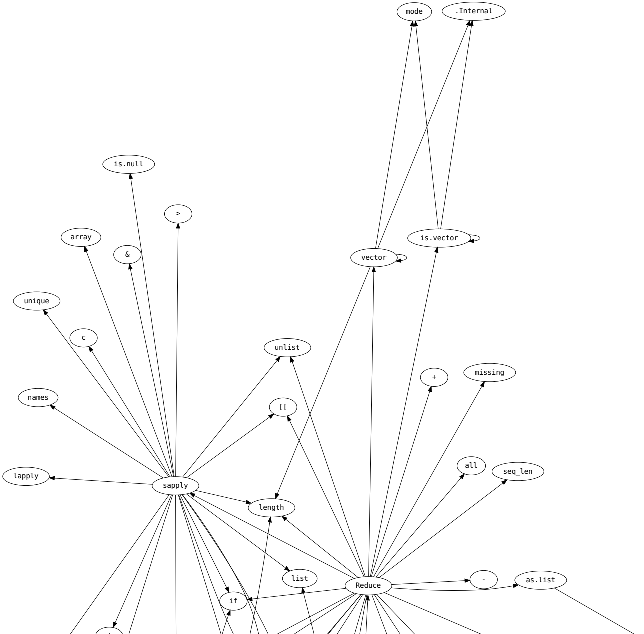
roxygen generates Rd

```
shell$ R CMD roxygen -d softThresh
```

generates/overwrites `softThresh/man/soft.threshold.Rd`

There is also the R function `roxygenize` (see its help page for details)

roxygen can also generate call graphs (complicated setup)



Rdoc puts docs in headers as well

(similar to roxygen, but less documentation and editor support)

```
## @RdocFunction soft.threshold
## @title "Soft-thresholding"
## \description{
##  Apply the soft-threshold function to a vector.
## }
## @synopsis
## \arguments{
##   \item{x}{A vector of numeric data.}
##   \item{lambda}{The largest absolute value
##      that will be mapped to zero.}
## }
## \value{
## The vector of observations after applying the
## soft-thresholding.
## }
## @author
```

Documentation generation based on comments in headers

- 2 step process:
  1. Write: documentation written in comments.
  2. Compile: comments automatically translated to Rd files.
- Advantages:
  - Documentation closer to code.
  - Less chance of mismatch.
  - Fewer manual documentation updates when the code changes.
- Disadvantages:
  - Need to copy function argument names in the header.
  - The header is sometimes really big.
  - In reality, the docs are far away from the corresponding code.
- Can we come up with a system where the documentation is even closer to the actual code?

Outline

inlinedocs allows docs in comments adjacent to the code

softThresh/R/soft.threshold.R

```
soft.threshold <- function # Soft-thresholding
### Apply the soft-threshold function to a vector.
(x,
### A vector of numeric data.
 lambda=1
### The largest absolute value that will be mapped to zero.
 ){
  stopifnot(lambda>=0)
  ifelse(abs(x)<lambda,0,x-sign(x)*lambda)
### The vector of observations after applying
### the soft-thresholding function.
}
```

another inlinedocs syntax for function arguments

softThresh/R/soft.threshold.R

```
soft.threshold <- function # Soft-thresholding
### Apply the soft-threshold function to a vector.
(x,        ##<< A vector of numeric data.
 lambda=1 ##<< The largest absolute value that
          ##   will be mapped to zero.
 ){
  stopifnot(lambda>=0)
  ifelse(abs(x)<lambda,0,x-sign(x)*lambda)
### The vector of observations after applying
### the soft-thresholding function.
}
```

inlinedocs: comment code wherever it is relevant

softThresh/R/soft.threshold.R

```
soft.threshold <- function # Soft-thresholding
### Apply the soft-threshold function to a vector.
(x,        ##<< A vector of numeric data.
 lambda=1 ##<< The largest absolute value that
          ##   will be mapped to zero.
 ){
  stopifnot(lambda>=0)
  ##details<< lambda must be non-negative.
  ifelse(abs(x)<lambda,0,x-sign(x)*lambda)
### The vector of observations after applying
### the soft-thresholding function.
}
```

inlinedocs::package.skeleton.dx() generates Rd files

```
R> library(inlinedocs)
R> package.skeleton.dx("softThresh")
```

produces `softThresh/man/soft.threshold.Rd`

How to write example code?

roxygen: in comments (not executable)

```
##' @examples
##' x <- seq(-5,5,l=50)
##' y <- soft.threshold(x)
##' plot(x,y)
soft.threshold <- function(x,lambda=1){...}
```

inlinedocs: in code (executable)

```
soft.threshold <- function(x,lambda=1){...}
attr(soft.threshold,"ex") <- function(){
  x <- seq(-5,5,l=50)
  y <- soft.threshold(x)
  plot(x,y)
}
```

inlinedocs for documentation generation

- 2 step write/compile process for documentation generation.
- Write the documentation in comments **right next to** the corresponding code.
- Takes advantage of function argument names, etc. defined in the code.
- Resulting code base is very easy to maintain.
- Almost eliminates the possibility of code and documentation conflicts.
- AND: support for S4 methods, named list documentation, easily extensible syntax.

Outline

General package structure

Documenting a function in several ways

- Filling in `package.skeleton` templates by hand
- Doc generation from headers using roxygen and R.oo::Rdoc
- Doc generation from inline comments using inlinedocs

Package publication, conclusions, and references

To publish your package

- Write your code in `pkgdir/R/code.R`
- Write a `pkgdir/DESCRIPTION`
- Write (or generate) documentation `pkgdir/man/*.Rd`
- `R CMD check pkgdir` (until no errors or warnings)
- `R CMD build pkgdir` (makes `pkgdir.tar.gz`)
- Upload `pkgdir.tar.gz` to ftp://cran.r-project.org/incoming
    - user: anonymous
    - password: your@email
    - send email to cran@r-project.org
- If it passes the CRAN checks, then it is posted to the CRAN website for anyone to download and install using `install.packages()`

References for learning more about package development

- The definitive guide: help.start() then Writing R Extensions
- The built-in package generator: ?package.skeleton
- roxygen
  - library(roxygen)
  - ?roxygenize
  - `http://roxygen.org/roxygen.pdf`
- R.oo:Rdoc
  - library(R.oo)
  - ?Rdoc (not very much documentation)
  - `http://www.aroma-project.org/developers`
- inlinedocs
  - library(inlinedocs)
  - ?inlinedocs
  - `http://inlinedocs.r-forge.r-project.org`
- Contact me directly: toby.hocking AT inria.fr, `http://cbio.ensmp.fr/~thocking/`