Sustainable R package development using documentation generation
http://inlinedocs.r-forge.r-project.org

Toby Dylan Hocking
toby.hocking AT inria.fr

9 June 2011
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 modified by the R community, 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, dependencies, 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.**
Why make an R package?

▶ It seems pretty complicated to make a package, but in fact it is simple and comes with many benefits.

▶ Advantages for you:
  1. Installation from any internet-connected computer using `install.packages()` from the R command line. This includes dependencies!
  2. Compilation of C/C++/Fortran code on the CRAN servers, so Windows/Mac users can install your package even if they do not have a compiler.

▶ Advantages for the R community:
  1. Your package will be stored on CRAN, so others can make packages that depend on yours.
  2. Your code becomes open-source, so others can modify your code.
How to write the package?

- Do it yourself! Read “Writing R Extensions,” only 163 pages in PDF form, as of R 2.13.0, 13 May 2011.
- Luckily, there are several functions that use documentation generation to simplify the package-writing process.
  - `package.skeleton()`
  - `roxygen::roxygenize()`
  - `R.oo::Rdoc$compile()`
  - `inlinedocs::package.skeleton.dx()`
R source files and DESCRIPTION metadata are used to construct documentation Rd files.
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
Example: soft-thresholding function

Soft-thresholding function, $\lambda = 1$

\[
f(x, \lambda) = \begin{cases} 
0 & |x| < \lambda \\
 x - \lambda \text{sign}(x) & \text{otherwise}
\end{cases}
\]
R implementation of soft-thresholding function

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

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

`softThresh/R/soft.threshold.R`

```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'\}}
  %% ...}
```

% Add one or more standard keywords, see file 'KEYWORDS' in the
% R documentation directory.

\keyword{ ~kwd1 }
\keyword{ ~kwd2 }% __ONLY ONE__ keyword per line
**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
softThresh/R/soft.threshold.R

```r
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)

```r
## @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

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
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.
}
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)

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

inlinedocs: in code (executable)

```r
soft.threshold <- structure(function(x,lambda=1){
  ...
},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/