

Topics

- we are going to see:
 - how to create a barebones R package
- we are going to skip
 - how to create documentation for a package
 - how to tidy up a package
 - how to do profiling on a package
 - how to provide test data files
 - how to submit to CRAN
- read the suggested reading for details on these aspects of package writing in R

Create A Package I

- package prerequisite:
 - if you are an *administrator* on a machine or server you don't need to do anything here
 - otherwise, i.e., if you are a *regular user* issue the following in the command prompt:
 - * Linux/UNIX/MAC

```
$ mkdir ~/Rlib
```
 - * Windows

```
C:\> mkdir Rlib
```
- this will create a directory called Rlib at the top level of your account

Create A Package II

- suppose we want to create a package called MH1 (Metropolis-Hastings)
- go to a directory where you want to work on and start up R
- issue the following command at the R prompt:

```
> doMH1 <- function (args) { }; package.skeleton('MH1', c('doMH1'))
```

- you should see output of the form:

```
Creating directories ...
Creating DESCRIPTION ...
Creating READMEs ...
Saving functions and data ...
Making help files ...
Created file named './MH1/man/doMH1.Rd'.
Edit the file and move it to the appropriate directory.
Done.
Further steps are described in ./MH1/README
```

- note it creates some directories, namely, MH1, MH1/man, MH1/src, MH1/R and some files e.g. MH1/R/doMH1.R therein

Create A Package III

- using a command line go to the sub-directory MH1 and edit the DESCRIPTION file in it
- initially it will look like:

Package: MH1

Type: Package

Title: What the package does (short line)

Version: 1.0

Date: 2005-07-28

Author: Who wrote it

Maintainer: Who to complain to <yourfault@somewhere.net>

Description: More about what it does (maybe more than one line)

License: What license is it under?

Create A Package IV

- the edited DESCRIPTION file may look like:

```
Package: MH1
Type: Package
Title: Does Metropolis Hastings algorithm
Version: 1.0
Date: 2005-07-28
Author: Gopi Goswami
Maintainer: <goswami@stat.harvard.edu>
Description: More about what it does (later, feeling lazy now!)
License: GPL
```

- make sure that at lest the Package field is properly changed

Create A Package V

- put all your C (or C++) source files, i.e., .h and .c (or .H and .C) files in the directory MH1/src
- put all your R source files, i.e., .R files in the directory MH1/R and of course edit the doMH1() function to add more functionality
- now create a file called zzz.R in the MH1/R directory and dump the following R code in it:

```
.First.lib <-  
  function (libname, pkgname)  
{  
  library.dynam(pkgname, pkgname, lib.loc=libname)  
}
```

- the .First.lib function is called whenever a package is loaded in R
- here we are loading the compiled C (or C++) code using the function library.dynam() whenever the package is loaded, this is much like the function dyn.load() we used to use in the .C, .Call and .External

interface

Create A Package VI

- now go to the directory where the MH1 directory lives and issue the following command in the system command prompt and sit back:
 - administrator:

```
R CMD INSTALL MH1
```
 - regular user:
 - * Linux/UNIX/MAC

```
R CMD INSTALL --library=$HOME/Rlib MH1
```
 - * Windows

```
R CMD INSTALL --library=C:\Rlib MH1
```
- if it fails most likely you got error messages from gcc pointing out where you messed up, correct those and re-issue the above command and stay in this loop until success!

Create A Package VII

- assuming you corrected all the errors, issue the following in the R command prompt:

- administrator:

```
> detach('package:MH1')  
> library(MH1)
```

- regular user:

```
> detach('package:MH1')  
> library(MH1, lib.loc='~/Rlib')
```

- the `library()` function loads the library
- the `detach()` function unloads the library, so don't use the this command for the very first time you load the library, use it when you make changes to your library and you want to reload it
- now you should be able to use the functions from the package:

```
> dd1 <- doMH1(100000, c(0, 1, 2.4), c(0), 0)  
> c(summary(dd1), sd(dd1))
```

Create A Package I

- you could also do the following to achieve other objectives (what are those? take a look at the code) while loading the package:

```
.First.lib <-  
  function (libname, pkgname)  
  {  
    ## figure out this year automatically  
    this.year <- substr(as.character(Sys.Date( )), 1, 4)  
  
    ## echo output to screen  
    cat("##\n## Metropolis-Hastings Package (MH2)\n")  
    cat("## Copyright (C) 2003-", this.year,  
        " Your Name\n", sep="")  
    cat("##\n## Support provided by Prof. Scary Guy\n")  
    ## assuming you would need the package MASS for your package  
    require(MASS, quietly=TRUE)  
  
    library.dynam(pkgname, pkgname, lib.loc=libname)  
  }
```

- this example is modified from code in Kevin Quinn and Andrew Martin's MCMCpack package

Create A Package I

- we have shown you how to create a in-house barebones package, you need to write documentation, provide data sets etc. and finally *build* a package before you can submit it to CRAN
- you build a package by issuing the following in the command prompt (Linux/UNIX/MAC/Windows):

```
$> R CMD build MH1
```
- this creates a file called `MH1_1.0.tar.gz` which is ready for submission/distribution

Code Files

```
../problem-set/MH1/main.c  
../problem-set/MH1/objects.c  
../problem-set/MH1/objects.h  
../problem-set/MH1/doMH1.R  
../problem-set/MH1/zzz.R
```