Creación de paquetes, informes y viñetas

Manuel Febrero Bande
y
Manuel Oviedo de la Fuente

USC
UNIVERSIDADE DE SANTIAGO DE COMPOSTELA
DEPARTAMENTO DE ESTATÍSTICA E INVESTIGACIÓN OPERATIVA

Table of Contents

Índice

Razones para crear un paquete

- Forma de mantener colecciones de funciones y datos en R que se pueden cargar y descargar en memoria de forma sencilla.
- Excelente forma de compartir con tus colegas o toda la comunidad tus ideas, código o utilidades.
- Manera de forzarnos a escribir documentación, descubrir errores y chequear que todo va bien.

Razones para crear un paquete

- Forma de mantener colecciones de funciones y datos en R que se pueden cargar y descargar en memoria de forma sencilla.
- Excelente forma de compartir con tus colegas o toda la comunidad tus ideas, código o utilidades.
- Manera de forzarnos a escribir documentación, descubrir errores y chequear que todo va bien.

Razones para crear un paquete

- Forma de mantener colecciones de funciones y datos en R que se pueden cargar y descargar en memoria de forma sencilla.
- Excelente forma de compartir con tus colegas o toda la comunidad tus ideas, código o utilidades.
- Manera de forzarnos a escribir documentación, descubrir errores y chequear que todo va bien.

Ingredientes

¿Qué necesito para empezar?

- Una idea: Una técnica novedosa, una reestructuración diferente de código existente, una colección de funciones que hacen más cómodo el trabajo, etc.
- Código y datos: Colección de código y datos que se quieren empaquetar.
- Herramientas: R instalado y en el PATH y opcionalmente un compilador de `TeX` (MikTeX)
  - Windows. `Rtools` `http://cran.r-project.org/bin/windows/Rtools/` MinGW (compiladores de Fortran, C) y utilidades tipo-unix. `C:\Rtools\bin;C:\Rtools\MinGW\bin; RHOME=C:\Archivos de programa\R\R-3.0.2; $RHOME$\bin; $RHOME$\bin\i386; $RHOME$\bin\x64;`
  - Linux. Compiladores de Fortran y C. (`r-base-dev`)

Ingredientes

¿Qué necesito para empezar?

- Una idea: Una técnica novedosa, una reestructuración diferente de código existente, una colección de funciones que hacen más cómodo el trabajo, etc.
- Código y datos: Colección de código y datos que se quieren empaquetar.
- Herramientas: R instalado y en el PATH y opcionalmente un compilador de `TeX` (MikTeX)
  - Windows. `Rtools` `http://cran.r-project.org/bin/windows/Rtools/` MinGW (compiladores de Fortran, C) y utilidades tipo-unix. `C:\Rtools\bin;C:\Rtools\MinGW\bin; RHOME=C:\Archivos de programa\R\R-3.0.2; $RHOME$\bin; $RHOME$\bin\i386; $RHOME$\bin\x64;`
  - Linux. Compiladores de Fortran y C. (`r-base-dev`)

Ingredientes

¿Qué necesito para empezar?

- Una idea: Una técnica novedosa, una reestructuración diferente de código existente, una colección de funciones que hacen más cómodo el trabajo, etc.
- Código y datos: Colección de código y datos que se quieren empaquetar.
- Herramientas: R instalado y en el PATH y opcionalmente un compilador de TeX (MikTeX)
  - Windows. `Rtools` `http://cran.r-project.org/bin/windows/Rtools/` MinGW (compiladores de Fortran, C) y utilidades tipo-unix. `C:\Rtools\bin;C:\Rtools\MinGW\bin; RHOME=C:\Archivos de programa\R\R-3.0.2; $RHOME$\bin; $RHOME$\bin\i386; $RHOME$\bin\x64;`
  - Linux. Compiladores de Fortran y C. (`r-base-dev`)

Creando un paquete (manera más sencilla)

1. Leer (o al menos abrir) el documento ''Writing R extensions''disponible en la ayuda de R. `http://cran.r-project.org/doc/manuals/R-exts.html`
2. Carga en memoria (R workspace) todas las funciones y conjuntos de datos del paquete.
3. Borra cualquier objeto que no quieras incluir.
4. Muevete al directorio donde quieres crear el paquete. `setwd(directorio)`
5. Usa la función `package.skeleton`

Hagasmoslo! I

- Ubícate en el directorio donde quieras crear el paquete (`setwd('C:/tudirectorio' ))`
- Limpia todos los objetos que tengas en memoria (`rm(list=ls(all=TRUE))`)

```
source("http://eio.usc.es/pub/febrero/Paquete/codigo.R")  #Cargamos las fun

package.skeleton("mipaquete")  # Creamos el directorio

> Creating directories ...
> Creating DESCRIPTION ...
> Creating NAMESPACE ...
> Creating Read-and-delete-me ...
> Saving functions and data ...
> Making help files ...
> Done.
> Further steps are described in './mipaquete/Read-and-delete-me'.
```

Función package.skeleton

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

- `name`: Elige un buen nombre para tu desarrollo, no el por defecto.
- `list`: Lista de objetos de R que quieres incluir (en formato carácter).
- `environment`: Nombre del entorno donde están los objetos.
- `path`: ¿Dondé lo quieres poner?
- `force`: Si el directorio existe, ¿lo machaco?
- `namespace`: Crea un objeto `NAMESPACE` y exporta todos los objetos.
- `code_files`: Nombre de los ficheros con el código.

Y ahora, ¿qué?

Después de ejecutar `package.skeleton` se crea un directorio con toda la información necesaria. El archivo `Read-and-delete-me` contiene las instrucciones.

Estructura del directorio

- `DESCRIPTION`: Fichero que debe ser editado conteniendo la información del paquete, autor, licencia y dependencias.
- `man/`: Subdirectorio de los ficheros de ayuda
- `R/`: Subdirectorio con código R
- `data/`: Subdirectorio de conjuntos de datos
- `src/*`: Subdirectorio para código C, Fortran o C++
- `inst/*`: Lo de este directorio se copia directamente al instalar
- `exec/*`: Ejecutables Perl o Java
- `tests/*`: Tests de validación (funciona en tu sistema?)
- `demo/*`: Directorio para incluir demo

¿Qué hacer despues de `package.skeleton`

1. Editar el fichero `DESCRIPTION`
2. Revisar y/o editar el fichero `NAMESPACE`
3. Editar los ficheros de ayuda en `man/`
4. Colocar el código de C/C++/Fortran en `src/`
5. Si es necesario, añade una función `.First.lib()` para cargar alguna librería compartida
6. Ejecuta `R CMD build` para construir el fichero `.tar`
7. Ejecuta `R CMD check` para chequear el paquete

Archivo DESCRIPTION

```
*Package: mipaquete
Type: Package
*Title: What the package does (short line)
*Version: 0.5-1
Date: 2013-10-03
*Author: Manuel Febrero
*Maintainer: Manuel Febrero <manuel.febrero@usc.es>
Authors@R: c(person("M.","Febrero",role=c("aut","cre"),
email="manuel.febrero@usc.es"),
person("M.","Otro",role="ctb"))
Depends: R(>= 1.8.0), fda.usc, fda
Suggests: MASS (Paquetes que uso en ejemplos)
Enhances: --Mejoro algún otro paquete?
Imports: --Otros paquetes de los que use el NAMESPACE
*Description: More about what it does (maybe more than one line)
*License: What license is it under? (GPL-2)
URL: http://www.r-project.org, http://paquete.direccion.com
BugReports: http://elpaquete.notiene.bugs.com
LazyData: true
VignetteBuilder: knitr
```

Archivo NAMESPACE I

```
# Refer to all C/Fortran routines by their name prefixed by C_
#useDynLib(mipaquete, .registration = TRUE, .fixes = "C_")  #

#exportPattern("^[^\\.]") Todo menos ocultos

importFrom(fda.usc, Ker.norm, Ker.epa, Ker.tri)
import(MASS) # Todo lo que tiene el paquete

#exportPattern("^[^\\.]")
#Solo lo que quiera exportar
export(meannp, varnp, xydata)

S3method(print,xydata)
S3method(plot, xydata)
S3method(plot, npmean)
```

Archivo NAMESPACE II

```
S3method(plot, npvar)
S3method(is, xydata)
#S3method(Ops, xydata)
#S3method("[", xydata)
#S3method("!=", xydata)
#S3method("*", xydata)
#S3method("+", xydata)
```

Ficheros de ayuda I

Ficheros de ayuda

- Se escriben en ficheros de texto con la extensión `.Rd`
- Las distintas secciones empiezan con $\backslash seccion\{\}$
- Formato similar a LaTeX pero con muchas menos opciones.
- Es posible incluir ecuaciones matemáticas, figuras, tablas y listas que pueden aparecer de forma diferente en la versión PDF y HTML.
- Estos ficheros se convierten a HTML o LaTeX cuando se crea el paquete
- Para crear ayuda con caracteres especiales el fichero debe incluir `\inputencoding{utf8}` (el mismo nombre que tenga en el paquete de LaTeX– `inputenc`
- Puedes crear tu esqueleto de ayuda de un objeto individual con `prompt(objeto)`

Comandos especiales para formatear ficheros Rd

Para formatear la ayuda se puede usar lo siguiente. Siempre \comando{}

- texto: \emph, \strong, \bold, \code, \preformatted, \kbd, \samp, \verb
- Comillas: \sQuote, \dQuote
- Referencias: \file, \email, \url, \href{direccion}{texto}, \var, \env, \option, \command, \dfn, \reference, \link,\acronym
- Listas y tablas: \enumerate{\item }, \itemize, \describe, \tabular{rlc}{\tab,\cr }
- Matemáticas y figuras: \eqn{latex}{ASCII}, \deqn{latex}{ASCII}, \figure{image.jpg}
- Condicional: \if{format}{text}, \ifelse{format}{text}{alternativa}

Formato de ayuda para funciones I

Formato de ayuda para funciones

- \name{*name*}: Nombre de la función
- \alias{*otro*}: Otras entradas que llevan a la misma ayuda.
- \title{*titulo*}: Título de la función (<65 char)
- \description{...}: Descríbelo. Puedes usar varías líneas.
- \usage{($fun(arg1, arg2, ...)$}: Sintaxis de la función con argumentos. \S3method{gen}{clase}
- \arguments{...}: Descripción de cada argumento.
- \details{...}: Detalle preciso de lo que hace la función.
- \value{...}: Descripción de lo que devuelve. Si es una lista describe cada componente.
- \references{...}: Referencias a la literatura.
- \author{...}: Autor de la ayuda. Usa \email{} o \url{}

Formato de ayuda para funciones II

- \note{...}: Notas. Tienes algo más que añadir?
- \seealso{...}: Referencia a otros objetos de R. Usa \code{\link[pkg]{obj}}
- \examples{...}: Ejemplo de uso. Admite \dontrun{} y \dontshow{}
- \keyword{*key*}: Elige de la lista de R /doc/KEYWORDS

Ejemplo Rd función I

```
\name{xydata}
\alias{xydata}
\title{Crea un objeto de la clase xydata}
\description{ A partir de dos vectores esta función construye un objeto
de la clase xydata}
\usage{xydata(x = NULL, y = NULL)}
\arguments{
  \item{x}{Objeto x}
  \item{y}{Objeto y}
}
\details{Esta función es el constructor de la clase xydata.}
\value{Objeto de la clase xydata
\item{x }{Componente x del objeto}
\item{y }{Componente y del objeto}
}
\references{\url{http://eio.usc.es/pub/febrero}}
\author{ Manuel Febrero \email{manuel.febrero@usc.es}}
\note{}
```

Ejemplo Rd función II

```
\seealso{ \code{\link{print.xydata}}, \code{\link{plot.xydata}}}
\examples{
t=runif(1000)
y=(t-0.5)^2+sin(2*pi*t)/4+rnorm(1000,sd=.1)
xy=xydata(t,y)
plot(xy,pch=19,col=sample(1:5,1000,replace=TRUE))
xy
}
\keyword{ classes }
\keyword{ utilities }% __ONLY ONE__ keyword per line
```

Formato de ayuda para datos I

Formato de ayuda para objetos de datos

- \name{*name*}: Nombre del objeto de datos
- \alias{*otro*}: Otras entradas que llevan a la misma ayuda.
- \docType{*data*}: Siempre data
- \title{*titulo*}: Título del objeto (<65 char)
- \description{...}: Descripción. Puedes usar varías líneas.
- \data{*name*}: Cómo se carga – LazyLoad?.
- \format{...}: Descripción del conjunto de datos. Si es un data.frame debe describirse cada variable.
- \source{...}: Origen de los datos.
- \references{...}: Referencias a la literatura.
- \examples{...}: Ejemplo de uso.
- \keyword{*key*}: Elige de la lista de R /doc/KEYWORDS

Ejemplo Rd dataset I

```
\name{xy}
\alias{xy}
\docType{data}
\title{ Un ejemplo de xydata}
\description{Un ejemplo simulado}
\usage{data(xy)}
\format{The format is:
List of 2
 $ x: num [1:1000] 0.474 0.075 0.234 0.458 0.931 ...
 $ y: num [1:1000] 0.125 0.257 0.155 -0.098 0.138 ...
 - attr(*, "class")= chr "xydata"
}
\source{Los obtuve de esta web \url{esta web}}
\examples{
data(xy)
## maybe str(xy) ; plot(xy) ...
}
\keyword{datasets}
```

Formato de ayuda para Paquetes I

Formato de ayuda para paquetes

- \name{*name*}: Nombre
- \alias{*otro*}: Nombre + otros posibles nombres.
- \docType{*package*}: Siempre package
- \title{*titulo*}: Título del paquete (<65 char)
- \description{...}: Descripción. Puedes usar varías líneas.
- \author{...}: Descripción del conjunto de datos. Si es un data.frame de cada variable.
- \references{...}: Referencias a la literatura.
- \examples{...}: Ejemplo de uso.
- \keyword{*key*}: Elige de la lista de R /doc/KEYWORDS
- \seealso{...}: Referencia a otros paquetes.

Ejemplo Rd package I

```
\name{mipaquete-package}
\alias{mipaquete-package}
\alias{mipaquete}
\docType{package}
\title{Estimación no paramétrica de media y varianza}
\description{Este paquete tiene dos rutinas que estiman media y varianza
en un modelo de regresión no paramétrico}
\details{
\tabular{ll}{
Package: \tab mipaquete\cr
Type: \tab Package\cr
Version: \tab 1.0\cr
Date: \tab 2013-10-03\cr
License: \tab GPL-2\cr
}
Las funciones principales son \code{meannp} y \code{varnp}}
\author{ Manuel Febrero
Maintainer: Yo mismo <micorreo@somewhere.net>
```

Ejemplo Rd package II

```
}
\references{}
\keyword{ package }
\keyword{ smooth  }
\seealso{\code{\link[KernSmooth]{locpoly}} ~~
}
\examples{
t=runif(1000)
y=(t-0.5)^2+sin(2*pi*t)/4+rnorm(1000,sd=.1)
xy=xydata(t,y)
resm=meannp(xy,h=0.05)
plot(resm)
resv=varnp(xy,h=0.05)
plot(resv)
}
```

Ya casi está! I

- Colocate en el directorio que tiene como subdirectorio el del paquete

```
R CMD build mipaquetedir  # Se crea fichero .tar.gz
R CMD check --as-cran mipaquetedir  # Se chequea como si fuese CRAN
---------------------------------------
* using log directory 'C:/Users/febrero/Mis documentos/My Dropbox/
   Presentaciones/Paquetes/mipaquete.Rcheck'
* using R version 3.0.2 (2013-09-25)
* using platform: i386-w64-mingw32 (32-bit)
* using session charset: ISO8859-1
* checking for file 'mipaquete/DESCRIPTION' ... OK
* checking extension type ... Package
* this is package 'mipaquete' version '0.5-1'
* checking CRAN incoming feasibility ... NOTE
Maintainer: 'Manuel Febrero <manuel.febrero@usc.es>'
New submission
* checking package namespace information ... OK
* checking package dependencies ... OK
* checking if this is a source package ... OK
```

Ya casi está! II

```
* checking if there is a namespace ... OK
* checking for executable files ... OK
* checking for hidden files and directories ... NOTE
Found the following hidden files and directories:
  .Rhistory
  .Rproj.user
These were most likely included in error. See section
'Package structure' in the 'Writing R Extensions' manual.

CRAN-pack knows about all of these
* checking for portable file names ... OK
* checking whether package 'mipaquete' can be installed ... OK
* checking installed package size ... OK
* checking package directory ... OK
* checking DESCRIPTION meta-information ... OK
* checking top-level files ... NOTE
Non-standard file found at top level:
  'mipaquete.Rproj'
* checking for left-over files ... OK
```

Ya casi está! III

```
* checking index information ... OK
* checking package subdirectories ... OK
* checking R files for non-ASCII characters ... OK
* checking R files for syntax errors ... OK
* checking whether the package can be loaded ... OK
* checking whether the package can be loaded with stated dependencies
* checking whether the package can be unloaded cleanly ... OK
* checking whether the namespace can be loaded with stated dependencie
* checking whether the namespace can be unloaded cleanly ... OK
* checking dependencies in R code ... OK
* checking S3 generic/method consistency ... WARNING
print:
  function(x, ...)
print.xydata:
  function(xy, ...)

plot:
  function(x, ...)
plot.xydata:
```

Ya casi está! IV

```
  function(xy, ...)

plot:
  function(x, ...)
plot.npmean:
  function(xy.npmean, ...)

plot:
  function(x, ...)
plot.npvar:
  function(xy.npvar, vpar, ...)

See section 'Generic functions and methods' of the 'Writing R
Extensions' manual.
* checking replacement functions ... OK
* checking foreign function calls ... OK
* checking R code for possible problems ... OK
* checking Rd files ... NOTE
prepare_Rd: NW.Rd:14: Dropping empty section \details
```

Ya casi está! V

```
prepare_Rd: mipaquete-package.Rd:19: Dropping empty section \reference
prepare_Rd: xydata.Rd:24: Dropping empty section \note
* checking Rd metadata ... OK
* checking Rd line widths ... OK
* checking Rd cross-references ... OK
* checking for missing documentation entries ... OK
* checking for code/documentation mismatches ... OK
* checking Rd \usage sections ... OK
* checking Rd contents ... OK
* checking for unstated dependencies in examples ... OK
* checking contents of 'data' directory ... OK
* checking data for non-ASCII characters ... OK
* checking data for ASCII and uncompressed saves ... OK
* checking examples ... OK
* checking PDF version of manual ... OK
WARNING: There was 1 warning.
NOTE: There were 4 notes.
------------------------------
```

Ojo con la instalación en los paquetes de TeX necesarios.

Ya casi está! VI

```
R CMD check --timings mipaquete
--------mipaquete-Ex.timings (Tiempo de cada ejemplo <100sg)
name user system elapsed
NW 0.05 0.00 0.05
is.xydata 0 0 0
meannp 0.02 0.00 0.02
mipaquete-package 0.55 0.00 0.55
plot.npmean 0.01 0.00 0.01
plot.npvar 0.24 0.00 0.24
plot.xydata 0.00 0.02 0.01
print.xydata 0 0 0
varnp 0.26 0.00 0.27
xy 0.02 0.01 0.09
xydata 0.02 0.00 0.02
-----------------------------
R CMD check mipaquetes*.tar.gz # Se chequea todo pero del tarball
R CMD INSTALL mipaquete # Instala el paquete en tu directorio
R CMD INSTALL --build mipaquete # Crea el fichero .zip
```

Ya casi está! VII

```
R CMD Rdconv  --help # Convierte Rd a otros formatos: Texto, HTML, LaT
R CMD Rd2pdf  --help # Genera documentación en PDF
R CMD Sweave  --help # Documentación con .Rnw .Snw
R CMD Stangle --help # Extraer codigo de un fichero .Rnw
```

- Ya está listo para enviar al CRAN.
  `http://cran.r-project.org/web/packages/policies.html`
  1. Mediante formulario web
     `http://CRAN.R-project.org/submit.html`
  2. Mediante ftp anónimo (`ftp://CRAN.R-project.org/incoming`) dejando el fichero `.tar.gz` acompañando un correo de texto plano a -CRAN@R-project.org- con el título 'CRAN submission PACKAGE VERSION'

Ejemplo en MS-DOS: Rd2pdf, Rdconv, Sweave, Stangle I

```
C:\Users\>R CMD Rd2pdf mipaquete2
Hmm ... looks like a package
Converting Rd files to LaTeX
Creating pdf output from LaTeX ...
Saving output to 'mipaquete2.pdf' ...
Done

C:\Users>cd mipaquete2/man/

C:\Users\mipaquete2\man>R CMD Rdconv -t html meannp.Rd > mean.html

C:\Users\mipaquete2\man>R CMD Rdconv --type=html mean
np.Rd > mean2.html

C:\Users\mipaquete2\inst\doc>R CMD Stangle mipaquete.Rnw
Writing to file mipaquete.R
```

Ejemplo en MS-DOS: Rd2pdf, Rdconv, Sweave, Stangle II

```
C:\Users\mipaquete2\inst\doc>R CMD Sweave mipaquete.Rnw
Writing to file mipaquete.tex
Processing code chunks with options ...
 1 : echo keep.source term verbatim (mipaquete.Rnw:30)
Loading required package: MASS
 2 : echo keep.source term verbatim (mipaquete.Rnw:36)
 3 : echo keep.source term verbatim pdf  (mipaquete.Rnw:46)
 4 : echo keep.source term verbatim (mipaquete.Rnw:59)
 5 : echo keep.source term verbatim pdf  (mipaquete.Rnw:66)
 6 : echo keep.source term verbatim pdf  (mipaquete.Rnw:72)

You can now run (pdf)latex on 'mipaquete.tex'
```

Ejemplo en R: Rd2pdf, Rdconv, Sweave, Stangle

```
library(utils)
# setwd('C:\\Users\\moviedo\\SgapeioCrearPaquete')
Sweave("./mipaquete2/inst/doc/mipaquete.Rnw")


> Writing to file mipaquete.tex
> Processing code chunks with options ...
>  1 : echo keep.source term verbatim (mipaquete.Rnw:33)
>  2 : echo keep.source term verbatim (mipaquete.Rnw:39)
>  3 : echo keep.source term verbatim pdf  (mipaquete.Rnw:49)
>  4 : echo keep.source term verbatim (mipaquete.Rnw:62)
>  5 : echo keep.source term verbatim pdf  (mipaquete.Rnw:69)
>  6 : echo keep.source term verbatim pdf  (mipaquete.Rnw:75)
>  7 : echo keep.source term verbatim (mipaquete.Rnw:84)
>
> You can now run (pdf)latex on 'mipaquete.tex'


## Crear el pdf
tools::texi2pdf("mipaquete.tex")
## y desde MS-DOS (si MikTex esta disponible) Rcmd texify --pdf
## Sweave-test-1.tex
## Creacion de un fichero R source file desde el codigo de loschunks
Stangle("./mipaquete2/inst/doc/mipaquete.Rnw")


> Writing to file mipaquete.R


source("mipaquete.R")
```

Depurando el código

Encontrar cuellos de botella en fichero `.timmings`.

Tiempo de ejecución

```
Rprof("fichero.out")
res<-mifunción(arg1,arg2,arg3,...)
Rprof(NULL)
```

El fichero se puede analizar con `R CMD Rprof fichero.out` o con `summaryRprof`.

Memoria

- El comando `Rprof` admite el argumento `memory.profiling=TRUE` para analizar la memoria ocupada.
- El comando `Rprofmem(fichero, threshold=1000)` guarda un análisis de las veces que se reserva un objeto de memoria mayor que threshold (en bytes).
- El comando `tracemem(obj)` muestra un mensaje cada vez que `obj` se copia, duplica u opera para obtener un nuevo objeto.

Ejemplo: Depurando el código en R

```
library(fda.usc)
res <- rproc2fdata(200, 1:100, sigma = "brownian")
Rprof("fichero.out")
res <- metric.lp(res)
Rprof(NULL)
Rprofmem("Rprofmem.out", threshold = 1000)
summaryRprof("fichero.out")$by.self
```

```
>                self.time self.pct total.time total.pct
> "int.simpson2"      0.30    44.12       0.48     70.59
> "metric.lp"         0.16    23.53       0.68    100.00
> "-"                 0.06     8.82       0.06      8.82
> "*"                 0.04     5.88       0.04      5.88
> "+"                 0.04     5.88       0.04      5.88
> "!="                0.02     2.94       0.02      2.94
> "%*%"               0.02     2.94       0.02      2.94
> "("                 0.02     2.94       0.02      2.94
> ":"                 0.02     2.94       0.02      2.94
```

```
# noquote(readLines('fichero.out', n = 5))
```

```
[1] sample.interval=20000             "array" "metric.lp"
[3] "int.simpson2" "metric.lp"       "metric.lp"
[5] "%*%" "int.simpson2" "metric.lp"
```

Ejemplo: Depurando el código en MS–DOS I

```
C:\Users>R CMD Rprof fichero.out

Each sample represents 0.02 seconds.
Total run time: 0.72 seconds.

Total seconds: time spent in function and callees.
Self seconds: time spent in function alone.

   %       total       %        self
 total    seconds     self    seconds    name
 100.0      0.72      30.6      0.22     "metric.lp"
  50.0      0.36      41.7      0.30     "int.simpson2"
  11.1      0.08      11.1      0.08     "-"
   2.8      0.02       2.8      0.02     "!="
   2.8      0.02       2.8      0.02     "%*%"
   2.8      0.02       2.8      0.02     "*"
   2.8      0.02       2.8      0.02     "+"
   2.8      0.02       2.8      0.02     "^"
```

Ejemplo: Depurando el código en MS–DOS II

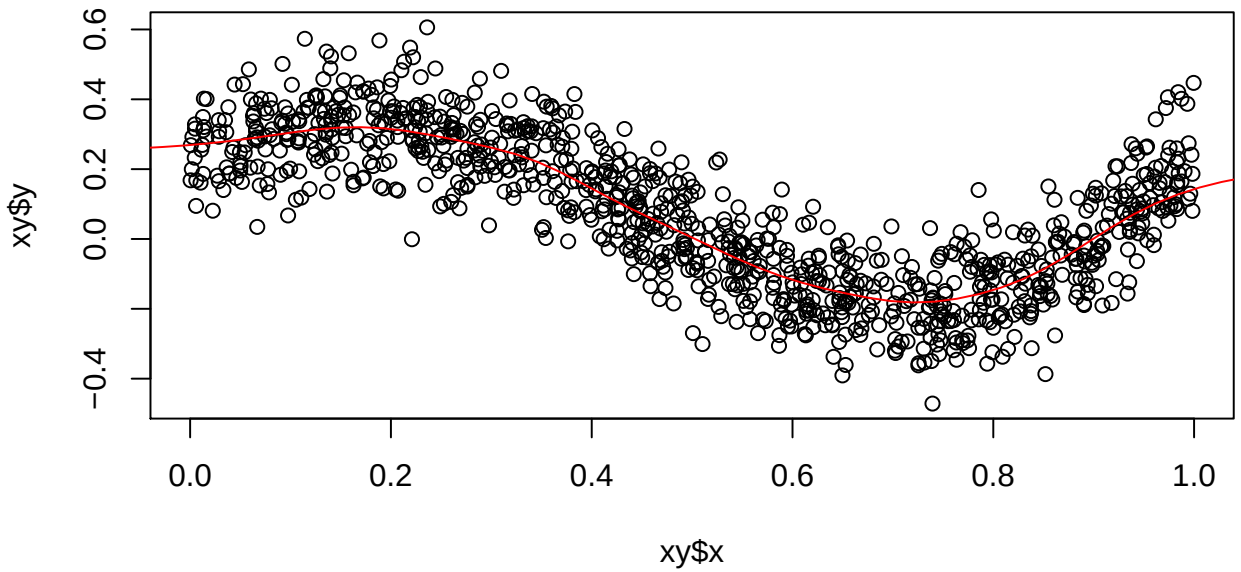
```
 2.8      0.02       2.8      0.02     "array"


 %         self       %       total
self     seconds     total   seconds    name
41.7       0.30       50.0       0.36     "int.simpson2"
30.6       0.22      100.0       0.72     "metric.lp"
11.1       0.08       11.1       0.08     "-"
 2.8       0.02        2.8       0.02     "!="
 2.8       0.02        2.8       0.02     "%*%"
 2.8       0.02        2.8       0.02     "*"
 2.8       0.02        2.8       0.02     "+"
 2.8       0.02        2.8       0.02     "^"
 2.8       0.02        2.8       0.02     "array"
```

Buenas prácticas

- Adaptarse a los tipos de objetos de R cuando sea posible.
- Adaptarse a los métodos genéricos de R (`summary`, `plot`, `predict`).
- Cuando no sea posible lo anterior, definir nuevos objetos, métodos respetando la filosofía de R
- Comprimir adecuadamente los ficheros de datos (`tools:::resaveRdaFiles`).
- Imitar la filosofía de funciones de R similares (`formula`, `model.frame`, `htest`, `lm`, `...`)
- Depurar el código y una vez depurado, volver a depurar.
- Escribir ayudas inteligibles y bien documentadas.
- Una vez publicado el paquete, actualizar como máximo cada 2 meses.

Ejemplo: métodos genéricos de R

```
library(mipaquete)
library(fda.usc)
data(xy)
resm = meannp(xy, h = 0.05)
plot(resm, col = 2)
```



Añadiendo código de otros lenguajes

- Para añadir código Fortran, C o C++ simplemente se crea el directorio `src` y se incluye allí las fuentes.
- –Linux– El código se compila en el momento de la instalación creando una librería compartida con extensión `so`.
- –Windows– El código se compila en el momento de la instalación (o cuando se crea el zip) con versiones para i386 y x64.
- El código debe ser lo más general posible evitando la llamada a librerías específicas que no puedan ser instaladas en cualquier arquitectura.
- En el directorio `src` también se pueden incluir ficheros de configuración más complejos propios de una compilación en fases (`configure.ac, Makevars, ...`)

Creación paquete utilizando Roxygen

`Roxygen` es un paquete de R que permite escribir la ayuda de cada función en el mismo fichero donde se define la función (con campos que empiezan por `' @`.
Se llama antes de la generación del paquete, para construir los ficheros `.Rd` siguiendo las pautas del paquete. Se puede utilizar en:

- MS–DOS: mediante `'R CMD roxygen'`
- R: mediante la función `roxygenize()`

Para mayor detalle véase `http://roxygen.org/`.

RStudio como entorno integrado

- RStudio (`www.rstudio.com`) presenta un entorno integrado donde se pueden editar todos los ficheros necesarios para realizar un paquete.
- Está disponible tanto para Linux como para Windows.
- Un paquete se crea en RStudio como un Proyecto (Nuevo o Existente) apuntando al directorio donde se han creado los ficheros.
- RStudio incluye un par de ficheros y directorios ocultos que debieran ser eliminados cuando ya no se necesiten.
- También permite reorganizar la ayuda y usar `roxygen`.

Índice

Viñetas

Una viñeta es un documento que muestra las excelencias del paquete. Típicamente un manual paso a paso.

- Por defecto, el código de la viñeta se coloca en el directorio `inst\doc` o en el directorio `vignettes`.
- El código fuente (fichero con extensión `.Rnw, .Snw`) se compila y genera el PDF cuando se genera el fichero comprimido
- La viñeta consiste principalmente en código LaTeX que se mezcla con código R en *chunks*.
- Un chunk comienza siempre con <<>>= y termina con @.
- La herramienta por defecto en R es `Sweave` aunque la librería `knitr` permite muchas más opciones (por ejemplo, para usar con `beamer`).

Sweave I

Opciones de Sweave:

- `<<split=FALSE>>`: Divide la salida en varias partes. (TRUE no se usa en viñetas de paquete).
- `<<echo=TRUE>>`: Se imprime o no el código.
- `<<label=nombre>>`: Nombre para el chunk que puede usarse más adelante.
- `<<prefix=TRUE>>`: Todos los ficheros generados tendrán un prefijo común.
- `<<prefix.string=figures/nombre>>`: Prefijo de las figuras.
- `<<fig=TRUE>>`: Se incluye una figura.
- `<<pdf=TRUE, png=TRUE, eps=TRUE, jpg=TRUE>>`: Formato de las figuras.
- `<<eval=TRUE>>`: Se evalúa (o no) el código.
- `<<results=verbatim|tex|hide>>`: Como se escriben los resultados
- `<<width=6>>`: Ancho en pulgadas de los gráficos.
- `<<height=6>>`: Alto en pulgadas de los gráficos.
- `\Sexpr{expr}`: Se usa para escribir una expresión en medio del texto.

Tipos de documentos y viñetas:

`\documentclass[opt1,opt2,...]{tipo}`

- article: para artículos científicos, documentación de programas.
- beamer: para presentaciones.
- report: para documentos extensos con varios capítulos, tesis.
- book: para libros, letter: para cartas,...

```
\documentclass[a4paper]{article}
\title{ Ejemplo 1: Sweave}
\author{ Autores }
\begin{document}
\maketitle
<<>>=
library(mipaquete)
data(xy)
resm=meannp(xy,h=0.05)
class(resm)
names(resm)
@
\end{document}
```

Opciones de Sweave para figuras:

- `<<fig=TRUE>>`: Se incluye una figura.
- `<<pdf=TRUE, png=TRUE, eps=TRUE, jpg=TRUE>>`: Formato de las figuras.
- `<<width=6>>`: Ancho en pulgadas de los gráficos.
- `<<height=6>>`: Alto en pulgadas de los gráficos.

```
<<echo=T,fig=TRUE,width=5,height=4>>=
library(mipaquete)
data(xy)
resm=meannp(xy,h=0.05)
plot(resm)
@
```

Veamos el ejemplo `mipaquete.Rnw`.

knitr

El paquete knitr permite la generación de informes dinámicos con R, combinando características de otros paquetes en un solo paquete (knitr $\approx$ Sweave + cacheSweave + pgfSweave + weaver + animation::saveLatex + R2HTML::RweaveHTML + highlight::HighlightWeaveLatex + ...).

Para generar viñetas automáticamente en la construcción del paquete con `knitr` debe aparecer en el fichero `DESCRIPTION` la línea `VignetteBuilder: knitr` y especificar `knitr` como una dependencia (en Suggests o Depends). Editores para knitr

`http://yihui.name/knitr/demo/editors/`:

- RStudio: Descargar la última versión y compilar PDF con un solo clic a través , `http://yihui.name/knitr/demo/rstudio/`.
- Texmaker,TeXStudio: Se puede definir un comando personalizado para procesar documentos Rnw.
- Work with Emacs, TeXShop, WinEdt and TextMate, etc.

knitr I

knitr permite una personalización más extensa y muchas más opciones.
La función `Sweave2knitr` convierte código de un formato al otro.

Opciones principales de `knitr`:

- `<<eval=TRUE|c(1,3:4)>>`: Se evalúa (o no) el código o las líneas que se evalúan.
- `<<echo=TRUE|c(1,3:4)>>`: Se imprime o no el código o las expresiones que se imprimen.
- `<<results='markup'|'asis'|'hold'|'hide'>>`: Como se escriben los resultados
- `<<warnings=FALSE>>`:Se escriben o no los warnings.
- `<<error=TRUE|FALSE>>`: Se muestran o no los errores.
- `<<split=TRUE>>`: Se dividen o no los resultados.
- `<<tidy=TRUE>>`: Decora el código al escribirlo. `tidy.opts=list(keep.blank.line=FALSE, width.cutoff=60)` permite elegir los detalles.
- `<<prompt=TRUE>>`: Se incluye el prompt en los resultados.
- `<<comment=''>>`: Carácter para los comentarios.

knitr II

- `<<size='normalsize'>>`: Tamaño de los resultados.
- `<<background='F7F7F7'>>`: Color de fondo.
- `<<cache=TRUE>>`: Se guarda en memoria el resultado.

Véase una lista completa de las opciones en `http://yihui.name/knitr/options`.
Los ejemplos creados para mipaquete: `mipaquete-knitr.Rnw` y `mipaquete-beamer.Rnw`.

Ejemplo en knitr de una viñeta

```
%\VignetteIndexEntry{Intro to mipaquete}
%\VignetteEngine{knitr::knitr}

\documentclass[a4paper]{article}
\title{ Ejemplo: knitr}
\author{ Autores }
\begin{document}
\maketitle
<<>>=
library(knitr)
library(mipaquete)
data(xy)
resm=meannp(xy,h=0.05)
@
\end{document}
```

Opciones de knitr para figuras

- `<<fig.path='figure/'>>`: Directorio donde se guardan las figuras
- `<<fig.keep='high'|'all'|'none'|'first'|'last'>>`: ¿Qué figura se guarda?
- `<<fig.show='asis'|'hold'|'animate'|'hide'>>`: Como mostrar los plots.
- `<<dev='CairoPDF'>>`: Un vector con los dispositivos gráficos: bmp, postscript, pdf, png, svg, jpeg, pictex, tiff, win.metafile, cairo_pdf, cairo_ps, CairoJPEG, CairoPNG, CairoPS, CairoPDF, CairoSVG, CairoTIFF, Cairo_pdf, Cairo_png, Cairo_ps, Cairo_svg, tikz. Las opciones de cada dispositivo se incluyen en `dev.args`.
- `<<fig.width=5, fig.height=6>>`: Ancho y Alto en pulgadas de los gráficos.
- `<<out.width='.8\linewidth', out.height='\linewidth'>>`: Reescalado de los gráficos en el texto.
- `<<out.extra='angle=90'>>`: Lista de opciones para la salida.
- `<<fig.align='center'>>`: Alineamiento.
- `<<fig.env='figure', fig.cap='Titulo'>>`: Opciones para figuras.

```
<<fig.width=5, fig.height=4, fig.align = 'center'>>=
plot(resm)
@
```

....y ahora una animación con knitr

- <<fig.show='asis'|'hold'|'animate'|'hide'>>: Como mostrar los plots.

```
<<echo=T,fig.show = 'animate',fig.width=5,fig.height=4>>=
h<-round(seq(0.001,.3,len=20),3)
for (i in 1:20){
resm=meannp(xy,h=h[i])
plot(resm,main=paste("h=",h[i],sep=""),lwd=2,col=2)
legend("bottomleft",legend=paste("h=",h[i],sep=""))
}
@
```

No olvidar poner \usepackage{animate}

Funciones gancho de knitr: Hooks

Son funciones que permiten personalizar la salida de knitr mediante el objeto knit_ hooks. El uso básico es `knit_hooks$set(param = FUN)`, donde `param` es el nombre de una opción chunk, y `FUN` es un función. Hay dos tipos de anzuelos: chunk hooks y output hooks.

Funciones gancho de knitr: chunk hooks

Funciones que se ejecutan antes o después de un trozo de código cuando la opción CHUNK es no nula,

```
foo_hook = function(before, options, envir) {
    if (before) {       ## code to be run before a chunk }
    } else {            ## code to be run after a chunk }
}
```

Por ejemplo, para poner margenes más reducidos (arriba y derecha),

```
<<setup, include=FALSE>>=
knit_hooks$set(small.mar = function(before, options, envir) {
    if (before) par(mar = c(4, 4, .1, .1))  })
@
```

LLamamos a la función gancho en el siguiente CHUNK,

```
<<miplot, small.mar=TRUE>>=
plot(resm,col=2)
@
```

Funciones gancho de knitr: output hooks

Sirven para para modificar y pulir el resultado de los chunks.

- `source`: el código fuente
- `output`: lo que sale por el terminal de R excepto advertencias, mensajes y errores
- `warning`: mensajes de adevertencia de `warning()`
- `message`: mensajes de aviso de `message()`
- `error`: los errores del `stop()`
- `plot`: salida de gráficos
- `inline`: salida del código en línea R
- `chunk`: toda la salida de un chunk
- `document`: la salida de todo el documento

Funciones gancho de knitr: output hooks

```
<<setup, include=FALSE>>=
hook_output = knit_hooks$get("output")
knit_hooks$set(output = function(x, options) {
    if (!is.null(n <- options$out.lines)) {
        x = unlist(stringr::str_split(x, "\n"))
        if (length(x) > n) {
            # truncate the output
            x = c(head(x, n), "....\n")
        }
        # paste first n lines together
        x = paste(x, collapse = "\n")
    }
    hook_output(x, options)
})
opts_chunk$set(out.lines = 4)
@
```

Creación de HTML con el formato Markdown

Markdown está dirigido principalmente a las páginas HTML y es fácil de aprender y escribir. Basta con copiar el siguiente texto en un fichero Rmd en RStudio,

````
## El argumento `before` es de tipo logico
## Si : `before == TRUE` ejecuta el codigo antes del chunk.

```{r}
knit_hooks$set(foo1 = function(before, options, envir) {
  if (before) {
    'Antes del chunk'
  } else {
    'Despues del chunk'
  }
})
```
````

El ejemplo original puede encontrarse en `https://github.com/yihui/knitr-examples/blob/master/045-chunk-hook.Rmd`

Creación de HTML con el formato Markdown

Otros enlaces con ejemplos de uso de markdown

`http://www.rstudio.com/ide/docs/authoring/using_markdown`
`http://rpubs.com/`

Como convertir un Markdown en LaTex: Pandoc

Los archivos Markdown se pueden convertir a LaTeX a través Pandoc y publicar un PDF utilizando la clase LaTeX de Chapman & Hall,
`http://johnmacfarlane.net/pandoc/installing.html`.
`http://yihui.name/en/2013/10/markdown-or-latex/`

Otras aplicaciones interesantes: Shiny

Shiny es una aplicacion de RStudio muy simple que permite a los usuarios de R convertir sus análisis en aplicaciones web interactivas faciles de usar (controles amigables como: menús desplegables y campos de texto).

`http://www.rstudio.com/shiny/`

Índice

Bibliografía

[Sweave] Friedich Leisch (2012). Sweave User Manual. http://www.stat.uni-muenchen.de/~leisch/Sweave/Sweave-manual.pdf.

[Rexts] R Core Team (2013). Writing R Extensions. http://cran.r-project.org/doc/manuals/r-devel/R-exts.html.

[knitr] Yihui Xie (2013). knitr: A General-Purpose Tool for Dynamic Report Generation in R. http://yihui.name/knitr.