

# Crear paquetes en R

Oscar Perpiñán Lamigueiro

Marzo de 2013

## Introducción

Crear un paquete en R

Publicar un paquete

# Definiciones previas

## Creating R Packages: a tutorial

- ▶ **Package:** An extension of the R base system with code, data and documentation in standardized format.
- ▶ **Library:** A directory containing installed packages.
- ▶ **Repository:** A website providing packages for installation.
- ▶ **Source:** The original version of a package with human-readable text and code.
- ▶ **Binary:** A compiled version of a package with computer-readable text and code, may work only on a specific platform.
- ▶ **Base packages:** Part of the R source tree, maintained by R Core.
- ▶ **Recommended packages:** Part of every R installation, but not necessarily maintained by R Core.
- ▶ **Contributed packages:** All the rest.

Crear paquetes en R

Oscar Perpiñán  
Lamigueiro

Introducción

Crear un paquete en R

Publicar un paquete

# ¿Por qué crear y publicar un paquete para R?

- ▶ Es una herramienta cómoda para mantener colecciones coherentes de funciones y datos.
- ▶ Permite publicar código de forma que pueda ser empleado por otros siguiendo unas estructuras comunes.
- ▶ La estructura de un paquete obliga a organizar, limpiar y documentar el código.
- ▶ Al distribuir las herramientas para que otros puedan usarlas, se obtiene realimentación sobre lo publicado, de forma que se aumenta su robustez, se amplían sus funcionalidades y se conecta con otras herramientas y proyectos.
- ▶ “As we enjoy great advantages from the inventions of others, we should be glad of an opportunity to serve others by any invention of ours, and this we should do freely and generously.”

Crear paquetes en R

Oscar Perpiñán  
Lamigueiro

Introducción

Crear un paquete en R

Publicar un paquete

# Algunos consejos genéricos

Crear paquetes en R

Oscar Perpiñán  
Lamigueiro

Introducción

Crear un paquete en R

Publicar un paquete

Extraído de [Best Practices for Scientific Computing](#)

- ▶ Write programs for people, not computers.
- ▶ Automate repetitive tasks
- ▶ Use the computer to record history
- ▶ Make incremental changes
- ▶ Use version control
- ▶ Don't repeat yourself (or others)
- ▶ Plan for mistakes
- ▶ Optimize software only after it works correctly
- ▶ Document design and purpose, not mechanics
- ▶ Collaborate

## Referencias

- ▶ Writing R Extensions
- ▶ Creating R Packages: a tutorial

Introducción

Crear un paquete en R

Publicar un paquete

- ▶ Las fuentes de un paquete de R están contenidas en un directorio que contiene al menos:
  - ▶ Los ficheros DESCRIPTION y NAMESPACE
  - ▶ Los subdirectorios:
    - ▶ R: código en ficheros .R
    - ▶ man: páginas de ayuda de las funciones, métodos y clases contenidos en el paquete.
- ▶ Esta estructura puede ser generada con `package.skeleton`.



# DESCRIPTION

Crear paquetes en R

Oscar Perpiñán  
Lamigueiro

Introducción

Crear un paquete en R

Publicar un paquete

- El fichero DESCRIPTION contiene la información básica sobre el paquete con un formato preestablecido:

```
Package: pkgname
Version: 0.5-1
Date: 2004-01-01
Title: My First Collection of Functions
Authors@R: c(person("Joe", "Developer", role = c("aut", "cre"),
                  email = "Joe.Developer@some.domain.net"),
             person("Pat", "Developer", role = "aut"),
             person("A.", "User", role = "ctb",
                  email = "A.User@whereever.net"))
Author: Joe Developer and Pat Developer, with contributions from A. User
Maintainer: Joe Developer <Joe.Developer@some.domain.net>
Depends: R (>= 1.8.0), nlme
Suggests: MASS
Description: A short (one paragraph) description of what
  the package does and why it may be useful.
License: GPL (>= 2)
URL: http://www.r-project.org, http://www.another.url
```

- Los campos Package, Version, License, Title, Author y Maintainer son obligatorios.

- ▶ R usa un sistema de gestión de *espacio de nombres* que permite al autor del paquete especificar:
  - ▶ las variables del paquete que se exportan (y son, por tanto, accesibles a los usuarios)
  - ▶ las variables que se importan de otros paquetes.
  - ▶ las clases y métodos S3 y S4 que deben registrarse.
- ▶ Este mecanismo queda definido en el contenido del fichero NAMESPACE.

- ▶ El NAMESPACE controla la estrategia de búsqueda de variables que utilizan las funciones del paquete:
  - ▶ En primer lugar busca entre las creadas localmente (por el código de la carpeta R/).
  - ▶ En segundo lugar busca entre las variables importadas explícitamente de otros paquetes.
  - ▶ En tercer lugar busca en el NAMESPACE del paquete base.
  - ▶ Por último busca siguiendo el camino habitual (ver el resultado de `search()`)

- ▶ Para exportar las variables `f` y `g`:

```
export(f, g)
```

- ▶ Para importar **todas** las variables del paquete `pkgExt`:

```
import(pkgExt)
```

- ▶ Para importar las variables `var1` y `var2` del paquete `pkgExt`:

```
importFrom(pkgExt, var1, var2)
```

- ▶ Para registrar el método S3 `print` para la clase `myClass`:

```
S3method(print, foo)
```

- ▶ Para registrar las clases S4 `class1` y `class2`:

```
exportClasses(class1, class2)
```

- ▶ Para registrar los métodos S4 `method1` y `method2`:

```
exportMethods(method1, method2)
```

- ▶ Para importar métodos y clases S4 de otro paquete:

```
importClassesFrom(package, ...)  
importMethodsFrom(package, ...)
```

- ▶ Las páginas de ayuda de los objetos R se escriben usando el formato “R documentation” (Rd), un lenguaje similar a  $\text{\LaTeX}$ .
- ▶ Es aconsejable seguir estas orientaciones: [Guidelines for Rd files](#)
- ▶ Para generar el esqueleto de un fichero Rd es aconsejable usar:
  - ▶ prompt: [genérica](#)
  - ▶ promptClass y promptMethods: [clases y métodos](#).
  - ▶ promptPackage: [paquete](#)
  - ▶ promptData: [datos](#)
- ▶ Todos los comandos disponibles están en el documento [Parsing Rd files](#).

```
\name{load}
\alias{load}
\title{Reload Saved Datasets}
\description{
  Reload the datasets written to a file with the function
  \code{save}.
}
\usage{
  load(file, envir = parent.frame())
}
\arguments{
\item{file}{a connection or a character string giving the
  name of the file to load.}
\item{envir}{the environment where the data should be
  loaded.}
}
\seealso{
  \code{\link{save}}.
}
\examples{
  ## save all data
  save(list = ls(), file= "all.RData")

  ## restore the saved values to the current environment
  load("all.RData")

  ## restore the saved values to the workspace
  load("all.RData", .GlobalEnv)
}
\keyword{file}
```

Introducción

Crear un paquete en R

Publicar un paquete



## ► Comprobar

```
R CMD check myPackage/
```

## ► Construir

```
R CMD build myPackage/
```

## ► Publicar (o actualizar) en un repositorio

- Comprobar un directorio (desde línea de comandos):

```
R CMD check myPackage/
```

- Comprobar un paquete ya construido (desde línea de comandos):

```
R CMD check myPackage.tar.gz
```

- Esta comprobación incluye más de 20 puntos de prueba detallados en el manual [Writing R extensions](#).

## Fuente o binario

Se puede construir un fichero fuente en formato *tarball* (independiente de la plataforma, habitual en sistemas Unix) o en forma binaria (dependiente de la plataforma, habitual para Windows y Mac).

- ▶ Fuente en formato *tarball*
  - ▶ El resultado es un fichero *tarball* `myPackage.tar.gz` que se puede distribuir a cualquier sistema.

```
R CMD build myPackage/
```

- ▶ Comprimido binario
  - ▶ El resultado es una copia comprimida de la versión **instalada** del paquete: depende del sistema operativo.

```
R CMD INSTALL --build myPackage/
```

# Comprobar y construir en sistemas Windows

Crear paquetes en R

Oscar Perpiñán  
Lamigueiro

Introducción

Crear un paquete en R

Publicar un paquete

- ▶ Para paquetes sin código compilado (C, Fortran), también se puede usar `R CMD check` y `R CMD build` en un sistema Windows.
- ▶ Para generar un binario hay que usar `R CMD INSTALL --build`.
  - ▶ Es posible que haya que modificar las variables de entorno `TEMP` y `TMP` de forma que **sólo** contengan caracteres ASCII.
- ▶ Para paquetes con código compilado, o en caso de problemas con los comandos anteriores, hay que usar [Rtools](#).
- ▶ Se pueden instalar fuentes *tarball* con (ver [R installation and administration](#)):

```
install.packages(myPackage.tar.gz, type='source')
```

- ▶ El principal repositorio de paquetes estables es **CRAN**.
  - ▶ Publicar en este repositorio conlleva la aceptación de unas **condiciones**.
  - ▶ Para publicar en CRAN hay que subir el fichero fuente *tarball* resultado de `R CMD build` via FTP anónimo a la dirección <ftp://CRAN.R-project.org/incoming/> y enviar un correo en texto plano a [CRAN@R-project.org](mailto:CRAN@R-project.org).
  - ▶ Es imprescindible haber comprobado el fichero con `R CMD check --as-cran` antes de subirlo al servidor FTP. El resultado de esta comprobación **no** debe contener errores ni advertencias (warnings).
  - ▶ Más detalle en el apartado Submission de las **condiciones de CRAN** y en el manual **R Extensions**.

- ▶ Otros repositorios destacables son:
  - ▶ [R-Forge](#) (versiones de desarrollo)
  - ▶ [Bioconductor](#) (paquetes de bioinformática)
  - ▶ [Proyectos R en Github](#) (versiones de desarrollo)
  - ▶ [RForge](#) (versiones de desarrollo)