Cómo crear paquetes de R

Marcelino de la Cruz Rot1

1. Departmento de Biología y Geología, Física y Química Inorgánica. ESCET. Universidad Rey Juan Carlos. Móstoles.

Autor para correspondencia: Marcelino de la Cruz Rot [[marcelino.delacruz@urjc.es](mailto:marcelino.delacruz@urjc.es)]

# Palabras clave

Ciencia reproducible.

# Keywords

Reproducible research.

# Introducción. ¿Por qué crear un paquete de R ?**[COMPLETAR]**

Algunas ideas:  
- Facilitar la organización, documentación y mantenimiento de nuestro código y datos.  
- Distribuir nuestro código y datos de una manera compacta.  
- Etc.

Existen numerosos recursos en internet con consejos y recetas para crear paquetes de R. La referencia fundamental. sin embargo, es el manual oficial [“Writing R Extensions”](https://cloud.r-project.org/doc/manuals/r-release/R-exts.html), que acompaña cada instalación de R, centrado en la construcción manual, paso a paso, de los paquetes. Además, Wickham (2015) proporciona una visión alternativa basada en la automatización de las tareas necesarias para la construcción y, sobre todo, el matenimiento de paquetes. Para cualquier neófito que quiera llegar a publicar exitosamente sus paquetes en el [repositorio oficial de R](https://cran.r-project.org/), lo más conveniente sería comenzar siguiendo la [aproximación tradicional](https://cloud.r-project.org/doc/manuals/r-release/R-exts.html) .

# Creación de un paquete.

El paquete se podría crear manualmente pero lo más común es emplear funciones dedicadas como package.skeleton(). Imaginemos que hemos creado las funciones sqrtfun1() y sqrtfun2() y la matriz datosX y queremos distribuirlo todo en un nuevo paquete, al que vamos a denominar paqueteR.

sqrtfun1 <- function(x) sqrt(x)  
sqrtfun2 <- function(x) x^(1/2)  
datosX<- matrix(runif(24), nrow=6, ncol=4)

**METER un S3 methods (ej. plot) para ejemplo de 1) cómo incluir en el fichero de ayuda el método y 2) como importar en el namespace (from graphics).**

Con la función package.skeleton() creamos un prototipo del paquete indicando qué objetos (funciones y/o datos) de nuestro workspace queremos distribuir.

package.skeleton(name="paqueteR", list=c("sqrtfun1", "sqrtfun2", "datosX"))

Alternativamente, si queremos incluir todos los objetos presentes en nuestro workspace podríamos escribir simplemente:

package.skeleton(name="paqueteR")

El prototipo de paquete consiste en un conjunto de carpetas y archivos colocados dentro de la nueva carpeta “paqueteR” (con el nombre que le hemos dado al paquete), que se ha creado en el directorio de trabajo.

getwd()  
dir()  
setwd("./paqueteR")  
dir()  
setwd("..")

Podemos usar el explorador de archivos de Windows (o la herramienta correspondiente en otros sistemas operativos) para explorar la estructura del paquete.  
**[METER FIGURA ESTRUCTUTRA DIRECTORIO]**  
Dentro de “paqueteR” habrá tres carpetas (data, man y R) y tres archivos (DESCRIPTION, NAMESPACE y Read-and-delete-me). El fichero Read-and-delete-me nos proporciona un resumen de los pasos necesarios para acabar de construir el paquete (y debería ser borrado antes de continuar). En el caso de un paquete sencillo como paqueteR, dichos pasos serían:  
1) Editar los ficheros de ayuda presentes en la carpeta man.  
2) Editar los ficheros NAMESPACE y DESCRIPTION.  
3) Ejecutar Rcmd build para construir el archivo comprimido tar (es decir, para crear lo que se considera normalmete el código fuente del paquete y se distribuiría a los ususarios).  
4) Ejecutar Rcmd check para comprobar que el paquete cumple los estándares de R y no presenta ningún error.

La carpeta R presentará tantos archivos como funciones hayamos incluido en nuestro paquete (todos con la extensión .R). Se trata de archivos de texto con el código de las funciones, y no necesitan ser editados (a no ser que queramos incluir el código de todas las funciones en un único archivo). La carpeta data incluirá tantos archivos de datos (con la extensión .rda) como objetos de datos hayamos incluído en el programa. Se trata de archivos binarios que no debemos editar.

# Edición de los ficheros de ayuda.

Dentro de la carpeta man (abreviatura de *man*ual) aparecerán varios ficheros con la extensión .Rd, tantos como objetos (funciones y datos) hayamos incluido en el paquete más uno adicional con el nombre del paquete con el añadido -package.Rd. Se trata de archivos de texto (pueden ser editados con cualquier editor de texto) escritos con el formato “R documentation” (Rd), un lenguaje de marcas semejante a (La)TeX pero más sencillo. Por ejemplo, si abrimos con un editor el archivo sqrtfun1.Rd observarenos la siguiente estructura: **[QUITAR ESTE CHUNK Y QUE LO VEA EL USUSARIO EN SU EQUIPO]**

\name{sqrtfun1}  
\alias{sqrtfun1}  
%- 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{  
sqrtfun1(x)  
}  
%- maybe also 'usage' for other objects documented here.  
\arguments{  
 \item{x}{  
%% ~~Describe \code{x} 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'}  
%% \item{comp2 }{Description of 'comp2'}  
%% ...  
}  
\references{  
%% ~put references to the literature/web site here ~  
}  
\author{  
%% ~~who you are~~  
}  
\note{  
%% ~~further notes~~  
}  
  
%% ~Make other sections like Warning with \section{Warning }{....} ~  
  
\seealso{  
%% ~~objects to See Also as \code{\link{help}}, ~~~  
}  
\examples{  
##---- Should be DIRECTLY executable !! ----  
##-- ==> Define data, use random,  
##-- or do help(data=index) for the standard data sets.  
  
## The function is currently defined as  
function (x)   
sqrt(x)  
}  
% Add one or more standard keywords, see file 'KEYWORDS' in the  
% R documentation directory.  
\keyword{ ~kwd1 }% use one of RShowDoc("KEYWORDS")  
\keyword{ ~kwd2 }% \_\_ONLY ONE\_\_ keyword per line

Podemos observar que el contenido del fichero está estructurado mediante una serie de macros, definidos cada uno mediante la barra \, un nombre (el nombre del macro), y las llaves {} (un único o dos pares) entre las cuales pueden aparecer uno o varios argumentos. Por ejemplo, el macro \name{sqrtfun1} define el nombre del fichero de ayuda, y ya viene especificado directamente al usar package.skeleton(). Otros, como \title{} o \description{}, que respectivamente definirían el título de la página de ayuda y una descripción breve de lo que hace la función o funciones documentadas en la misma deberían ser definidas manualmente. En el momento de redactar esta nota (R 3.6-1), desde un punto de vista estrictamente formal (es decir, para que podamos construir el paquete), tan sólo sería necesario que estuviesen correctamente definidos los macros \name{},\alias{} (con el que se indican todos los objetos documentados en el fichero, ver ejemplo más abajo), \title{}, \description{}, y \usage{} (que también viene definido por defecto), pero **para aumentar la utilidad del paquete convendría definir (la mayoría de) el resto de macros** o incluso crear algunos nuevos (como sugiere el comentario sobre la creación de la sección \Warning{}). Por lo tanto, una versión elemental del fichero de ayuda para sqrtfun1() podría ser algo así:

\name{sqrtfun1}  
\alias{sqrtfun1}  
\title{  
Square Root of a Number  
}  
\description{  
Computes the square root of a vector.  
}  
\usage{  
sqrtfun1(x)  
}

Es importante notar que aunque para poder construir el paquete bastaría que los manuales tuvieran la estructura anterior, para poder pasar los filtros de CRAN y poder subir el paquete al repositorio oficial de R deberíamos además definir los macros \arguments{} y \examples{}. Además, el título debería estar *capitalized* (es decir, con la primera letra de los **S**ustantivos, **V**erbos y **A**djetivos en mayúscula y con las **p**reposiciones, **c**onjunciones y **a**rtículos en minúsculas), y la descripción debería acabar en un punto final.

Por otro lado, teniendo en cuenta que la función sqrtfun1() es bastante similar y/o está bastante relacionada con la función sqrtfun2(), podemos aprovechar para documentar ambas en el mismo fichero de ayuda. Esto nos ahorra trabajo (menos teclear y menos posibilidad de errores) y puede facilitar al usuario final elegir la función más adecuada para us necesidades. Para documentar otros objetos en el mismo fichero de ayuda debemos añadir tantos macros \alias{} como objetos adicionales queramos incluir, y en el caso de que sean funciones, aumentar el macro \usage{} incluyendo la instrucción de uso de las nuevas funciones. En nuestro ejemplo:

\name{sqrtfun1}  
\alias{sqrtfun1}  
\alias{sqrtfun2}  
\title{  
Square Root of a Number  
}  
\description{  
Computes the square root of a vector.  
}  
\usage{  
sqrtfun1(x)  
sqrtfun2(x)  
}

En adición a los macros anteriores, que definen secciones dentro de la página de ayuda, otros macros permiten formatear el texto, indicar el set de caracteres que se empleará, etc. Más detalles se pueden consultar en el documento [“Parsing Rd files”](https://developer.r-project.org/parseRd.pdf) disponible en la página de desarrolladores de R.

El fichero denominado paqueteR-package.Rd no es estrictamente necesario en el caso de un paquete pequeño como paqueteR: lo sería en el caso de paquetes más grandes y con muchas funciones, a modo de guía-resumen sobre la funcionalidad del mismo. Por lo tanto podríamos borrarlo en vez de editarlo.

Es bastante común leer o escuchar que la edición y mantenimiento de los ficheros de ayuda de R es un proceso frustrante, penoso, etc, y por ello hay quien ha inventado algún paquete como roxygen2 (Wickham et al., 2018) que permitiría crearlos automáticamente a partir de los comentarios (texto precedido del símbolo #') que acompañan al código de las funciones. Sin embargo, esto no libera al creador del paquete de conocer algunos fundamentos del lenguaje ´Rd´ (los que por otra parte le permitirían editar con soltura cualquier archivo Rd). En el caso del mantenimiento (es decir, actualización) de paquetes, el *sufrimiento* puede ser inexistente si se siguen un par de reglas básicas: 1) definir un nuevo \alias{} para cada nueva función que se adicione al paquete y se documente en un archivo Rd pre-existente y 2) incluir una nueva entrada en la sección \usage{} (y en su caso los nuevos items necesarios en la sección \arguments{}).

## Documentación de métodos S3.

En al caso de que hayamos incluido métodos S3 en nuestro paquete (plot, print, summary, etc), hay que documentarlos también. Lo más conveniente sería hacerlo en la página de ayuda de la función con la que estén relacionados. Por ejemplo, en De la Cruz (2019) habíamos creado un método print.miopolar() y un método plot.miopolar() para, respectivamente, describir y dibujar ordenaciones polares. Dado que estos métodos afectan a los objetos resultantes de la función opolar(), deberíamos documentarlos en el fichero de ayuda de esta función. Por lo tanto, en el fichero opolar.Rd, aparecería lo siguiente:

\name{opolar}  
\alias{opolar}  
\alias{print.miopolar}  
\alias{plot.miopolar}  
\title{  
 Polar Ordination  
}  
\description{  
 Computes and plots polar ordinations in the sense of Bray and Curtis.  
}  
\usage{  
 opolar(x)  
 \method{print}{miopolar}(x,...)  
 \method{plot}{miopolar}(x,...)  
}  
\arguments{  
 \item{x}{ Either a matrix (as argument to \code{opolar}) or a \code{miopolar} object (as argument to the \code{plot} and \code{print} methods)}  
 \item{dots}{Additional arguments passed to the \code{plot} and \code{print} methods)}  
}  
[...]

Es decir, si elegimos documentar los métodos en la página de ayuda de otra función, deberemos definir un macro \alias{} para cada uno de ellos e incluir una refencia a cada uno de ellos en la sección \{usage}. En este caso, debemos usar el macro \method{}{}, que entre el primer par de llaves lleva el tipo de método (plot o print) y en el segundo, la clase de objetos a los que aplican (en este caso a objetos de clase miopar).

## Codificación de caracteres especiales (*encoding*).

Uno de los problemas más frecuentes con los que los ecoinformáticos hispanohablantes deben enfrentarse a la hora de documentar un paquete es el de los caracteres propios del español (por ejemplo, la **ñ** o las vocales con tilde) que no aparecen en el código ASCII básico que se emplea por defecto en el lenguaje Rd. Estos caracteres pueden aparecer en el nombre del autor de las funciones o en las referencias bibliográficas que incluyamos en la documentación. Para solucionarlo debemos definir (al principio del archivo a ser posible, o en todo caso antes de que empiecen a aparecer secciones con caracteres especiales) el macro \encoding{latin1} (que señalará al intérprete de los ficheros Rd la manera apropiada de procesarlos) y, cada vez que aparezca una frase o palabra con dichos caracteres, usaremos el macro \{}{}, incluyendo entre el primer par de llaves el texto con los caracteres hispanos y en el segundo par de llaves una versión del mismo escrita en caracteres ASCII. Por ejemplo:

\name{sqrtfun1}  
\alias{sqrtfun1}  
[...]  
\encoding{latin1}  
\references{  
\enc{Núñez-Gigüela, A. and Tranquila-Marión, B.}{Nunez-Giguela, A. and Tranquila-Marion, B.} (2020) \enc{Cómo}{Como} escribir notas \enc{ecoinformáticas}{ecoinformaticas}. \emph{Ecosistemas}.  
}

De esta manera, cuando el intérprete de los ficheros Rd detecte la existencia de un juego de caracteres con la codificación latin1 en el sistema operativo donde se va a instalar el paquete, usará la primera versión (y en caso opuesto, usara la versión ASCII) para construir las páginas de ayuda.

## Ejemplos, tiempo, don test, dont run, etc. **[COMPLETAR]**

## Comprobación de los ficheros de ayuda.

La redacción de los ficheros de ayuda es uno de los pasos de la construcción de un paquete en la que se suelen cometer más errores (a veces simplemente por fallos al teclear). Para asegurarse de que el fichero está correctamente redactado, y que se convertirá adecuadamente a los diferentes formatos (txt, html o pdf) en los que se mostrará cuando se invoque la ayuda, conviene comprobarlo empeando las facilidades que proporciona R. Bien usando la consola del sistema operativo y tecleando

rcmd Rdconv -t "txt" ./paqueteR/man/sqrtfun1.Rd

o, directamente desde la consola de R empleando la función Rcmd()del paquete tools,

library(tools)  
Rcmd('Rdconv -t "txt" ./paqueteR/man/sqrtfun1.Rd')

se nos presentará en pantalla una versión en formato texto del fichero de ayuda, que podríamos salvar tambien como un fichero. Por ejemplo, si escribimos:

Rcmd('Rdconv -t "html" -o "sqrtfun1.html" ./paqueteR/man/sqrtfun1.Rd')

se creará un fichero denominado ´sqrtfun1.html´. En ambos casos, podemos examinar el resultado para confirmar que todo está correcto y se presenta como queremos o, en caso de que haya errores, realizar las correcciones pertinentes. Esto nos ahorrará problemas y errores en el paso final de construcción del paquete. Nótese que en los ejemplos anteriores estamos indicado la ruta (el *path*) del fichero Rd que queremos convertir (./paqueteR/man/sqrtfun1.Rd) asumiendo que nuestro directorio de trabajo (en el que tenemos abierta la consola del sitema operativo o el que está usando R y que podemos consular con get.wd()) es el mismo en el que está la carpeta paqueteR.

# Edición del archivo DESCRIPTION

Cuando usamos la función package.skeleton() para construir el prototipo de un paquete, por defecto se nos crea un archivo de texto denominado DESCRIPTION. La información que incluida en este archivo es la que aparecerá en la página del paquete si finalmente lo subimos al servidor de CRAN (por ejemplo como en la del paquete [logihist](https://CRAN.R-project.org/package=logihist) o cualquier otro). Por lo tanto es necesario editarlo cuidadosamente. En su interior aparecen esbozados los campos cuya información se debe proporcionar. Excepto los campos Package:, que automáticamente incluye el nobre del paquete y Type: (que indica que se trata de un paquete) es necesario modificar todos los demás.

Package: paqueteR  
Type: Package  
Title: What the package does (short line)  
Version: 1.0  
Date: 2019-08-31  
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?

Nótese que siempre hay un espacio en blanco entre los dos puntos al final del nombre del campo y la información que se incluye en el mismo. En Title: hay que incluir una pequeña descripción (una línea) de lo que hace el paquete. Al igual que en la sección \title{} de los ficheros Rd, el título debe estar “capitalizado”, es decir, con la primera letra de los sustantivos, adjetivos, verbos y adeberbios en mayúscula (y la de los otros tipos de palabras en minúsculas) y **no** acabar en un punto final. No se debe repetir el nombre del paquete ni incluir frases del tipo " A package to…". La Version: por defecto aparece como 1.0, pero podemos poner otro número a discreción. Si se trata de un paquete publicado en CRAN, las sucesivas modificaciones que hagamos deben obligatoriamente tener un número mayor (esto es lógico, pero nada impide caotizar la numeración en nuestras versiones de andar por casa). Son válidas también numeraciones del tipo Version: 0.1-1. La fecha que aparece por defecto (Date:) es la del día en que creamos el prototipo del paquete con package.skeleton(). Sería más preciso indicar la fecha en la que dimos por concluida la construcción del paquete (después de editar y corregir todos los errores que pudieran surgir). En el campo Author: podemos escribir nuestro nombre. En el caso de paquetes con varios autores los podemos añadir separados por una coma. En la [sección 1.1.1](https://cloud.r-project.org/doc/manuals/r-release/R-exts.html#The-DESCRIPTION-file) del manual pueden verse formas alternativas de indicar el papel de cada colaborador. El campo Maintainer: debe incluir el nombre de la persona encargada del mantenimiento del paquete, junto con una dirección válida de correo electrónico (encerrada entre los símbolos <>). En el caso de los paquetes almacenados en el servidor de [CRAN](https://cloud.r-project.org/web/packages/available_packages_by_name.html) este *e-mail* se empleará para notificar cualquier incidencia, error o necesidad de modificación, por lo que es importante especificarlo cuidadosamente. **En el campo Description: se… [COMPLETAR]**

**En el campo License: … [COMPLETAR]**

Si dentro de alguna función de nuestro paquete [usamos funciones de otros paquetes](https://cloud.r-project.org/doc/manual/R-exts.html#Package-Dependencies) (exceptuando las del paquete base) es necesario indicarlo en el archivo DESCRIPTION. Para ello podemos usar el campo Imports: o el campo Depends:, indicando a continuación, separados por comas, el paquete o paquetes a los que pertencen dichas funciones. Si lo indicamos dentro del campo Depends:, cada vez que el usuario final cargue nuestro paquete (con la función library()), cargará también los paquetes indicados en este campo [pondrá el paquete en el árbol de búsqueda]. Si por el contrario lo indicamos con el campo Imports:, importará sólo los *namespaces* (ver debajo) de dichos paquetes. Esto es más eficiente computacionalmente, especialmente si usamos funciones de muchos paquetes diferentes. Un mismo paquete sólo debe aparecer en uno de estos campos (es decir, o bien en Imports: o bien en Depends:), aunque podría aparecer dos veces en cada campo si fuese necesario indicar el rango de versiones del paquete ajeno que resultarían funcionales para el nuestro. A modo de resumen, si nuestras funciones se van a usar conjuntamente con las de otro paquete de las que depende incluiríamos dicho paquete en Depends:; por el contrario, si en nuestro paquete empleamos alguna función aislada de otro paquete, del que no vamos a usar ninguna otra función, lo más sensato sería incluir dicho paqute en Imports:.

## Algunos ejempos de fichero **DESCRIPTION** por si fueran necesarios

Package: replicatedpp2w  
Type: Package  
Title: Two-Way ANOVA-Like Method to Analyze Replicated Point Patterns  
Version: 0.1-2  
Date: 2017-11-24  
Author: Marcelino de la Cruz Rot  
Depends: spatstat.utils  
Maintainer: Marcelino de la Cruz <marcelino.delacruz@urjc.es>  
Description: Test for effects of both individual factors and their interaction on replicated spatial patterns in a two factorial design, as explained in Ramon et al. (2016) <doi:10.1111/ecog.01848>.  
Imports: spatstat  
License: GPL (>=2)

Package: Rramas  
Type: Package  
Title: Matrix Population Models  
Version: 0.1-6  
Date: 2019-04-20  
Depends: diagram  
Author: Marcelino de la Cruz  
Maintainer: Marcelino de la Cruz <marcelino.delacruz@urjc.es>  
Description: Analyzes and predicts from matrix population models (Caswell 2006) <doi:10.1002/9781118445112.stat07481>.  
License: GPL (>=2)  
LazyLoad: yes

Package: overlapptest  
Type: Package  
Title: Test Overlapping of Polygons Against Random Rotation  
Version: 1.1  
Date: 2019-02-05  
Author: Marcelino de la Cruz Rot  
Depends: spatstat  
Suggests: maptools ==> VEGAN  
Maintainer: Marcelino de la Cruz <marcelino.delacruz@urjc.es>  
Description: Tests the observed overlapping polygon area in a collection of polygons against a null model of random rotation, as explained in De la Cruz et al. (2017) <doi:10.13140/RG.2.2.12825.72801>.  
License: GPL (>=2)

# Edición del archivo NAMESPACE **[COMPLETAR]**

El archivo NAMESPACE define el *espacio de nombres* del paquete. Se trata de una guía que indica a R 1) si una función del paquete está disponible (una función exportable) o no (una función interna) y 2) dónde se encuentran las funciones de otros paquetes que se emplean dentro del código de nuestras funciones.

## Algunos ejempos de fichero **NAMESPACE** por si fueran necesarios

# Export regular functions  
export("K2w")  
  
# imports  
import(spatstat)  
importFrom(spatstat.utils, short.deparse)  
# ToDo: "importFrom" only required functions   
  
## Registration of S3 methods  
import(stats)  
import(graphics)  
S3method(print, k2w)  
S3method(plot, k2w)

# Export regular functions  
export("as.tmatrix", "decline", "estambi", "estdemo", "explosion", "project1", "projectn")  
  
# Import all packages listed as Imports or Depends  
# ToDo: "importFrom" only required functions   
import(diagram)  
importFrom("grDevices", "dev.new", "rainbow")  
  
## Registration of S3 methods  
import(stats)  
import(graphics)  
S3method(plot, rmas)  
S3method(plot, summary.rmas)  
S3method(plot, summary.rmas.risk)  
S3method(plot, tmatrix)  
S3method(print, summary.tmatrix)  
S3method(print, tmatrix)  
S3method(summary, rmas)  
S3method(summary, rmas.risk)  
S3method(summary, tmatrix)

# Export regular functions  
export("centroidiam", "check.ventana", "pval", "rotawin", "test.auto.intersection", "test.intersection",  
"test.auto.intersection.p", "test.intersection.p")  
  
# imports  
import(spatstat)  
importFrom("stats", "dist", "median", "runif")  
importFrom("parallel", "makeCluster", "clusterExport", "parSapplyLB", "stopCluster")  
# ToDo: "importFrom" only required functions

# Otras carpetas: inst, src, doc, etc **[COMPLETAR]**

# Automatización con Roxigen, Rdpack, etc. **[COMPLETAR]**

# Agradecimientos

REMEDINAL TE-CM (S2018/EMT-4338).

###### REFERENCIAS

Cruz, M. de la. 2019. Cómo escribir funciones en R. *Ecosistemas*.

Wickham, H. 2015. *R Packages: Organize, Test, Document, and Share Your Code.* O’Reilly.

Wickham, H., Danenberg, P., Eugster, M. 2018. *roxygen2: In-Line Documentation for R*.

###### PIES DE FIGURA

**Figura 1**. Legenda de la figura 1.

###### FIGURE LEGENDS

**Figure 1**. English legend of FIg. 1.

###### FIGURA 1