

Desarrollo de Paquetes

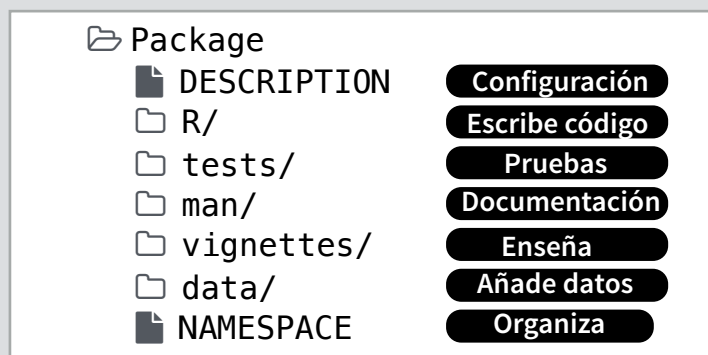
con *devtools* Hoja de Referencia



Estructura de Paquetes

Un paquete es una costumbre, un convenio informal, para organizar archivos en carpetas.

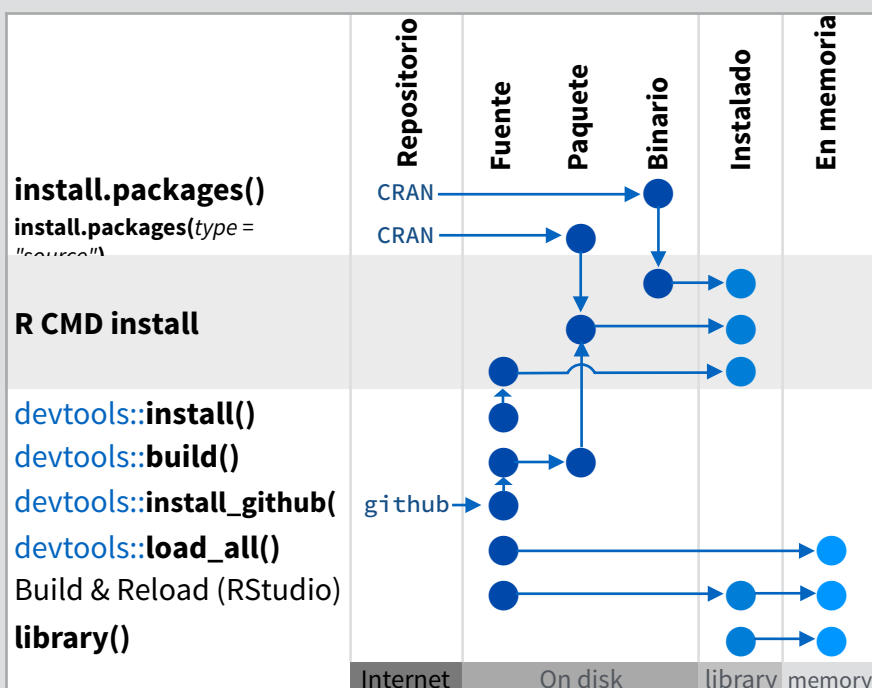
Esta hoja muestra como trabajar con las 7 partes mas comunes de un paquete de R:



El contenido de un paquete se puede guardar en disco como:

- **fuelle** - una carpeta con subcarpetas como arriba
- **paquete** - un solo archivo comprimido (.tar.gz)
- **binario** - un solo archivo comprimido optimizado para un sistema operativo (SO) específico

También se puede instalar en una librería R (cargado a memoria durante una sesión de R) or archivado en línea en un repositorio. Usa las funciones abajo para mover de un estado al otro.



`devtools::add_build_ignore("file")`

Añade archivos a `.Rbuildignore`, una lista de archivos que no se incluirán al construir (build) el paquete.

Configuración (DESCRIPTION)

El archivo `DESCRIPTION` describe tu trabajo y configura como tu paquete va a funcionar con otros paquetes.

- ✓ Debes tener un archivo `DESCRIPTION`
- ✓ Añade los paquetes de los cuales el tuyo depende con `devtools::use_package()`
Añade un paquete al campo `Imports` (o el campo `Suggests` si el segundo argumento es "Suggests").

CCO	MIT	GPL-2
Sin restricciones	Licencia MIT aplica si tu código es compartido por otros.	Licencia GPL-2 aplica a tu código y todo el código que alguien incluye en el paquete si es compartido por otros.

```
Package: mipaquete
Title: Título del paquete
Version: 0.1.0
Authors@R: person("Hadley", "Wickham", email = "hadley@me.com", role = c("aut", "cre"))
Description: Lo que el paquete hace (en un párrafo)
Depends: R (>= 3.1.0)
License: GPL-2
LazyData: true
Imports:
  dplyr (>= 0.4.0),
  ggvis (>= 0.2)
Suggests:
  knitr (>= 0.1.0)
```

Imports: paquetes que tu paquete necesita tener para funcionar. R los instalará cuando instalas tu paquete.

Suggests: paquetes que no son esenciales para el tuyo. Usuarios los pueden instalar manualmente, si gustan, o no instalarlos.

Escribe código (R/)

Todo el código R de tu paquete va a `R/`. Un paquete con solamente una carpeta `R/` directory es un paquete útil.

- ✓ Crea un nuevo proyecto de paquete con `devtools::create("camino/a/nombre")`
Crea una plantilla para desarrollar un paquete.
- ✓ Guardo tu código en `R/` como scripts (extensión `.R`)

Flujo de Trabajo

1. Modifica tu código.
2. Carga tu código con uno de `devtools::load_all()`
Re-carga todos los cambios guardado en `R/` a memoria.
Ctrl/Cmd + Shift + L (abreviado de teclado)
Guarda todos los documentos y llama `load_all()`.
3. Experimenta en la consola.
4. Repite.

- Usa un estilo consistente con r-pkgs.had.co.nz/r.html#style.
- Haz clic en la función y aprieta **F2** para abrir su definición.
- Busca una función con **Ctrl +**

Visita r-pkgs.had.co.nz

Aprende mas en <http://r-pkgs.had.co.nz> • devtools 1.6.1 • Actualizado: 1/15
RStudio® es una marca registrada de of RStudio, Inc. • Todos los derechos reservados
info@rstudio.com • 844-448-1212 • [rstudio.com](https://www.rstudio.com)
Traducido por Frans van Dunné • innovateonline.nl

Pruebas (tests/)

Usa `tests/` para guardar pruebas unitarias que te informarán en el caso de interrupciones en tu código.

- ✓ Añade una carpeta `tests/` e importa `testthat` con `devtools::use_testthat()`
Configura el paquete para usar pruebas automatizadas con `testthat`
- ✓ Escribe pruebas con `context()`, `test()`, y expectativas
- ✓ Guarda tus pruebas como archivos `.R` en `tests/testthat/`

Flujo de Trabajo

1. Modifica tu código o pruebas.
2. Prueba tu código con uno de `devtools::test()`
Corre todas las pruebas guardadas en `tests/`.
Ctrl/Cmd + Shift + T (abreviado de teclado)
3. Repite hasta que todas las pruebas pasan

Prueba ejemplo

```
context("Arithmetic")

test_that("Math works", {
  expect_equal(1 + 1, 2)
  expect_equal(1 + 2, 3)
  expect_equal(1 + 3, 4)
})
```

<code>expect_equal()</code>	es igual dentro de pequeño margen de tolerancia numerica?
<code>expect_identical()</code>	es exactamente igual?
<code>expect_match()</code>	coincide con caracteres especificos o expresion regular?
<code>expect_output()</code>	imprime salida especificada?
<code>expect_message()</code>	muestra mensaje especificado?
<code>expect_warning()</code>	muestra advertencia especificada?
<code>expect_error()</code>	arroja el error especificado?
<code>expect_is()</code>	salida hereda de una clase especifica?
<code>expect_false()</code>	devuelve FALSE?
<code>expect_true()</code>	devuelve TRUE?

Documentación (man/)

man/ contiene la documentación de tus funciones, y las páginas de ayuda para tu paquete.

- ✓ Usa comentarios *roxygen* para documentar cada función a la par de su definición
- ✓ Documenta el nombre de cada conjunto de datos exportado
- ✓ Incluye ejemplos ilustrativos para cada función

Flujo de Trabajo

1. Añade comentarios *roxygen* a tus archivos .R
2. Convierte comentarios *roxygen* en documentación usando uno de

devtools::document()

Convierte comentarios *roxygen* a archivos .Rd y los coloca en man/. También ensambla NAMESPACE.

Ctrl/Cmd + Shift + D (abreviación de teclado)

3. Abre páginas de ayuda con ? para tener una vista previa de la documentación.
4. Repite

Etiquetas de formato para .Rd

\emph{}	\code{\link{}}	\dontrun{}
\strong{}	\link{}	\dontshow{}
\code{}	\link[package]{}	\donttest{}
\pkg{}	\linkS4class{}	
\email{}	\deqn{}	\tabular{lcr}
\href{}	\eqn{}	\tab
\url{}		\cr

El paquete roxygen

roxygen te permite escribir documentación entre las líneas de tu código en los archivos .R con una sintaxis abreviada.

- Añade documentación *roxygen* como líneas de comentario comenzando con #.
- Coloca las líneas de comentarios directamente encima del código que define el objeto documentado.
- Coloca un etiqueta @ inmediatamente después de # para definir una sección específica de la documentación.
- Líneas in etiquetas serán usados para generar un título, descripción y sección de detalles (en ese orden).

```
#' Suma dos números.
#
#' @param x A número.
#' @param y A número.
#' @return La suma de \code{x} y \code{y}.
#' @examples
#' suma(1, 1)
#' @export
suma <- function(x, y) {
  x + y
}
```

Etiquetas roxygen comunes

@aliases	@inheritParams	@seealso
@concepts	@keywords	@format
@describeIn	@param	@source data
@examples	@rdname	@include
@export	@return	@slot S4
@family	@section	@field RC

Añade datos (data/)

La carpeta data/ te permite incluir datos con tu paquete.

- ✓ Guarda datos en uno de las carpetas **data/**, **R/Sysdata.rda**, **inst/extdata**.
- ✓ Siempre usa **LazyData: true** en tu archivo DESCRIPTION.
- ✓ Guarda como archivos .Rdata (sugerencia)

devtools::use_data()

Añade un objeto de datos a data/ (R/Sysdata.rda if **internal = TRUE**)

devtools::use_data_raw()

Añade un script de R para limpiar datos en data-raw/. Incluye data-raw/ en .Rbuildignore.

Guarda datos en

- **data/** para hacerlos accesibles a usuarios del paquete.
- **R/sysdata.rda** para mantener los datos internos, para uso por tus funciones.
- **inst/extdata** para dar acceso a datos crudos al cargar y ejecutar ejemplos. Puedes acceder los datos con **system.file()**.

Organiza (NAMESPACE)

El archivo NAMESPACE te ayuda a crear un paquete auto-contenido: no interfiere con otros paquetes y otros paquetes no interfieren con el.

- ✓ Exporta funciones para usuarios colocando **@export** en sus comentarios *roxygen*.
- ✓ Importa objetos de otros paquetes con **package::object** (recomendado) o **@import**, **@importFrom**, **@importClassesFrom**, **@importMethodsFrom** (no siempre recomendado)

Flujo de Trabajo

1. Modifica tu código o prueba.
2. Documenta tu paquete (**devtools::document()**)
3. Verifica NAMESPACE
4. Repite hasta que NAMESPACE es correcto

Comparte tu paquete

r-pkgs.had.co.nz/release.html

Enseña (vignettes/)

vignettes/ contiene los documentos que enseñan a tus usuarios como resolver problemas reales con tus herramientas.

- ✓ Crea una carpeta vignettes/ y una plantilla con **devtools::use_vignette()**
Añade una plantilla de un vignette como vignettes/my-vignette.Rmd.
- ✓ Añade encabezado YAML a tus vignettes (ejemplo a la derecha)
- ✓ Escribe el cuerpo de tus vignettes en R Markdown (rmarkdown.rstudio.com)

```
---
title: "Titulo del Vignette"
author: "Autor del Vignette"
date: "`r Sys.Date()`"
output: rmarkdown::html_vignette
vignette: >
  %\VignetteIndexEntry{Titulo del Vignette}
  %\VignetteEngine{knitr::rmarkdown}
  \usepackage[utf8]{inputenc}
---
```