Desarrollo de Paquetes : : GUÍA RÁPIDA

Estructura de Paquetes

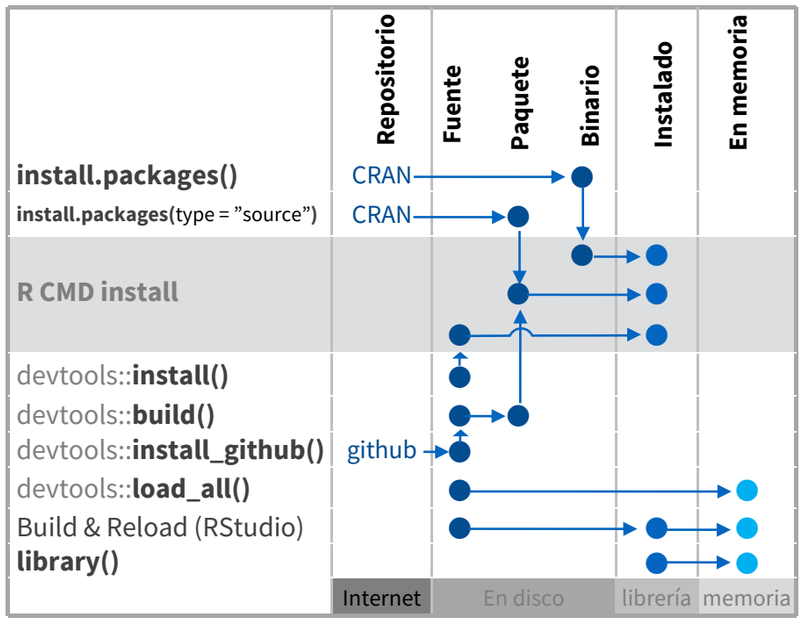
Un paquete es una estructura de carpetas para organizar archivos. Esta hoja muestra como trabajar con los 7 elementos más comunes de un paquete de R:

Package	
DESCRIPTION	CONFIGURACIÓN
R/	ESCRIBE CÓDIGO
tests/	PRUEBAS
man/	DOCUMENTACIÓN
vignettes/	ENSEÑA
data/	AÑADE DATOS
NAMESPACE	ORGANIZA

El contenido de un paquete puede guardarse en disco como:

- **fuente** – una carpeta con subcarpetas como arriba
- **paquete** – un único archivo comprido (*.tar.gz*)
- **binario** – un único archivo comprimido optimizado para un sistema operativo específico

También puede estar instalado en una librería de R (cargado en memoria durante una session de R) o archivado en un repositorio en línea. Usa las siguientes funciones para cambiar entre estos estados.



devtools::**use_build_ignore(***"archivo"***)**

Agrega el *archivo* a .Rbuildignore, una lista de archivos que no serán incluídos cuando se construya el paquete.

Configuración (DESCRIPTION)

El archivo DESCRIPTION describe tu trabajo, define cómo va a funcionar tu paquete con otros y que *copyright* se aplica.

- Debes tener un archivo DESCRIPTION
- Agrega los paquetes de los cuales depende con: devtools::**use_package()**
  Agrega un paquete al campo *Imports* o *Suggests*

CC0	MIT	GPL-2
Uso sin restricciones.	Aplica licencia MIT a tu código si es compartido por otros.	Aplica licencia GPL-2 a tu código, *y todo el código que alguien incluya,* 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 (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* para funcionar. R va a instalarlos cuando instales tu paquete.

Suggests: paquetes que no son esenciales para el funcionamiento de tu paquete. Los usuarios pueden decidir instalarlos o no.

Escribe código (R/)

Todo el código de R en tu paquete va en R/. Un paquete con solo la carpeta R/ ya es un paquete muy útil.

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

FLUJO DE TRABAJO

1. Edita tu código.
2. Carga tu código con una de estas opciones:
   devtools::**load_all()**
   Re-carga todos los archivos *guardados* en R/ en memoria.
   Ctrl/Cmd + Shift + L (atajo de teclado)
   Guarda todos los archivos abiertos y luego ejecuta 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 una función y aprieta **F2** para acceder su definición.
- Busca una función con **Ctrl + .**

Visita r-pkgs.had.co.nz para aprender mucho más sobre escribir y publicar paquetes de R

Pruebas (tests/)

Usa tests/ para guardar las pruebas que van a indicarte si tu código no funciona.

- Agrega el directorio **tests/**
- Importa **testthat** con devtools::**use_testthat()**, para configurar el paquete y que use pruebas automáticas
- Escribe pruebas con **context()**, **test()**, y los resultados esperados
- Guarda tus pruebas como archivos .R en **tests/testthat/**

FLUJO DE TRABAJO

1. Modifica tu código o pruebas.
2. Prueba tu código con alguna de estas opciones:
   devtools::**test()**
   Ejecuta todas las pruebas en tests/
   Ctrl/Cmd + Shift + T
   (atajo de teclado)
3. Repite hasta pasar todas las pruebas.

Ejemplo de Prueba

```
context("Arithmética")

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

Función	Prueba
expect_equal()	es igual con una pequeña torelancia numérica?
expect_identical()	es exactamente igual?
expect_match()	coincide con ciertos caracteres o con una expresión regular?
expect_output()	imprime la salida especificada?
expect_message()	muestra el mensaje especificado?
expect_warning()	muestra la advertencia especificada?
expect_error()	devuelve el error especificado?
expect_is()	la salida tiene una clase específica?
expect_false()	devuelve FALSE?
expect_true()	devuelve TRUE?

Documentación (🗀 man/)

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

- ☑ Usa comentarios de *roxygen* para documentar cada función junto con su definición.
- ☑ Documenta el nombre de cada set de datos exportado.
- ☑ Incluye ejemplos útiles para cada función.

FLUJO DE TRABAJO

1. Agrega comentarios de *roxygen* en tus archivos .R
2. Convierte comentarios de *roxygen* en documentación con alguna de estas opciones:

   devtools::**document()**

   Convierte comentarios de *roxygen* en archivos .Rd y los guarda en 🗀 man/. Construye el archivo NAMESPACE.

   Ctrl/Cmd + Shift + D (Atajo de teclado)
3. Abre páginas de ayuda con **?** para previsualizar la documentación
4. Repite

ETIQUETAS DE FORMATO PARA .Rd

```
\emph{italizada}
\strong{negrita}
\code{función(args)}
\pkg{paquete}

\dontrun{código}
\dontshow{código}
\donttest{código}

\deqn{a + b (bloque)}
\eqn{a + b (en linea)}

\email{nombre@dominio.com}
\href{url}{display}
\url{url}

\link[=dest]{display}
\linkS4class{clase}
\code{\link{función}}
\code{\link[paquete]{función}}

\tabular{lcr}{
  izquierda \tab centrado \tab derecha \cr
  celda     \tab celda    \tab celda   \cr
}
```

ROXYGEN2

El paquete **roxygen2** te permite escribir documentación entre las líneas de tu código en los archivos .R con una sintaxis abreviada. devtools usa roxygen2 para crear la documentación.

- Agrega documentación de *roxygen* como líneas de comentarios que empiezan con **#'**.
- Ubica las líneas de comentarios que define el objeto a documentar por encima de tu código.
- Ubica etiquetas de *roxygen* **@** luego de **#'** para definir secciones específicas en la documentación.
- Las líneas sin etiquetas seran usadas para generar un título, una descripción y los detalles (en ese orden).

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

ETIQUETAS COMUNES DE ROXYGENTAGS

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

Enseña (🗀 vignettes/)

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

- ☑ Crea una carpeta 🗀 vignettes/ y una plantilla de viñeta con: devtools::**use_vignette()**
  Agrega la plantilla como /mi-viñeta.Rmd.
- ☑ Agrega el encabezado YAML a tu viñeta (como en el ejemplo)
- ☑ Escribe el contenido de tu viñeta en *R Markdown* (rmarkdown.rstudio.com)

```
---
title: "Título de la viñeta"
author: "Autor o autora de la viñeta"
date: "`r Sys.Date()`"
output: rmarkdown::html_vignette
vignette: >
  %\VignetteIndexEntry{Título de la viñeta}
  %\VignetteEngine{knitr::rmarkdown}
  \usepackage[utf8]{inputenc}
---
```

Agrega datos (🗀 data/)

La carpeta 🗀 datos/ te permite incuir set de datos en tu paquete.

- ☑ Guarda los datos como archivos .Rdata (sugerencia)
- ☑ Guarda los datos en una de las carpetas **data/**, **R/Sysdata.rda**, **inst/extdata**
- ☑ Usa siempre **LazyData: true** en tu archivo DESCRIPTION.

devtools::use_data()
Agrega un objetos de datos a data/ (R/Sysdata.rda si **internal = TRUE**)

devtools::use_data_raw()
Agrega un Script de R para limpiar los datos en un set de datos en data-raw/. Incluye data-raw/ en .Rbuildignore.

Guarda los datos en:

- **data/** para que estén accesibles para los usuarios del paquete.
- **R/sysdata.rda** para mantener los datos como internos y que los usen tus funciones.
- **inst/extdata** para que los datos crudos esten disponibles para cargar y ejecutar ejemplos. Acede a éstos datos con **system.file()**

Organiza (📄 NAMESPACE)

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

- ☑ Exporta funciones para usuarios usando **@export** en los comentarios de *roxygen*
- ☑ Importa objetos desde otros paquetes con **paquete::objeto** (recomendado) o **@import**, **@importFrom**, **@importClassesFrom**, **@importMethodsFrom** (no siempre recomendado)

FLUJO DE TRABAJO

1. Modifica tu código o pruebas.
2. Documenta tu paquete (devtools::**document()**)
3. Revisa el archivo NAMESPACE
4. Repite hasta que el archivo NAMESPACE es correcto

COMPARTE TU PAQUETE

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

RStudio