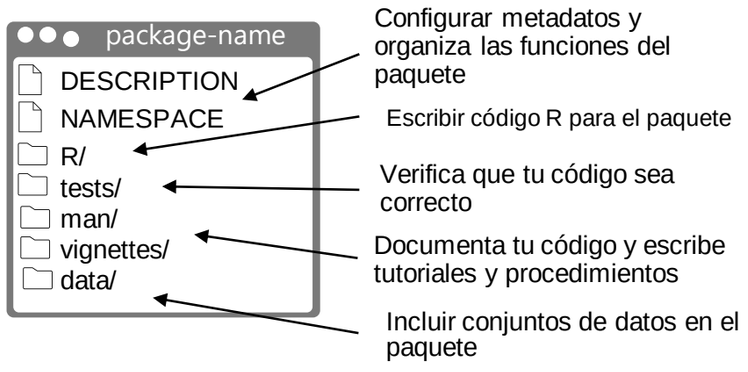
Desarrollo de Paquetes : : GUÍA RÁPIDA

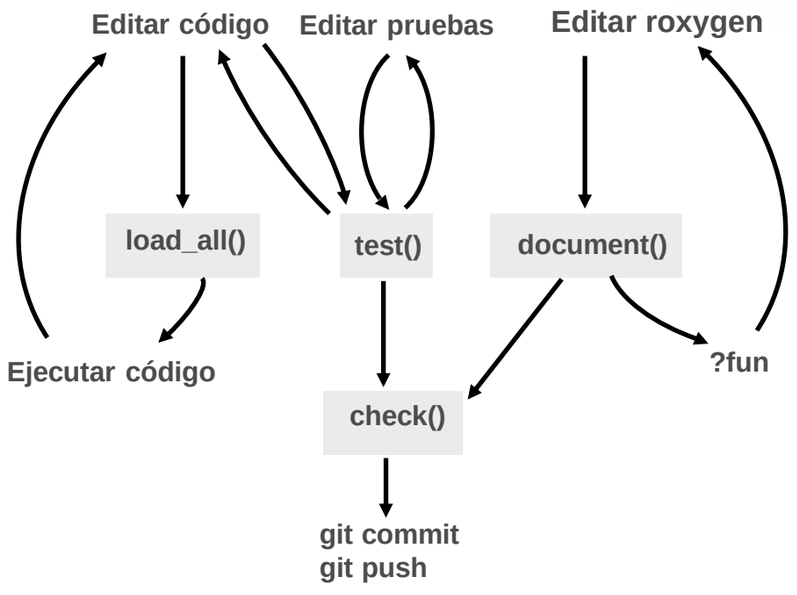
Estructura del Paquete

Un paquete es una convención para organizar archivos en directorios. Esta guía rápida muestra cómo trabajar con las 7 partes más comunes de un paquete R:



Hay varios paquetes útiles para el desarrollo de paquetes, incluido usethis, que automatiza fácilmente muchas de las tareas más repetitivas. Instale y cargue devtools, que reúne varios de estos paquetes para acceder a todo en un solo paso.

Flujo de trabajo



DESCRIPTION

El archivo DESCRIPTION describe su trabajo, configura cómo funcionará su paquete con otros paquetes y aplica una licencia.

- ☑ Elija una licencia con **use_mit_license(), use_gpl3_license(), use_proprietary_license().**
- ☑ Agregue los paquetes que necesite con **use_package().**

Importe los paquetes que el paquete necesita para funcionar. R los instalará cuando instale el paquete.

use_package(x, type = "imports"**)**

Sugiera los paquetes que necesitan los desarrolladores de su paquete. Los usuarios pueden instalar o no, como quieran.

use_package(x, type = "suggests"**)**

Empezando

Una vez por máquina:

- Configure con use_devtools() para que devtools siempre se cargue en sesiones interactivas de R

```
if (interactive()) {
   require("devtools", quietly = TRUE)
   # adjunta automáticamente usethis
}
```

- **create_github_token()** — Configurar las credenciales de GitHub
- **git_vaccinate()** — Ignora los archivos especiales comunes

Una vez por paquete:

- **create_package()** — Creación de un proyecto con scaffolding de paquetes
- **use_git()** — Activar git
- **use_github()** — Conectar con GitHub
- **use_github_action()** — Configurar comprobaciones automatizadas de paquetes

¿Tienes problemas con git? Obtenga un informe de situación con **git_sitrep().**

- **load_all()** (Ctrl/Cmd + Shift + L) — Cargar código
- **document()** (Ctrl/Cmd + Shift + D) — Recompilar documentos y NAMESPACE
- **test()** (Ctrl/Cmd + Shift + T) — Ejecución de pruebas
- **check()** (Ctrl/Cmd + Shift + E) — Revisar paquete completo

R/

Todo el código R del paquete va en R/. Un paquete con solo un directorio R/ sigue siendo un paquete muy útil.

- ☑ Cree un nuevo paquete con **create_package(**"path/to/name"**).**
- ☑ Cree archivos R con **use_r(**"file-name"**).**

- Sigue la guía de estilo de tidyverse en **style.tidyverse.org**
- Haga clic en una función y presione **F2** para ir a su definición
- Busque una función o archivo con **Ctrl + .**

NAMESPACE

El archivo NAMESPACE le ayuda a hacer que su paquete sea autónomo: no interferirá con otros paquetes, y otros paquetes no interferirán con él.

- ☑ Funciones de exportación para los usuarios colocando **@export** en sus comentarios de roxygen.
- ☑ Utilice objetos de otros paquetes con **package::object** o **@importFrom package object** (recomendado) o **@import package** (use con cautela).
- ☑ Llame **document()** para generar NAMESPACE y **load_all()** para volver a cargarlo.

DESCRIPTION	NAMESPACE
Pone a disposición los paquetes	Hace que la función esté disponible
Obligatorio	Opcional (puede usar :: en su lugar)
use_package()	**use_import_from()**

 • info@posit.co • posit.co • Aprenda más en **r-pkgs.org** • Guía rápida en HTML en **pos.it/cheatsheets** • devtools 2.4.5 • usethis 2.2.2 • testthat 3.2.1.1 • roxygen2 7.3.1 • Actualizado: 2024-05

man/

La documentación se convertirá en las páginas de ayuda del paquete.

- ☑ Documente cada función con un bloque roxygen por encima de su definición en R/. En RStudio, Code > Insert Roxygen Skeleton helps (Ctrl/Cmd + Alt + Shift + R).
- ☑ Documente cada conjunto de datos con el bloque roxygen encima del nombre del conjunto de datos entre comillas.
- ☑ Documente el paquete con **use_package_doc().**
- ☑ Construya documentación en man/ a partir de bloques de roxygen con **document().**

vignettes/

- ☑ Cree una viñeta que se incluya con su paquete con **use_vignette().**
- ☑ Crea un artículo que solo aparezca en el sitio web con **use_article().**
- ☑ Escribe el cuerpo de tus viñetas en R Markdown.

Sitios web con pkgdown

- ☑ Usa GitHub y **use_pkgdown_github_pages()** para configurar pkgdown y configurar un flujo de trabajo automatizado mediante GitHub Actions and Pages.
- ☑ Si no usas GitHub, llama a **use_pkgdown()** para configurar pkgdown. A continuación, compile localmente con **pkgdown::build_site().**

tests/

- ☑ Configure la infraestructura de prueba con **use_testthat().**
- ☑ Cree un archivo de prueba con **use_test().**
- ☑ Escribe pruebas con **test_that()** y **expect_()**.
- ☑ Ejecute todas las pruebas con **test()** y ejecute pruebas para el archivo actual con **test_active_file().**
- ☑ Vea la cobertura de todos los archivos con **test_coverage()** y ver la cobertura del archivo actual con **test_coverage_active_file().**

ROXYGEN2

El paquete roxygen2 le permite escribir documentación en su archivo . R con sintaxis abreviada.

- Agregue la documentación de roxygen como comentarios que comiencen con **#'**.
- Coloque una etiqueta de roxygen @ justo después de **#'** para proporcionar una sección específica de documentación.
- Los párrafos sin etiquetar se utilizarán para generar un título, una descripción y una sección de detalles (en ese orden).

```
#' Add together two numbers
#'
#' @param x A number.
#' @param y A number.
#' @returns The sum of `x` and `y`.
#' @export
#' @examples
#' add(1, 1)
add <- function(x, y) {
  x + y
}
```

ETIQUETAS COMUNES DE ROXYGEN

@description	@family	**@returns**
@examples	@inheritParams	@seealso
@examplesIf	**@param**	
@export	@rdname	

README.Rmd + NEWS.md

- ☑ Cree un archivo de markdown README y NEWS con **use_readme_rmd()** y **use_news_md().**

Sentencia Expect	**Pruebas**
expect_equal()	¿Es igual? (dentro de la tolerancia numérica)
expect_error()	¿Arroja un error especificado?
expect_snapshot()	¿La salida no ha cambiado?

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

data/

- ☑ Registre cómo se preparó un conjunto de datos como un script de R y guarde ese script en data-raw/ con **use_data_raw().**
- ☑ Guarde un objeto de datos preparado en data/ con **use_data().**

Estados del Paquete

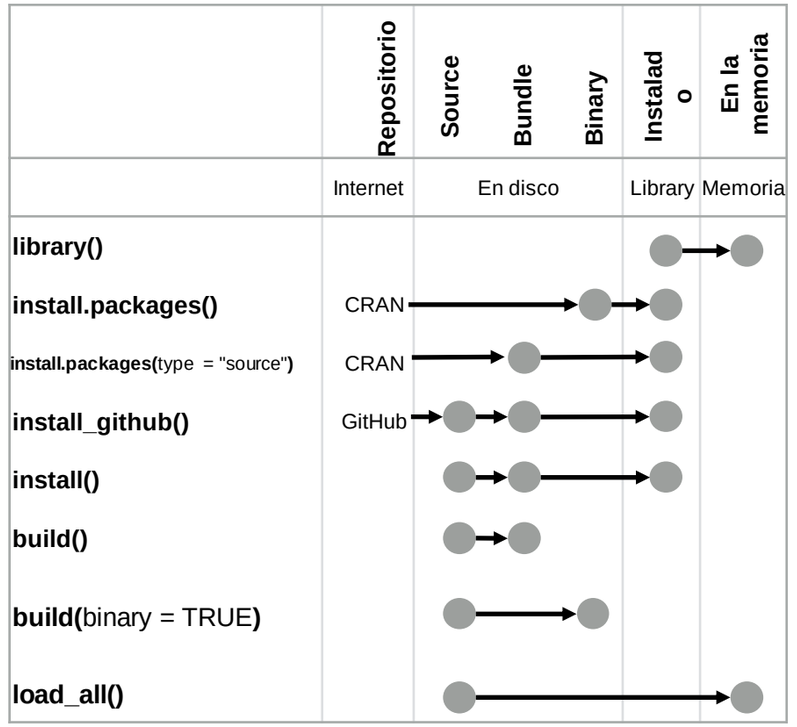
El contenido de un paquete se puede almacenar en el disco como un:

- **source** - un directorio con subdirectorios (como se muestra en Estructura del paquete)
- **bundle** - Un solo archivo comprimido (*.tar.gz*)
- **binary** - un único archivo comprimido optimizado para un sistema operativo específico

Los paquetes existen en esos estados de forma local o remota, por ejemplo, en CRAN o en GitHub.

A partir de esos estados, se puede instalar un paquete en una biblioteca de R y, a continuación, cargarlo en la memoria para usarlo durante una sesión de R.

Utilice las siguientes funciones para desplazarse entre estos estados.



Visite **r-pkgs.org** para obtener más información sobre la escritura y publicación de paquetes para R.