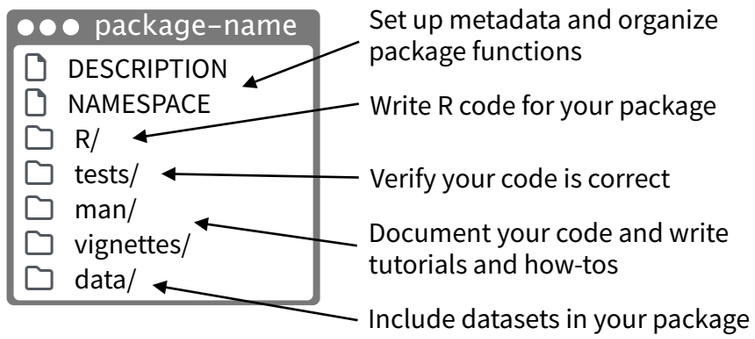
Package Development : : **CHEAT SHEET**

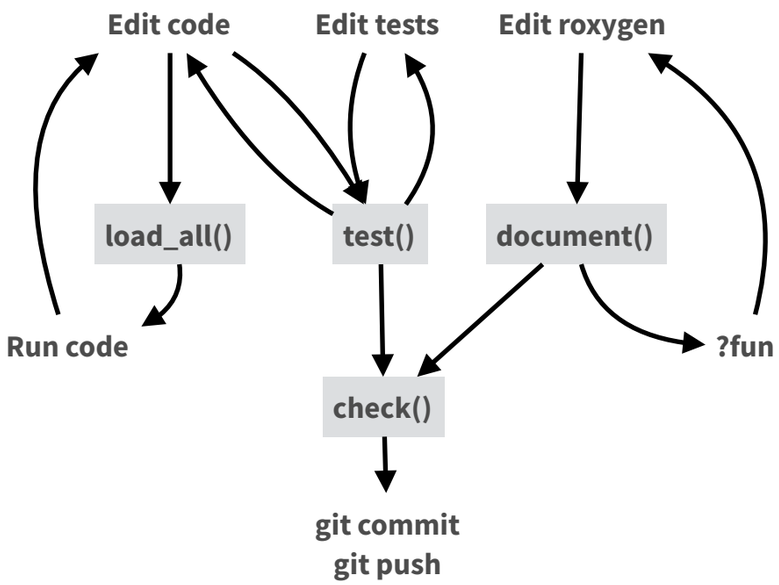
Package Structure

A package is a convention for organizing files into directories. This cheat sheet shows how to work with the 7 most common parts of an R package:



There are multiple packages useful to package development, including **usethis** which handily automates many of the more repetitive tasks. Install and load **devtools,** which wraps together several of these packages to access everything in one step.

Workflow



- **load_all()** (Ctrl/Cmd + Shift + L) — Load code
- **document()** (Ctrl/Cmd + Shift + D) — Rebuild docs and NAMESPACE
- **test()** (Ctrl/Cmd + Shift + T) — Run tests
- **check()** (Ctrl/Cmd + Shift + E) — Check complete package

Getting Started

Once per machine:

- Get set up with **use_r_profile()** so **devtools** is always loaded in interactive R sessions

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

- **create_github_token()** — Set up GitHub credentials
- **git_vaccinate()** — Ignores common special files

Once per package:

- **create_package()** — Create a project with package scaffolding
- **use_git()** — Activate git
- **use_github()** — Connect to GitHub
- **use_github_action()** — Set up automated package checks

Having problems with git? Get a situation report with **git_sitrep().**

R/

All of the R code in your package goes in R/. A package with just an R/ directory is still a very useful package.

- ☑ Create a new package project with **create_package(**"path/to/name"**).**
- ☑ Create R files with **use_r(**"file-name"**).**

- Follow the tidyverse style guide at **style.tidyverse.org**
- Click on a function and press **F2** to go to its definition
- Find a function or file with **Ctrl + .**

DESCRIPTION

The DESCRIPTION file describes your work, sets up how your package will work with other packages, and applies a license.

- ☑ Pick a license with **use_mit_license(), use_gpl3_license(), use_proprietary_license().**
- ☑ Add packages that you need with **use_package().**

Import packages that your package requires to work. R will install them when it installs your package.

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

Suggest packages that developers of your package need. Users can install or not, as they like.

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

NAMESPACE

The NAMESPACE file helps you make your package self-contained: it won't interfere with other packages, and other packages won't interfere with it.

- ☑ Export functions for users by placing **@export** in their roxygen comments.
- ☑ Use objects from other packages with **package::object** or **@importFrom package object** (recommended) or **@import package** (use with caution).
- ☑ Call **document()** to generate NAMESPACE and **load_all()** to reload.

DESCRIPTION	NAMESPACE
Makes **packages** available	Makes **function** available
Mandatory	Optional (can use :: instead)
use_package()	**use_import_from()**

man/

The documentation will become the help pages in your package.

- ☑ Document each function with a roxygen block above its definition in R/. In RStudio, Code > Insert Roxygen Skeleton helps (Ctrl/Cmd + Alt + Shift + R).
- ☑ Document each dataset with roxygen block above the name of the dataset in quotes.
- ☑ Document the package with **use_package_doc().**
- ☑ Build documentation in man/ from Roxygen blocks with **document().**

vignettes/

- ☑ Create a vignette that is included with your package with **use_vignette().**
- ☑ Create an article that only appears on the website with **use_article().**
- ☑ Write the body of your vignettes in R Markdown.

Websites with pkgdown

- ☑ Use GitHub and **use_pkgdown_github_pages()** to set up pkgdown and configures an automated workflow using GitHub Actions and Pages.
- ☑ If you're not using GitHub, call **use_pkgdown()** to configure pkgdown. Then build locally with **pkgdown::build_site().**

tests/

- ☑ Set up test infrastructure with **use_testthat().**
- ☑ Create a test file with **use_test().**
- ☑ Write tests with **test_that()** and **expect_()**.
- ☑ Run all tests with **test()** and run tests for current file with **test_active_file().**
- ☑ See coverage of all files with **test_coverage()** and see coverage of current file with **test_coverage_active_file().**

ROXYGEN2

The **roxygen2** package lets you write documentation inline in your .R files with shorthand syntax.

- Add roxygen documentation as comments beginning with **#'**.
- Place a roxygen **@** tag (right) after **#'** to supply a specific section of documentation.
- Untagged paragraphs will be used to generate a title, description, and details section (in that order).

```
#' 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
}
```

COMMON ROXYGEN TAGS

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

README.Rmd + NEWS.md

- ☑ Create a README and NEWS markdown files with **use_readme_rmd()** and **use_news_md().**

Expect statement	Tests
expect_equal()	Is equal? (within numerical tolerance)
expect_error()	Throws specified error?
expect_snapshot()	Output is unchanged?

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

data/

- ☑ Record how a data set was prepared as an R script and save that script to data-raw/ with **use_data_raw().**
- ☑ Save a prepared data object to data/ with **use_data().**

Package States

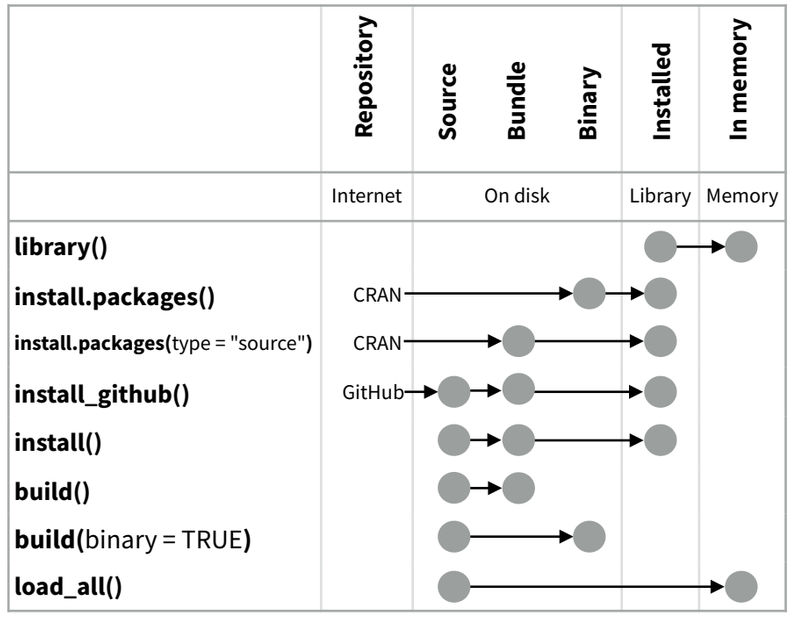
The contents of a package can be stored on disk as a:

- **source** - a directory with sub-directories (as shown in Package structure)
- **bundle** - a single compressed file (*.tar.gz*)
- **binary** - a single compressed file optimized for a specific OS

Packages exist in those states locally or remotely, e.g. on CRAN or on GitHub.

From those states, a package can be installed into an R library and then loaded into memory for use during an R session.

Use the functions below to move between these states.



Visit **r-pkgs.org** to learn much more about writing and publishing packages for R.