

# Package: Rapp (via r-universe)

June 11, 2026

**Title** Easily Build Command Line Applications

**Version** 0.4.0

**Description** Run simple 'R' scripts as command line applications, with automatic robust and convenient support for command line arguments. This package provides 'Rapp', an alternative 'R' front-end similar to 'Rscript', that enables this.

**License** MIT + file LICENSE

**URL** <https://github.com/r-lib/Rapp>

**BugReports** <https://github.com/r-lib/Rapp/issues>

**Encoding** UTF-8

**Suggests** testthat (>= 3.0.0), withr

**Config/testthat/edition** 3

**Config/testthat/parallel** true

**Config/testthat/start-first** help-snapshots, basics

**Imports** yaml12

**Config/roxygen2/version** 8.0.0

**NeedsCompilation** no

**Author** Tomasz Kalinowski [aut, cre]

**Maintainer** Tomasz Kalinowski <tomasz@posit.co>

**Repository** <https://cran.r-universe.dev>

**Date/Publication** 2026-06-11 16:32:16 UTC

**RemoteUrl** <https://github.com/cran/Rapp>

**RemoteRef** HEAD

**RemoteSha** e2b04b68d3438fcc5582001187112151b6d6e4be

## Contents

install_pkg_cli_apps . . . . .	2
run . . . . .	4
<b>Index</b>	<b>6</b>

---

install\_pkg\_cli\_apps *Install CLI launchers for package scripts*

---

### Description

install\_pkg\_cli\_apps() scans an installed package's exec/ directory for .R scripts whose shebang line invokes Rapp (for example, #!/usr/bin/env Rapp) or Rscript (for example, #!/usr/bin/env Rscript). Each discovered script gets a lightweight launcher in destdir that invokes Rapp or Rscript to run the app. The launcher encodes the absolute path to the R binary this function is called from, and its name defaults to the script basename without the .R extension.

### Usage

```
install_pkg_cli_apps(
  package = parent.pkg() %||% rownames(utils::installed.packages()),
  destdir = NULL,
  lib.loc = NULL,
  overwrite = NA
)
```

```
uninstall_pkg_cli_apps(package = parent.pkg(), destdir = NULL)
```

### Arguments

package	Package names to process. Defaults to the calling package when run inside a package; otherwise all installed packages.
destdir	Directory to write launchers to. See Details for defaults.
lib.loc	Additional library paths forwarded to <code>base::system.file()</code> while locating package scripts. Discovery happens at install time; written launchers embed absolute script paths.
overwrite	Whether to replace an existing executable. TRUE always overwrites, FALSE never overwrites non-Rapp executables, and NA (the default) prompts interactively and otherwise skips.

### Details

Optional `#!/ launcher`: front matter in the script lets authors set the installed launcher name and tune the Rscript flags. By default, package apps are invoked with `--default-packages=base, <pkg>`, where `<pkg>` is the package providing the executable. The top-level Rapp launcher installed by `install_pkg_cli_apps("Rapp")` invokes `Rscript -e Rapp::run()` without package app defaults.

Launchers are regenerated each time `install_pkg_cli_apps()` is called, and any obsolete launchers for the same package are removed. `RAPP_BIN_DIR` overrides the default destination. Launchers are POSIX shell scripts on Unix-like systems and .bat files on Windows. Front-matter options such as `name`, `vanilla`, `no-environ`, and `default-packages` map directly to the corresponding Rscript arguments.

When overwrite is NA, files previously written by Rapp are always replaced while other executables trigger a confirmation prompt (skipped in non-interactive sessions). A warning is emitted when skipping an existing executable.

If `destdir` is not provided, it is resolved in this order:

- env var `RAPP_BIN_DIR`
- env var `XDG_BIN_HOME`
- env var `XDG_DATA_HOME/./bin`
- the default location:
  - macOS and Linux: `~/.local/bin`,
  - Windows: `%LOCALAPPDATA%\Programs\R\Rapp\bin`

On Windows, the resolved `destdir` is explicitly added to `PATH` (it generally is not by default). On macOS, when the default `~/.local/bin` destination is not already on `PATH`, it is added to `~/.zprofile`, or `$ZDOTDIR/.zprofile` when `ZDOTDIR` is set. If the profile cannot be updated, a warning is emitted and installation continues. To disable adding it to the `PATH`, set envvar `RAPP_NO_MODIFY_PATH=1`.

On Linux, `~/.local/bin` is typically already on `PATH` if it exists. Note: some shells add `~/.local/bin` to `PATH` only if it exists at login. If `install_pkg_cli_apps()` created the directory, you may need to restart the shell for the new apps to be found on `PATH`.

Example setting launcher args:

```
#!/usr/bin/env Rapp
#| description: About this app
#| launcher:
#|   name: about
#|   vanilla: true
#|   default-packages: [base, utils, mypkg]
```

## Value

Invisibly returns the paths of launchers that were (re)written.

## Examples

```
## Not run:
# Install the launcher for the Rapp package itself: `Rapp`
install_pkg_cli_apps("Rapp")

## End(Not run)
```

---

run	<i>Run an R app.</i>
-----	----------------------

---

## Description

Run an R app.

## Usage

```
run(app, args = commandArgs(TRUE))
```

## Arguments

app	A filepath to an Rapp.
args	character vector of command line args.

## Details

See the package README for full details. <https://github.com/r-lib/Rapp>

## Value

Mainly called for its side effect. For advanced or testing use, it invisibly returns the evaluation environment where the app's expressions ran. If the app did not run (for example, when `--help` is used), it returns `NULL` invisibly.

## Examples

```
# For the example, place 'Rapp', the package examples, and 'R' on the PATH
old_path <- Sys.getenv("PATH")
Sys.setenv(PATH = paste(system.file("examples", package = "Rapp"),
                        system.file("exec", package = "Rapp"),
                        R.home("bin"),
                        old_path,
                        sep = .Platform$path.sep))

# Here is an example app:
# flip-coin.R
writeLines(readLines(
  system.file("examples/flip-coin.R", package = "Rapp")))

if(.Platform$OS.type != "windows") {
  # on macOS and Linux, you can call the app directly
  system("flip-coin.R")
  system("flip-coin.R --n 3")
} else {
  # On windows, there is no support for shebang '#!' style executables
  # but you can invoke 'Rapp' directly
  system("Rapp flip-coin.R")
}
```

*run*

5

```
    system("Rapp flip-coin.R --n 3")  
}
```

```
# restore PATH  
Sys.setenv(PATH = old_path)
```

# Index

`base::system.file()`, 2

`install_pkg_cli_apps`, 2

`run`, 4

`uninstall_pkg_cli_apps`  
    (`install_pkg_cli_apps`), 2