Package 'rok'

November 14, 2026

Title Fast Package and Project Manager

Version 0.2.0

Description Manages the packages and the R version of a project. Packages are resolved from dated snapshots of the 'Posit Package Manager' <https://packagemanager.posit.co> and recorded in a lockfile, so that a project can be reproduced as it was. The work is done by the 'rok' command-line program, which this package downloads from the project's 'GitHub' releases on first use, after asking for user consent.

License MIT + file LICENSE

URL https://github.com/yo5uke/rok

BugReports https://github.com/yo5uke/rok/issues

Encoding UTF-8

Depends R (>= 4.0.0)

Imports tools, utils

Config/roxygen2/markdown TRUE

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Yosuke Abe [aut, cre, cph]

Maintainer Yosuke Abe <yosuke.abe0507@gmail.com>

Repository CRAN

Date/Publication 2026-11-14 07:50:02 UTC

Contents

add 2
export_renv 3
import_renv 4
init 5
install_r 6
pin_r 6

remove . . . 7
rok_available . . . 8
rok_binary . . . 8
run . . . 9
setup . . . 9
status . . . 10
sync . . . 11
tree . . . 11
undo . . . 12
update . . . 13
why . . . 14

Index **15**

---

add *Add packages*

Description

Declares packages in `rok.toml`, then updates `rok.lock` and the project library. Accepts the same forms as pak: `"fixest"` from CRAN, `"owner/repo"` and `"owner/repo@ref"` from GitHub. With `repo`, the packages come from that CRAN-like repository (such as R-multiverse or an r-universe), which is declared in `rok.toml` if it is not yet.

Usage

```
add(..., version = NULL, latest = FALSE, repo = NULL, project = ".")
```

Arguments

`...`	Package names or GitHub repositories.
`version`	A version constraint for one package, such as `"< 0.13"`.
`latest`	Take the packages from the latest snapshot, keeping the project's snapshot.
`repo`	The URL of one CRAN-like repository to take the packages from, such as `"https://community.r-multiverse.org"`. CRAN's own URL means CRAN.
`project`	A directory of the project (default: the working directory).

Details

Some packages build from source better with known environment variables (such as `NOT_CRAN = "true"` for polars). Adding them is offered before anything is installed.

Value

The result (the changes), invisibly.

Examples

```
dir <- tempfile("my-analysis")
init(dir)
add("R6", project = dir)
add("jsonlite", version = ">= 1.8", project = dir)

# GitHub packages, optionally at a tag, branch or commit
add("gaborcsardi/praise", project = dir)

# A package from a CRAN-like repository
add("polars", repo = "https://community.r-multiverse.org", project = dir)
```

`export_renv` *Write renv.lock from rok.lock*

Description

Writes a minimal `renv.lock` (without `Hash`) that `renv::restore()` can use: each package names the dated Posit Package Manager snapshot it comes from, and GitHub packages their commit. Unmanaged packages are left out. To write it every time `rok.lock` changes, add `[export]` with `renv = true` to `rok.toml`.

Usage

```
export_renv(path = NULL, project = ".")
```

Arguments

`path`	Where to write it (default: `renv.lock` in the project, after asking if it exists).
`project`	A directory of the project (default: the working directory).

Value

The result, invisibly.

Examples

```
dir <- tempfile("my-analysis")
init(dir)
add("R6", project = dir)
export_renv(file.path(tempdir(), "renv.lock"), project = dir)
```

`import_renv` *Migrate a project from renv*

Description

Creates `rok.toml` and `rok.lock` from `renv.lock`, with exactly the same package versions (CRAN versions from the snapshot dates they were current, GitHub packages at the same commits). The packages to declare are the ones no other package needs and the ones the code uses; you are asked before anything is written. `renv.lock` and `renv/` are left as they are, and `undo()` reverts the migration.

Usage

```
import_renv(file = NULL, unsupported = NULL, project = ".")
```

Arguments

`file`	The renv lockfile (default: `renv.lock` in the project).
`unsupported`	What to do with packages from sources rok cannot install yet (such as Bioconductor): `"unmanaged"` keeps them as unmanaged packages, `"abort"` stops (asked if not given).
`project`	A directory of the project (default: the working directory).

Value

The result, invisibly.

Examples

```
dir <- tempfile("from-renv")
dir.create(dir)
writeLines(c(
  '{"R": {"Version": "4.6.1", "Repositories": []},',
  ' "Packages": {"R6": {"Package": "R6", "Version": "2.6.1", "Source": "Repository"}}}'
), file.path(dir, "renv.lock"))
import_renv(project = dir)
```

`init` *Create a project*

Description

Creates `rok.toml`, `rok.lock` and the startup hook in `path`. The project uses the minor version of the R that is running, and the latest snapshot date that Posit Package Manager has published. If the rok binary is not installed yet, `setup()` installs it first.

Usage

```
init(path = NULL, r = NULL, name = NULL)
```

Arguments

`path`	The project directory (created if needed). By default, the working directory, after asking.
`r`	The R version, such as `"4.6"` or `"latest"` (default: the running R).
`name`	The project name (default: the directory name).

Details

Without `path`, the project is created in the working directory, after asking (the same question also covers installing the binary, the first time). Without anyone to ask, give `path` or set `options(rok.yes = TRUE)`.

In an interactive session, R then switches to the project: in Positron or RStudio, R restarts (after asking if the global environment has objects), or the IDE opens the new project's folder; elsewhere, you are told to start R in the project's folder. In Positron, the project's R is also written to `.vscode/settings.json`, after asking.

Value

The result, invisibly.

Examples

```
dir <- tempfile("my-analysis")
init(dir)
list.files(dir, all.files = TRUE, no.. = TRUE)
```

install_r *Install R*

Description

Installs R without administrator rights. Without `version`, installs the version the project in the working directory needs (or, outside a project, the latest release). Before installing, it says which version and where it goes, and asks (without anyone to ask, set `options(rok.yes = TRUE)`).

Usage

```
install_r(version = NULL, system = FALSE)
```

Arguments

`version`	A version such as `"4.6"`, `"4.6.1"` or `"latest"`.
`system`	Only show the commands that install R for all users (they need `sudo`).

Value

The result, invisibly.

Examples

```
# Show how to install the latest R for all users (nothing is installed)
install_r("latest", system = TRUE)
```

pin_r *Change the project's R version*

Description

Moves the project to another minor version of R and resolves its packages again. If some packages have no binary for the new R, asks how to handle them. If the new R is not installed, offers to install it.

Usage

```
pin_r(version, strategy = NULL, project = ".")
```

Arguments

`version`	A version such as `"4.6"`, `"4.6.1"` or `"latest"`.
`strategy`	What to do with packages without binaries: `"move"`, `"build"`, `"date"` or `"today"` (asked if not given).
`project`	A directory of the project (default: the working directory).

Value

The result, invisibly.

Examples

```
dir <- tempfile("my-analysis")
init(dir)
pin_r("latest", project = dir)
```

`remove` *Remove packages*

Description

Removes packages from `rok.toml`, then updates `rok.lock` and the project library. The packages stay in rok's cache.

Usage

```
remove(..., project = ".")
```

Arguments

`...`	Package names (or `"owner/repo"` for GitHub packages).
`project`	A directory of the project (default: the working directory).

Details

When rok is attached, this function masks `base::remove()`. Calls meant for it still reach it: if an argument is an unquoted name (`remove(x)`), or `list`, `pos`, `envir` or `inherits` is given, the call goes to the `remove()` that rok masks. To remove packages whose names are in a variable, call `rok::remove(pkgs)`.

Value

The result (the changes), invisibly.

Examples

```
dir <- tempfile("my-analysis")
init(dir)
add("R6", project = dir)
remove("R6", project = dir)
```

rok_available *Whether the rok binary is ready*

Description

Checks that a rok binary with the same version as this package can be found (see `rok_binary()`). If not, `setup()` installs one.

Usage

```
rok_available()
```

Value

`TRUE` or `FALSE`.

Examples

```
rok_available()
```

rok_binary *Locate the rok binary*

Description

Looks for the binary in this order: the option `rok.binary`, the environment variable `ROK_BINARY`, the place `setup()` puts it, and the `PATH`. The binary must have the same version as this package.

Usage

```
rok_binary()
```

Value

The path of the binary. An error if no matching binary is found.

See Also

`rok_available()` to check without an error.

Examples

```
rok_binary()
```

run *Run a script with the project's R*

Description

Syncs the project, then runs `file` with `Rscript` in a separate process, using the project's R (or `r`) and library.

Usage

```
run(file, args = character(), r = NULL, project = ".")
```

Arguments

`file`	The script.
`args`	Arguments passed to the script.
`r`	The R version to use instead of the project's, such as `"4.5"`.
`project`	A directory of the project (default: the working directory).

Value

The exit status of the script, invisibly.

Examples

```
dir <- tempfile("my-analysis")
init(dir)
script <- file.path(dir, "hello.R")
writeLines('cat("Hello from", R.version.string, "\\n")', script)
run(script, project = dir)
```

setup *Install the rok binary*

Description

Downloads the rok binary that matches this package (or copies `binary`), and keeps a copy of this package where activated projects can load it. Everything goes under `tools::R_user_dir("rok", "data")`, after asking once. `init()` calls this when needed; run it yourself to repair an installation or to install without the network.

Usage

```
setup(binary = NULL)
```

Arguments

binary	The path of a rok binary to install instead of downloading one.

Value

A list with the paths of the binary and of the package library, invisibly.

Examples

```
setup()
```

---

status *Show the project's status*

Description

Shows what is out of sync between `rok.toml`, `rok.lock`, the library and R, without using the network, and returns the details invisibly.

Usage

```
status(packages = FALSE, project = ".")
```

Arguments

packages	List every package.
project	A directory of the project (default: the working directory).

Value

A list, invisibly: `ok` (whether there is nothing to fix), `problems` (a data frame with one row per problem) and `packages` (a data frame with one row per package), among others.

Examples

```
dir <- tempfile("my-analysis")
init(dir)
s <- status(project = dir)
s$ok
```

sync *Make the library match the lockfile*

Description

Updates `rok.lock` first if `rok.toml` changed.

Usage

```
sync(locked = FALSE, project = ".")
```

Arguments

`locked`	Fail instead of updating `rok.lock` when it is out of date.
`project`	A directory of the project (default: the working directory).

Value

The result, invisibly.

Examples

```
dir <- tempfile("my-analysis")
init(dir)
sync(project = dir)
```

tree *Show the dependency tree*

Description

Show the dependency tree

Usage

```
tree(pkg = NULL, depth = NULL, project = ".")
```

Arguments

`pkg`	Show only this package's dependencies (default: every declared package).
`depth`	How many levels to show.
`project`	A directory of the project (default: the working directory).

Value

The lines of the drawing, invisibly.

Examples

```
dir <- tempfile("my-analysis")
init(dir)
add("R6", project = dir)
tree(project = dir)
```

---

undo *Undo the last change*

Description

Restores `rok.toml` and `rok.lock` as they were before the last `add()`, `remove()`, `update()` or `pin_r()`, and the library with them.

Usage

```
undo(project = ".")
```

Arguments

`project`	A directory of the project (default: the working directory).

Value

The result, invisibly.

Examples

```
dir <- tempfile("my-analysis")
init(dir)
add("R6", project = dir)
undo(project = dir)
```

`update` *Update packages*

Description

Without packages, moves the project's snapshot date to the latest one (or `to`) and resolves again. With packages, updates only those (and what they need) from the latest snapshot, keeping the project's date.

Usage

```
update(..., to = NULL, dry_run = FALSE, project = ".")
```

Arguments

`...`	Packages to update (default: all).
`to`	The snapshot date to move to.
`dry_run`	Show what would change without changing anything.
`project`	A directory of the project (default: the working directory).

Details

When rok is attached, this function masks `stats::update()`. Calls meant for it still reach it: if the first argument is not a character vector (a fitted model or a formula, as in `update(fit, . ~ . + x)`), the call goes to the `update()` that rok masks, as if rok were not attached.

Value

The result (the changes), invisibly.

Examples

```
dir <- tempfile("my-analysis")
init(dir)
add("R6", project = dir)
update(dry_run = TRUE, project = dir)
update("R6", project = dir)

# Models are still updated by stats::update()
fit <- lm(dist ~ speed, data = cars)
update(fit, . ~ . + I(speed^2))
```

why *Why a package is installed*

Description

Shows the packages that need pkg, up to the declared ones.

Usage

```
why(pkg, project = ".")
```

Arguments

pkg	A package name.
project	A directory of the project (default: the working directory).

Value

The lines of the drawing, invisibly.

Examples

```
dir <- tempfile("my-analysis")
init(dir)
add("jsonlite", project = dir)
why("jsonlite", project = dir)
```

Index

add, 2
add(), *12*

base::remove(), *7*

export_renv, 3

import_renv, 4
init, 5
init(), *9*
install_r, 6

pin_r, 6
pin_r(), *12*

remove, 7
remove(), *12*
rok_available, 8
rok_available(), *8*
rok_binary, 8
rok_binary(), *8*
run, 9

setup, 9
setup(), *5*, *8*
stats::update(), *13*
status, 10
sync, 11

tree, 11

undo, 12
undo(), *4*
update, 13
update(), *12*

why, 14