Package 'courieR'

May 30, 2026

Type Package

Title Migrate Installed R Packages Between R Versions

Version 0.2.0

Description Detects all R installations on the current machine and migrates installed R packages between them. Provides find_routes() to discover R versions, manifest() to scan package libraries via 'subprocess', inventory() to compare two libraries, and ship() to install packages into a target R version using 'pak'. Includes a Shiny dashboard (open_hub()) for interactive one-way and two-way migration.

License MIT + file LICENSE

URL https://github.com/lennon-li/courieR

BugReports https://github.com/lennon-li/courieR/issues

Language en-US

Encoding UTF-8

RoxygenNote 8.0.0

Config/testthat/edition 3

Imports processx (>= 3.8.0), callr (>= 3.7.0), pak (>= 0.7.0), jsonlite (>= 1.8.0), desc (>= 1.4.0), fs (>= 1.6.0), cli (>= 3.6.0), data.table (>= 1.14.0), shiny (>= 1.8.0), bslib (>= 0.7.0), bsicons (>= 0.1.2), DT (>= 0.31), stringr (>= 1.5.0)

Suggests testthat (>= 3.0.0), withr (>= 3.0.0), mockery (>= 0.4.4), knitr (>= 1.45), rmarkdown (>= 2.26)

VignetteBuilder knitr

NeedsCompilation no

Author Lennon Li [aut, cre]

Maintainer Lennon Li <yeli@biostats.ai>

Repository CRAN

Date/Publication 2026-05-30 13:20:02 UTC

Contents

`dispatch` *Run an R command in the background and log output*

Description

Run an R command in the background and log output

Usage

```
dispatch(
  project_path,
  expr,
  phase,
  label,
  rscript_path = NULL,
  timeout_sec = 600L
)
```

Arguments

`project_path`	Path to the project
`expr`	Expression to run (as a quoted expression or function)
`phase`	Character: "baseline" or "post_migration"
`label`	Character: "document", "test", or "check"
`rscript_path`	Optional path to Rscript
`timeout_sec`	Timeout in seconds

Value

A list with process info

Examples

```
tmp <- tempdir()
job <- dispatch(tmp, "message('hello')", "baseline", "document")
Sys.sleep(1)
job$process$is_alive()
```

`find_routes` *Detect R installations on the system*

Description

Scans the current machine for every R installation it can find, across multiple sources per platform, and returns a tidy data frame of results.

Usage

```
find_routes(search_paths = NULL)
```

Arguments

`search_paths`	An optional character vector of additional paths to search. Each element may be a directory containing `bin/Rscript` (or `bin/x64/Rscript.exe` on Windows), or a direct path to an `Rscript` executable.

Details

Detection sources by platform:

Windows

- HKLM registry (`SOFTWARE\R-core\R`) — standard admin installs via the CRAN Windows installer.
- HKCU registry (`SOFTWARE\R-core\R`) — non-admin installs that register under the current user hive only.
- `%ProgramFiles%\R` — directory scan for admin installs not in the registry.
- `%LOCALAPPDATA%\Programs\R` — rig-managed and other user-local installs.
- `%USERPROFILE%\Documents\R` — installs placed in the user's Documents folder.
- rig (`rig list`) — any additional versions managed by rig that were not found by path scanning.

macOS

- `/Library/Frameworks/R.framework/Versions` — system-wide CRAN installer.

- `~/Library/Frameworks/R.framework/Versions` — user-local framework installs (no admin required).
- Homebrew: `/opt/homebrew/opt/r` (Apple Silicon) and `/usr/local/opt/r` (Intel).
- rig (`rig list`) — rig-managed versions.

Linux

- `/opt/R` — rig system-wide installs.
- `~/.local/share/rig/R` — rig user-local installs.
- conda environments (active `$CONDA_PREFIX`).
- System `Rscript` on `$PATH`.

Symlinks are resolved via `fs::path_real()` so that duplicate entries from different detection sources pointing to the same executable are collapsed.

Value

A data frame with one row per unique R installation and the following columns:

version Character. R version string, e.g. `"4.4.1"`.

rscript_path Character. Absolute path to the `Rscript` executable.

is_current Logical. `TRUE` for the R session running courieR.

source Character. Detection source label (e.g. `"registry-hklm"`, `"registry-hkcu"`, `"programfiles"`, `"appdata"`, `"documents"`, `"homebrew"`, `"rig"`, `"search_paths"`).

Examples

```
routes <- find_routes()
routes[, c("version", "rscript_path", "is_current")]

# include a non-standard install
routes <- find_routes(search_paths = "/opt/custom-r/bin/Rscript")
```

---

inspect_shipment *Detect project characteristics*

Description

Detect project characteristics

Usage

```
inspect_shipment(project_path)
```

Arguments

`project_path`	Path to the project

Value

A named list

Examples

```
res <- inspect_shipment(tempdir())
res$is_package
```

`inventory` *Compare two package libraries*

Description

Compare two package libraries

Usage

```
inventory(source_pkgs, target_pkgs)
```

Arguments

`source_pkgs`	data.table from `manifest`
`target_pkgs`	data.table from `manifest`

Value

A list of data.tables and a summary data.frame

Examples

```
src <- data.table::data.table(
  package  = c("dplyr", "ggplot2"),
  version  = c("1.1.4", "3.5.1"),
  priority = NA_character_
)
tgt <- data.table::data.table(
  package = "dplyr",
  version = "1.0.0"
)
inventory(src, tgt)
```

manifest *List packages installed in a library, optionally via a different R executable*

Description

List packages installed in a library, optionally via a different R executable

Usage

```
manifest(
  rscript_path = NULL,
  lib_path = NULL,
  format = c("data.table", "data.frame"),
  timeout_sec = 30L
)
```

Arguments

rscript_path	Path to the Rscript executable. Defaults to current session.
lib_path	Library path to query. Defaults to default `.libPaths()` of the target R.
format	Return format
timeout_sec	Timeout for subprocess

Value

data.table

Examples

```
pkgs <- manifest()
head(pkgs)
```

open_depot *Ensure the courier depot directory structure exists*

Description

Creates `.courier-depot/` and its subdirectories in the project path. Writes a `.gitignore` to prevent tracking of logs and artifacts.

Usage

```
open_depot(project_path)
```

Arguments

project_path	Path to the R project

Value

Invisibly returns the path to the .courier-depot directory.

Examples

```
depot <- open_depot(tempdir())
```

---

open_hub *Launch the courieR delivery hub dashboard*

Description

Launch the courieR delivery hub dashboard

Usage

```
open_hub(project_path = NULL, port = NULL, launch.browser = TRUE)
```

Arguments

project_path	Optional path to pre-fill in the app
port	Optional port to run the app on
launch.browser	Logical. Whether to open the browser

Value

Called for its side effect of launching a Shiny application.

Examples

```
if (interactive()) {
  open_hub()
}
```

parse_dispatch_log *Parse test log*

Description

Parse test log

Usage

```
parse_dispatch_log(log_path)
```

Arguments

log_path	Path to the log file

Value

data.table

Examples

```
tmp <- tempfile(fileext = ".log")
writeLines(c(
  "-- Failure (test-foo.R:1): addition works ----",
  "Expected 3, got 4."
), tmp)
parse_dispatch_log(tmp)
file.remove(tmp)
```

parse_inspection_log *Parse R CMD check log*

Description

Parse R CMD check log

Usage

```
parse_inspection_log(log_path)
```

Arguments

log_path	Path to the log file

Value

data.table

Examples

```
tmp <- tempfile(fileext = ".log")
writeLines(c(
  "* checking examples ... WARNING",
  "  An example result is marked with \\donttest."
), tmp)
parse_inspection_log(tmp)
file.remove(tmp)
```

`rate_shipment` *Classify shipment risk based on check and test results*

Description

Classify shipment risk based on check and test results

Usage

```
rate_shipment(baseline_results, post_results)
```

Arguments

`baseline_results`	data.table from baseline check
`post_results`	data.table from post-shipment check

Value

A list

Examples

```
baseline <- data.table::data.table(
  severity = character(), message = character(),
  file = character(), line = character()
)
post <- data.table::data.table(
  severity = "ERROR", message = "undefined symbol",
  file = "R/foo.R", line = "10"
)
rate_shipment(baseline, post)
```

rig_available *Check if rig is available*

Description

Check if rig is available

Usage

```
rig_available()
```

Value

Logical

Examples

```
rig_available()
```

rig_install *Install R via rig*

Description

Install R via rig

Usage

```
rig_install(version, wait = TRUE)
```

Arguments

version	R version
wait	Logical

Value

The result of `processx::run()`.

Examples

```
if (interactive() && rig_available()) {
  rig_install("4.5.0", wait = FALSE)
}
```

rig_list *List rig installations*

Description

List rig installations

Usage

```
rig_list()
```

Value

data.frame

Examples

```
if (rig_available()) rig_list()
```

ship *Ship packages between R installations*

Description

Ship packages between R installations

Usage

```
ship(
  source_path,
  target_path,
  packages = NULL,
  dry_run = FALSE,
  upgrade = FALSE,
  ...
)
```

Arguments

source_path	Rscript path of the source installation
target_path	Rscript path of the target installation
packages	Optional character vector of packages to ship. If NULL, ships all non-base missing/outdated packages.
dry_run	Logical. If TRUE, return plan without installing.
upgrade	Logical. Passed to pak
...	Extra arguments

Value

A list with shipment results

Safety

`ship()` installs packages into the target R library via `pak::pkg_install()` running in a subprocess. Set `dry_run = TRUE` to preview the migration plan without installing anything. When `dry_run = FALSE` (the default), pak from the current R session is used to install into the target library, which is the correct design for a migration tool (the source R need not have pak installed). All subprocess calls are confined to the target library path; no files are written outside the target library or the R temporary directory.

Examples

```
routes <- find_routes()
if (nrow(routes) >= 2) {
  result <- ship(
    source_path = routes$rscript_path[1],
    target_path = routes$rscript_path[2],
    dry_run = TRUE
  )
  print(result$plan)
}
```

`take_inventory` *Scan project dependencies*

Description

Scan project dependencies

Usage

```
take_inventory(project_path)
```

Arguments

`project_path`	Path to the project

Value

A data.table

Examples

```
take_inventory(tempdir())
```

wrap *Generate a pak specification for a package*

Description

Generate a pak specification for a package

Usage

```
wrap(package, version = NULL, source_hint = NULL, github_ref = NULL)
```

Arguments

package	Package name
version	Optional version constraint or exact version
source_hint	Optional hint: "CRAN", "Bioconductor", "GitHub", "local"
github_ref	Optional GitHub ref like "owner/repo@ref"

Value

A character vector of pak specs

Examples

```
wrap("dplyr")
wrap("dplyr", version = "1.1.4")
wrap("mypackage", source_hint = "Bioconductor")
wrap("r-lib/rlang", source_hint = "GitHub", github_ref = "r-lib/rlang")
```

Index