Package: courieR (via r-universe)

June 9, 2026

Type Package

Title Migrate Installed R Packages Between R Versions

Version 0.3.0

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 source-to-target migration.

License MIT + file LICENSE

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

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

Language en-US

Encoding UTF-8

Roxygen list(markdown = TRUE)

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), shinyjs (>= 2.1.0), bslib (>= 0.7.0), bsicons (>= 0.1.2), DT (>= 0.31)

Suggests stringr (>= 1.5.0), testthat (>= 3.0.0), withr (>= 3.0.0), mockery (>= 0.4.4), knitr (>= 1.45), rmarkdown (>= 2.26), pkgdown (>= 2.0.0)

VignetteBuilder knitr

Config/pak/sysreqs cmake make libuv1-dev zlib1g-dev

Repository https://lennon-li.r-universe.dev

Date/Publication 2026-06-09 16:31:19 UTC

RemoteUrl https://github.com/lennon-li/courier

RemoteRef HEAD

RemoteSha 91d8e2a6882f3ea60e3707c4275751794ed1ad2e

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.

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

Takes two package manifests (from `manifest()`) and classifies every source package as missing, outdated, newer, or the same relative to the target.

Usage

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

Arguments

source_pkgs	`data.table` or `data.frame` from `manifest()`, representing the installation you are copying packages *from*.
target_pkgs	`data.table` or `data.frame` from `manifest()`, representing the installation you are copying packages *into*.

Value

A named list with the following elements:

`missing` Packages in source that are absent from target.

`outdated` Packages where the source version is newer than the target version.

`newer` Packages where the target already has a newer version than the source.

`same` Packages at identical versions in both installations.

`comparison` Full merged `data.table` of all source packages with a `status` column (`"missing"`, `"outdated"`, `"newer"`, or `"same"`).

`summary` One-row `data.frame` with counts: `missing`, `outdated`, `newer`, `same`, `total_source`.

Each data.table includes columns `package`, `version.x` (source version), `version.y` (target version), and `source`.

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*

Description

Runs a subprocess under the given R executable and returns all user-installed packages. Base and recommended packages are excluded automatically.

Usage

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

Arguments

rscript_path	Full path to an Rscript executable. Defaults to the current R session. Use find_routes() to get paths for other installations.
lib_path	Library path to query within the target R. Defaults to the first element of .libPaths() in that R installation.
format	"data.table" (default) or "data.frame".
timeout_sec	Maximum seconds to wait for the subprocess. Increase this on slow machines or network-mounted drives. Default 30.

Value

A data.table (or data.frame) with one row per user-installed package and columns: package, version, source ("CRAN", "GitHub", "Bioconductor", or "unknown"), remotetype, remoteusername, remoterepo, libpath. Base and recommended packages are never included in the output.

Examples

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

---

migrate *Migrate packages between two R installations in one call*

Description

Convenience wrapper around find_routes() and ship(). Matches installations by version string (e.g. "4.5.2") or full Rscript path, then runs the migration. Use ship() directly if you need fine-grained control.

Usage

```
migrate(from, to, dry_run = FALSE, upgrade = TRUE, mode = "online", ...)
```

Arguments

from	Version string or Rscript path of the source installation (packages are copied *from* here).
to	Version string or Rscript path of the target installation (packages are installed *into* here).
dry_run	If TRUE, return the plan without installing anything. Default FALSE.
upgrade	If TRUE, packages already in the target but at an older version are upgraded as well. Default TRUE.
mode	Transfer mode passed to ship(): "online" (default — reinstall via pak), "offline" (file-copy only, skip packages without a valid source path), or "preserve" (copy for exact version, fall back to a pinned pak spec on failure).
...	Additional arguments passed to ship().

Value

The same named list returned by `ship()`: `plan`, `results`, `comparison`, `dry_run`, `elapsed_sec`.

Examples

```
## Not run:
  # dry run first
  migrate("4.5.2", "4.6.0", dry_run = TRUE)

  # for real
  result <- migrate("4.5.2", "4.6.0")
  table(result$results$status)

## End(Not run)
```

`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 dashboard*

Description

Opens the Shiny dashboard in your browser. The dashboard detects all R installations on the machine and lets you compare and sync packages between any two of them without writing any code.

Usage

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

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

Arguments

`project_path`	Reserved; currently unused.
`port`	Port to run the Shiny app on. `NULL` picks a random available port.
`launch.browser`	Whether to open the system browser automatically. Default `TRUE`.

Details

`hub()` is a short alias for `open_hub()`.

Value

Called for its side effect of launching a Shiny application.

Examples

```
if (interactive()) {
  hub()        # short form
  open_hub()   # same thing
}
```

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

Compares the package libraries of two R installations and transfers missing or outdated packages into the target.

Usage

```
ship(
  source_path,
  target_path,
  packages = NULL,
  dry_run = FALSE,
  upgrade = FALSE,
  log_callback = NULL,
  mode = c("online", "offline", "preserve"),
  ...
)
```

Arguments

`source_path`	Full path to the `Rscript` executable of the source installation (the one you are copying packages *from*). Use `find_routes()` to discover available paths.
`target_path`	Full path to the `Rscript` executable of the target installation (the one you are installing packages *into*). The target R must have `pak` installed for `mode = "online"` or pak fallbacks.
`packages`	Character vector of package names to act on. If `NULL` (the default), all packages that are missing from or outdated in the target are included.
`dry_run`	If `TRUE`, build and return the installation plan without installing anything. Use this to review what will happen before committing to a sync.
`upgrade`	If `TRUE`, packages already present in the target but at an older version than the source are upgraded. If `FALSE` (the default), only packages missing from the target are installed.
`log_callback`	Optional function of one argument. When provided, it is called with a single character string for each progress message emitted during package transfer.
`mode`	Transfer mode: `"online"` reinstalls via pak (default), `"offline"` copies package directories by file and skips packages without a valid source path, and `"preserve"` copies first then falls back to a pinned pak spec for packages that could not be copied.
`...`	Reserved for future arguments.

Value

A named list with the following elements:

`plan` `data.table` of planned actions with columns `package`, `action` (`"install"` or `"upgrade"`), `mode`, `version.x` (source version), `version.y` (target version, `NA` if the package is missing), and `pak_spec` (the spec passed to pak).

`results` `data.table` of per-package outcomes with columns `package`, `status` (`"success"`, `"skipped"`, or `"error"`), and `message`.

`comparison` The raw `inventory()` comparison table.

`dry_run` `TRUE` if no packages were installed.

`elapsed_sec` Total wall-clock time in seconds.

Safety

`ship()` can install packages into the target R library via `pak::pkg_install()` running in a subprocess, or copy package directories directly for offline/preserve transfers. Set `dry_run = TRUE` to preview the migration plan without installing or copying anything. The source R need not have pak installed. Subprocess calls and file copies are confined to the target library path and 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