Package: harness (via r-universe)

June 9, 2026

Title Curated Agentic Harnesses for R Professional Roles

Version 0.2.0

Description A bootstrapper that launches a command-line coding agent of the user's choice in a terminal tab pre-configured for a professional R role. Each role is described by a curated harness: a subset of community skills, a system prompt, a folder layout, and quality gates. The package does not run an agent loop and does not call a language model; it discovers the chosen coder binary, generates its configuration, links the curated skills, and opens the terminal. Code written by the agent is run manually by the user, by design, so that every generated script passes through a human audit gate before execution.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

Depends R (>= 4.2)

Imports jsonlite, yaml

Suggests rstudioapi, testthat (>= 3.0.0), withr

Config/testthat/edition 3

URL https://github.com/pcbrom/harness

BugReports https://github.com/pcbrom/harness/issues

Repository https://pcbrom.r-universe.dev

Date/Publication 2026-06-09 18:19:44 UTC

RemoteUrl https://github.com/pcbrom/harness

RemoteRef HEAD

RemoteSha cb6866e109f62de8b3b9b3b43bd3ba8a1f6dc27f

Contents

`adapters` *List the registered adapters*

Description

List the registered adapters

Usage

```
adapters()
```

Value

A character vector of adapter names.

Examples

```
adapters()
```

available_roles *List the available curated roles*

Description

Returns the names of the harnesses bundled with the package, taken from the `inst/harness/<role>.yml` catalogue.

Usage

```
available_roles()
```

Value

A character vector of role names, sorted alphabetically.

Examples

```
available_roles()
```

clone_community_skills *Clone the community-skills catalogue*

Description

Clones the external community-skills repository into a discoverable location so that `community_skills_path()` finds it on the next call. The catalogue is an external dependency and is never bundled with this package; this helper only automates the checkout. It is never run when the package is loaded.

Usage

```
clone_community_skills(
  dest = file.path(path.expand("~"), ".community-skills"),
  url = community_skills_url(),
  shallow = TRUE,
  quiet = FALSE
)
```

Arguments

`dest`	Destination directory. Defaults to `~/.community-skills`, one of the paths searched by `community_skills_path()`.
`url`	The git remote to clone from.
`shallow`	When `TRUE` (the default), performs a shallow clone.
`quiet`	Suppress git and progress messages.

Value

The absolute path to the checkout, invisibly.

Examples

```
## Not run:
clone_community_skills()

## End(Not run)
```

`community_skills_path` *Locate the community-skills checkout*

Description

Searches, in order, the `COMMUNITY_SKILLS_PATH` environment variable, `~/.community-skills/` and `~/projects/community-skills/`. The community skills catalogue is an external dependency and is never bundled with this package.

Usage

```
community_skills_path()
```

Value

The absolute path to the checkout, or `NA_character_` when none is found.

Examples

```
community_skills_path()
```

`launch` *Launch a curated coding session*

Description

Loads the harness for `role`, configures the chosen `adapter` for the project, scaffolds the role's folder layout, and opens the coder in a terminal tab. Inside RStudio the terminal is created with `rstudioapi::terminalCreate`; otherwise an external terminal emulator is used, and when none is available the launch command is reported for the user to run.

Usage

```
launch(
  adapter = "claude",
  role,
  project_dir = getwd(),
  scaffold = TRUE,
  dry_run = FALSE,
  config_home = NULL,
  skills_path = NULL,
  binary = NULL
)
```

Arguments

adapter	The coder to launch. See adapters().
role	The professional role. See available_roles().
project_dir	The project root. Defaults to the working directory.
scaffold	When TRUE (the default), creates the role's folder layout.
dry_run	When TRUE, configures everything but does not open a terminal. Useful for inspection and testing.
config_home	Override the adapter configuration home (mainly for testing).
skills_path	Override the community-skills checkout path.
binary	Override the discovered coder binary path.

Details

The package never runs an agent loop and never executes code produced by the agent. Generated scripts are run manually by the user.

Value

An object of class harness_launch, invisibly.

Examples

```
## Not run:
launch("claude", role = "data-scientist")

## End(Not run)
```

role *Load a curated role*

Description

Reads the harness for `name` from the catalogue, validates it against the schema, and returns it as a `harness_role` object.

Usage

```
role(name)
```

Arguments

`name`	A role name, as returned by `available_roles()`.

Value

An object of class `harness_role`.

Examples

```
ds <- role("data-scientist")
ds$skills
```

role_config *Show the full configuration of a role*

Description

Prints the complete harness configuration for a role: description, execution policy, skills, folder layout, quality gates, package dependencies and the full system prompt, followed by the path to the source YAML. Unlike the compact `print` method, this includes the system prompt in full.

Usage

```
role_config(name)
```

Arguments

`name`	A role name, or a `harness_role` object.

Value

The `harness_role` object, invisibly.

Examples

```
role_config("data-scientist")
```

role_list *Tabulate the available roles*

Description

Returns a data frame with one row per curated role, summarising its version, the number of skills it declares, and the first line of its description.

Usage

```
role_list()
```

Value

A data frame with columns `role`, `version`, `skills` and `description`.

Examples

```
role_list()
```

role_skills *List the skills of one or more roles*

Description

Returns a long data frame with one row per role-skill pair. With `available = TRUE`, each skill is checked against the community-skills checkout and an `available` column reports whether its SKILL.md is present.

Usage

```
role_skills(role_name = NULL, available = FALSE)
```

Arguments

`role_name`	A role name, or `NULL` (the default) for every role.
`available`	When `TRUE`, add an `available` column reporting whether the skill is present in the community-skills checkout.

Value

A data frame with columns `role` and `skill`, plus `available` when requested.

Examples

```
role_skills("data-scientist")
utils::head(role_skills())
```

scaffold_layout *Scaffold the folder layout of a role*

Description

Creates the directories declared in the harness `layout` under `project_dir`. Existing directories are left untouched. The function never writes code, runs a script, or removes anything; it only ensures the audit folders exist.

Usage

```
scaffold_layout(role_name, project_dir = getwd(), create = FALSE)
```

Arguments

`role_name`	A role name, as returned by `available_roles()`.
`project_dir`	The project root under which to create the layout.
`create`	When `FALSE` (the default), the function reports the directories that would be created without touching the filesystem. Set to `TRUE` to create them.

Value

A data frame with one row per layout entry, invisibly.

Examples

```
tmp <- tempfile("proj")
dir.create(tmp)
scaffold_layout("data-scientist", tmp, create = TRUE)
```

send_selection_to_coder *Send the editor selection to the coder terminal*

Description

An RStudio addin. It reads the current editor selection, asks for a short note, and sends the note with a `file:line` reference to the coder running in the harness terminal that `launch()` opened. Bind it to a keyboard shortcut through Tools, Modify Keyboard Shortcuts, Addins.

Usage

```
send_selection_to_coder()
```

Details

The function forwards text the user wrote; it does not run an agent loop and does not call a language model. It requires RStudio and an open harness terminal.

Value

The message sent, invisibly, or `NULL` when cancelled.

`setup` *Validate the environment for a role*

Description

Checks that a community-skills checkout exists, reports which curated skills for the role are present in that checkout, and which declared CRAN packages are installed. Optionally scaffolds the role's folder layout. The function never installs packages and never runs generated code.

Usage

```
setup(role_name = NULL, project_dir = getwd(), scaffold = FALSE)
```

Arguments

`role_name`	A role name, or `NULL` to validate only the checkout.
`project_dir`	The project root for optional layout scaffolding.
`scaffold`	When `TRUE`, creates the role's folder layout under `project_dir`.

Value

An object of class `harness_setup`, invisibly.

Examples

```
## Not run:
setup("data-scientist")

## End(Not run)
```

status *Report the harness environment status*

Description

Summarises the discoverable environment: the community-skills checkout, the bundled roles, and the registered adapters with their binary availability. The function performs no side effects.

Usage

```
status()
```

Value

An object of class `harness_status`, invisibly printed by default.

Examples

```
status()
```

update_community_skills *Update the community-skills catalogue*

Description

Runs a fast-forward `git pull` on an existing community-skills checkout so that the curated skills track the upstream repository. The update is user-initiated. It can also run when the package is attached, but only when the user opts in through the `harness.auto_update` option or the `HARNESS_AUTO_UPDATE` environment variable; the default does nothing on load.

Usage

```
update_community_skills(dest = community_skills_path(), quiet = FALSE)
```

Arguments

`dest`	The checkout to update. Defaults to the discovered path from `community_skills_path()`.
`quiet`	Suppress git and progress messages.

Value

The absolute path to the checkout, invisibly.

Examples

```
## Not run:
update_community_skills()

## End(Not run)
```

Index