Package: readmit (via r-universe)

November 2, 2026

Title Readmission Risk Pool Platform

Version 0.0.1.9000

Description Provides the R implementation of the Readmission Risk Pool (RRP) analytical platform for reassessing the remaining probability of a first hospital readmission through 30 days after discharge. Distributed as the 'readmit' package, it creates user-owned projects, applies shared eligibility rules at explicit assessment times, retains completed assessments as history, and generates CSV analytical products for a project-owned interactive application. Project authors supply clinical source-mapping and risk-provider functions; no clinically validated prediction model is included.

License Apache License (>= 2)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Suggests bslib, plotly, reactable, shiny, testthat (>= 3.0.0), yaml

Config/testthat/edition 3

URL https://readmit.centralstatz.com/, https://github.com/centralstatz/readmit

BugReports https://github.com/centralstatz/readmit/issues

Depends R (>= 4.1.0)

Imports DBI, duckdb

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

Date/Publication 2026-11-02 18:39:27 UTC

RemoteUrl https://github.com/centralstatz/readmit

RemoteRef HEAD

RemoteSha 6c25cd33a563434d243014e6976c1403929a3fc3

Contents

rrp_inspect_history 2
rrp_project_init 3
rrp_project_status 4
rrp_project_validate 4
rrp_refresh_products 5
rrp_run 7
rrp_state_init 8

Index **9**

`rrp_inspect_history` *Inspect completed RRP assessments*

Description

Read retained assessment rows and their run context without sourcing or invoking project-authored analytical functions. Both optional filters use exact identifier matching and may be combined.

Usage

```
rrp_inspect_history(path, run_id = NULL, episode_id = NULL)
```

Arguments

`path`	Path to an existing RRP project with initialized state.
`run_id`	Optional single nonblank run ID to select.
`episode_id`	Optional single nonblank episode ID to select.

Details

Initialize state with `rrp_state_init()` before inspecting history. Rows are ordered by run creation time, run ID, and episode ID. Reads use a read-only connection and never create state or modify history. A completed run with zero episodes contributes no assessment row. `rrp_run()` adds completed assessments; `rrp_refresh_products()` derives CSVs from history.

Value

An ordinary data frame with assessment identity `run_id`, `episode_id`, `patient_id`, and `index_encounter_id`; UTC times `admission_time`, `discharge_time`, and `followup_window_end`; outcome `disposition` and `probability`; provider `failure_kind` and `failure_reason`; selected event `terminal_event_id`, `terminal_event_type`, `terminal_occurred_at`, and `terminal_available_at`; and run context `project_id`, UTC `as_of_time`, UTC `created_at`, `source_mapping_script`, and `risk_provider_script`. A probability is present for accepted outcomes; failure and event fields are missing when inapplicable. Empty or unmatched filtered results retain the same columns and types, including UTC `POSIXct` timestamps.

Examples

```
project <- tempfile("rrp-inspect-")
rrp_project_init(project, "example-rrp")
rrp_state_init(project)
rrp_inspect_history(project)
unlink(project, recursive = TRUE)
```

`rrp_project_init` *Create an RRP project*

Description

Copy the installed project template into a new, user-owned directory. The project contains editable configuration, branding, source-mapping and risk-provider scripts, and a Shiny application. Initialization does not create analytical state or products. The project owner must implement the two analytical scripts before running assessments.

Usage

```
rrp_project_init(path, project_id)
```

Arguments

`path`	Destination for the new project. Its parent directory must exist, and the destination itself must not exist, even as an empty directory.
`project_id`	A nonempty project identifier beginning with a letter or digit and containing only letters, digits, `.`, `_`, or `-`. It fills the project identity and initial branding fields; edit `_brand.yml` afterward to change the displayed name or styling.

Details

The project-owned application runs separately with ordinary Shiny tooling, for example `shiny::runApp("path/to/project` after its dependencies are installed as described in the project's `README.md`.

Value

The normalized project path, invisibly.

Examples

```
project <- file.path(tempdir(), "example-rrp-project")
if (!file.exists(project)) {
  rrp_project_init(project, "example-rrp-project")
}
```

rrp_project_status *Summarize an RRP project's retained history*

Description

Read project identity and completed-run status without sourcing the project's analytical scripts, executing an assessment, or changing state. This also works before analytical state has been initialized.

Usage

```
rrp_project_status(path)
```

Arguments

path	Path to an existing RRP project directory.

Details

The latest run is the most recently recorded completed run by `created_at`, not the largest analytical `as_of_time`. Ties in recorded creation time are broken by `run_id`. Reads use a read-only connection and do not initialize state.

Value

An ordinary named list containing character `project_id`, logical `state_initialized`, integer `run_count`, character `latest_run_id`, and UTC `POSIXct` `latest_as_of_time`. The latest fields are typed missing values when there are no runs.

Examples

```
project <- tempfile("rrp-status-")
rrp_project_init(project, "example-rrp")
rrp_project_status(project)
unlink(project, recursive = TRUE)
```

rrp_project_validate *Check an RRP project's authoring structure*

Description

Check that an existing project has readable `rrp-project.dcf` and `rrp-authoring.dcf` files and that their configured scripts define `rrp_produce_canonical(project_root, as_of_time)` and `rrp_calculate_risk(project_root, request)`. A newly initialized project's deliberate function stubs pass this check.

Usage

```
rrp_project_validate(path)
```

Arguments

path	Path to an existing RRP project directory.

Details

The project-authored scripts are sourced, so their top-level R code can run. Neither authoring function is called. This is a structural check; it does not test source data, model behavior, analytical readiness, or the project's Shiny application.

Value

TRUE, invisibly, when the essential configuration and authoring functions load successfully. Missing or unusable resources raise an error.

Examples

```
project <- tempfile("rrp-validation-")
rrp_project_init(project, "example-rrp")
rrp_project_validate(project)
unlink(project, recursive = TRUE)
```

---

rrp_refresh_products *Refresh an RRP project's analytical product CSVs*

Description

Derive the current clinically eligible risk pool, observed accepted-risk trajectories, and operational scope from retained assessment history. The selected completed run has the greatest analytical as_of_time; execution created_at and then run_id break ties. A later-executed historical backfill does not replace a newer analytical snapshot.

Usage

```
rrp_refresh_products(path)
```

Arguments

path	Path to an existing RRP project with initialized analytical state and at least one completed assessment run.

Details

Files are written under `state/products/` as `current-remaining-risk.csv`, `remaining-risk-trajectory.csv`, and `operational-scope-summary.csv`. The current-risk file includes clinically eligible provider-failure episodes with a missing probability, never zero or an earlier estimate. Refresh reads history only: it does not run the source mapper or provider, add a run, change retained history, or edit the project-owned app. All files are staged and checked before replacement. Replacement of the three files is not fully atomic if a filesystem error occurs during the final replacement sequence. Call `rrp_run()` to retain an assessment first; `rrp_inspect_history()` reads its stored rows.

Value

Invisibly, a list with `selected_run_id`, UTC `as_of_time`, and a named character vector `files` of generated CSV paths. Its keys are `current_risk` (`current-remaining-risk.csv`), `trajectory` (`remaining-risk-trajectory.csv`), and `scope` (`operational-scope-summary.csv`); the paths identify files, not their in-memory contents.

Examples

```
project <- tempfile("rrp-products-")
rrp_project_init(project, "example-rrp")
time <- as.POSIXct("2026-01-10 12:00:00", tz = "UTC")
episode <- data.frame(
  episode_id = "episode-1", patient_id = "patient-1",
  index_encounter_id = "encounter-1",
  admission_time = time - 2 * 86400, discharge_time = time - 86400
)
events <- data.frame(
  terminal_event_id = character(), episode_id = character(),
  event_type = character(), occurred_at = time[FALSE],
  available_at = time[FALSE]
)
saveRDS(list(discharge_episode = episode, terminal_event = events),
        file.path(project, "canonical.rds"))
# Replace both generated stubs; this constant risk is illustrative only.
writeLines(
  c("rrp_produce_canonical <- function(project_root, as_of_time) {",
    "  readRDS(file.path(project_root, 'canonical.rds'))", "}"),
  file.path(project, "R", "produce-canonical.R")
)
writeLines(
  "rrp_calculate_risk <- function(project_root, request) list(probability = 0.2)",
  file.path(project, "R", "calculate-risk.R")
)
rrp_state_init(project)
rrp_run(project, time)
products <- rrp_refresh_products(project)
names(products$files)
file.exists(products$files)
unlink(project, recursive = TRUE)
```

rrp_run *Run an RRP assessment and retain its history*

Description

Run the configured project source mapper at one analytical time, determine which discharge episodes remain eligible for first readmission in the remaining 30-day follow-up window, and call the project risk provider once per eligible episode. Accepted estimates, provider failures, and ineligible outcomes are retained together as one completed run.

Usage

```
rrp_run(path, as_of_time)
```

Arguments

path	Path to an existing RRP project with initialized analytical state.
as_of_time	One finite UTC POSIXct analytical instant. This same time is used for source production, eligibility, provider requests, and history.

Details

Call rrp_state_init() before the first run. The project owner must implement the configured rrp_produce_canonical() and rrp_calculate_risk() functions. A provider failure affects its episode, while a source or persistence failure stops the run without a partial completed record. Inspect retained rows with rrp_inspect_history() and generate analytical products separately with rrp_refresh_products().

Value

An ordinary list with run_id, as_of_time, episodes_processed, accepted, provider_failures, and ineligible. Counts are integers; the run ID belongs to a successfully committed run.

Examples

```
project <- tempfile("rrp-run-")
rrp_project_init(project, "example-rrp")
time <- as.POSIXct("2026-01-10 12:00:00", tz = "UTC")
episode <- data.frame(
  episode_id = "episode-1", patient_id = "patient-1",
  index_encounter_id = "encounter-1",
  admission_time = time - 2 * 86400, discharge_time = time - 86400
)
events <- data.frame(
  terminal_event_id = character(), episode_id = character(),
  event_type = character(), occurred_at = time[FALSE],
  available_at = time[FALSE]
)
```

```
saveRDS(list(discharge_episode = episode, terminal_event = events),
        file.path(project, "canonical.rds"))
# Replace both generated stubs; this constant risk is illustrative only.
writeLines(
  c("rrp_produce_canonical <- function(project_root, as_of_time) {",
    "  readRDS(file.path(project_root, 'canonical.rds'))", "}"),
  file.path(project, "R", "produce-canonical.R")
)
writeLines(
  "rrp_calculate_risk <- function(project_root, request) list(probability = 0.2)",
  file.path(project, "R", "calculate-risk.R")
)
rrp_state_init(project)
assessment <- rrp_run(project, time)
assessment$accepted
unlink(project, recursive = TRUE)
```

`rrp_state_init` *Initialize persistent history for an RRP project*

Description

Create `state/history.duckdb` and its empty `runs` and `assessments` tables for an existing project. This is separate from project creation and does not execute assessments or generate analytical products.

Usage

```
rrp_state_init(path)
```

Arguments

`path`	Path to an existing RRP project that passes structural validation.

Details

Initialization refuses to overwrite any existing `state/` directory. Project-authored scripts are loaded during structural validation, but their source-mapping and risk-provider functions are not called.

Value

The normalized project path, invisibly.

Examples

```
project <- tempfile("rrp-state-")
rrp_project_init(project, "example-rrp")
rrp_state_init(project)
unlink(project, recursive = TRUE)
```

Index

rrp_inspect_history, 2
rrp_inspect_history(), *6, 7*
rrp_project_init, 3
rrp_project_status, 4
rrp_project_validate, 4
rrp_refresh_products, 5
rrp_refresh_products(), *2*, *7*
rrp_run, 7
rrp_run(), *2*, *6*
rrp_state_init, 8
rrp_state_init(), *2*, *7*