Package: cttiR (via r-universe)

November 3, 2026

Title Reproducible Research Project Builder

Version 0.1.1.9000

Description Creates deterministic research projects from a name, research type and goal, routes them to a reviewed standard or specialist workflow, and generates a stage library whose package calls are covered by reviewed, revision-scoped approvals. Maintains an immutable knowledge catalog of package APIs and documentation with verified local, GitHub, CRAN and Bioconductor refresh and rollback, a curated resource catalog, grounded answers, project synchronization, audits with allowlisted repairs, an optional local model runtime and a local Shiny interface.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

SystemRequirements curl and tar with zstd support (optional, for managed local runtime acquisition on Linux x86_64)

Imports DBI, RSQLite, digest, fs, httr2, jsonlite, jsonvalidate, processx, ps, uuid, yaml

Suggests broom, broom.mixed, callr (>= 3.7.0), colorspace, DelayedArray, dplyr, ggplot2, git2r, knitr, lme4, Matrix, nlme, patchwork, pkgload, png, RColorBrewer, readr, reflowR (>= 0.1.0.9000), renv (>= 1.0.0), rmarkdown, S4Vectors, Seurat, SeuratObject, shiny (>= 1.7.0), shinytest2, SingleCellExperiment, SummarizedExperiment, survival, targets, testthat (>= 3.2.0), viridisLite, withr, workflowr

Config/testthat/edition 3

Config/Needs/check CTTIR/reflowR@910f4688c8fa32540ade7d000ac83ab2ca0e1d1d

Roxygen list(markdown = TRUE)

URL https://github.com/CTTIR/cttiR

BugReports https://github.com/CTTIR/cttiR/issues

Config/roxygen2/version 8.0.0

VignetteBuilder knitr

Config/Needs/website CTTIR/themakR, pkgdown

Config/pak/sysreqs cmake make libuv1-dev libssl-dev libnode-dev

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

Date/Publication 2026-11-03 08:40:48 UTC

RemoteUrl https://github.com/CTTIR/cttiR

RemoteRef HEAD

RemoteSha f9d818e3d28734a3a3cade500b106ae5ebdd9ce0

Contents

ask . . . 3
audit . . . 4
check_analysis_data . . . 5
configure . . . 6
doctor . . . 6
evidence_resolve . . . 7
evidence_search . . . 7
packages . . . 8
project . . . 9
resources . . . 10
rollback_knowledge . . . 12
runtime_cleanup . . . 12
runtime_stop . . . 13
runtime_storage . . . 13
search . . . 14
setup . . . 15
setup_app . . . 16
setup_release . . . 17
study_analyze . . . 18
study_dependency_lock . . . 19
study_paired_spec . . . 19
study_project . . . 20
study_render . . . 21
study_run . . . 22
study_spec . . . 22
study_stage . . . 23
study_validate . . . 24
study_validate_project . . . 24
sync . . . 25
update . . . 26
validate_config . . . 29
validate_spec . . . 30

Index **31**

ask *Ask for grounded workflow advice*

Description

Reads the question as untrusted text and matches it to registered workflow capabilities and exact `package::export` names in the pinned catalog revision. Supported capabilities return ordered steps, prerequisites, the approved package revisions and a reviewed, namespaced code snippet that is validated against the approved function table before it is returned. Nothing is executed, no model is called and no question text leaves the machine. Unsupported procedures return a precise gap and the nearest approved alternatives instead of invented code. Method families without a reviewed adapter (competing risks, ordinal or multinomial models, GEE, count models, quantile regression, propensity scores and weighting, Bayesian models, generalized additive models, machine learning and meta-analysis) are named in the gap and never answered with a nearby engine; such a question gets no code for any part of it. A question that names an absent or unapproved function gets no code. Instruction-shaped text, including lookalike letters and override markers, is treated as data.

Usage

```
ask(question, path = NULL, verified_only = TRUE)
```

Arguments

`question`	Nonempty question, optionally naming `package::export`.
`path`	Optional exact project root selecting its pinned catalog.
`verified_only`	If `TRUE`, every returned API claim and code call must be covered by a complete workflow approval of the pinned revision. If `FALSE`, statically verified but unapproved APIs and candidate capabilities may be listed, clearly labelled; nonexistent exports are never presented as real.

Value

A `cttir_answer` with `answer`, `steps`, `prerequisites`, `packages`, `data_requirements` (per-block inputs, outputs, column aliases, expected classes, placeholders and preceding producers), `code`, `citations`, `verification_levels`, `evidence`, `symbols`, `gaps`, `alternatives` (approved capabilities offered only as context when a named function does not exist or the method family is unsupported) and `limitations`. Citations name the pinned revision (versioned source archive or commit). Code uses placeholders (`<...>`) for mapped data and is illustrative; generated projects use the full reviewed stage library.

audit *Inspect installation, resources and project integrity*

Description

Runs registered checks with stable IDs (`INS-`, `KB-`, `RES-`, `PRJ-`, `STD-` and `INT-`). Default inspection is local and read-only: it reads package resources, catalog metadata, runtime state files and project metadata. Project control files are compared with what `project()` regenerates from the accepted specification and lock. All project R code (scripts outside `data/`, `renv/` and hidden directories, and R chunks, chunk options and inline R of R Markdown and Quarto documents) is parsed, never evaluated: unparseable files, forbidden or unsafe calls and clearly unresolved calls fail `STD-003`; constructs that the package's reviewed generated code uses in the same file are not reported again. Static checks cannot follow names built at run time and are no sandbox. It never opens referenced study datasets or dereferences paths bound in `.cttir/local.yml`, runs study pipelines, installs dependencies, starts services, contacts a model daemon or queries remote servers. Declared Imports are checked with `requireNamespace()`, which may load trusted installed namespaces. When the project root contains `.git` and Git is available, tracked file names under ignored private paths are listed with `git ls-files` (names only; no contents are read).

Usage

```
audit(
  path = NULL,
  scope = c("installation", "knowledge", "project"),
  repair = FALSE,
  output = NULL,
  strict = FALSE,
  live = FALSE
)
```

Arguments

`path`	Optional exact existing project root.
`scope`	Nonempty subset of `installation`, `knowledge`, `project`, `integration`. Integration opts into temporary synthetic files.
`repair`	Apply the documented reversible repair allowlist.
`output`	Optional directory for JSON and Markdown reports.
`strict`	Throw `cttir_audit_failed` after reporting required failures or required unverified checks. The condition contains the full report.
`live`	Allow a bounded CPU structured-output probe of an already owned runtime when the integration scope is selected, and a loopback metadata query for model locality. Never installs, pulls or starts a runtime.

Details

Overall status precedence: a required `fail`, then a required `not_tested`, then any warning or optional finding, then `pass`; when every check is `not_applicable` the overall status is `not_tested`. Which checks are required derives from the project's declared readiness and workflow.

With `repair = TRUE`, only an allowlist of reversible repairs is attempted, each only when its triggering check failed: rolling back an interrupted project write whose recorded writer has stopped, restoring a missing managed file whose generated content matches its accepted baseline, and repointing a corrupt active catalog pointer to the most recently written retained manifest that passes verification (under the catalog writer lock, journaled, never deleting snapshots). Edited files are never overwritten. Each repair records before/after hashes, its outcome and a recheck; a failed repair restores the prior bytes. All checks except integration are rerun after an applied repair.

The integration scope explicitly creates synthetic temporary projects and an isolated temporary catalog store and removes them afterwards; it never uses the configured catalog store for its fixture.

Reports are written only to an explicit `output` directory, always under new unique names. Report text is portable: paths under the project root become relative, the home directory is shown as ~, and Markdown control characters in messages are escaped.

Value

A `cttir_audit` with a check table (`id`, `scope`, `status`, `required`, `message`, `description`, `severity`, `read_effects`, `repair_id`, `evidence` as JSON text, `duration_seconds`), overall status and reason, repairs, `repair_plan` (the allowlisted repairs that `repair = TRUE` would attempt for the final checks, with their target files and current hashes; computed read-only), package identity, effects and limitations.

`check_analysis_data` *Check explicitly supplied analysis data*

Description

Checks mapped columns in a plain data frame without opening data files, changing data, evaluating column names, fitting models or approving a workflow. Only the currently planned independent linear/binomial, right-censored survival, and continuous grouped models have data checks. Complete-case exclusions are reported only when explicitly configured; no imputation or deletion occurs.

Usage

```
check_analysis_data(data, spec)
```

Arguments

`data`	A plain base R data frame, at most one million rows and 200 columns.
`spec`	A project specification list or local specification file.

Value

A list with state, coded issues, aggregate missingness and row counts. A passed check is not model, diagnostic or scientific approval. Individual records, identifier values and excluded row numbers are never returned.

`configure` *Configure an existing project with an explicit preview*

Description

Opens the local application on the Configure view of an existing project. Changes are collected from the optional questionnaire and advanced JSON, previewed with `sync()` (field-level requested changes and the file plan) and applied only on explicit request. Edited managed files are reported as conflicts and nothing is written while conflicts remain.

Usage

```
configure(path = ".", launch.browser = interactive())
```

Arguments

`path`	Exact existing project root.
`launch.browser`	Open the browser when the application is run.

Value

A local Shiny application using `sync()` for preview and application.

`doctor` *Brief read-only diagnostics*

Description

Runs the default audit scopes through the same check engine as `audit()`, never with repairs, live probes or reports.

Usage

```
doctor(path = NULL)
```

Arguments

`path`	Optional exact project root.

Value

A `cttir_audit` from the shared inspection engine.

evidence_resolve *Resolve One Exact Local Evidence Citation*

Description

Resolve One Exact Local Evidence Citation

Usage

```
evidence_resolve(directory, manifest_sha256, evidence_id)
```

Arguments

directory	Local immutable evidence-registry directory.
manifest_sha256	Trusted SHA-256 of its manifest, retained externally.
evidence_id	Exact evidence-v1 identifier returned by evidence_search().

Value

One verified record, or an error for missing, ambiguous, restricted or changed evidence. This does not certify that a claim follows from its text.

evidence_search *Search an Externally Pinned Local Evidence Registry*

Description

Searches captured books, package documentation, method guidance and legal references. Selected source bytes and extraction identities are checked at retrieval. A match does not establish that a claim is supported, that a legal source is currently applicable, or that code is approved for execution.

Usage

```
evidence_search(
  directory,
  manifest_sha256,
  query,
  limit = 8L,
  family = NULL,
  package = NULL,
  version = NULL
)
```

Arguments

directory	Local immutable evidence-registry directory.
manifest_sha256	Trusted SHA-256 of its manifest, retained externally.
query	Plain-text search terms, at most 8000 bytes.
limit	Maximum number of passages, from 1 to 50.
family	Optional books, packages, uzh, legal or cited_references filter.
package	Optional exact package name.
version	Optional exact source version; requires package.

Value

Search results with evidence IDs, locators, versions, hashes and explicit coverage limitations. Restricted documents are excluded.

---

packages *Inspect cataloged package revisions*

Description

Inspect cataloged package revisions

Usage

```
packages(path = NULL)
```

Arguments

path	Optional exact project root selecting its pinned API catalog.

Details

Workflow approvals are reviewer decisions keyed to an exact package `source_hash`. `update()` attaches the bundled reviewed decisions and, when set, those in the JSON file named by `options(cttiR.approvals = path)` to the matching extracted revision inside the immutable snapshot, so project pins and `rollback_knowledge()` carry them. A decision counts only when that revision stores every required topic and document under a rights basis, every required callable is a statically verified documented export and a recorded fixture passed. Any source change, including documentation only, requires a new decision. Decisions never install or run anything.

Value

A data frame of package identity, revision, extraction coverage and verification limits. S3 counts are missing for historical snapshots without a method index. `approved` counts callables covered by a complete reviewed workflow approval of that exact revision and `approved_roles` lists the approved roles (comma separated, empty when none). No namespaces are loaded to inspect installed versions.

`project` *Create an offline research project scaffold*

Description

The foundation build uses deterministic templates, without package installation or execution of study code. It makes no network calls and no model inference unless the local planner policy enables one loopback exchange with an owned local model (see Details). The planned standard workflow uses the reviewed reflowR layout. Advanced workflow options that are not implemented raise a typed error instead of being ignored.

Usage

```
project(
  name,
  type,
  goal,
  path = getwd(),
  config = NULL,
  options = list(),
  dry_run = FALSE
)
```

Arguments

`name`	Nonempty project title, kept as written. A portable child-directory slug is derived once: ASCII transliteration where possible; when letters have no transliteration (for example CJK script) or the slug would exceed 80 characters, a short hash of the name is appended (`project_<hash>` when nothing transliterates).
`type`	One of `primary_research`, `secondary_research`, `methods`, `review`, `software`, `mixed`, or `other`.
`goal`	Nonempty research objective.
`path`	Existing parent directory; never the project directory itself.
`config`	Optional named configuration list or local JSON/YAML file.
`options`	Named configuration list taking precedence over `config`.
`dry_run`	If TRUE, return a plan without writing any files.

Details

Explicit `workflow` options add reviewed integrations. `pipeline = "targets"` writes `_targets.R` and adds targets to the dependency manifest. `environment = "renv"` with `prepare_environment = TRUE` prepares a project library in an isolated R process after the scaffold has been published, honoring `network` (`"offline"` by default). `git = TRUE` runs `git init` in the new project root only, never inside another work tree, and never stages or commits. Failures of these steps keep the scaffold and are reported as readiness blockers with a recovery command. `reporting = "workflowr"` with

`git = TRUE` creates one root workflowr site. `reporting_preset` selects `"minimal"` or `"standard"` (default). Rendering is explicit and checks current computation receipts. Use `sync()` to preview migration from the current generic bundle. With workflowr, targets scheduling checks content identities and reuses fresh study results before explicit rendering.

Creation stages the scaffold next to the target and publishes it with one rename while holding a creation lock that records its process. A lock left by a stopped process on this host (or an ownerless lock from an older version, after 24 hours) is recovered on the next attempt: only the lock and that attempt's staging directory are removed, and `recovered` reports it.

Optional `analysis$mapping` records `data_source_id`, `outcome`, `predictors`, `subject`, `time`, `event`, `event_value`, `non_event_value`, `estimand`, `time_origin`, `time_unit` and `missing_data`. Dataset IDs must refer to `data_sources`. Column names are literal data, not R expressions. An explicitly empty predictor list denotes an intercept-only intention. Missing-data choices are `unknown`, `fail` and `complete_case`; these record intent and never remove observations during planning. Predictive and causal designs are marked as requiring additional supported adapters. Scientific and revision-specific workflow approval remain separate prerequisites.

Unset aim, outcome family, unit structure and modality are filled from goal keywords by default. Only when `options(cttiR.planner = "local_llm")` is set and the owned runtime prepared by `setup()` and its recorded model digest verify, one bounded read-only exchange with that local model (at most one repair) proposes the same unset fields, also in dry runs. It never starts, pulls or installs anything; explicit values always win; failures fall back to the keyword rules, and replies showing signs of embedded instructions leave the fields unknown. The reason is recorded in `spec$decisions`.

Value

A `cttir_project` containing path, spec, plan, readiness, manifest and warnings. An identical repeat is read-only and preserves user edits. `readiness$level` becomes `environment_ready` only when `renv.lock` and the project library match the pinned dependencies; `readiness$environment` and `readiness$git` report the evidence and any recovery command. The environment state is derived on each machine from `renv.lock`, the project library and `.cttir/environment.json`; `cttir-lock.json` pins versions but never records an installed environment. `readiness$analysis` reports candidate routing, missing fields and capability gaps. It never opens data or executes a model, even with `analysis$approved`.

Examples

```
p <- project("Example", "methods", "Plan a reproducible comparison",
  path = tempdir(), dry_run = TRUE
)
print(p)
```

`resources` *Browse the bundled research package candidates*

Description

Metadata reflects the supplied 2026-09-26 research snapshot. Candidate status does not establish API verification or tested workflow support. This query is read-only and offline; it does not load or install cataloged packages.

Usage

```
resources(
  query = NULL,
  domain = NULL,
  repository = NULL,
  path = NULL,
  limit = 50L
)
```

Arguments

`query`	Optional nonempty text; literal case-insensitive substring search.
`domain`	Optional exact category filter, such as `cytometry`.
`repository`	Optional exact repository filter, such as `Bioconductor`.
`path`	Optional exact project root whose resource snapshot must match.
`limit`	Positive integer row limit, at most 10000.

Value

A data frame with source, observed version, verification and maturity fields, plus each observation's recorded `freshness` and `fetch_status` in the selected snapshot (for example `source_unavailable` and `unavailable_observation` after an optional index outage). Filters combine with AND. Exact package names rank first. Nullable author and maintainer fields preserve literal DESCRIPTION text with its matching hash. Missing metadata, including historical snapshots, stays unknown. These fields do not verify ownership; an index-only refresh does not establish new author or maintainer evidence. The `authors_r_roles` list column parses literal `person()` declarations without evaluating R code; unsupported expressions remain unresolved. `supplemental_maintainer_evidence_json` contains separately observed, version-matched DESCRIPTION metadata with its own source URL and hash. Its `relation_to_seed` distinguishes matching hashes from independent observations; it never replaces the original observation's source hash.

Examples

```
resources("cytometry", limit = 3L)
```

rollback_knowledge *Restore a retained composite catalog without changing project pins*

Description

Restores the API snapshot, resource snapshot and recorded Bioconductor release policy of a retained composite snapshot together. The target must verify completely. A damaged active snapshot does not block restoring a verified one: the damage is reported in `active_problem` and `warnings`, the damaged files are kept for inspection, and the impact against them is not computed.

Usage

```
rollback_knowledge(version, dry_run = TRUE)
```

Arguments

`version`	Composite snapshot ID returned by `update()`. Unknown IDs fail with a message listing the retained IDs.
`dry_run`	Preview only; the default makes no changes.

Value

A `cttir_update` containing IDs, API impact, activation outcome, `active_problem` (`NA` when the active snapshot verified) and the restored `bioc_release` policy (`NA` release when none is recorded).

runtime_cleanup *Preview or remove reviewed unreferenced model downloads*

Description

Only files directly under the owned `release-downloads` directory can be removed. Retained release pins, model blobs, runtimes and other files are protected. Preview first and pass its named `expected` hashes to a separate consenting call. A setup lock prevents concurrent package acquisition; all selected hashes are rechecked before deletion. Failed or partial cleanup leaves a durable receipt. No process is stopped and no model tag is removed.

Usage

```
runtime_cleanup(files, expected = NULL, dry_run = TRUE, consent = FALSE)
```

Arguments

`files`	Exact download basenames to inspect or remove.
`expected`	Named SHA-256 vector returned by the preview; required for removal and checked again under the setup lock.
`dry_run`	Preview without writes (default `TRUE`).
`consent`	Explicit consent to irreversible removal (default `FALSE`).

Value

A list with state, selected paths/bytes, expected hashes and, after a removal attempt, a receipt path. Omitted files are never selected implicitly.

`runtime_stop` *Preview or stop the verified owned local daemon*

Description

A separate consenting call must supply the identity from the preview. Ownership, binary, model directory and process creation time are rechecked under the setup lock. Sends a graceful termination signal only to that process. No lookup by port/name, forced kill, model deletion or service-manager change is performed. All files and release pins remain available for a subsequent explicit setup. Windows supports the read-only preview only. Applying a stop on Windows is refused before creating locks or receipts or sending signals, because this graceful termination operation is unavailable there.

Usage

```
runtime_stop(dry_run = TRUE, consent = FALSE, expected = NULL)
```

Arguments

`dry_run`	Preview without signals or writes (default `TRUE`).
`consent`	Explicit consent to stop the previewed daemon (default `FALSE`).
`expected`	Exact identity checksum returned by the preview.

Value

A list with state, process identity and, after a stop attempt, a durable receipt. `stopping` means the process remained alive after five seconds; it is not a claim that all resources or child processes were released.

`runtime_storage` *Inspect managed model storage without contacting a daemon*

Description

Lists release downloads, runtime archives, model blobs and retained release manifests. Runtime libraries, logs and model-tag metadata are outside this size inventory. All retained release manifests protect their model downloads, including older versions kept for rollback. Model blobs are always retained. This call performs no HTTP requests, starts no process and does not create storage.

Usage

```
runtime_storage()
```

Value

A list with storage root, file inventory, inventoried bytes, current setup lock state and locally verified owned process identity when available.

`search` *Search revision-scoped APIs and stored documentation*

Description

Search revision-scoped APIs and stored documentation

Usage

```
search(query, packages = NULL, path = NULL, limit = 20L)
```

Arguments

`query`	Nonempty literal search text, optionally `package::export`.
`packages`	Optional character vector of exact package names.
`path`	Optional exact project root selecting its pinned API catalog.
`limit`	Positive integer result limit, at most 10000.

Value

A data frame containing stable IDs, revision, snippet, score and source evidence. S3 declarations are labelled separately, including private implementations; static evidence does not establish installed dispatch or tested workflow approval. Reexports (`reexport_declared`) and S4/S7 declarations (`static_declaration_only`) are never callable evidence. Only exports covered by a complete reviewed approval of the exact revision are labelled `workflow_approved` with `approved = TRUE`.

`setup` *Explicitly prepare an owned local runtime*

Description

Acquisition uses a pinned publisher archive and checksum, currently for Linux x86_64 only. No system services, existing daemons or package libraries are modified. The runtime uses a retained user data directory and disables cloud execution. Unsupported platforms return blockers; they are not claimed tested. Calling this function explicitly permits its reported downloads and startup. Dry runs never create directories or contact a daemon. Offline mode reuses verified local artifacts and never pulls a model.

Usage

```
setup(model = "auto", install_ollama = TRUE, offline = FALSE, dry_run = FALSE)
```

Arguments

`model`	Plain local model tag (`name` or `name:tag`, no registry host, namespace or URL), or `auto` for the smallest qualified model whose tested settings and recorded CPU, available RAM and disk budgets fit. Unknown or insufficient resources block automatic acquisition. Budgets are conservative admission checks, not reserved resources or process memory limits.
`install_ollama`	Allow portable acquisition when the runtime is absent.
`offline`	Forbid acquisition and model downloads.
`dry_run`	Return the plan without persistent changes or HTTP requests.

Details

Machine preferences may set `options(cttiR.runtime_dir = path)` and `options(cttiR.ollama_endpoint = "http://127.0.0.1:11434")`. Other endpoints are rejected. The local HTTP client disables proxies and redirects.

Value

A `cttir_setup` with steps, runtime/model identity, actions and blockers. The offline `preflight` reports acquisition and planning blockers separately, published full download sizes (unknown for unrecorded models), available disk space, recorded admission budgets and verified local process ownership when available. File presence alone does not establish integrity or readiness. Explicit unqualified model preparation remains possible; it does not qualify that model for planning. `download_estimate` reports estimated acquisition bytes, with zero for reusable components and unknown model bytes until the owned daemon is queried. Dry runs never query it. Published sizes are estimates rather than measured transfer: cached archives and shared model blobs can reduce bytes, and mutable model tags can change size. Existing files still need verification.

`setup_app` *Open the local project builder*

Description

Returns the local cttiR application. Printing it in an interactive R session launches it on the loopback interface. Views: Home, Create (Fast and Detailed share one draft), Open project/Configure, Ask, Knowledge (package coverage, catalog update preview/apply and rollback preview), Resources, Audit (report export and the documented repair allowlist) and Runtime setup. Every operation calls the same exported functions as the R API (`project()`, `sync()`, `ask()`, `search()`, `packages()`, `resources()`, `update()`, `rollback_knowledge()`, `audit()`, `setup()`) in a bounded background R process; read-only work can be cancelled and duplicate submissions are refused. Creation and configuration require a reviewed preview of the current draft and catalog. Scientific analysis and returned code are never executed by the general project builder.

The separate Strict synthetic examples view offers two fixed, noncausal examples: an independent continuous-outcome association and a paired mean change. Preview, compilation, synthetic analysis and rendering are separate explicit actions through `study_project()`, `study_run()` and `study_render()`. It accepts no data upload, arbitrary code or model plan, and installs no packages. The prepared external library must already exist. One compiled example is retained per application session. Each successful step retains its trusted external declaration in server-side session state; download it outside the project for continuation through the R API after session loss. The interface never recovers authority from project files and never marks scientific or real-study approval. General Open/Configure remains for projects created by `project()`, not strict example projects.

Usage

```
setup_app(mode = c("fast", "detailed"), launch.browser = interactive())
```

Arguments

`mode`	Initial `fast` or `detailed` view.
`launch.browser`	Open the browser when the application is run.

Details

Fast asks only for name, research type and goal (plus the parent directory) and shows the resolved workflow profile, routing reason and selected tools. Detailed adds optional questions from a declarative registry; every question can stay unanswered, changing an answer marks dependent answers for review, and answers that no longer apply are not used. Answers are passed as `options`, the advanced JSON as `config`.

Drafts can be exported to JSON and restored explicitly. They retain answers but exclude the parent directory, local data bindings and accepted plans. Restored drafts require a new preview; Configure drafts are bound to the existing project identity. Exported text may contain user-supplied sensitive information. Drafts are not automatically saved. The interface is available in English and German.

Value

A `shiny.appobj`. Shiny and callr must be installed. The background workers use the installed cttiR, which must match the running version. Run locally only; authentication and remote multi-user deployment are not supported.

`setup_release` *Preview or install an exact local model release*

Description

Validates a locally supplied release manifest against a separately trusted SHA-256 value. The default preview performs no HTTP requests or writes. Installation requires both `dry_run = FALSE` and `consent = TRUE`. The model can be imported offline from a verified GGUF, or downloaded from the immutable Hugging Face revision in the manifest. Downloads and failed imports are retained for diagnosis and retry. Existing model tags are never overwritten.

Usage

```
setup_release(
  manifest,
  manifest_sha256,
  model_file = NULL,
  dry_run = TRUE,
  consent = FALSE,
  offline = FALSE,
  install_ollama = TRUE
)
```

Arguments

`manifest`	Path to a local release JSON manifest.
`manifest_sha256`	Trusted SHA-256 of that exact manifest, obtained through a separately reviewed release record rather than trusting a download alone.
`model_file`	Optional local GGUF matching the manifest for offline import.
`dry_run`	Return the setup preview without side effects (default `TRUE`).
`consent`	Explicit noninteractive consent to the previewed acquisition, owned daemon startup and import. Required for a non-preview call.
`offline`	Forbid model and runtime downloads.
`install_ollama`	Permit acquisition of the pinned runtime when absent.

Details

Acquisition is supported on Linux x86_64. Other platforms return explicit blockers. The function starts or reuses only the package's verified, owned, cloud-disabled runtime, never changes login services and never modifies an unmanaged daemon. Successful import establishes artifact identity and a bounded structured-output probe; it does not grant planner or assistance qualification. Previous release artifacts remain available for rollback by selecting their original manifests. Research-project dependency preparation is separate.

Value

A `cttir_setup` containing storage/resource requirements, runtime/model identity, steps and blockers. No qualification follows from manifest labels.

`study_analyze` *Execute a Synthetic Study Recipe*

Description

Execute a Synthetic Study Recipe

Usage

```
study_analyze(spec, data, units)
```

Arguments

`spec`	A closed specification from `study_spec()` or `study_paired_spec()`.
`data`	Optional data frame. Association observations need unique, nonmissing IDs. Paired observations need exactly one measurement per pair ID at each declared occasion. Continuous columns must be plain numeric vectors, with one value per row. Association IDs must be plain character or numeric vectors and remain unique when converted to text. Supply character IDs to retain leading zeros or distinguish closely spaced numeric identifiers.
`units`	Named character vector or list declaring the actual units of every continuous analysis column; these must equal the planned units.

Value

A list with coefficient table, sample size, residual degrees of freedom, diagnostics and explicit limitations. Association results also include rank and residual standard deviation; paired results include pair count and within-pair differences. Passing numerical computation does not grant scientific approval.

study_dependency_lock *Snapshot the Prepared Study Environment*

Description

Snapshot the Prepared Study Environment

Usage

```
study_dependency_lock()
```

Value

A dependency lock containing the installed R and transitive package versions. This only reads the external library; it installs nothing and does not assert a download origin or provide a restoration manifest.

study_paired_spec *Declare a Synthetic Paired Mean-Change Study*

Description

Declares two measurements per independent pair, matched by an explicit ID. The contrast is after minus before, with a two-sided test against zero. Only synthetic, noncausal clinical examples are supported. Declaring pair independence does not establish it scientifically.

Usage

```
study_paired_spec(
  study_id,
  name,
  question,
  population,
  pair_id,
  occasion,
  outcome,
  before,
  after,
  independent_pairs,
  research = list(domain = "clinical", data_origin = "secondary", design =
    "observational"),
  confidence = 0.95,
  evidence = list()
)
```

Arguments

`study_id`	Stable identifier using letters, digits and underscores.
`name`	Plain-text study name.
`question`	Prespecified research question.
`population`	Explicit synthetic population description.
`pair_id`	Column containing complete character pair identifiers.
`occasion`	Character column identifying the measurement occasion.
`outcome`	A continuous variable declaration with `name`, `type = "continuous"` and `unit`.
`before, after`	Distinct character labels in the occasion column.
`independent_pairs`	Required explicit declaration, exactly `TRUE`. Clustered or otherwise dependent pairs require a different recipe.
`research`	List with domain, data_origin and design.
`confidence`	Confidence interval coverage, between 0.8 and 0.99.
`evidence`	List of evidence declarations, retained without asserting support.

Value

A closed synthetic paired specification. No data are read or analyzed.

`study_project` *Compile a Synthetic Study into a Strict Workflow Project*

Description

Compile a Synthetic Study into a Strict Workflow Project

Usage

```
study_project(directory, spec, data, dependency_lock, author, units, seed = 1L)
```

Arguments

`directory`	New project directory whose parent already exists. Accepted platform aliases are resolved to their physical paths before compilation; user-created symbolic links remain unsupported.
`spec`	A closed specification from `study_spec()` or `study_paired_spec()`.
`data`	Optional data frame. Association observations need unique, nonmissing IDs. Paired observations need exactly one measurement per pair ID at each declared occasion. Continuous columns must be plain numeric vectors, with one value per row. Association IDs must be plain character or numeric vectors and remain unique when converted to text. Supply character IDs to retain leading zeros or distinguish closely spaced numeric identifiers.

dependency_lock
: An externally prepared dependency lock, for example from study_dependency_lock(). No packages are restored automatically.

author
: Plain-text author name.

units
: Named character vector or list declaring the actual units of every continuous analysis column; these must equal the planned units.

seed
: Fixed nonnegative integer seed for the workflow.

Value

A list with directory and an external trusted declaration. Retain the declaration outside the project and pass successive updated declarations to study_run(), study_render() and study_validate_project(). Creation initializes a study-owned Git repository and stages only generated analysis Rmd files. It does not commit, publish or approve a real study. Declared measurements are written as 17-digit decimal values and checked for an exact round trip before any project is created.

---

study_render *Render All Stages from Pinned Study Results*

Description

Render All Stages from Pinned Study Results

Usage

```
study_render(directory, declaration, timeout = 600)
```

Arguments

directory
: Existing study project.

declaration
: Trusted external declaration returned by the compiler or the latest successful run/render. Never obtain this from project files.

timeout
: Maximum child R rendering time in seconds, between 30 and 3600.

Value

A list with directory, validation and the updated external declaration pinning stage receipts and all generated report files. No Git commit or publication occurs. Rendering uses a clean child R session in the prepared external library and never reruns statistical estimation.

study_run *Run the Fixed Synthetic Study Recipe*

Description

Run the Fixed Synthetic Study Recipe

Usage

```
study_run(directory, declaration)
```

Arguments

directory	Existing study project.
declaration	Trusted external declaration returned by the compiler or the latest successful run/render. Never obtain this from project files.

Value

A list with directory, numerical result and an updated external declaration pinning the exact run receipt. Preserve the updated declaration.

study_spec *Declare a Synthetic Independent Association Study*

Description

Creates a closed, noncausal specification for the initial reviewed-code development recipe. This version supports synthetic inputs only; it does not approve a real study or infer scientific decisions from data.

Usage

```
study_spec(
  study_id,
  name,
  question,
  population,
  observation_id,
  outcome,
  exposure,
  covariates = list(),
  research = list(domain = "clinical", data_origin = "secondary", design =
    "observational"),
  confidence = 0.95,
  evidence = list()
)
```

Arguments

`study_id`	Stable identifier using letters, digits and underscores.
`name`	Plain-text study name.
`question`	Prespecified research question.
`population`	Explicit synthetic population description.
`observation_id`	Name of the unique observational-unit identifier.
`outcome`, `exposure`	Lists with name, type (continuous) and unit.
`covariates`	List of continuous or categorical declarations. Categorical declarations contain name, type, levels and reference.
`research`	List with domain, data_origin and design.
`confidence`	Confidence interval coverage, between 0.8 and 0.99.
`evidence`	List of evidence declarations, retained without asserting support.

Value

A validated specification. No data are opened or analyzed.

`study_stage` *Present a Verified Recorded Study Stage*

Description

Present a Verified Recorded Study Stage

Usage

```
study_stage(stage, directory = ".")
```

Arguments

`stage`	Fixed stage identifier from the compiled profile.
`directory`	Study directory. Used by canonical stage files only.

Value

Invisibly the stage view receipt. Requires the external trusted declaration installed by `study_render()` in a clean child R session.

study_validate *Validate a Study Contract and Optional Data*

Description

Validate a Study Contract and Optional Data

Usage

```
study_validate(spec, data = NULL, units = NULL)
```

Arguments

spec	A closed specification from `study_spec()` or `study_paired_spec()`.
data	Optional data frame. Association observations need unique, nonmissing IDs. Paired observations need exactly one measurement per pair ID at each declared occasion. Continuous columns must be plain numeric vectors, with one value per row. Association IDs must be plain character or numeric vectors and remain unique when converted to text. Supply character IDs to retain leading zeros or distinguish closely spaced numeric identifiers.
units	Named character vector or list declaring the actual units of every continuous analysis column; these must equal the planned units.

Value

Invisibly a list with valid, recipe_id, specification_sha256 and, when data are supplied, n. Invalid or unsupported input raises an error.

study_validate_project *Validate a Compiled Study and Its Externally Pinned Results*

Description

Validate a Compiled Study and Its Externally Pinned Results

Usage

```
study_validate_project(
  directory,
  declaration,
  require_run = FALSE,
  require_render = FALSE
)
```

Arguments

directory	Existing study project.
declaration	Trusted external declaration returned by the compiler or the latest successful run/render. Never obtain this from project files.
require_run	Require current externally pinned numerical results.
require_render	Require current externally pinned HTML and all applicable stages.

Value

A list with valid, reason, structural validation and scientific_approval set to FALSE. Stale or unsupported material yields valid = FALSE.

---

sync *Preview or apply project configuration changes*

Description

Preserves identity, catalog pins and edits unless an explicit reporting migration requests the compatible local catalog refresh described below. A file changed both by its user and by the new rendering is never overwritten: a managed file yields a conflict, and no writes occur while any conflict is present; a user-owned file (protocol, metadata, publication and analysis text) is kept and the rendered update is written to `.cttir/proposed/<path>` for review. Edited files whose rendering did not change are simply kept. Unedited user-owned files, such as `metadata/data-registry.yml` or `publication.yml`, are updated like managed files. Applied changes have per-file backups and a journal. The operation is not a multi-file atomic filesystem transaction. Interrupted writers require review; a writer lock left by a stopped process that had not started its journal is released. Migration from `reporting = "generic"` to `"workflowr"` is supported for the current template with `git = TRUE`. Preview first, then apply with `dry_run = FALSE`. Edited site configuration conflicts; edited manuscripts remain user-owned. For older catalog pins, `knowledge = list(refresh = TRUE)` explicitly selects the active local catalog only if every existing dependency retains its source revision. This option performs no network update.

Usage

```
sync(path = ".", config = NULL, options = list(), dry_run = TRUE)
```

Arguments

path	Exact project root.
config	Optional configuration list or YAML/JSON path.
options	Configuration overrides taking precedence over `config`.
dry_run	Return a read-only plan by default.

Details

For current standard projects, workflow options pipeline, environment, prepare_environment, network, git and table_backend may change; the profile stays fixed. An applied sync then runs the explicit environment and Git steps; dry runs only read their state. renv.lock is derived output and is never planned as a template file. A lockfile whose project library is absent on this machine is not rewritten: the recovery command restores it with renv::restore().

cttir-project.yml may be edited by hand. project() and audit() then report the edit, and sync(path) previews it against the accepted copy in .cttir/accepted-spec.yml: the edit is checked with the same rules as configuration input (schema version, identity, provenance and decisions stay as recorded; entries are not removed; publication paths are stable). sync(path, dry_run = FALSE) accepts it; config and options passed in the same call take precedence over the hand edit. Schema 1 projects remain readable. Synchronization previews their migration to schema 2; applying it preserves the original specification under .cttir/migrations/ and retains the selected catalog and dependency pins. The migration result records before/after hashes and changed JSON pointers.

Value

A cttir_sync with actions (with a reason per file), conflicts, changed files, preserved edited files, pending_updates (kept user files with a proposed update), spec_edit (a reviewed hand edit and its changed JSON pointers), recovered stale locks, journal, the computed readiness level, blockers, environment and git status and an analysis configuration assessment. Recording mappings does not approve or execute an analysis.

---

update *Refresh explicitly registered package sources and resource metadata*

Description

Builds complete immutable API and resource snapshots before one atomic pointer activation. Never installs or upgrades packages, runs install.packages() or BiocManager, executes downloaded or source code, changes project pins or promotes workflow approvals. Removed exports do not fall back to bundled revisions. A failed required source or catalog preserves both previous catalogs; the typed error carries a failed cttir_update in $report.

Usage

```
update(
  sources = NULL,
  packages = NULL,
  mode = c("local", "remote"),
  dry_run = FALSE,
  include_embeddings = FALSE,
  prune = FALSE,
  catalogs = c("knowledge", "resources"),
  discover = FALSE,
```

```
  bioc_version = NULL
)

update_knowledge(...)
```

Arguments

sources	Optional vector of registered source IDs.
packages	Optional vector of exact package names within selected sources.
mode	`local` reads configured local sources without any network access and marks remote currency unknown. `remote` also fetches registered public GitHub, CRAN and Bioconductor sources and refreshes resource observations from the official repository indices.
dry_run	Compute the candidate and diff in disposable temporary storage. Remote previews download only into that storage and delete it afterwards.
include_embeddings	Request derived embeddings. No embedding backend is yet configured; lexical search remains usable with a warning.
prune	When `TRUE`, registry records declaring `retired = TRUE` with a nonempty `retirement_evidence` are removed from the new current API snapshot and recorded as tombstones. Without evidence the update is refused. When `FALSE` (default), retired or unavailable packages stay current with a stale freshness flag.
catalogs	Nonempty subset of `knowledge` and `resources`.
discover	When `TRUE` (remote mode only), adds a bounded number of new quarantined `discovered_candidate` rows matching configured keywords.
bioc_version	Optional exact Bioconductor release such as `"3.23"`. `NULL` keeps the catalog-store policy or derives the release compatible with the running R. Aliases such as `release`/`devel` and unknown releases fail.
...	Arguments forwarded unchanged to `update()`.

Details

Registry. Register trusted sources with `options(cttiR.sources = list(...))`. Each record has an `id` and exactly one location:

- `path`: a trusted local package source directory.
- `github = "owner/repository"` with `package`; optional `ref` (commit, tag or branch, default `HEAD`) and `subdir`. Fetches resolve one commit and verify each file against its Git blob hash. At most 5000 tree entries, 250 files, 20 MB in total and one MB per file; 30-second request timeouts and a two-minute source budget. Truncated trees fail closed.
- `cran = "pkg"`, optional exact `version` and `md5`. The current version is fetched from `https://cloud.r-project.org/s` and verified against the `MD5sum` in `https://cloud.r-project.org/src/contrib/PACKAGES`. Other versions use `.../src/contrib/Archive/<pkg>/<pkg>_<ver>.tar.gz`; CRAN publishes no checksum for those, so a registered `md5` is verified when given and otherwise the locally computed MD5 is labelled as unverified.

- `bioc = "pkg"`, optional `bioc_version` and `version`: the tarball listed in `https://bioconductor.org/packages/<re` verified against its `MD5sum`. An exact earlier version of the same release is fetched from that release's `Archive/` directory, verified against a registered `md5` when given and otherwise labelled unverified.
- `r_distribution`: a local released R source tree (directory containing `VERSION` and `COPYING`) with an exact `package`. This reads `src/library/<package>/DESCRIPTION.in`, replacing only the literal `@VERSION@` marker in memory; development versions are refused and the tree is not authenticated as canonical.

MD5 values are repository integrity metadata, not publisher authenticity. Repository fetches use only those fixed HTTPS endpoints, never follow redirects or send credentials, time out after 60 seconds per request, and bound each index to 50 MB and each tarball to 30 MB (120 MB uncompressed). Archives are parsed in memory and fully listed first; absolute or parent paths, links, devices, duplicate or out-of-package members refuse the whole archive. Only DESCRIPTION, NAMESPACE, top-level `R/*.R`, `man`, vignettes, `inst/doc`, `inst/CITATION` and top-level README/NEWS/licence files are written to a temporary stage; documentation files above one MB are omitted with their size and hash recorded, and oversize API files fail closed.

Optional records may set `optional = TRUE`: an unavailable optional remote source keeps its previous revision marked `source_unavailable` and the result is `partial`; a required failure aborts before activation. A record may also declare `retired = TRUE` with `retirement_evidence` (see `prune`). Local mode skips remote registrations unless they are explicitly selected, in which case it reports that remote mode is required.

Resources. In remote mode the resource catalog reads the CRAN index and the selected Bioconductor release software index once each per run and refreshes observations of CRAN and Bioconductor software packages already in the resource database (version, licence, R and dependency floors, compilation flag, observation time and fetch status). Curated purpose, tiers, notes and verification states are preserved. Absence from an index is recorded as `missing_from_selected_index`, never as retirement. Version, licence and dependency-floor changes and dual CRAN/Bioconductor listings with different versions are reported in `resource_changes`. An unavailable index fails the update unless declared optional via `options(cttiR.resource_refresh = list(optional = c("cran", "bioconductor")))`; its previous observations are then kept as `source_unavailable`. The same option takes `exclude`, exact package names that are never refreshed or discovered. Discovery uses `options(cttiR.discovery = list(keywords = ..., biocViews = ..., limit = 25L))` (limit at most 100) against package names and any Title/biocViews fields present in the fetched indices; the current official PACKAGES files carry neither field, so matching is effectively by name. Discovered rows are quarantined and never promoted. An explicit `bioc_version` that differs from the stored policy stages a separate observation set (release in each observation ID) and records the new policy in the composite snapshot only after a successful applied run, so `rollback_knowledge()` restores the policy together with both catalogs; project pins keep their resource snapshots. Upstream-only, base R, historical Bioconductor and data packages are not refreshed by index. A resources-only run observes CRAN and Bioconductor registrations through the indices without downloading their tarballs.

Not supported. Discovery from Bioconductor VIEWS or package pages, versions from other Bioconductor releases, alternate CRAN paths (such as recommended package copies), private repositories, conditional HTTP requests, embeddings and automatic promotion of candidates.

Documentation text is stored only for records with `documentation_rights`, a nonempty description of the reviewed rights basis covering all selected documents; otherwise only hashes and inventory are kept. Documents are stored literally, never rendered or evaluated (one MB per file, ten MB

per package); binary assets and rendered `inst/doc` HTML/PDF up to 25 MB are hashed but never stored. Reviewed workflow approvals (see `inst/extdata/approvals.json` and `options(cttiR.approvals)`) attach only to the exact extracted revision; the result reports approvals added, removed or invalidated in `approval_diff`. Missing vignettes are reported as absent from the source, not unpublished. Sources are parsed statically; local sources receive a content-derived revision, including same-version edits. Packages absent from the resource registry are indexed as APIs only.

Observation times are part of the resource snapshot, so a preview and its applied run can have different composite IDs even when API content matches. Repeating an unchanged update retains its ID. Applied updates can recover a stopped local writer lock only after checking its owner, the active snapshot and any activation journal; the old lock is retained under `recovered-locks`. Active or unknown writers, corrupt pointers and conflicting journals are refused, and previews never recover locks. An active snapshot that fails verification is refused with the verified retained IDs that `rollback_knowledge()` can restore.

Value

A `cttir_update` with `status` (`planned`, `unchanged`, `succeeded`, `partial`; failures raise with a `failed` report), previous/new composite IDs, per-catalog IDs, per-source status and report, repository index reports, the exact repository URLs requested, `resource_changes`, API and documentation diffs, new `tombstones`, the Bioconductor release policy, activation and warnings.

`validate_config` *Validate customization input*

Description

Validates a named list or a local JSON/YAML file without executing expressions, making network requests or creating files. Unknown keys are rejected. Version 1 configuration is normalized to version 2 in memory; the input file is not rewritten. Explicit interoperability pins must match the active catalog.

Usage

```
validate_config(config)
```

Arguments

`config`	A named list or path to a JSON/YAML file.

Value

Invisibly, the normalized configuration list. Invalid input raises a `cttir_schema_error` or `cttir_input_error` condition.

Examples

```
validate_config(list(research = list(data_origin = "existing_dataset")))
```

`validate_spec` *Validate a resolved project specification*

Description

Validate a resolved project specification

Usage

```
validate_spec(spec)
```

Arguments

`spec`	A resolved specification list or local JSON/YAML file path.

Value

Invisibly, the normalized specification. Also checks unique IDs, portable slugs and conservative scientific approval prerequisites.

Index

ask, 3
ask(), *16*
audit, 4
audit(), *6*, *16*

check_analysis_data, 5
configure, 6

doctor, 6

evidence_resolve, 7
evidence_search, 7
evidence_search(), *7*

packages, 8
packages(), *16*
project, 9
project(), *4*, *16*

requireNamespace(), *4*
resources, 10
resources(), *16*
rollback_knowledge, 12
rollback_knowledge(), *8*, *16*, *28*, *29*
runtime_cleanup, 12
runtime_stop, 13
runtime_storage, 13

search, 14
search(), *16*
setup, 15
setup(), *10*, *16*
setup_app, 16
setup_release, 17
study_analyze, 18
study_dependency_lock, 19
study_dependency_lock(), *21*
study_paired_spec, 19
study_paired_spec(), *18*, *20*, *24*
study_project, 20
study_project(), *16*
study_render, 21
study_render(), *16*, *21*, *23*
study_run, 22
study_run(), *16*, *21*
study_spec, 22
study_spec(), *18*, *20*, *24*
study_stage, 23
study_validate, 24
study_validate_project, 24
study_validate_project(), *21*
sync, 25
sync(), *6*, *10*, *16*

update, 26
update(), *8*, *12*, *16*, *27*
update_knowledge *(*update*)*, 26

validate_config, 29
validate_spec, 30