Package: rmoriebricklayer (via r-universe)

June 25, 2026

Title Reproducible Data Bundles with Provenance and Fallback

Version 0.1.0

Description Tools for building brick-proof, reproducible data bundles. Resolves open-data sources through CKAN package_show and package_search endpoints, records and verifies provenance with SHA256 digests and Wayback Machine snapshots, validates downloaded data against a pinned schema, and falls back to schema-driven synthetic data when the real source is unreachable. Run records are captured in a manifest plus a plain-language summary so any result can be traced back to its inputs.

License AGPL-3

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports digest, jsonlite, stats, tools, utils

Suggests testthat (>= 3.0.0)

URL https://github.com/rootcoder007/rmorie-bricklayer

BugReports https://github.com/rootcoder007/rmorie-bricklayer/issues

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Repository https://rootcoder007.r-universe.dev

Date/Publication 2026-06-24 23:52:34 UTC

RemoteUrl https://github.com/rootcoder007/rmorie-bricklayer

RemoteRef main

RemoteSha 6c241cd101ef8e7301d6e7cecbc06213cb0632ae

RemoteSubdir bricklayer

Contents

`apply_schema_validation` *Apply Schema Validation, Stopping on Fatal Issues*

Description

Runs `validate_schema()` and acts on the result: fatal issues raise an error via `stop()`, warning-severity issues emit a `warning()`.

Usage

```
apply_schema_validation(df_raw, provenance)
```

Arguments

`df_raw`	The data frame to validate.
`provenance`	A provenance list as returned by `load_provenance()`.

Value

Invisibly, `TRUE` if no issues were found and `FALSE` otherwise. Errors if any fatal issue is present.

download_data *Download a File*

Description

Thin wrapper around `utils::download.file()` that returns the target path invisibly so it composes in pipelines.

Usage

```
download_data(url, target_path, mode = "wb", quiet = FALSE)
```

Arguments

`url`	URL to download.
`target_path`	Destination path on disk.
`mode`	Write mode passed to `utils::download.file()`; defaults to `"wb"` (binary) for cross-platform safety.
`quiet`	Logical; suppress progress output. Defaults to `FALSE`.

Value

The `target_path`, returned invisibly.

friendly_download *Download a File With Diagnostic Error Messages*

Description

Wraps `utils::download.file()` and, on failure, prints plain-language guidance for the most common academic and corporate network problems (rate limiting, TLS-inspection VPNs, DNS failures, timeouts, HTTP 403). Optionally retries from a Wayback Machine snapshot URL.

Usage

```
friendly_download(url, target_path, attempt_wayback = NULL)
```

Arguments

`url`	URL to download.
`target_path`	Destination path on disk.
`attempt_wayback`	Optional Wayback Machine snapshot URL tried as a fallback if the primary download fails. `NULL` (default) or an empty string disables the fallback.

Value

TRUE if either the primary download or the Wayback fallback succeeds, otherwise FALSE.

---

load_provenance *Load a Pinned Data-Provenance Record*

Description

Reads a data_provenance.json file describing a project's pinned data source: the CKAN endpoint, resource name pattern, expected SHA256, Wayback snapshot, schema, and synthetic-data recipe.

Usage

```
load_provenance(path)
```

Arguments

path	Path to the provenance JSON file.

Value

The parsed provenance as a nested list (via jsonlite::fromJSON() with simplifyVector = FALSE), or NULL if the file does not exist.

---

make_manifest *Construct a Reproducibility Manifest*

Description

Creates an empty manifest object that accumulates cross-check entries via record() and is later serialized with write_manifest_json().

Usage

```
make_manifest(meta)
```

Arguments

meta	A named list of run metadata (e.g. project, author, run_at, os, r_version, synthetic).

Value

A manifest list with elements meta and an empty results list.

make_synthetic_column *Generate One Synthetic Column From a Spec*

Description

Builds a single synthetic data column according to a column spec drawn from a provenance synthetic recipe. Supported types are "sample" (categorical, optionally weighted), "bernoulli" (two-label draw with optional per-row base rate), "poisson" (counts with a floor), "id_pattern" (templated IDs, optionally per-year sequenced), and "sequence" (a running integer sequence).

Usage

```
make_synthetic_column(spec, n, ctx = list(), base_p = NULL)
```

Arguments

spec	A list describing the column; recognised fields depend on spec$type (e.g. values, weights, p, p_with_baserate, labels, lambda, min, pattern, year_col, from).
n	Number of values to generate.
ctx	Named list of already-generated columns, letting later columns (such as id_pattern with a year_col) reference earlier ones. Defaults to an empty list.
base_p	Optional numeric vector of per-row latent propensities used by the "bernoulli" type to add row-level variation.

Value

A vector of length n for the requested column type. Errors on an unknown type.

make_synthetic_csv *Generate a Synthetic CSV From a Schema Recipe*

Description

Generates a reproducible synthetic data set from the schema$synthetic_recipe block of a provenance object and writes it to a CSV. Columns are produced in declaration order so later columns can reference earlier ones, a shared per-row latent propensity drives any Bernoulli columns, and an optional row-replication block expands per-person rows.

Usage

```
make_synthetic_csv(schema, out_path, n_rows = NULL, seed = NULL)
```

Arguments

schema	The synthetic recipe (a list with columns, and optional n_rows, seed, and row_replication).
out_path	Path where the CSV is written.
n_rows	Number of rows (persons, if replicating) to generate. Defaults to schema$n_rows, then 50000.
seed	Random seed for reproducibility. Defaults to schema$seed, then 91735246.

Value

Invisibly, a list with path, rows (rows written), and seed used.

---

record *Record a Cross-Check Result in a Manifest*

Description

Appends one named cross-check entry to a manifest, classifying it as PASS, DIFFER, or INFO, printing a formatted line to the console, and returning the updated manifest.

Usage

```
record(
  manifest,
  name,
  observed,
  expected,
  tol = 1e-04,
  group = "general",
  synthetic = FALSE
)
```

Arguments

manifest	A manifest as returned by make_manifest().
name	Unique name for this cross-check; used as the result key.
observed	The observed value (numeric or otherwise).
expected	The expected value to compare against.
tol	Numeric tolerance; a numeric pair within tol is PASS. Defaults to 0.0001.
group	Optional grouping label for the entry. Defaults to "general".
synthetic	Logical; if TRUE the entry is marked INFO because comparison against synthetic data is not meaningful.

Value

The updated manifest, returned so calls can be chained.

resolve_via_ckan *Resolve a Download URL via CKAN package_show*

Description

Queries the CKAN `package_show` endpoint recorded in a provenance object and returns the URL of the first resource whose name matches the provenance's name-match pattern. CKAN powers data.ontario.ca, data.gov.uk, data.gov, and most government open-data portals, so this recovers the current download URL even if the underlying resource UUID has been replaced.

Usage

```
resolve_via_ckan(provenance)
```

Arguments

`provenance`	A provenance list as returned by `load_provenance()`. Must contain `dataset$ckan_api_endpoint` and `resource$name_match_pattern`.

Value

The matched resource URL as a character string, or `NULL` if the endpoint is missing, the request fails, CKAN reports failure, or no resource name matches.

resolve_via_ckan_search *Resolve a Download URL via CKAN package_search*

Description

Fallback for `resolve_via_ckan()` when the dataset slug has changed. Derives the CKAN portal base URL from the provenance's `package_show` endpoint, runs a `package_search` query (from `resource$search_query`, or derived from the name-match pattern), and returns the URL of the first matching resource, preferring CSV format when specified.

Usage

```
resolve_via_ckan_search(provenance)
```

Arguments

`provenance`	A provenance list as returned by `load_provenance()`. Uses `resource$search_query`, `resource$name_match_pattern`, `resource$format`, and `dataset$ckan_api_endpoint`.

Value

The matched resource URL as a character string, or `NULL` if no query or base URL can be derived, the request fails, or nothing matches.

sha256_file *Compute a File's SHA256 Digest*

Description

Returns the SHA256 digest of a file as a lowercase hex string, using the digest package. Used to record and verify data provenance.

Usage

```
sha256_file(path)
```

Arguments

path	Path to the file to hash.

Value

The SHA256 digest as a character string.

validate_schema *Validate a Data Frame Against a Provenance Schema*

Description

Checks a raw data frame against the schema block of a provenance object: required columns, row-count bounds, and allowed categorical value sets. Returns the issues found rather than raising, so the caller decides how to react.

Usage

```
validate_schema(df_raw, provenance)
```

Arguments

df_raw	The data frame to validate.
provenance	A provenance list as returned by load_provenance(). The schema block may contain expected_columns, structural_invariants (min_data_rows, max_data_rows), and expected_value_sets (a named list of allowed values per column).

Value

A named list of issues; each issue is a list with severity ("fatal" or "warning") and a human-readable message. A zero-length list means the data frame is clean.

verify_sha256 *Verify a File's SHA256 Against an Expected Digest*

Description

Computes the SHA256 digest of a file (via the digest package) and compares it to the expected value pinned in provenance.

Usage

```
verify_sha256(path, expected_sha)
```

Arguments

path	Path to the file to hash.
expected_sha	The expected SHA256 digest, as a lowercase hex string.

Value

A list with actual (computed digest), expected (the value passed in), and match (logical; TRUE if they are identical).

write_manifest_json *Write a Manifest to JSON*

Description

Serializes a manifest to a pretty-printed JSON file via the jsonlite package.

Usage

```
write_manifest_json(manifest, path)
```

Arguments

manifest	A manifest as returned by make_manifest() / built up with record().
path	Destination path for the JSON file.

Value

The path, returned invisibly.

`write_summary_txt` *Write a Plain-Language Run Summary*

Description

Writes a human-readable `SUMMARY.txt` into the output directory, covering run metadata, the exact absolute paths used, result counts, the files produced, and optional notes, contact, and licence lines.

Usage

```
write_summary_txt(
  manifest,
  output_dir,
  paths,
  what_was_done = NULL,
  contact = NULL,
  licence = NULL
)
```

Arguments

`manifest`	A manifest as returned by `make_manifest()`; its `meta` supplies project/author/run details.
`output_dir`	Directory to write `SUMMARY.txt` into and to list produced files from.
`paths`	A named list of absolute paths to report (e.g. `bundle`, `input`, `results`, `analysis_script`, `provenance`).
`what_was_done`	Optional character vector of bullet points describing what the run did.
`contact`	Optional contact string appended to the summary.
`licence`	Optional licence string appended to the summary.

Value

The path to the written `SUMMARY.txt`, returned invisibly.

Index