Package 'pensar'

May 9, 2026

Type Package

Title LLM Wiki Engine

Version 0.4.2

Description Maintains a persistent, compounding knowledge base from source documents. Ingests articles, chat logs, briefings, and messages into a structured vault of markdown files with 'YAML' frontmatter and wikilinks. The vault is designed for large language model (LLM) agents to summarize, cross-reference, and maintain; humans curate sources, edit, and ask questions.

License Apache License (>= 2)

URL https://github.com/cornball-ai/pensar

BugReports https://github.com/cornball-ai/pensar/issues

SystemRequirements pandoc (for vault_export()), git (for vault_commit())

Imports yaml

Suggests saber, tinytest

Encoding UTF-8

NeedsCompilation no

Author Troy Hernandez [aut, cre] (ORCID: <https://orcid.org/0009-0005-4248-604X>), cornball.ai [cph]

Maintainer Troy Hernandez <troy@cornball.ai>

Repository CRAN

Date/Publication 2026-05-04 19:00:02 UTC

Contents

`backlinks` *Backlink discovery*

Description

Find pages that link to a given page via wikilinks. Find backlinks to a page

Scans all markdown files in the vault for `[[wikilinks]]` that reference the target page.

Usage

```
backlinks(page, vault = default_vault())
```

Arguments

`page`	Page name (without `.md` extension).
`vault`	Path to the vault directory.

Value

A data.frame with columns `source` (page name) and `file` (path relative to the vault).

Examples

```
v <- tempfile("vault-")
init_vault(v, rproj = FALSE, agent_instructions = FALSE)
ingest("See [[seed]] for context.", type = "articles",
       source = "demo", vault = v)
backlinks("seed", vault = v)
unlink(v, recursive = TRUE)
```

`ingest` *Source ingestion*

Description

Ingest content into a pensar vault. Ingest content into the vault

Writes content to `raw/{type}/`, generates a filename from source and date, adds YAML frontmatter, updates `index.md`, and appends to `log.md`.

Usage

```
ingest(content, type = c("articles", "chats", "briefings", "matrix"), source,
       title = NULL, tags = NULL, vault = default_vault())
```

Arguments

`content`	Character string or character vector (lines) of content.
`type`	Content type: `"articles"`, `"chats"`, `"briefings"`, or `"matrix"`.
`source`	Short identifier for the content source (e.g., URL, session ID, project name).
`title`	Optional title. If `NULL`, derived from source.
`tags`	Optional character vector of tags.
`vault`	Path to the vault directory.

Value

The path to the written file, invisibly.

Examples

```
v <- tempfile("vault-")
init_vault(v, rproj = FALSE, agent_instructions = FALSE)
ingest("Hello, world.", type = "articles", source = "demo",
       vault = v)
status(v)
unlink(v, recursive = TRUE)
```

ingest_briefing *Briefing ingestion*

Description

Generate a saber briefing and ingest it into the vault. Generate and ingest a saber briefing

Calls `saber::briefing(project)` to produce a project briefing and ingests it into the vault as a `briefings` raw source. Requires the `saber` package.

Usage

```
ingest_briefing(project = NULL, vault = default_vault())
```

Arguments

`project`	Project name. If `NULL`, inferred from the git root of the current working directory.
`vault`	Path to the vault directory.

Value

The path to the ingested briefing file, invisibly. Returns `NULL` invisibly if `saber` is not installed or the project cannot be inferred.

Examples

```
## Not run:
# Requires saber and a git project so a briefing can be generated.
v <- tempfile("vault-")
init_vault(v, rproj = FALSE, agent_instructions = FALSE)
ingest_briefing(project = "pensar", vault = v)

## End(Not run)
```

init_vault *Vault initialization*

Description

Create and seed a pensar vault. Initialize a pensar vault

Creates the vault directory structure and seeds the control files: `schema.md`, `index.md`, `log.md`, and (by default) agent instruction files for Claude Code and Codex.

Usage

```
init_vault(path = default_vault(), rproj = TRUE, agent_instructions = TRUE)
```

Arguments

path	Path to the vault directory. No implicit default: pass an explicit path, or configure one via PENSAR_VAULT, use_vault(), or a walk-up schema.md marker. Per CRAN policy pensar will not silently write to the user's home filespace.
rproj	If TRUE (default), also write an RStudio project file ({basename(path)}.Rproj). The project file makes a vault stored under a hidden directory (e.g., one configured via PENSAR_VAULT pointing at ~/.local/share/...) easy to open as an RStudio project, since RStudio's GUI normally refuses to create projects inside hidden folders. Code indexing is disabled in the project file since the vault contents are markdown, not R source. The file is a harmless ~14-line INI stub; delete it anytime if you prefer not to use RStudio. Pass rproj = FALSE to skip it entirely.
agent_instructions	If TRUE (default), write CLAUDE.md and AGENTS.md with identical content orienting an AI agent to work in this vault (CLI reminders, editing rules, ingest workflow). If you don't plan to start an AI agent session in the vault, pass FALSE.

Value

The vault path, invisibly.

Examples

```
v <- tempfile("vault-")
init_vault(v, rproj = FALSE, agent_instructions = FALSE)
list.files(v, recursive = TRUE)
unlink(v, recursive = TRUE)
```

---

lint *Vault lint*

Description

Health check for a pensar vault. Vault health check

Scans the vault for orphan pages (no incoming wikilinks), broken wikilinks (pointing to nonexistent pages), and tag clusters with no wiki synthesis.

Usage

```
lint(vault = default_vault(), min_cluster_size = 3L)
```

Arguments

vault	Path to the vault directory.
min_cluster_size	Minimum number of raw pages sharing a tag to suggest a wiki page. Default 3.

Value

A list with class pensar_lint.

Examples

```
v <- tempfile("vault-")
init_vault(v, rproj = FALSE, agent_instructions = FALSE)
ingest("Refers to [[absent]].", type = "articles", source = "demo",
       vault = v)
lint(v)
unlink(v, recursive = TRUE)
```

---

log_entry *Vault log*

Description

Append-only operation log for a pensar vault. Append a log entry

Appends a structured entry to log.md with timestamp, operation type, and message.

Usage

```
log_entry(message, operation = "note", vault = default_vault())
```

Arguments

message	Description of what happened.
operation	Operation type (e.g., "init", "ingest", "lint").
vault	Path to the vault directory.

Value

Invisible NULL.

Examples

```
v <- tempfile("vault-")
init_vault(v, rproj = FALSE, agent_instructions = FALSE)
log_entry("Reviewed wiki/seed.md", operation = "review", vault = v)
unlink(v, recursive = TRUE)
```

outlinks *Outlink discovery*

Description

Find the pages a given page cites via wikilinks. Find outlinks from a page

Scans a single page for `[[wikilinks]]` and returns the targets. Mirror of `backlinks()` in the forward direction.

Usage

```
outlinks(page, vault = default_vault())
```

Arguments

`page`	Page name (without `.md` extension).
`vault`	Path to the vault directory.

Value

A data.frame with columns `target` (page name) and `exists` (logical: whether the target page exists in the vault).

Examples

```
v <- tempfile("vault-")
init_vault(v, rproj = FALSE, agent_instructions = FALSE)
fp <- ingest("Cites [[seed]] and [[missing]].", type = "articles",
             source = "demo", vault = v)
outlinks(tools::file_path_sans_ext(basename(fp)), vault = v)
unlink(v, recursive = TRUE)
```

show_page *Page inspection*

Description

Drill down into a page: content, outlinks, and backlinks. Show a page with its connections

Returns the page content alongside its outgoing and incoming wikilinks. Use this when you need to review or edit a page: the outlinks show what raw sources the page cites; the backlinks show what depends on it.

Usage

```
show_page(page, vault = default_vault())
```

Arguments

page	Page name (without .md extension).
vault	Path to the vault directory.

Value

A list with class pensar_page.

Examples

```
v <- tempfile("vault-")
init_vault(v, rproj = FALSE, agent_instructions = FALSE)
fp <- ingest("Body text.", type = "articles", source = "demo",
             vault = v)
show_page(tools::file_path_sans_ext(basename(fp)), vault = v)
unlink(v, recursive = TRUE)
```

---

status *Vault status*

Description

Summary stats for a pensar vault. Vault status summary

Returns page counts by category, total pages, and wikilink count.

Usage

```
status(vault = default_vault())
```

Arguments

vault	Path to the vault directory.

Value

A list with class pensar_status.

Examples

```
v <- tempfile("vault-")
init_vault(v, rproj = FALSE, agent_instructions = FALSE)
status(v)
unlink(v, recursive = TRUE)
```

update_index *Vault index*

Description

Regenerate the vault index as a markdown catalog. Update the vault index

Scans all markdown files in the vault and regenerates `index.md` as a categorized catalog with wikilinks and titles.

Usage

```
update_index(vault = default_vault())
```

Arguments

`vault`	Path to the vault directory.

Value

The path to `index.md`, invisibly.

Examples

```
v <- tempfile("vault-")
init_vault(v, rproj = FALSE, agent_instructions = FALSE)
ingest("Body.", type = "articles", source = "demo", vault = v)
update_index(v)
unlink(v, recursive = TRUE)
```

---

use_vault *Remember a vault path for this R session*

Description

Sets `options("pensar.vault")` so subsequent pensar calls resolve to `path` without repeating the argument. Persist by adding `pensar::use_vault("~/wiki")` to `~/.Rprofile` as a global default. Both `PENSAR_VAULT` and a project-local `schema.md` found via walk-up will override this option (see `default_vault` resolution order).

Usage

```
use_vault(path)
```

Arguments

`path`	Path to your pensar vault directory.

Value

The resolved path, invisibly.

Examples

```
v <- tempfile("vault-")
init_vault(v, rproj = FALSE, agent_instructions = FALSE)
use_vault(v)
status()
options(pensar.vault = NULL)
unlink(v, recursive = TRUE)
```

---

vault_commit *Vault git operations*

Description

Auto-commit and push for pensar vaults that are git repos. Commit vault changes to git

No-op if the vault is not a git repo or if there are no changes. Stages all changes (respecting `.gitignore`), commits with the given message, and optionally pushes to remotes.

Honors the `PENSAR_AUTO_PUSH` environment variable: if set to `"0"` or `"false"` (case-insensitive), skips the push step. Otherwise, pushes to every configured remote.

Usage

```
vault_commit(message, vault = default_vault(), push = NULL)
```

Arguments

`message`	Commit message.
`vault`	Path to the vault directory.
`push`	If `NULL` (default), honors `PENSAR_AUTO_PUSH`. Pass `TRUE` or `FALSE` to override.

Value

`TRUE` if a commit was made, `FALSE` otherwise (invisibly).

Examples

```
v <- tempfile("vault-")
init_vault(v, rproj = FALSE, agent_instructions = FALSE)
# Returns FALSE invisibly: no .git in this temp vault.
vault_commit("noop", vault = v, push = FALSE)
unlink(v, recursive = TRUE)
```

vault_export *Static HTML export*

Description

Render the vault to a directory of static HTML files. Export the vault to static HTML

Renders every markdown page in the vault to HTML, resolving `[[wikilinks]]` to relative anchor tags. Output is a standalone site that can be served from any static file server or opened via `file://`.

The rendered site is regenerable from the vault, so it defaults to the R user cache directory (`tools::R_user_dir("pensar","` rather than living inside the vault itself. Pass a different `out_dir` to override, or set the `PENSAR_SITE_DIR` environment variable to change the default globally (e.g., point it at a Syncthing folder so edits propagate to other devices on export).

Requires the pandoc command-line tool to be available.

Usage

```
vault_export(vault = default_vault(), out_dir = default_site_dir())
```

Arguments

`vault`	Path to the vault directory.
`out_dir`	Destination directory. No default: pass an explicit path or set `PENSAR_SITE_DIR` (per CRAN policy, pensar will not silently render into a home-filespace location).

Value

The output directory path, invisibly.

Examples

```
if (nzchar(Sys.which("pandoc"))) {
    v <- tempfile("vault-")
    init_vault(v, rproj = FALSE, agent_instructions = FALSE)
    ingest("Body.", type = "articles", source = "demo", vault = v)
    vault_export(v, out_dir = tempfile("site-"))
    unlink(v, recursive = TRUE)
}
```

vault_graph *Vault wikilink graph*

Description

Render the vault's wikilink graph as SVG via saber. Render a vault's wikilink graph as SVG

Scans every markdown page in the vault (excluding control files), extracts `[[wikilinks]]` as edges, and renders the result via `saber::graph_svg()`. Node tooltips carry the page type, tags, and date from YAML frontmatter; broken wikilinks (targets with no matching page) appear as external nodes with a distinct tooltip.

Usage

```
vault_graph(vault = default_vault(), width = 1600L, height = 1200L, ...)
```

Arguments

`vault`	Path to the vault directory.
`...`	Passed through to `saber::graph_svg()` (e.g., `iterations`, `seed`).
`width, height`	Viewport in pixels. Defaults (1600 x 1200) are larger than `saber::graph_svg()`'s defaults since vaults tend toward many nodes.

Value

Character vector of SVG lines. Write with `writeLines()`.

Examples

```
## Not run:
# Requires a version of 'saber' that exports graph_svg().
v <- tempfile("vault-")
init_vault(v, rproj = FALSE, agent_instructions = FALSE)
ingest("Cites [[other]].", type = "articles", source = "demo",
       vault = v)
svg <- vault_graph(v)
writeLines(svg, tempfile(fileext = ".svg"))

## End(Not run)
```

Index