

Package ‘saber’

June 25, 2026

Type Package

Title Context Engineering for Large Language Model Agents

Version 0.7.2

Date 2026-06-24

Description Context-engineering primitives for Artificial Intelligence (AI) coding agents working in R. Assembles agent context from memory and instruction files ('AGENTS.md', 'CLAUDE.md'), traces function call blast radius across projects, generates project briefings, parses source into Abstract Syntax Tree (AST) symbol indices, discovers dependency graphs, and introspects installed packages.
Zero dependencies.

License Apache License (>= 2)

Depends R (>= 4.4.0)

URL <https://github.com/cornball-ai/saber>

BugReports <https://github.com/cornball-ai/saber/issues>

Suggests tinytest

Encoding UTF-8

RoxygenNote 7.3.2

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 2026-06-25 05:20:02 UTC

Contents

agent_context	2
blast_radius	3

briefing	4
default_exclude	5
find_downstream	6
fn_graph	6
graph_svg	7
pkg_exports	8
pkg_graph	9
pkg_help	10
pkg_internals	10
projects	11
symbols	11

Index 13

agent_context	<i>Agent context assembly</i>
---------------	-------------------------------

Description

Assemble context from memory, instructions, and identity files for AI coding agents. Load context files for an AI coding agent

Returns assembled context (memory, project/global instructions, agent identity files) tailored to a specific consumer. Files autoloaded by the agent natively are skipped to avoid duplication.

Defaults per agent:

- "claude" - skips Claude Code project memory and CLAUDE.md files (autoloaded by 'Claude Code'). Loads Codex memories, AGENTS.md, USER.md, and SOUL.md when present.
- "codex" - skips AGENTS.md and Codex memories (autoloaded by Codex). Loads Claude Code project memory, CLAUDE.md, USER.md, and SOUL.md when present.
- "corteza" or NULL - loads everything available.

Project and global instructions are resolved by trying both naming conventions and picking the file relevant to the consumer:

- Project: CLAUDE.md or AGENTS.md
- Global: ~/.claude/CLAUDE.md or <workspace_dir>/USER.md

Override the defaults with the include_* parameters.

Usage

```
agent_context(agent = NULL, project_dir = getwd(), workspace_dir = NULL,
             memory_base = file.path(path.expand("~"), ".claude", "projects"),
             claude_global_path = file.path(path.expand("~"), ".claude", "CLAUDE.md"),
             include_memory = NULL, include_project = NULL,
             include_global = NULL, include_soul = NULL,
             max_memory_lines = 100L)
```

Arguments

agent	Consumer identifier: "claude", "codex", "corteza", or NULL (interactive / unknown). The legacy "llamar" identifier is accepted as an alias for "corteza".
project_dir	Project directory to scan for CLAUDE.md / AGENTS.md.
workspace_dir	Optional directory containing SOUL.md and USER.md (e.g. ~/.corteza/workspace). If NULL, those files are skipped.
memory_base	Base directory for 'Claude Code' project memory files.
claude_global_path	Path to the global 'Claude Code' instructions file. Defaults to ~/.claude/CLAUDE.md.
include_memory	Override default for memory inclusion. Use TRUE/FALSE to force, or NULL for the agent default.
include_project	Override default for project instructions (CLAUDE.md / AGENTS.md).
include_global	Override default for global instructions (~/.claude/CLAUDE.md / USER.md).
include_soul	Override default for SOUL.md inclusion.
max_memory_lines	Maximum lines to include from each memory source.

Value

Character string of assembled context, or empty string if no context applies.

Examples

```
# Codex agent in current project
saber::agent_context(agent = "codex")

# Corteza with workspace files
saber::agent_context(agent = "corteza",
                    workspace_dir = "~/corteza/workspace")

# Force-include memory regardless of agent default
saber::agent_context(agent = "claude", include_memory = TRUE)
```

blast_radius

Blast radius analysis

Description

Find all callers of a function across projects. Find callers of a function across projects

Given a function name and project, finds all internal callers within that project and all callers in downstream projects (projects whose DESCRIPTION lists this one in Depends, Imports, or LinkingTo).

With include = c("r", "examples", "vignettes") the search can be extended to references in the target project's roxygen @examples blocks and vignette code chunks (Rmd, qmd, Rnw). Documentation scanning is target-project only; it does not walk downstream projects' docs.

Usage

```
blast_radius(fn, project = NULL, include = "r", scan_dir = path.expand("~"),
             cache_dir = file.path(tools::R_user_dir("saber", "cache"), "symbols"),
             exclude = default_exclude())
```

Arguments

fn	Character. Function name to search for.
project	Character. Project name (or path to project directory).
include	Character vector. Any of "r" (R source, default), "examples" (roxygen@examples blocks in the target project), and "vignettes" (code chunks in the target project's vignettes).
scan_dir	Directory to scan for downstream projects.
cache_dir	Directory for symbol cache files.
exclude	Character vector of directory basenames to skip when scanning for downstream projects.

Value

A data.frame with columns: caller, project, file, line, source. source is one of "r", "example", "vignette".

Examples

```
# Create a minimal project
d <- file.path(tempdir(), "blastpkg")
dir.create(file.path(d, "R"), recursive = TRUE, showWarnings = FALSE)
writeLines("helper <- function(x) x + 1", file.path(d, "R", "helper.R"))
writeLines("main <- function(x) helper(x * 2)", file.path(d, "R", "main.R"))

# Find all callers of helper()
blast_radius("helper", project = d, scan_dir = tempdir(),
             cache_dir = tempdir())

# Include roxygen @examples and vignettes from the target project
blast_radius("helper", project = d, include = c("r", "examples", "vignettes"),
             scan_dir = tempdir(), cache_dir = tempdir())
```

 briefing

Project briefings

Description

Generate project context for AI coding agents. Generate a project briefing

Produces a concise markdown briefing combining DESCRIPTION metadata, downstream dependents, and recent git commits. Written to the user cache directory so both the agent and user see the same context.

For runtime context (memory, identity files, project instructions), see [agent_context](#).

Usage

```
briefing(project = NULL, scan_dir = path.expand("~/"),
         briefs_dir = file.path(tools::R_user_dir("saber", "cache"), "briefs"))
```

Arguments

project Project name. If NULL, inferred from the current working directory basename.

scan_dir Directory to scan for project directories.

briefs_dir Directory to write briefing markdown files.

Value

The briefing text (character string), returned invisibly. Emitted via message() and written to briefs_dir/{project}.md.

Examples

```
d <- file.path(tempdir(), "briefpkg")
dir.create(file.path(d, "R"), recursive = TRUE, showWarnings = FALSE)
writeLines(c("Package: briefpkg", "Title: Demo", "Version: 0.1.0"),
           file.path(d, "DESCRIPTION"))
briefing("briefpkg", scan_dir = tempdir(),
        briefs_dir = file.path(tempdir(), "briefs"))
```

default_exclude	<i>Default directories to exclude when scanning for projects</i>
-----------------	--

Description

Returns a character vector of directory basenames that are skipped when scanning for downstream projects. Override by passing a custom exclude vector to [blast_radius](#).

Usage

```
default_exclude()
```

Value

Character vector of directory basenames.

Examples

```
default_exclude()
```

find_downstream	<i>Find projects that depend on a given package</i>
-----------------	---

Description

Scans DESCRIPTION files in project directories under scan_dir for Depends, Imports, or LinkingTo fields that reference package.

Usage

```
find_downstream(package, scan_dir = path.expand("~"),
                exclude = default_exclude())
```

Arguments

package	Character. Package name to search for.
scan_dir	Directory to scan for project directories.
exclude	Character vector of directory basenames to skip.

Value

Character vector of project names that depend on package.

Examples

```
d <- file.path(tempdir(), "dsdir")
dir.create(d, showWarnings = FALSE)
pkg <- file.path(d, "child")
dir.create(pkg, showWarnings = FALSE)
writeLines(c("Package: child", "Version: 0.1.0", "Imports: parent"),
           file.path(pkg, "DESCRIPTION"))
find_downstream("parent", scan_dir = d)
```

fn_graph	<i>Code intelligence: function call graph</i>
----------	---

Description

Render a project's internal function call graph as interactive SVG. Render a function call graph for an R project

Pulls the AST symbol index via [symbols](#) and renders internal function-to-function call edges as SVG. External calls (into other packages) are dropped by default.

Usage

```
fn_graph(project_dir, include_external = FALSE, ...,
         cache_dir = file.path(tools::R_user_dir("saber", "cache"), "symbols"))
```

Arguments

`project_dir` Path to the project directory (an R package root).

`include_external` If TRUE, also include nodes for functions called from other packages. Default FALSE.

`...` Passed through to `graph_svg` (e.g., width, height, iterations, seed).

`cache_dir` Directory for the underlying `symbols` cache. Pass `tempdir()` (or any non-default path) when running examples / tests to avoid writing to the user's persistent cache. Name-only argument — must be passed by name.

Value

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

Examples

```
d <- file.path(tempdir(), "fngdemo")
dir.create(file.path(d, "R"), recursive = TRUE, showWarnings = FALSE)
writeln(c("Package: demo", "Version: 0.1.0"),
        file.path(d, "DESCRIPTION"))
writeln("add <- function(x, y) x + y", file.path(d, "R", "add.R"))
writeln("double <- function(x) add(x, x)",
        file.path(d, "R", "double.R"))
svg <- fn_graph(d, cache_dir = tempdir())
writeln(svg, tempfile(fileext = ".svg"))
```

graph_svg

Force-directed graph rendering

Description

Render a graph as static, interactive SVG using a base R Fruchterman-Reingold force simulation. Render a graph as static SVG

Runs a Fruchterman-Reingold force simulation to lay out nodes, then emits SVG with baked coordinates. Output is interactive via native browser features: hover tooltips from `<title>` elements, click navigation from `<a xlink:href>` wrappers, CSS `:hover` highlighting. No JavaScript.

Suitable for graphs up to roughly a few hundred nodes. The repulsion step is vectorized via `outer()` but allocates an $n \times n$ matrix per iteration; larger graphs should pre-filter.

Usage

```
graph_svg(edges, nodes = NULL, width = 1200L, height = 900L, iterations = 50L,
          seed = 1L)
```

Arguments

edges	Data frame with from and to columns holding node ids.
nodes	Optional data frame with id, label, href columns. Optionally a tooltip column (plain text, may contain newlines) that overrides the default hover text (which is the label). id must cover every node mentioned in edges. If NULL, ids from edges are used as labels with no hrefs or tooltips.
width	SVG viewport width in pixels.
height	SVG viewport height in pixels.
iterations	Force-simulation steps. 50 is usually enough.
seed	Integer seed for the random initial layout (output is deterministic given the same seed).

Value

Character vector, one SVG element per line. Write with `writeln()`.

Examples

```
edges <- data.frame(from = c("a", "a", "b"),
                   to = c("b", "c", "c"))
svg <- graph_svg(edges)
writeln(svg, tempfile(fileext = ".svg"))
```

pkg_exports

Package introspection

Description

Query installed R packages for exported functions, internal functions, and help documentation. List exported functions of a package

Returns a data.frame of exported functions with their argument signatures.

Usage

```
pkg_exports(package, pattern = NULL)
```

Arguments

package	Character. Package name.
pattern	Optional regex to filter function names.

Value

A data.frame with columns: name, args.

Examples

```
pkg_exports("tools")
pkg_exports("tools", pattern = "^Rd")
```

pkg_graph

Project discovery: package dependency graph

Description

Render the dependency graph across a set of R packages as interactive SVG. Render a package-level dependency graph

Discovers R packages under scan_dir via [projects](#), parses each one's Depends and Imports fields, and renders edges between packages that both live in scan_dir. External CRAN dependencies are dropped.

Usage

```
pkg_graph(scan_dir = path.expand("~/"), packages = NULL,
          include_suggests = FALSE, ...)
```

Arguments

scan_dir	Directory to scan for project directories.
packages	Optional character vector limiting the graph to these packages.
include_suggests	If TRUE, also include edges for packages in each project's Suggests field. Default FALSE (only Depends and Imports).
...	Passed through to graph_svg .

Value

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

Examples

```
d <- file.path(tempdir(), "pkgdemo")
dir.create(file.path(d, "parent"), recursive = TRUE, showWarnings = FALSE)
dir.create(file.path(d, "child"), showWarnings = FALSE)
writeln(c("Package: parent", "Title: P", "Version: 0.1.0"),
        file.path(d, "parent", "DESCRIPTION"))
writeln(c("Package: child", "Title: C", "Version: 0.1.0",
          "Imports: parent"),
        file.path(d, "child", "DESCRIPTION"))
svg <- pkg_graph(scan_dir = d)
writeln(svg, tempfile(fileext = ".svg"))
```

pkg_help *Get help for a package topic as markdown*

Description

Extracts help documentation and converts it to clean markdown.

Usage

```
pkg_help(topic, package, format = c("md", "hugo"))
```

Arguments

topic	Character. The help topic name.
package	Character. Package name.
format	Character. Output format: "md" (default) for plain markdown, or "hugo" for Hugo-compatible markdown with YAML front matter.

Value

Character string of markdown help text.

Examples

```
cat(pkg_help("md5sum", "tools"))
```

pkg_internals *List internal (non-exported) functions of a package*

Description

Returns functions defined in a package namespace but not exported.

Usage

```
pkg_internals(package, pattern = NULL)
```

Arguments

package	Character. Package name.
pattern	Optional regex to filter function names.

Value

A data.frame with columns: name, args.

Examples

```
pkg_internals("tools", pattern = "^check")
```

 projects

Project discovery

Description

Discover R package projects and map their dependency relationships. Discover R package projects
Scans a directory for subdirectories containing a DESCRIPTION file and returns their metadata.

Usage

```
projects(scan_dir = path.expand("~/"), exclude = default_exclude())
```

Arguments

scan_dir Directory to scan for project directories.
exclude Character vector of directory basenames to skip.

Value

A data.frame with columns: package, title, version, path, depends, imports.

Examples

```
d <- file.path(tempdir(), "scandir")
dir.create(d, showWarnings = FALSE)
pkg <- file.path(d, "mypkg")
dir.create(pkg, showWarnings = FALSE)
writeLines(c("Package: mypkg", "Title: Demo", "Version: 0.1.0"),
           file.path(pkg, "DESCRIPTION"))
projects(scan_dir = d)
```

 symbols

Code intelligence: AST symbol index

Description

Parse R source files into structured function definitions and call relationships.

Usage

```
symbols(project_dir,
        cache_dir = file.path(tools::R_user_dir("saber", "cache"), "symbols"))
```

Arguments

`project_dir` Path to the project directory.
`cache_dir` Directory for symbol cache files.

Value

A list with components:

defs `data.frame(name, file, line, exported)`

calls `data.frame(caller, callee, file, line)`

Examples

```
# Create a minimal project with R source files
d <- file.path(tempdir(), "demopkg")
dir.create(file.path(d, "R"), recursive = TRUE, showWarnings = FALSE)
writeLines("add <- function(x, y) x + y", file.path(d, "R", "add.R"))
writeLines("double <- function(x) add(x, x)", file.path(d, "R", "double.R"))

idx <- symbols(d, cache_dir = tempdir())
idx$defs # function definitions
idx$calls # call relationships (double calls add)
```

Index

[agent_context](#), [2](#), [4](#)

[blast_radius](#), [3](#), [5](#)

[briefing](#), [4](#)

[default_exclude](#), [5](#)

[find_downstream](#), [6](#)

[fn_graph](#), [6](#)

[graph_svg](#), [7](#), [7](#), [9](#)

[pkg_exports](#), [8](#)

[pkg_graph](#), [9](#)

[pkg_help](#), [10](#)

[pkg_internals](#), [10](#)

[projects](#), [9](#), [11](#)

[symbols](#), [6](#), [7](#), [11](#)