

# Package ‘saber’

April 5, 2026

**Type** Package

**Title** Code Analysis and Project Context for R

**Version** 0.3.0

**Description** Parses R source files into Abstract Syntax Tree (AST) symbol indices, traces function callers across projects, discovers project dependency graphs, generates project briefings, and provides package introspection tools. Designed for AI coding agents that need structured code understanding.

**License** Apache License (>= 2)

**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>>)

**Maintainer** Troy Hernandez <troy@cornball.ai>

**Repository** CRAN

**Date/Publication** 2026-04-05 07:00:02 UTC

## Contents

blast_radius	2
briefing	3
default_exclude	4
find_downstream	4
pkg_exports	5
pkg_help	6
pkg_internals	6
projects	7
symbols	7

---

blast_radius	<i>Blast radius analysis</i>
--------------	------------------------------

---

### 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).

### Usage

```
blast_radius(fn, project = NULL, 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).
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.

### 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())
```

---

briefing	<i>Project briefings</i>
----------	--------------------------

---

## Description

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

Produces a concise markdown briefing combining DESCRIPTION metadata, downstream dependents, 'Claude Code' memory (skipped when agent = "claude" since 'Claude Code' autoloads it), and recent git commits. Written to the user cache directory so both the agent and user see the same context.

## Usage

```
briefing(project = NULL, scan_dir = path.expand("~/"), agent = NULL,
         memory_base = file.path(path.expand("~/"), ".claude", "projects"),
         briefs_dir = file.path(tools::R_user_dir("saber", "cache"), "briefs"),
         max_memory_lines = 30L)
```

## Arguments

project	Project name. If NULL, inferred from the current working directory basename.
scan_dir	Directory to scan for project directories.
agent	Which coding agent is calling: "claude", "codex", or NULL (interactive / unknown). When "claude", the briefing skips 'Claude Code' memory (which 'Claude Code' autoloads separately). Other values include it.
memory_base	Base directory for 'Claude Code' project memory files.
briefs_dir	Directory to write briefing markdown files.
max_memory_lines	Maximum lines to include from the memory file.

## Value

The briefing text (character string), returned invisibly. Printed to stdout 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)
```

---

pkg\_exports

*Package introspection*

---

**Description**

Query installed R packages for exports, internals, and help. 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\_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	<i>Project discovery</i>
----------	--------------------------

---

**Description**

Discover R package projects and 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	<i>AST symbol index</i>
---------	-------------------------

---

**Description**

Parse R source files to extract function definitions and calls.

**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

blast\_radius, [2](#), [4](#)  
briefing, [3](#)  
  
default\_exclude, [4](#)  
  
find\_downstream, [4](#)  
  
pkg\_exports, [5](#)  
pkg\_help, [6](#)  
pkg\_internals, [6](#)  
projects, [7](#)  
  
symbols, [7](#)