Package: gsocproposal (via r-universe)

October 12, 2026

Title Submit Proposals to Google Summer of Code

Version 0.1.0

Description Provides a single function to determine whether a Google Summer of Code proposal can still be submitted. The submission deadline is stored in a package-level environment variable so that it can be updated without modifying source documentation. Documentation is generated dynamically: the deadline is resolved at install time and the time-remaining message is computed at render time using Rd Sexpr macros.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports tools

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Repository https://adit-0132.r-universe.dev

Date/Publication 2026-07-07 18:33:05 UTC

RemoteUrl https://github.com/adit-0132/multilingual-docs

RemoteRef publish

RemoteSha 23ff48d0c32c365d3e88fb364972c0612f02be55

Contents

submit_proposal 2
walk_rdb_tree 4

Index **6**

submit_proposal *Submit a Google Summer of Code Proposal* and *Showcase Rd Macros*

Description

Checks whether the current system date falls on or before the package submission deadline. Returns TRUE if submissions are still open and FALSE once the deadline has passed.

The deadline is not hard-coded in the documentation. It is read from the package environment .pkg_env$deadline at **install time**:

Deadline: 2025-03-31

When you view this help page, the following message is computed *live* at **render time**:

The deadline has passed, try next year!

Usage

```
submit_proposal()

## S3 method for class 'gsoc_proposal'
print(x, ...)
```

Arguments

x	An object of class gsoc_proposal. Dispatches via print.
...	Further arguments passed to or from other methods. Argument names with special characters (such as ...) must round-trip safely.

Details

This section showcases ordinary inline markup so we can see it render and, later, translate.

Inline text markup: Emphasis with *emph*, **strong**, and **bold**. Code and objects: submit_proposal(), a variable *deadline*, a package **gsocproposal**, a file 'R/globals.R', an option '--no-save', an environment variable LANGUAGE, a command R CMD INSTALL, and a keystroke Ctrl+C. Quotations: 'single' and "double"; a definition *Sexpr*; an acronym CRAN; a citation *R Core Team*; verbatim #ifdef; literal sample 'x <- 1'. The R language. A non-ASCII fallback: Björk. A literal percent sign: 50%. A user-defined macro that takes an argument: The deadline is **2025-03-31**.

Lists: An itemized list:

- Build stage runs at R CMD build.
- Install stage runs at R CMD INSTALL.
- Render stage runs every time ?submit_proposal is opened.

An enumerated list:

1. First.
2. Second.

A described list:

build Baked into the source tarball.

install Baked into the installed '`.rdb`' database.

render Evaluated live on every help lookup.

A table:

Stage	Runs during	Live in .rdb?
build	R CMD build	no
install	R CMD INSTALL	no (baked to text)
render	every help lookup	yes

Math: Inline math $d = t_1 - t_0$ and display math:

$$\text{open} \iff \text{Sys.Date()} \leq \text{deadline}.$$

Platform conditionals (#ifdef / #ifndef): The lines below are preprocessor-style conditionals. The # must be the first character on its line, the block must be wholly nested inside a macro, and each block is closed by `#endif` (parseRd.pdf sections 2.3 and 3). This paragraph is included on **everything except Windows** (Unix, macOS).

Value

A single logical value. `TRUE` if `Sys.Date()` is less than or equal to the deadline stored in `.pkg_env$deadline`; `FALSE` otherwise.

Sexpr stage and results showcase

Every `Sexpr` variant in one place. `stage` controls *when* the code runs; `results` controls *how* the value is inserted.

build [evaluated at build stage]

install (default stage) R 4.6.1

render 2026-10-12

`results=verbatim` prints as if at the console:

```
[1] 2
```

`echo=TRUE` echoes the source too:

```
>  x <- 40 + 2; x
[1] 42
```

`results=rd` treats the returned string as Rd markup and splices it in place: *(this fragment was generated and re-parsed as Rd)*

`results=hide` evaluates for its side effect and inserts nothing:

Note

Output-format conditionals choose content per renderer. LaTeX output is shown here. A raw passthrough for HTML only: .

References

R Core Team. *Writing R Extensions*. https://cran.r-project.org/doc/manuals/R-exts.html. Murdoch, D. (2010). Parsing Rd files.

See Also

Sys.Date for the system date, as.Date for date parsing, and this page for a self link.

Examples

```
# Check whether the submission window is currently open.
submit_proposal()

## Not run:
# Not run during checks (e.g. needs interaction or a network):
browseURL("https://summerofcode.withgoogle.com")

## End(Not run)


# Run interactively but skipped on CRAN's timing-sensitive checks:
Sys.sleep(0)
```

`walk_rdb_tree` *Walk the compiled Rd_db tree of a help topic*

Description

Pretty-prints the parse tree of a help topic as stored in the *installed* `.rdb` database (via Rd_db), one node per line, indented by depth. Unlike walking the source `.Rd` with parse_Rd, this shows the post-install tree: build- and install-stage `Sexpr` are baked to plain text, `#ifdef` blocks are resolved, render-stage `Sexpr` stay live (with their `Rd_option`), and user macros keep a `USERMACRO` trace.

Usage

```
walk_rdb_tree(topic, package = "gsocproposal")
```

Arguments

`topic`	Help topic to inspect, given **unquoted** (e.g. `walk_rdb_tree(sexpr_render)`) or as a string (`walk_rdb_tree("sexpr_render")`). A trailing `.Rd` is optional.
`package`	Installed package to read from. Defaults to `"gsocproposal"`.

Value

Invisibly, the `Rd` object for `topic`; called for the side effect of printing the tree.

Examples

```
walk_rdb_tree(sexpr_render)
walk_rdb_tree(ifdef)
```

Index

∗ **Sexpr**
    submit_proposal, 2
∗ **documentation**
    submit_proposal, 2
∗ **dynamic documentation**
    submit_proposal, 2
∗ **utilities**
    submit_proposal, 2

as.Date, *4*

gsocproposal-macros *(*submit_proposal*)*, 2

parse_Rd, *4*
print, *2*

Rd_db, *4*

Sexpr, *3*
submit_proposal, 2
Sys.Date, *4*

walk_rdb_tree, 4