Package 'cargo'

July 2, 2022

Title Develop R Packages using Rust

Version 0.2.15

Description A framework is provided to develop R packages using 'Rust' <https://www.rust-lang.org/> with minimal overhead, and more wrappers are easily added. Help is provided to run 'Cargo' <https://doc.rust-lang.org/cargo/> in a manner consistent with CRAN policies. Rust code can also be embedded directly in an R script. The package is not official, affiliated with, nor endorsed by the Rust project.

URL https://github.com/dbdahl/cargo-framework (repository), https://raw.githubusercontent.com/dbdahl/cargo-framework/main/cargo/inst/doc/Writing_R_Extensions_in_Rust.pdf (paper)

BugReports https://github.com/dbdahl/cargo-framework/issues

License MIT + file LICENSE | Apache License 2.0

Depends R (>= 4.0.0)

Suggests roxygen2 (>= 7.2.0), testthat (>= 3.1.4)

SystemRequirements Cargo [Rust package manager]

Encoding UTF-8

RoxygenNote 7.2.0

Config/testthat/edition 3

NeedsCompilation no

Author David B. Dahl [aut, cre] (<https://orcid.org/0000-0002-8173-1547>)

Maintainer David B. Dahl <dahl@stat.byu.edu>

Repository CRAN

Date/Publication 2022-07-01 22:20:02 UTC

R topics documented:

`api_documentation` *Browse API Documentation*

Description

This function opens in a web browser the documentation of the API for the Cargo Framework.

Usage

```
api_documentation(pkgroot = ".")
```

Arguments

`pkgroot`	The root directory of the package.

Value

`NULL`, invisibly.

`install` *Install Rust Toolchain*

Description

This function downloads the '`rustup`' installer, run it, and adds targets to compile for all the CRAN build machines.

Usage

```
install(force = FALSE)
```

Arguments

`force`	If `TRUE`, installation proceeds without asking for user confirmation.

Value

Invisibly, TRUE if successful and FALSE otherwise.

---

new_package *Make a Skeleton for a New Package*

Description

A new Rust-based package using the cargo framework is created at the supplied path and the package is installed.

Usage

```
new_package(path, ...)
```

Arguments

path	A path where the package is created. The name of the package is taken as the last element in the file path.
...	Extra arguments that are currently ignored.

---

prebuild *Prepare for Building the Package Source*

Description

This function generates documentation and/or Rust registration code, depending on the value of what.

Usage

```
prebuild(
  pkgroot = ".",
  what = c("register_calls", "documentation", "vendor")[1:2]
)
```

Arguments

pkgroot	The root directory of the package.
what	A character vector indicating the desired action. If it contains "register_calls", the function (re)generates Rust code. If it contains "documentation", the function (re)generates documentation. If it contains "vendor", the Rust dependencies are (re)vendored.

Details

If a package's usage of `base::.Call()` changes, rerun this function to update the `src/rust/src/registration.rs` file.

Likewise, if a package's documentation changes, run this function to generate documentation using `roxygen2::roxygenise()` and then automatically move it from `man` to `man`. When the source package is installed, examples in the documentation are adjusted according based on whether the Rust can be compiled at that time. Lines in the examples that start with `#' # R_CARGO` are deleted if Rust can be compiled, otherwise, the line is preserved (with `# R_CARGO` itself removed).

Value

`NULL`, invisibly.

`run` *Run Cargo*

Description

This function runs Cargo (Rust's package manager) with the `...` arguments passed as command line arguments.

Usage

```
run(
  ...,
  minimum_version = ".",
  methods = c("envir", "path", "cache"),
  environment_variables = list(),
  rustflags = NULL,
  use_packageStartupMessage = FALSE,
  must_be_silent = FALSE,
  no_prompting = FALSE,
  stdout = "",
  stderr = ""
)
```

Arguments

`...`	Character vector of command line arguments passed to the `cargo` command.
`minimum_version`	A character string representing the minimum version of Rust that is needed. Or a path to the root of a package (i.e., the directory containing the DESCRIPTION file), in which case the value is found from the field: `SystemRequirements: Cargo (>= XXXX)`.

`methods` A character vector potentially containing values `"envir"`, `"path"`, and `"cache"`. This indicates the methods to use (and their order) when searching for a suitable Cargo installation. `"envir"` indicates to try to use the values of the `CARGO_HOME` and `RUSTUP_HOME` environment variables. `"path"` indicates to try to use the directories `.cargo` and `.rustup` in the user's home directory. `"cache"` indicates to try to use the directories from the cargo package's own installation as given by the `tools::R_user_dir('cargo', 'cache')`.

`environment_variables` A named character vector providing environment variables which should be temporarily set while running Cargo. Note that `RUSTUP_HOME` and `CARGO_HOME` are automatically set by this function.

`rustflags` A character vector from which the `CARGO_ENCODED_RUSTFLAGS` environment variables is constructed and then temporarily set. Or, if `NULL`, this environment variable is left unchanged.

`use_packageStartupMessage` Should essential messages be displayed using `base::packageStartupMessage()`?

`must_be_silent` Should all messages be suppressed (regardless of the value of `use_packageStartupMessage`)?

`no_prompting` Prohibit prompting the user?

`stdout` See argument of the same name in `base::system2()`.

`stderr` See argument of the same name in `base::system2()`.

Value

The same value and behavior as the `base::system2()` function, except a non-zero exit code will be given in Cargo is not found.

Examples

```
if ( run("--version", must_be_silent=TRUE) != 0 ) {
   message("Cargo is not installed. Please run cargo::install() in an interactive session.")
}
```

`rust_fn` *Define an R Function Implemented in Rust*

Description

This function takes Rust code as a string from the last unnamed argument, takes variable names for all other unnamed arguments, compiles the Rust function, and wraps it as an R function.

Usage

```
rust_fn(
  ...,
  dependencies = character(0),
  minimum_version = "1.31.0",
  verbose = FALSE,
  cached = TRUE,
  longjmp = TRUE,
  invisible = FALSE
)
```

Arguments

`...`	Rust code is taken as a string from the last unnamed argument, and variable names come for all other unnamed arguments. See example.
`dependencies`	A character vector of crate dependencies, e.g., `c('rand = "0.8.5"','rand_pcg = "0.3.1"')`.
`minimum_version`	A character string representing the minimum version of Rust that is needed. Or a path to the root of a package (i.e., the directory containing the DESCRIPTION file), in which case the value is found from the field: `SystemRequirements: Cargo (>= XXXX)`.
`verbose`	If `TRUE`, Cargo prints compilation details. If `FALSE`, Cargo is run in quiet mode, except for the first time this function is run. If `"never"`, Cargo is always run in quiet mode. In any case, errors in code are always shown.
`cached`	Should Cargo use previously compiled artifacts?
`longjmp`	Should the compiled function use the faster (but experimental) longjmp functionality when Rust code panics?
`invisible`	Should the compiled function return values invisibly?

Value

An R function implemented with the supplied Rust code.

`shlib_get` *Cache a Shared Library*

Description

This function retrieves the path to a cached shared library developed with the Cargo framework.

Usage

```
shlib_get(pkgname)
```

Arguments

pkgname	A string giving the name of the package.

Value

A string giving the path to the cached shared library, or NULL if the library is not cached.

See Also

shlib_set

Examples

```
shlib_get("my_package")
```

---

shlib_set *Cache a Shared Library*

Description

This function caches a shared library developed with the Cargo framework.

Usage

```
shlib_set(
  pkgname,
  path,
  force = FALSE,
  use_packageStartupMessage = FALSE,
  no_prompting = FALSE
)
```

Arguments

pkgname	A string giving the name of the package.
path	A string giving the path to the shared library.
force	If TRUE, installation proceeds without asking for user confirmation.
use_packageStartupMessage	Should essential messages be displayed using base::packageStartupMessage()?
no_prompting	Prohibit prompting the user?

Value

A logical indicating whether the library was successfully cached.

See Also

shlib_get

Examples

```
shlib_set("my_package","/some/path/to/a/shared_library.so")
```

Index