Package ‘cargo’

August 22, 2021

Title Develop R Packages using Rust

Version 0.1.38

Description A framework is provided to transparently 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.

URL https://github.com/dbdahl/cargo-framework (repository),
https://arxiv.org/pdf/2108.07179.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 (>= 7.1.1), testthat (>= 3.0.4)

Encoding UTF-8

RoxygenNote 7.1.1

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 2021-08-22 04:40:02 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.

`cross_compile` *Cross Compile Static Library for CRAN*

Description

This function cross compiles the Rust static library for CRAN's target platforms. The package developer can then uploaded these to a web server. Then, if a particular CRAN build machine does not have a sufficient installation of the Rust toolchain, the package's '`tools/staticlib.R`' script can download the appropriate static library.

Usage

```
cross_compile(
  destination_directory,
  pkgroot = ".",
  target = "CRAN",
  minimum_version = file.path(pkgroot, "DESCRIPTION"),
  verbose = TRUE
)
```

Arguments

`destination_directory`	An existing directory where the static libraries should be added.
`pkgroot`	The root directory of the package.
`target`	A character vector giving Rust targets (e.g. `"x86_64-pc-windows-gnu"`). The value `"CRAN"` is replaced by all targets for CRAN build machines.
`minimum_version`	A character string representing the minimum version of Rust that is needed. Or a path to the DESCRIPTION file, in which case the value is found from the field: `SystemRequirements: Cargo (>= XXXX)`.
`verbose`	Should Cargo be run in non-quiet mode?

Value

`NULL`, invisibly.

See Also

target

`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.

`register_calls` *Generate Rust Code to Register Rust Functions*

Description

This function generates Rust code to register Rust functions accessed in R through .Call(). If a package's usage of .Call() functions changes, rerun this function to update the `src/rustlib/src/registration.rs` file.

Usage

```
register_calls(pkgroot = ".")
```

Arguments

`pkgroot`	The root directory of the package.

Value

`NULL`, invisibly.

`run` *Run Cargo*

Description

This function finds and runs Cargo (Rust's package manager) with the `...` arguments passed as command line arguments but, by default, runs according to CRAN policies. First, it does not write to the user's file system (e.g., `~/.cargo`). Second, it only uses at most two parallel jobs.

Usage

```
run(..., minimum_version = file.path("..", "DESCRIPTION"), verbose = TRUE)
```

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 DESCRIPTION file, in which case the value is found from the field: `SystemRequirements: Cargo (>= XXXX)`.
`verbose`	Should Cargo be run in non-quiet mode?

Details

To enable caching, set the `R_CARGO_SAVE_CACHE` environment variable to `TRUE`. Then, if defined, the `R_CARGO_HOME` environment variable will be used as the cache location. Otherwise, Cargo uses its default behavior (usually writing to `~/.cargo` unless the `CARGO_HOME` environment variable is set). Regardless of the location, the user is responsible to maintaining and clearing the cache when using the `R_CARGO_SAVE_CACHE` environment variable.

To enable a specific number of parallel jobs, set the `R_CARGO_BUILD_JOBS` environment variable to the desired integer. If `R_CARGO_BUILD_JOBS` is `0`, Cargo will use its default behavior (usually using all the cores unless the `CARGO_BUILD_JOBS` environment variable is set or the `--jobs` argument is provided).

Value

A logical equaling `TRUE` if and only if the minimum version is available and the exit status of the command is zero (indicating success). The function is designed to never throw a warning or error.

See Also

`base::Sys.setenv()`

Examples

```
run(minimum_version="1.54")
```

`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 = "0.8.4"','rand_pcg = "0.3.1"')`.
`minimum_version`	A character string representing the minimum version of Rust that is needed. Or a path to 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.

`setup_rust` *Setup Rust Toolchain*

Description

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

Usage

```
setup_rust(force = FALSE)
```

Arguments

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

Value

Invisibly, `TRUE` if successful and `FALSE` otherwise.

`target` *Determine the Rust Build Target*

Description

This function tries to determine the appropriate Rust target for this instance of R. Or, it gives the targets necessary for CRAN build machines.

Usage

```
target(cran = FALSE)
```

Arguments

`cran`	Are targets for all CRAN build machines desired?

Value

If `cran=FALSE`, a string giving a Rust target, or `""` if this cannot be determined. If `cran=TRUE`, a character vector giving the targets necessary for CRAN build machines.

See Also

cross_compile

Examples

```
target()
```

Index