Package: hellorust.multiversion (via r-universe)

May 28, 2026

Type Package

Title Minimal Rust SIMD Multiversioning Example for R

Version 1.3.0

Description Template R package with minimal setup to use Rust code in R without hacks or frameworks, plus a Rust SIMD example selected by C runtime dispatch. Includes basic examples of importing cargo dependencies, spawning threads and passing numbers or strings from Rust to R. The SIMD demo keeps scalar and optional SSE2, AVX2, AVX-512, or NEON kernels in one Rust static library and lets C choose the best Rust function using Rust-reported compiled and CPU features. Cargo crates are automatically 'vendored' in the R source package to support offline installation. This project was first presented at 'Erum2018' to showcase R-Rust integration <https://jeroen.github.io/erum2018/>; for a real world use-case, see the 'gifski' package on 'CRAN'.

License MIT + file LICENSE

URL https://github.com/sounkou-bioinfo/hellorust-multiversion

BugReports https://github.com/sounkou-bioinfo/hellorust-multiversion/issues

SystemRequirements Cargo (Rust's package manager), rustc

Suggests bench, tinytest

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/pak/sysreqs libclang-dev

Repository https://sounkou-bioinfo.r-universe.dev

Date/Publication 2026-05-28 17:37:04 UTC

RemoteUrl https://github.com/sounkou-bioinfo/hellorust-multiversion

RemoteRef HEAD

RemoteSha 7b499277219c381fd7d05c21cc1758134d0a5c23

Contents

`hello` *Hello Rust!*

Description

Minimal examples of calling rust functions in R via C.

Usage

```
hello()

random()

runthreads()
```

Details

These functions call out to rust functions defined in the `myrustlib` cargo crate which is embedded in this package. They return values generated in Rust, such as a UTF-8 string or random number. In addition, `runthreads` is an example of a multi-threaded rust function.

Value

a value generated in Rust (a string, random number, and NULL respectively).

Examples

```
hello()
random()
runthreads()
```

simd_count_nonzero *Count non-zero bytes with Rust SIMD kernels and C dispatch*

Description

This is a Rust/C/R multiversioning demo. The SIMD kernels and CPU feature probes live in Rust, while the C layer only caches a function pointer and dispatches to the best Rust backend whose compiled feature and runtime CPU feature are both available. The package still builds one R shared library; it does not create one shared library per SIMD backend.

`"auto"` selects the fastest available Rust backend from the intersection of compiled features and runtime CPU features. `"scalar"` is always available and is useful for correctness checks and scalar-vs-SIMD benchmarks.

Usage

```
simd_count_nonzero(x)

simd_set_backend(
  backend = c("auto", "scalar", "sse2", "avx2", "avx512", "neon")
)

simd_info()
```

Arguments

x	A raw vector.
backend	One of `"auto"`, `"scalar"`, `"sse2"`, `"avx2"`, `"avx512"`, or `"neon"`.

Value

The number of non-zero bytes in `x`.

The selected backend name, invisibly.

A named list with character-vector `compiled` and `supported` backend sets, the runtime `selected` backend, and logical fields for Rust-reported compiled and CPU features.

Examples

```
simd_count_nonzero(as.raw(c(0, 1, 2, 0, 3)))
simd_set_backend("scalar")
simd_set_backend("auto")
simd_info()
```

Index