Package: RsimdDispatch (via r-universe)

May 31, 2026

Title Runtime 'SIMD' Dispatch Templates for 'C' Code in 'R' Packages

Version 0.1.2.9001

Description Provides templates and a working example for runtime Single Instruction Multiple Data ('SIMD') dispatch in 'C' code used by 'R' packages. Packages can stage scalar and architecture-specific kernel objects during configuration, then select compiled and CPU-supported implementations at runtime through guarded operation tables. The package also vendors the header-only 'SIMDe' library for downstream packages through the 'LinkingTo' field.

License GPL (>= 2)

Copyright See inst/AUTHORS and inst/LICENCE.note for bundled SIMDe authorship and licensing details.

SystemRequirements GNU make

Suggests bench, knitr, rmarkdown, tinytest

VignetteBuilder knitr

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

URL https://github.com/sounkou-bioinfo/RsimdDispatch, https://sounkou-bioinfo.github.io/RsimdDispatch/

BugReports https://github.com/sounkou-bioinfo/RsimdDispatch/issues

Config/pak/sysreqs make

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

Date/Publication 2026-05-31 18:35:08 UTC

RemoteUrl https://github.com/sounkou-bioinfo/RsimdDispatch

RemoteRef HEAD

RemoteSha 855851c6b68c89ed0d267891066359cec73208a6

Contents

`convolve1d` *Full one-dimensional convolution with the selected SIMD backend*

Description

Demonstration numeric kernel for the runtime dispatch template. `convolve1d()` computes the same full convolution as a simple nested-loop R definition. For each pair of positions it adds `a[i] * b[j]` to `out[i + j - 1]`. SIMD backends vectorize the inner multiply-add over `b` and the shifted output window.

Usage

```
convolve1d(a, b)
```

Arguments

`a, b`	Numeric vectors.

Value

A numeric vector of length `length(a) + length(b) - 1`, or `numeric(0)` when either input is empty.

Examples

```
convolve1d(c(1, 2, 3), c(10, 100))
```

count_nonzero *Count non-zero bytes with the selected SIMD backend*

Description

Demonstration kernel for the runtime dispatch template. `count_nonzero()` counts bytes that are not `00` in a raw vector using the currently selected backend. The default backend is `"auto"`, which selects the best compiled backend supported by the current CPU/runtime.

Usage

```
count_nonzero(x)
```

Arguments

`x`	A raw vector.

Value

A `double` scalar giving the count of non-zero bytes. The return type is `double` rather than `integer` to accommodate vectors longer than `.Machine$integer.max` without overflow.

Examples

```
count_nonzero(as.raw(c(0, 1, 0, 2)))
```

simd_backend *Report the currently selected SIMD backend*

Description

Report the currently selected SIMD backend

Usage

```
simd_backend()
```

Value

A character scalar naming the selected backend summary.

Examples

```
simd_backend()
```

simd_dispatch_template_path *Configure an R package for C runtime SIMD dispatch*

Description

`use_simd_dispatch()` copies the dispatch scaffold into an R package and performs the package-name and C-prefix substitutions needed for a working package. It writes package files, updates `DESCRIPTION`, `.Rbuildignore`, and `.gitignore`, and returns the copied paths invisibly.

Usage

```
simd_dispatch_template_path()

use_simd_dispatch(
  path = ".",
  pkg = NULL,
  prefix = NULL,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

`path`	Package root where the template should be copied.
`pkg`	R package name. If `NULL`, the name is read from `DESCRIPTION`.
`prefix`	C symbol prefix used to replace `sd_` in the copied sources. The default is a sanitized lowercase package name.
`overwrite`	Whether to overwrite existing files.
`quiet`	Whether to suppress progress messages.

Value

Invisibly returns copied file paths.

Developer utility

This function is intended for package authors. It is not needed at runtime by users of packages that already include generated dispatch code.

Examples

```
simd_dispatch_template_path()
```

simd_info *Report runtime SIMD dispatch diagnostics*

Description

Returns the requested backend, selected backend, compiled backends, CPU-supported backends, operation-level backend availability, operation-level selected backends, SIMDe-native backends, and target architecture information. Calling this initializes the lazy auto-dispatch selection if it has not already been initialized. For SIMDe provenance (version, commit, date) use `simde_info()`.

Usage

```
simd_info()
```

Value

A named list with the following elements:

`dispatch_mode` Character scalar. Internal dispatch implementation strategy string (stable across patch releases).

`requested_backend` Character scalar. The last value passed to `simd_set_backend()`, or `"auto"` if not explicitly set.

`selected_backend` Character scalar. Summary of the currently active backend: a backend name, `"mixed"`, or `"unavailable"`.

`compiled_backends` Character vector. Backends compiled into the shared library (not necessarily supported by the current CPU).

`cpu_supported_backends` Character vector. Backends whose required CPU features are present at runtime.

`available_backends` Character vector. Backends that are both compiled and CPU-supported (i.e. usable).

`simde_native_backends` Character vector. Backends for which the compiler emitted native intrinsics rather than SIMDe emulation.

`operations` Character vector. Names of registered operations (e.g. `"count_nonzero"`, `"convolve1d"`).

`operation_backends` Named list of character vectors. For each operation, the backends that provide a kernel for it.

`operation_selected_backends` Named character vector. For each operation, the backend currently selected for dispatch (or `NA` if none is resolved).

`cpu_sse2` Logical scalar. `TRUE` if SSE2 is detected at runtime.

`cpu_sse41` Logical scalar. `TRUE` if SSE4.1 is detected.

`cpu_avx2` Logical scalar. `TRUE` if AVX2 is detected.

`cpu_avx512` Logical scalar. `TRUE` if AVX-512 (F/BW/VL) is detected.

`cpu_neon` Logical scalar. `TRUE` if NEON/AdvSIMD is detected.

`cpu_wasm_simd128` Logical scalar. `TRUE` if WASM SIMD128 is available.

`target_arch` Character scalar. Compiler target CPU architecture (e.g. `"x86_64"`, `"aarch64"`).

`target_os` Character scalar. Compiler target OS family (e.g. `"linux"`, `"macos"`, `"windows"`, `"emscripten"`).

Examples

```
names(simd_info())
```

`simd_set_backend` *Select the runtime SIMD backend*

Description

Select the backend used by subsequent calls to dispatched demo kernels such as `count_nonzero()` and `convolve1d()`. `RsimdDispatch` keeps all compiled variants in one shared object and switches a guarded resolved operation table. This makes same-process benchmarking possible.

Usage

```
simd_set_backend(backend = "auto")
```

Arguments

`backend`	Character scalar. Use `"auto"` to select the best available backend, or one of `simd_info()$available_backends` for an explicit choice.

Value

The selected backend summary string, invisibly. Possible values:

`"<name>"` All operations resolved to the same named backend.

`"mixed"` Different operations resolved to different backends (only possible under `"auto"`).

`"unavailable"` No operation could be resolved for the requested backend.

Examples

```
old <- simd_backend()
simd_set_backend("scalar")
simd_set_backend("auto")
```

simde_info *Report vendored SIMDe provenance*

Description

simde_info() reports the version, upstream repository, pinned commit, and commit date for the bundled header-only SIMDe library.

Usage

```
simde_info()
```

Value

A named list of character scalars describing the vendored SIMDe copy.

Examples

```
simde_info()[c("version", "commit")]
```

Index