

# Package ‘RcppXsimd’

January 22, 2021

**Type** Package

**Title** Xsimd C++ Header-Only Library Files

**Version** 7.1.6

**Date** 2021-01-05

**Description** This header-only library provides modern, portable C++ wrappers for SIMD intrinsics and parallelized, optimized math implementations (SSE, AVX, NEON, AVX512). By placing this library in this package, we offer an efficient distribution system for Xsimd <<https://github.com/xtensor-stack/xsimd>> for R packages using CRAN.

**License** BSD\_3\_clause + file LICENSE

**Imports** Rcpp (>= 1.0.0)

**LinkingTo** Rcpp

**RoxygenNote** 7.1.1

**Encoding** UTF-8

**Suggests** testthat

**NeedsCompilation** yes

**Author** Marc A. Suchard [aut, cre],  
Andrew J. Holbrook [aut],  
Observational Health Data Sciences and Informatics [cph],  
Johan Mabille [cph, ctb] (author and copyright holder of Xsimd library  
under a BSD-3 license),  
Sylvain Corlay [cph, ctb] (author and copyright holder of Xsimd library  
under a BSD-3 license),  
Alexander J. Lee [cph, ctb] (author and copyright holder of  
FeatureDetector library under a CC0 1.0 license)

**Maintainer** Marc A. Suchard <[msuchard@ucla.edu](mailto:msuchard@ucla.edu)>

**Repository** CRAN

**Date/Publication** 2021-01-21 23:30:10 UTC

## R topics documented:

getAVX512Flags . . . . . 2

getAVXFlags . . . . .	2
getNEONFlags . . . . .	3
getSimdFeatures . . . . .	3
getSSEFlags . . . . .	4
RcppXsimd . . . . .	4
supportsAVX . . . . .	4
supportsAVX512 . . . . .	5
supportsNEON . . . . .	6
supportsSSE . . . . .	7
<b>Index</b>	<b>8</b>

---

getAVX512Flags	<i>Concatenate supported AVX512 compiler flags for system CPU</i>
----------------	---

---

**Description**

Concatenate supported AVX512 compiler flags for system CPU

**Usage**

```
getAVX512Flags()
```

**Value**

String for compiler flags

---

getAVXFlags	<i>Concatenate supported AVX compiler flags for system CPU</i>
-------------	--

---

**Description**

Concatenate supported AVX compiler flags for system CPU

**Usage**

```
getAVXFlags()
```

**Value**

String for compiler flags

---

getNEONFlags	<i>Concatenate supported NEON compiler flags for system CPU</i>
--------------	---

---

**Description**

Concatenate supported NEON compiler flags for system CPU

**Usage**

```
getNEONFlags()
```

**Value**

String for compiler flags

---

getSimdFeatures	<i>Poll OS and CPU for SIMD feature support</i>
-----------------	---

---

**Description**

Execute CPUID to poll operating system and central processing unit for single instruction, multiple data feature support.

**Usage**

```
getSimdFeatures()
```

**Value**

List of operating system (OS) and hardware (HW) feature support; see CPUID Wiki page for flag definitions

**References**

<https://en.wikipedia.org/wiki/CPUID>

---

getSSEFlags	<i>Concatenate supported SSE compiler flags for system CPU</i>
-------------	--

---

**Description**

Concatenate supported SSE compiler flags for system CPU

**Usage**

```
getSSEFlags()
```

**Value**

String for compiler flags

---

RcppXsimd	<i>RcppXsimd: Rcpp wrapper to Xsimd</i>
-----------	---

---

**Description**

The RcppXsimd package wrappers the header-only C++ Xsimd library that provides parallelized math implementations using SIMD

---

supportsAVX	<i>Determine if CPU supports AVX SIMD instructions</i>
-------------	--

---

**Description**

Determine if CPU supports AVX SIMD instructions

**Usage**

```
supportsAVX()
```

**Value**

Boolean

**Examples**

```

## Not run:

if (supportsAVX()) {
  Sys.setenv(PKG_CPPFLAGS = getAVXFlags())
  Rcpp::sourceCpp(verbose = TRUE, code='
    // [[Rcpp::plugins(cpp14)]]
    // [[Rcpp::depends(RcppXsimd)]]

    #include <Rcpp.h>
    #include "xsimd/xsimd.hpp"

    // [[Rcpp::export]]
    void demoAVX() {
      xsimd::batch<double, 4> a(1.0);
      xsimd::batch<double, 4> b(1.0);
      Rcpp::Rcout << a << " + " << b << " = " << (a + b) << std::endl;
    }')
  demoAVX()
} else {
  message("AVX is not supported")
}

## End(Not run)

```

---

 supportsAVX512

*Determine if CPU supports AVX512 SIMD instructions*


---

**Description**

Determine if CPU supports AVX512 SIMD instructions

**Usage**

```
supportsAVX512()
```

**Value**

Boolean

**Examples**

```

## Not run:

if (supportsAVX512()) {
  Sys.setenv(PKG_CPPFLAGS = getAVX512Flags())
  Rcpp::sourceCpp(verbose = TRUE, code='
    // [[Rcpp::plugins(cpp14)]]

```

```

// [[Rcpp::depends(RcppXsimd)]]

#include <Rcpp.h>
#include "xsimd/xsimd.hpp"

// [[Rcpp::export]]
void demoAVX512() {
  xsimd::batch<double, 8> a(1.0);
  xsimd::batch<double, 8> b(1.0);
  Rcpp::Rcout << a << " + " << b << " = " << (a + b) << std::endl;
}'
demoAVX512()
} else {
  message("AVX512 is not supported")
}

## End(Not run)

```

---

 supportsNEON

*Determine if CPU supports NEON SIMD instructions*


---

## Description

Determine if CPU supports NEON SIMD instructions

## Usage

```
supportsNEON()
```

## Value

Boolean

## Examples

```

## Not run:

if (supportsNEON()) {
  Sys.setenv(PKG_CPPFLAGS = getNEONFlags())
  Rcpp::sourceCpp(verbose = TRUE, code='
// [[Rcpp::plugins(cpp14)]]
// [[Rcpp::depends(RcppXsimd)]]

#include <Rcpp.h>
#include "xsimd/xsimd.hpp"

// [[Rcpp::export]]
void demoNEON() {
  xsimd::batch<double, 2> a(1.0);
  xsimd::batch<double, 2> b(1.0);

```

```

    Rcpp::Rcout << a << " + " << b << " = " << (a + b) << std::endl;
  }')
  demoNEON()
} else {
  message("NEON is not supported")
}

## End(Not run)

```

---

 supportsSSE

*Determine if CPU supports SSE SIMD instructions*


---

## Description

Determine if CPU supports SSE SIMD instructions

## Usage

```
supportsSSE()
```

## Value

Boolean

## Examples

```

## Not run:

if (supportsSSE()) {
  Sys.setenv(PKG_CPPFLAGS = getSSEFlags())
  Rcpp::sourceCpp(verbose = TRUE, code='
    // [[Rcpp::plugins(cpp14)]]
    // [[Rcpp::depends(RcppXsimd)]]

    #include <Rcpp.h>
    #include "xsimd/xsimd.hpp"

    // [[Rcpp::export]]
    void demoSSE42() {
      xsimd::batch<double, 2> a(1.0);
      xsimd::batch<double, 2> b(1.0);
      Rcpp::Rcout << a << " + " << b << " = " << (a + b) << std::endl;
    }')
  demoSSE42()
} else {
  message("SSE4.2 is not supported")
}

## End(Not run)

```

# Index

[getAVX512Flags](#), 2  
[getAVXFlags](#), 2  
[getNEONFlags](#), 3  
[getSimdFeatures](#), 3  
[getSSEFlags](#), 4

[RcppXsimd](#), 4

[supportsAVX](#), 4  
[supportsAVX512](#), 5  
[supportsNEON](#), 6  
[supportsSSE](#), 7