

Package ‘strider’

October 14, 2022

Type Package

Title Strided Iterator and Range

Version 1.3

Date 2020-6-1

Description The strided iterator adapts multidimensional buffers to work with the C++ standard library and range-based for-loops. Given a pointer or iterator into a multidimensional data buffer, one can generate an iterator range using `make_strided` to construct strided versions of the standard library's `begin` and `end`. For constructing range-based for-loops, a `strided_range` class is provided. These help authors to avoid integer-based indexing, which in some cases can impede algorithm performance and introduce indexing errors. This library exists primarily to expose the header file to other R projects.

License MIT + file LICENSE

Imports Rcpp (>= 0.12.13)

LinkingTo Rcpp, BH

Suggests knitr, rmarkdown, testthat, microbenchmark, ggplot2, dplyr, covr, BH

VignetteBuilder knitr

URL <https://github.com/thk686/strider>

BugReports <https://github.com/thk686/strider/issues>

SystemRequirements C++11

RoxygenNote 6.1.1

Encoding UTF-8

NeedsCompilation yes

Author Tim Keitt [aut, cre]

Maintainer Tim Keitt <tkeitt@gmail.com>

Repository CRAN

Date/Publication 2020-06-03 06:30:02 UTC

R topics documented:

convolve2	2
row_sums	3

Index	4
--------------	----------

convolve2	<i>Convolve Matrices</i>
-----------	--------------------------

Description

Demonstration of fast matrix convolution C++

Usage

```
convolve2(a, b)
```

Arguments

a	a numeric matrix
b	a numeric matrix

Details

A very efficient matrix convolution implementation that demonstrates the use of the strided pointer and strided range concepts. Performance will be improved if the smaller matrix is given as the second argument.

See Also

[convolve](#)

Examples

```
a = matrix(c(1, 2, 1,
             1, 1, 1), 2, 3, byrow = TRUE)
b = matrix(c(0, 0, 0,
             0, 0, 0,
             0, 1, 0,
             0, 0, 0), 4, 3, byrow = TRUE)
convolve2(a, b)
```

`row_sums`*Fast row sums*

Description

Demonstration of fast row and columns sums in C++

Usage

```
row_sums(x)
```

```
col_sums(x)
```

Arguments

`x` a numeric matrix

Details

A very efficient row summing algorithm that demonstrates the use of the strided pointer concept. The `row_sum` algorithm is roughly twice as fast as [rowSums](#). The `col_sum` algorithm matches [colSums](#) for speed.

See Also

[rowSums](#), [colSums](#)

Examples

```
row_sums(matrix(1:9, 3))  
col_sums(matrix(1:9, 3))
```

Index

`col_sums (row_sums)`, 3

`colSums`, 3

`convolve`, 2

`convolve2`, 2

`row_sums`, 3

`rowSums`, 3