

Package: natcpp (via r-universe)

July 20, 2026

Type Package

Title Fast C++ Primitives for the 'NeuroAnatomy Toolbox'

Version 0.3.1

Description Fast functions implemented in C++ via 'Rcpp' to support the 'NeuroAnatomy Toolbox' ('nat') ecosystem. These functions provide large speed-ups for basic manipulation of neuronal skeletons over pure R functions found in the 'nat' package. The expectation is that end users will not use this package directly, but instead the 'nat' package will automatically use routines from this package when it is available to enable large performance gains.

License GPL (>= 3)

URL <https://github.com/natverse/natcpp>, <https://natverse.org/natcpp/>

BugReports <https://github.com/natverse/natcpp/issues>

Imports Matrix, methods, Rcpp (>= 1.0.6)

Suggests spelling, testthat (>= 3.0.0)

LinkingTo Rcpp, RcppThread

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Gregory Jefferis [aut, cre] (ORCID:
<<https://orcid.org/0000-0002-0587-9355>>)

Maintainer Gregory Jefferis <jefferis@gmail.com>

Repository <https://cran.r-universe.dev>

Date/Publication 2026-07-20 17:10:13 UTC

RemoteUrl <https://github.com/cran/natcpp>

RemoteRef HEAD

RemoteSha 8fbdfb5a5ddbbad379f87b8d8fc26f99206c8ed1

Contents

c_coords21dindex	2
c_EdgeListFromSegList	3
c_ijkpos	3
c_listlengths	4
c_ListofMatrixRows	5
c_seglengths	5
c_sub2ind	6
c_topntail	6
c_weighted_jaccard_dense	7
c_weighted_jaccard_sparse	8
Index	10

c_coords21dindex	<i>Convert physical coordinates to 1d indices into image array</i>
------------------	--

Description

Convert physical coordinates to 1d indices into image array

Usage

c_coords21dindex(xyz, origin, voxdims, dims, clamp = FALSE)

Arguments

xyz	Nx3 matrix or data.frame of physical coordinates
origin	Numeric: 3d coordinates of the origin
voxdims	Numeric: 3 numbers describing the voxel dimensions
dims	Integer dimensions of the 3d image array
clamp	Logical: whether or not to clamp values within the pixel boundaries of the image.

Value

Nx3 integer matrix of pixel coordinates

`c_EdgeListFromSegList` *Turn a segment list into an edgelist suitable for constructing an ngraph*

Description

Turn a segment list into an edgelist suitable for constructing an ngraph

Usage

```
c_EdgeListFromSegList(L)
```

Arguments

`L` a list containing integer vectors from `as.seglist`

Details

It is up to the caller to generate the `seglist`. Note that isolated points will be dropped since they have no edges.

Value

An integer matrix of `N` rows and 2 columns

Examples

```
## Not run:
library(nat)
# make a neuron with multiple subtrees
n=prune_vertices(Cell07PNs[[1]], 48L)
# Must use flatten=T if including all subtrees
sl=as.seglist(n, all = TRUE, flatten = TRUE)
c_EdgeListFromSegList(sl)

## End(Not run)
```

`c_ijkpos` *Convert physical coordinates to pixel coordinates*

Description

Convert physical coordinates to pixel coordinates

Usage

```
c_ijkpos(xyz, origin, voxdims, dims, clamp = FALSE)
```

Arguments

xyz	Nx3 matrix of physical coordinates
origin	Numeric: 3d coordinates of the origin
voxdims	Numeric: 3 numbers describing the voxel dimensions
dims	Integer dimensions of the 3d image array
clamp	Logical: whether or not to clamp values within the pixel boundaries of the image.

Value

Nx3 integer matrix of pixel coordinates

c_listlengths	<i>A simple function to compute the lengths of the elements of an R list</i>
---------------	--

Description

A simple function to compute the lengths of the elements of an R list

Usage

```
c_listlengths(L)
```

Arguments

L	a list
---	--------

Details

This is equivalent to the `base::lengths` however it is much faster for long lists (and somewhat slower for short ones).

Value

An integer vector containing the length of each element of L

c_ListofMatrixRows	<i>Convert a matrix into list of row vectors</i>
--------------------	--

Description

Convert a matrix into list of row vectors

Usage

```
c_ListofMatrixRows(object)
```

Arguments

object An integer, numeric, character or logical matrix of N rows and M columns

Details

Typically this will be for 3D coordinates but there are no limits on row length.

Value

a list containing N vectors of length M corresponding to the rows of object.

Examples

```
## Not run:
library(nat)
xyz=xyzmatrix(Cell07PNs)
mat2list = function(m) {
  um=unname(m)
  lapply(1:nrow(um), function(i) um[i,])
}
bench::mark(rcpp=c_ListofMatrixRows(xyz), r=mat2list(xyz))

## End(Not run)
```

c_seglengths	<i>Compute summed segment lengths or total cable</i>
--------------	--

Description

c_seglengths computes the summed segment length equivalent to `nat::seglengths(sumsegment = T)`

c_total_cable computes the summed total cable for a whole neuron. It's intended use is the `nat::summary.neuron` function.

Usage

```
c_seglengths(sl, x, y, z)
```

```
c_total_cable(sl, x, y, z)
```

Arguments

sl	A seglist with 1-indices into vectors x,y,z
x, y, z	Numeric vectors with 3D coordinate data (which could be columns from a data frame)

c_sub2ind	<i>Find 1D index given n-dimensional indices</i>
-----------	--

Description

Find 1D index given n-dimensional indices

Usage

```
c_sub2ind(dims, indices)
```

Arguments

dims	Integer dimensions of the array (usually 3d)
indices	Nx3 integer matrix of pixel coordinates

Value

numeric vector of linear indices into the array

c_topntail	<i>Find the first and last elements of all vectors in a list</i>
------------	--

Description

c_topntail returns an 2xN matrix containing the start and end of each of the vectors in the input list. Length 0 vectors are ignored, while length 1 vectors are duplicated

For c_topntail_list, a list of the same length as L having the same elements when their length is <=2 or the first and last elements when length>2.

Usage

```
c_topntail(L)
```

```
c_topntail_list(L)
```

Arguments

`L` a list containing integer vectors, typically a `segl` list

Value

For `c_topntail` an integer matrix. For `c_topntail_list` a list.

`c_weighted_jaccard_dense`

Dense weighted Jaccard similarity via C++

Description

Compute the full weighted Jaccard similarity matrix for a [dgCMatrix](#) using an adaptive dense accumulation strategy. For small output matrices, uses a feature-oriented loop; for large outputs, switches to a column-oriented loop for better cache performance.

Usage

```
c_weighted_jaccard_dense(
  x,
  transpose = FALSE,
  threads = 4L,
  triangle = FALSE,
  distance = FALSE
)
```

Arguments

<code>x</code>	A dgCMatrix (sparse column-compressed matrix)
<code>transpose</code>	If <code>FALSE</code> , compare columns; if <code>TRUE</code> , compare rows
<code>threads</code>	Number of threads (default 4). Set to 0 for all cores.
<code>triangle</code>	If <code>TRUE</code> , return only the lower triangle as a flat numeric vector in dist layout. If <code>FALSE</code> (default), return a full square matrix.
<code>distance</code>	If <code>TRUE</code> , return distance ($1 - \text{similarity}$) instead of similarity. Default <code>FALSE</code> .

Value

A dense numeric similarity matrix, or a numeric vector in [dist](#) layout when `triangle = TRUE`.

c_weighted_jaccard_sparse

Sparse weighted Jaccard similarity via C++

Description

Compute the weighted Jaccard similarity matrix for a [dgCMatrix](#), returning a sparse result. Uses `weighted_jaccard_sparse_fill` to compute min-sums only for column (or row) pairs that share at least one non-zero feature, then normalises to similarity. Only the upper triangle is computed, taking advantage of the symmetry of the Jaccard index.

Usage

```
c_weighted_jaccard_sparse(
  x,
  transpose = FALSE,
  display_progress = TRUE,
  threads = 4L,
  triangle = FALSE,
  distance = FALSE
)
```

Arguments

<code>x</code>	A dgCMatrix (sparse column-compressed matrix)
<code>transpose</code>	If FALSE (default), compare columns; if TRUE, compare rows.
<code>display_progress</code>	Whether to show a text progress bar (default TRUE).
<code>threads</code>	Number of threads for parallel computation (default 4). Set to 0 to use all available cores.
<code>triangle</code>	If TRUE, return a symmetric <code>dsCMatrix</code> (upper triangle only). If FALSE (default), return a general dgCMatrix .
<code>distance</code>	If TRUE, return distance (1 - similarity) instead of similarity. Default FALSE. A warning is issued since sparse distance matrices are typically dense.

Value

A sparse similarity (or distance) matrix: `dsCMatrix` when `triangle = TRUE`, [dgCMatrix](#) otherwise.

See Also

[c_weighted_jaccard_dense](#) for the dense equivalent

Examples

```
## Not run:
library(Matrix)
m <- sparseMatrix(i = c(1,2,1,2,3,3), j = c(1,1,2,2,2,3),
                  x = c(4,2,1,3,3,1), dims = c(3,3))
c_weighted_jaccard_sparse(m)

## End(Not run)
```

Index

c_coords21dindex, [2](#)
c_EdgeListFromSegList, [3](#)
c_ijkpos, [3](#)
c_listlengths, [4](#)
c_ListofMatrixRows, [5](#)
c_seglengths, [5](#)
c_sub2ind, [6](#)
c_topntail, [6](#)
c_topntail_list (c_topntail), [6](#)
c_total_cable (c_seglengths), [5](#)
c_weighted_jaccard_dense, [7](#), [8](#)
c_weighted_jaccard_sparse, [8](#)

dgCMatrix, [7](#), [8](#)
dist, [7](#)