

Package ‘natcpp’

September 26, 2026

Type Package

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

Version 0.3.2

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, RcppEigen, RcppThread

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Gregory Jefferis [aut, cre] (ORCID:
<<https://orcid.org/0000-0002-0587-9355>>),
libigl contributors [ctb, cph] (bundled libigl fast winding number code
(MPL-2.0); see src/vendor/README.md)

Maintainer Gregory Jefferis <jefferis@gmail.com>

Repository CRAN

Date/Publication 2026-09-26 07:40:02 UTC

Contents

c_coords21dindex	2
c_EdgeListFromSegList	3
c_ijkpos	3
c_listlengths	4
c_ListofMatrixRows	5
c_pointsinside	5
c_seglengths	7
c_sub2ind	7
c_topntail	8
c_weighted_jaccard_dense	8
c_weighted_jaccard_sparse	9
Index	11

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_pointsinside	<i>Test which points lie inside a triangle mesh (generalised winding number)</i>
----------------	--

Description

Robust point-in-mesh test based on the generalised (solid-angle) winding number. For a closed mesh the winding number is approximately ± 1 for interior points and 0 for exterior points, so $\text{abs}(w) > 0.5$ classifies points as inside. Unlike a closest-point signed-distance test it does not depend on surface normals and has no ray-casting tie-breaking, so it does not produce the spurious "outside point classified as inside" results that normal-based tests can give near thin protrusions or sharp features.

Usage

```
c_pointsinside(
  points,
  vertices,
  faces,
  method = c("auto", "bvh", "bruteforce"),
  threads = NULL,
  accuracy = 2
)
```

Arguments

points	An Nx3 matrix of query point coordinates (or anything coercible with <code>as.matrix</code>).
vertices	An Nx3 matrix of mesh vertex coordinates.
faces	An Nx3 integer matrix of 1-based vertex indices (one triangle per row), e.g. <code>t(mesh\$it)</code> for an rgl mesh3d.
method	Winding-number back end: "auto" (default), "bvh" or "bruteforce". See Details .
threads	Number of threads for parallel computation. The default NULL applies the package thread policy (respecting <code>getOption("Ncpus")</code> and the <code>OMP_THREAD_LIMIT</code> environment variable, else 2). Set to 0 to use all available cores.
accuracy	libigl accuracy-scale parameter for method = "bvh" (default 2); ignored by the brute-force back end.

Details

The mesh should be closed (watertight) and triangular; the result is independent of face orientation (winding). It is intended as the accelerated back end for `nat::pointsinside()`.

Two back ends are available, selected by method:

"bruteforce" A self-contained $O(P \times F)$ implementation (P points, F faces), parallelised over points with **RcppThread**. No setup cost, so it is fastest for small meshes.

"bvh" libigl's "Fast Winding Numbers for Soups and Clouds" (Barill et al. 2018): a bounding-volume hierarchy is built once over the mesh and each query point is then evaluated in $O(\log F)$, so it scales to millions of points on meshes of tens of thousands of faces. `accuracy` tunes the multipole approximation.

"auto" (the default) picks "bvh" only for large meshes queried by enough points to amortise building the hierarchy, and "bruteforce" otherwise (so a simple mesh such as a cuboid always uses brute force). The two back ends agree to within the winding-number tolerance.

Value

A logical vector of length `nrow(points)` (TRUE = inside).

Examples

```
# tetrahedron
V <- rbind(c(0,0,0), c(1,0,0), c(0,1,0), c(0,0,1))
F <- rbind(c(1,3,2), c(1,2,4), c(1,4,3), c(2,3,4))
c_pointsinside(rbind(c(.2,.2,.2), c(2,2,2)), V, F) # TRUE FALSE
```

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

Find the first and last elements of all vectors in a list

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 seglist

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](#), returning a dense matrix (or a [dist](#)-layout vector).

Usage

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

Arguments

x	A dgCMatrix (sparse column-compressed matrix)
transpose	If FALSE, compare columns; if TRUE, compare rows
threads	Number of threads for parallel computation. The default NULL applies the package thread policy (respecting <code>getOption("Ncpus")</code> and the <code>OMP_THREAD_LIMIT</code> environment variable, else 2). Set to 0 to use all available cores.
triangle	If TRUE, return only the lower triangle as a flat numeric vector in dist layout. If FALSE (default), return a full square matrix.
distance	If TRUE, return distance (1 - similarity) instead of similarity. Default FALSE.

Details

Uses an adaptive dense accumulation strategy: for small output matrices a feature-oriented loop, switching to a column-oriented loop for larger outputs for better cache performance.

Value

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

See Also

[c_weighted_jaccard_sparse](#) for the sparse 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_dense(m)

## End(Not run)
```

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 = NULL,
  triangle = FALSE,
  distance = FALSE
)
```

Arguments

x	A dgCMatrix (sparse column-compressed matrix)
transpose	If FALSE (default), compare columns; if TRUE, compare rows.
display_progress	Whether to show a text progress bar (default TRUE).
threads	Number of threads for parallel computation. The default NULL applies the package thread policy (respecting <code>getOption("Ncpus")</code> and the <code>OMP_THREAD_LIMIT</code> environment variable, else 2). Set to 0 to use all available cores.
triangle	If TRUE, return a symmetric <code>dsCMatrix</code> (upper triangle only). If FALSE (default), return a general dgCMatrix .
distance	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_pointsinside, [5](#)
c_seglengths, [7](#)
c_sub2ind, [7](#)
c_topntail, [8](#)
c_topntail_list (c_topntail), [8](#)
c_total_cable (c_seglengths), [7](#)
c_weighted_jaccard_dense, [8](#), [10](#)
c_weighted_jaccard_sparse, [9](#), [9](#)

dgCMatrix, [8–10](#)
dist, [8](#), [9](#)