

Package: RcppHNSW (via r-universe)

October 18, 2026

Title 'Rcpp' Bindings for 'hnsplib', a Library for Approximate Nearest Neighbors

Version 0.7.0.9001

Description 'Hnsplib' is a C++ library for Approximate Nearest Neighbors. This package provides a minimal R interface by relying on the 'Rcpp' package. See <https://github.com/nmslib/hnsplib> for more on 'hnsplib'. 'hnsplib' is released under Version 2.0 of the Apache License.

License GPL (>= 3)

Depends R (>= 3.4.0)

SystemRequirements C++17

URL <https://jlmelville.github.io/rcpphnsw/>,
<https://github.com/jlmelville/rcpphnsw>

BugReports <https://github.com/jlmelville/rcpphnsw/issues>

Imports methods, Rcpp (>= 0.11.3)

Suggests covr, pkgdown, testthat (>= 3.0.0)

LinkingTo Rcpp

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.0

Config/testthat/edition 3

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

Date/Publication 2026-10-18 15:25:46 UTC

RemoteUrl <https://github.com/jlmelville/rcpphnsw>

RemoteRef HEAD

RemoteSha ff0de6573ed911b16b032d12c1c29170f170b4c7

Contents

RcppHnsw-package	2
hnsw_build	3
hnsw_knn	5
hnsw_search	7

Index	9
--------------	----------

RcppHnsw-package	<i>Approximate nearest neighbor search with hnswlib</i>
------------------	---

Description

RcppHNSW provides an R interface to hierarchical navigable small-world graphs. Use `hnsw_knn()` for neighbors within one dataset, or `hnsw_build()` and `hnsw_search()` to query new data against an existing index.

Module classes

HnswEuclidean, HnswL2, HnswCosine, and HnswIp provide direct access to insertion, search, deletion, resizing, and saved indexes. They use Euclidean distance, squared Euclidean distance, one minus cosine similarity, and one minus inner product, respectively. An index returned by `hnsw_build()` is one of these objects, so you can use its methods directly too. See the [Module guide](#) for constructors and methods.

Labels start at one and follow insertion order. Deleted items are excluded from search and retrieval but still count towards `size()` and capacity. If insertion fails after modifying the index, or a native resize fails, discard the index and rebuild or reload it.

Numeric limits

Coordinates are stored as single-precision floats and must be finite and representable in that format. Cosine vectors are normalized to unit length and must have a nonzero norm after conversion.

To prevent distance overflow, each vector's sum of absolute coordinates is limited to approximately $4.6e18$, after conversion and cosine normalization. Loaded checkpoints must meet the same limit, including deleted items.

Saved indexes

`ann$save()` writes hnswlib's raw checkpoint format, whose compatibility depends on the hnswlib version and platform. Load with the original dimension and distance class. You can also load an HnswL2 checkpoint into HnswEuclidean, or vice versa.

Loading restores items, deletion state, and capacity, but resets search ef to 10. Set ef again with `setEf()` or `hnsw_search()`. An optional positive load capacity overrides the saved value when it is at least the stored item count; smaller values retain the saved capacity.

Author(s)

James Melville for the R interface; Yury Malkov for hnswlib itself.

Maintainer: James Melville jlmelville@gmail.com

References

<https://github.com/nmslib/hnswlib>

Malkov, Y. A., & Yashunin, D. A. (2020). Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4), 824-836. doi:10.1109/TPAMI.2018.2889473.

See Also

Useful links:

- <https://jlmelville.github.io/rcpphnsw/>
- <https://github.com/jlmelville/rcpphnsw>
- Report bugs at <https://github.com/jlmelville/rcpphnsw/issues>

hnsw_build

Build a nearest neighbor index

Description

Build an HNSW index that you can keep and query with [hnsw_search\(\)](#).

Usage

```
hnsw_build(  
  X,  
  distance = "euclidean",  
  M = 16,  
  ef = 200,  
  verbose = FALSE,  
  progress = "bar",  
  n_threads = 0,  
  grain_size = 1,  
  byrow = TRUE,  
  random_seed = 100  
)
```

Arguments

<code>X</code>	Numeric matrix to index, with one item per row, or per column when <code>byrow = FALSE</code> .
<code>distance</code>	Distance to use: • "euclidean": Euclidean distance. • "l2": squared Euclidean distance. • "cosine": one minus cosine similarity. • "ip": one minus inner product, $1 - \text{sum}(a * b)$; can be negative.
<code>M</code>	Number of graph links per item during construction. Larger values improve connectivity and use more memory. Must be between 2 and 10000.
<code>ef</code>	Candidate-list size during construction. Larger values improve index quality and increase build time. Must be a positive whole number and is not capped at the dataset size. This is the <code>ef_construction</code> parameter of hnsw_knn() .
<code>verbose</code>	If TRUE, log messages to the console.
<code>progress</code>	Unused; retained for compatibility.
<code>n_threads</code>	Maximum number of threads for batch insertion or search. Zero (the default) and one run serially.
<code>grain_size</code>	Minimum number of items per thread. Larger values limit threading overhead for small batches. Zero is treated as one.
<code>byrow</code>	If TRUE, items are rows of <code>X</code> ; otherwise, they are columns.
<code>random_seed</code>	Seed for hnswlib's index construction. R's <code>set.seed()</code> has no effect. Use serial construction for repeatable builds; parallel insertion order can vary with a fixed seed.

Details

Coordinates are stored in single precision; see [RcppHnsw-package](#) for numeric limits. If you want to add more items, resize, or save the index, see the [Module guide](#) for the available methods.

Value

An `HnswEuclidean`, `HnswL2`, `HnswCosine`, or `HnswIp` index, according to `distance`. Labels are one-based and follow the order in `X`.

See Also

[hnsw_knn\(\)](#) to build and search in one call.

Examples

```

irism <- as.matrix(iris[, -5])
ann <- hnsw_build(irism[1:100, ])
neighbors <- hnsw_search(irism[101:150, ], ann, k = 5)

```

hnsw_knn

*Find approximate nearest neighbors***Description**

Build an HNSW index and find k neighbors for every item in X.

Usage

```
hnsw_knn(
  X,
  k = 10,
  distance = "euclidean",
  M = 16,
  ef_construction = 200,
  ef = 10,
  verbose = FALSE,
  progress = "bar",
  n_threads = 0,
  grain_size = 1,
  byrow = TRUE,
  random_seed = 100
)
```

Arguments

X	Numeric matrix with one item per row, or per column when byrow = FALSE.
k	Number of neighbors per item. A positive whole number no larger than the number of items in X.
distance	Distance to use: • "euclidean": Euclidean distance. • "l2": squared Euclidean distance. • "cosine": one minus cosine similarity. • "ip": one minus inner product, $1 - \text{sum}(a * b)$; can be negative.
M	Number of graph links per item during construction. Larger values improve connectivity and use more memory. Must be between 2 and 10000.
ef_construction	Candidate-list size during construction. Larger values improve index quality and increase build time. Must be a positive whole number; raised to at least k.
ef	Candidate-list size during search. Increase it to improve recall at the cost of search time. Must be a positive whole number; the effective value is at least k and is not capped at the index size.
verbose	If TRUE, log messages to the console.
progress	Unused; retained for compatibility.

<code>n_threads</code>	Maximum number of threads for batch insertion or search. Zero (the default) and one run serially.
<code>grain_size</code>	Minimum number of items per thread. Larger values limit threading overhead for small batches. Zero is treated as one.
<code>byrow</code>	If TRUE, items are rows of X; otherwise, they are columns. Results follow the same orientation.
<code>random_seed</code>	Seed for hnswnn's index construction. R's <code>set.seed()</code> has no effect. Use serial construction for repeatable builds; parallel insertion order can vary with a fixed seed.

Details

If you are searching for neighbors within X, don't assume that the first neighbor is always the item itself: approximate search can miss self-matches. With inner-product distance, an item need not be its own nearest neighbor even in an exact search.

Coordinates are stored in single precision. See [RcppHnsw-package](#) for numeric limits. For help choosing the parameters, see the [parameter guide](#) which also covers data layout.

Value

A list with matrices `idx` (one-based neighbor indices) and `dist` (distances), ordered by increasing distance for each query. Both are $n \times k$ when `byrow = TRUE`, or $k \times n$ otherwise, where n is the number of items in X.

References

Malkov, Y. A., & Yashunin, D. A. (2020). Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4), 824-836. doi:10.1109/TPAMI.2018.2889473.

See Also

[hnswnn_build\(\)](#) and [hnswnn_search\(\)](#) to build once and search new data.

Examples

```
irism <- as.matrix(iris[, -5])
neighbors <- hnswnn(irism, k = 4)
neighbors$idx[1:5, ]
neighbors$dist[1:5, ]
```

hnsf_search	<i>Search a nearest neighbor index</i>
-------------	--

Description

Find neighbors in an existing index for each query in X.

Usage

```
hnsf_search(
  X,
  ann,
  k,
  ef = 10,
  verbose = FALSE,
  progress = "bar",
  n_threads = 0,
  grain_size = 1,
  byrow = TRUE
)
```

Arguments

X	Numeric query matrix with the same number of dimensions as the indexed data. Each row is a query, or each column when byrow = FALSE.
ann	An index from hnsf_build() or a Module constructor.
k	Number of neighbors per query. A positive whole number no larger than the number of undeleted items. <code>ann\$size()</code> includes deleted items.
ef	Candidate-list size during search. Increase it to improve recall at the cost of search time. Must be a positive whole number; the effective value is at least k and is not capped at the index size.
verbose	If TRUE, log messages to the console.
progress	Unused; retained for compatibility.
n_threads	Maximum number of threads for batch insertion or search. Zero (the default) and one run serially.
grain_size	Minimum number of items per thread. Larger values limit threading overhead for small batches. Zero is treated as one.
byrow	If TRUE, items are rows of X; otherwise, they are columns. Results follow the same orientation.

Details

The index's distance measure is used for all queries. Search is approximate; with inner-product distance, an item need not be its own nearest neighbor. Coordinates are stored in single precision; see [RcppHnsf-package](#) for numeric limits.

This call also updates ann's `ef`, `n_threads`, and `grain_size` settings. If you then use the Module methods directly, they will use these settings.

Value

A list with matrices `idx` (one-based labels in ann) and `dist` (distances), ordered by increasing distance for each query. Both are $n \times k$ when `byrow = TRUE`, or $k \times n$ otherwise, where n is the number of queries in X .

See Also

[hnsw_build\(\)](#) to create an index, [hnsw_knn\(\)](#) to build and search in one call.

Examples

```
irism <- as.matrix(iris[, -5])
ann <- hnsw_build(irism[1:100, ])
neighbors <- hnsw_search(irism[101:150, ], ann, k = 5)
```

Index

hnsw_build, [3](#)
hnsw_build(), [2](#), [6–8](#)
hnsw_knn, [5](#)
hnsw_knn(), [2](#), [4](#), [8](#)
hnsw_search, [7](#)
hnsw_search(), [2](#), [3](#), [6](#)
HnswCosine (RcppHnsw-package), [2](#)
HnswEuclidean (RcppHnsw-package), [2](#)
HnswIp (RcppHnsw-package), [2](#)
HnswL2 (RcppHnsw-package), [2](#)

Rcpp_HnswCosine-class
 (RcppHnsw-package), [2](#)
Rcpp_HnswEuclidean-class
 (RcppHnsw-package), [2](#)
Rcpp_HnswIp-class (RcppHnsw-package), [2](#)
Rcpp_HnswL2-class (RcppHnsw-package), [2](#)
RcppHNSW (RcppHnsw-package), [2](#)
RcppHNSW-package (RcppHnsw-package), [2](#)
RcppHnsw-package, [2](#), [4](#), [6](#), [7](#)