

Package ‘jvecfor’

August 14, 2026

Type Package

Title Fast K-Nearest Neighbor Search for Single-Cell Analysis

Version 1.1.0

biocViews SingleCell, GraphAndNetwork, Clustering, Classification

Description Drop-in replacement for BiocNeighbors::findKNN using the jvecfor Java library, which builds on the jvector library to leverage the Java Vector API for portable SIMD acceleration across AVX2, AVX-512, and ARM NEON hardware. jvecfor/jvector implements HNSW-DiskANN approximate search and VP-tree exact search. The package achieves approximately 2x speedup over Annoy-based search at $n \geq 50K$ cells while returning output structurally identical to BiocNeighbors, making it suitable for seamless integration into existing Bioconductor single-cell workflows. Convenience wrappers delegate shared nearest-neighbor (SNN) and k-nearest-neighbor (KNN) graph construction to the bluster package.

License GPL-3

Encoding UTF-8

Language en-US

LazyData false

Depends R ($\geq 4.6.0$)

Imports BiocNeighbors, BiocParallel, Matrix, bluster, data.table, methods, processx

SystemRequirements Java (≥ 20)

RoxygenNote 7.3.3

URL <https://github.com/gkanogiannis/jvecfor>

BugReports <https://github.com/gkanogiannis/jvecfor/issues>

Suggests BiocStyle, igraph, knitr, rmarkdown, testthat ($\geq 3.0.0$)

VignetteBuilder knitr

Config/testthat/edition 3

git_url <https://git.bioconductor.org/packages/jvecfor>

git_branch devel

git_last_commit df5d2ab

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-08-13

Author Anestis Gkanogiannis [aut, cre] (ORCID:
<<https://orcid.org/0000-0002-6441-0688>>)

Maintainer Anestis Gkanogiannis <anestis@gkanogiannis.com>

Contents

jvecfor-package	2
fastFindKNN	3
fastMakeKNNGraph	4
fastMakeSNNGraph	6
JvecforIndex-class	7
JvecforParam-class	7
jvecfor_setup	10
Index	11

jvecfor-package	<i>jvecfor: Fast K-Nearest Neighbor Search for Single-Cell Analysis</i>
-----------------	---

Description

Drop-in replacement for `BiocNeighbors::findKNN` using the jvecfor Java library (HNSW-DiskANN approximate and VP-tree exact methods). Achieves approximately 2x speedup over Annoy-based search at $n \geq 50K$ cells. Convenience wrappers delegate SNN/KNN graph construction to the bluster package.

Main functions

`fastFindKNN` KNN search – returns index + distance matrices.
`fastMakeSNNGraph` KNN -> SNN graph via bluster.
`fastMakeKNNGraph` KNN -> KNN graph via bluster.
`JvecforParam` BiocNeighbors parameter class for drop-in integration with scran, scater, etc.
`jvecfor_setup` Install a custom jvecfor JAR.

Options

`jvecfor.verbose` Logical. Enable Java/jvecfor progress logging globally. Default FALSE.
`jvecfor.jar` Character. Path to a custom jvecfor JAR file. Overrides the bundled JAR in `inst/java/`.

Author(s)

Maintainer: Anestis Gkanogiannis <anestis@gkanogiannis.com> (ORCID)

See Also

Useful links:

- <https://github.com/gkanogiannis/jvecfor>
- Report bugs at <https://github.com/gkanogiannis/jvecfor/issues>

fastFindKNN

*Fast K-Nearest Neighbor Search***Description**

Drop-in replacement for `BiocNeighbors::findKNN` using the `jvecfor` Java library. Supports HNSW-DiskANN approximate search (`type="ann"`) and VP-tree exact search (`type="knn"`).

Usage

```
fastFindKNN(
  X,
  k = 15L,
  type = c("ann", "knn"),
  metric = c("euclidean", "cosine", "dot_product"),
  num.threads = NULL,
  BPPARAM = BiocParallel::bpparam(),
  ef.search = 0L,
  M = 16L,
  oversample.factor = 1,
  pq.subspaces = 0L,
  get.distance = TRUE,
  verbose = getOption("jvecfor.verbose", FALSE)
)
```

Arguments

<code>X</code>	A numeric matrix, data.frame, or sparse matrix (<code>Matrix::dgCMatrix</code>) with rows = observations, cols = features. Sparse matrices are written in MatrixMarket format and densified in the Java backend, avoiding R-side memory allocation of the full dense matrix.
<code>k</code>	Integer. Number of nearest neighbors to find (excluding self). Default 15.
<code>type</code>	Character. "ann" for approximate (HNSW-DiskANN) or "knn" for exact (VP-tree). Default "ann".
<code>metric</code>	Character. Distance metric: "euclidean", "cosine", or "dot_product". Default "euclidean". Note: "dot_product" is only valid when type = "ann"; it is not a proper metric and cannot be used with the VP-tree exact search.
<code>num.threads</code>	Integer or NULL. Number of Java threads. If NULL, defaults to <code>BiocParallel::bpworkers(BPPARAM)</code> .
<code>BPPARAM</code>	A BiocParallelParam object controlling the thread count. Defaults to <code>bpparam()</code> .
<code>ef.search</code>	Integer. HNSW-DiskANN beam width override (0 = auto: $\max(k+1, 3k)$). Only meaningful when type = "ann". Default 0L.
<code>M</code>	Integer. HNSW-DiskANN maximum connections per node. Higher values (e.g. 32) improve recall for high-dimensional data at the cost of more memory and a slower build. Only meaningful when type = "ann". Default 16L.
<code>oversample.factor</code>	Numeric. Oversampling multiplier for the ANN beam width. When > 1.0, fetches <code>ceil(ef * oversample.factor)</code> candidates and returns the top k, improving recall at proportional cost. Only meaningful when type = "ann". Default 1.0.

pq.subspaces	Integer. Number of Product Quantization subspaces for approximate ANN scoring (0 = disabled). Typical value: $\text{ncol}(X) / 2$. Reduces search time approximately 4-8x with minimal recall loss. Only meaningful when <code>type = "ann"</code> . Default 0L.
get.distance	Logical. Return distance matrix alongside index? Default TRUE.
verbose	Logical. Pass <code>--verbose</code> to the Java process, enabling HNSW-DiskANN build progress logging on stderr. Overrides the <code>jvecfor.verbose</code> global option when set explicitly. Default <code>getOption("jvecfor.verbose", FALSE)</code> .

Value

A named list:

index $n \times k$ integer matrix of 1-indexed neighbor indices.

distance $n \times k$ numeric matrix of distances/similarities, or NULL if `get.distance=FALSE`.

Examples

```
set.seed(42)
X <- matrix(rnorm(200), nrow = 20, ncol = 10)

# Full examples require Java >= 20 on PATH
nn <- fastFindKNN(X, k = 3)
dim(nn$index)    # 20 x 3
dim(nn$distance) # 20 x 3

# High-recall HNSW-DiskANN with wider beam and more connections
nn2 <- fastFindKNN(X, k = 3, M = 32, ef.search = 100,
                   oversample.factor = 2.0)

# Dot-product similarity (ANN only)
nn3 <- fastFindKNN(X, k = 3, metric = "dot_product")
```

fastMakeKNNGraph

Build a K-Nearest Neighbor Graph

Description

Convenience wrapper that calls `fastFindKNN` then delegates graph construction to `bluster::neighborsToKNNGraph`.

Usage

```
fastMakeKNNGraph(
  X,
  k = 15L,
  type = c("ann", "knn"),
  metric = c("euclidean", "cosine", "dot_product"),
  num.threads = NULL,
  BPPARAM = BiocParallel::bpparam(),
  ef.search = 0L,
  M = 16L,
```

```

oversample.factor = 1,
pq.subspaces = 0L,
verbose = getOption("jvecfor.verbose", FALSE),
directed = FALSE,
...
)

```

Arguments

<code>X</code>	A numeric matrix, data.frame, or sparse matrix (<code>Matrix::dgCMatrix</code>) with rows = cells, cols = features/PCs.
<code>k</code>	Integer. Number of nearest neighbors. Default 15.
<code>type</code>	Character. "ann" or "knn". Default "ann".
<code>metric</code>	Character. "euclidean", "cosine", or "dot_product". Default "euclidean". See fastFindKNN for the dot_product restriction.
<code>num.threads</code>	Integer or NULL. Number of Java threads. If NULL, defaults to <code>BiocParallel::bpworkers(BPPARAM)</code> .
<code>BPPARAM</code>	A BiocParallelParam object controlling the thread count. Defaults to <code>bpparam()</code> .
<code>ef.search</code>	Integer. HNSW-DiskANN beam width override (0 = auto). Default 0L.
<code>M</code>	Integer. HNSW-DiskANN max connections per node. Default 16L.
<code>oversample.factor</code>	Numeric. Oversampling multiplier. Default 1.0.
<code>pq.subspaces</code>	Integer. PQ subspaces (0 = disabled). Default 0L.
<code>verbose</code>	Logical. Enable Java verbose logging. Default <code>getOption("jvecfor.verbose", FALSE)</code> .
<code>directed</code>	Logical. Build directed graph? Default FALSE.
<code>...</code>	Additional arguments forwarded to <code>bluster::neighborsToKNNGraph</code> .

Value

An `igraph` object (KNN graph).

Examples

```

set.seed(42)
X <- matrix(rnorm(5000), nrow = 100, ncol = 50)

# Full examples require Java >= 20 on PATH
g <- fastMakeKNNGraph(X, k = 10)
igraph::vcount(g) # 100

```

fastMakeSNNGraph

*Build a Shared Nearest-Neighbor Graph***Description**

Convenience wrapper that calls `fastFindKNN` then delegates graph construction to `bluster::neighborsToSNNGraph`.

Usage

```
fastMakeSNNGraph(
  X,
  k = 15L,
  type = c("ann", "knn"),
  metric = c("euclidean", "cosine", "dot_product"),
  num.threads = NULL,
  BPPARAM = BiocParallel::bpparam(),
  ef.search = 0L,
  M = 16L,
  oversample.factor = 1,
  pq.subspaces = 0L,
  verbose = getOption("jvecfor.verbose", FALSE),
  snn.type = "rank",
  ...
)
```

Arguments

<code>X</code>	A numeric matrix, data.frame, or sparse matrix (<code>Matrix::dgCMatrix</code>) with <code>rows = cells</code> , <code>cols = features/PCs</code> .
<code>k</code>	Integer. Number of nearest neighbors. Default 15.
<code>type</code>	Character. "ann" or "knn". Default "ann".
<code>metric</code>	Character. "euclidean", "cosine", or "dot_product". Default "euclidean". See fastFindKNN for the dot_product restriction.
<code>num.threads</code>	Integer or NULL. Number of Java threads. If NULL, defaults to <code>BiocParallel::bpworkers(BPPARAM)</code> .
<code>BPPARAM</code>	A BiocParallelParam object controlling the thread count. Defaults to <code>bpparam()</code> .
<code>ef.search</code>	Integer. HNSW-DiskANN beam width override (0 = auto). Default 0L.
<code>M</code>	Integer. HNSW-DiskANN max connections per node. Default 16L.
<code>oversample.factor</code>	Numeric. Oversampling multiplier. Default 1.0.
<code>pq.subspaces</code>	Integer. PQ subspaces (0 = disabled). Default 0L.
<code>verbose</code>	Logical. Enable Java verbose logging. Default <code>getOption("jvecfor.verbose", FALSE)</code> .
<code>snn.type</code>	Character. SNN weighting scheme passed to <code>bluster::neighborsToSNNGraph</code> : "rank", "jaccard", or "number". Default "rank".
<code>...</code>	Additional arguments forwarded to <code>bluster::neighborsToSNNGraph</code> .

Value

An igraph object (weighted, undirected SNN graph).

Examples

```
set.seed(42)
X <- matrix(rnorm(5000), nrow = 100, ncol = 50)

# Full examples require Java >= 20 on PATH
g <- fastMakeSNNGraph(X, k = 10)
igraph::vcount(g) # 100

# Higher recall with larger beam and more connections
g2 <- fastMakeSNNGraph(X, k = 10, M = 32, oversample.factor = 2.0)
```

JvecforIndex-class	<i>JvecforIndex: BiocNeighbors Index Class for jvecfor</i>
--------------------	--

Description

A [BiocNeighborIndex](#) subclass storing the data matrix and JvecforParam parameters. The actual Java HNSW/VP-tree index is built on-the-fly when [findKNN](#) is called.

Slots

data Numeric matrix (rows = observations, cols = features).

param A [JvecforParam](#) object.

names Row names from the original matrix, or NULL.

See Also

[JvecforParam](#)

JvecforParam-class	<i>JvecforParam: BiocNeighbors Parameter Class for jvecfor</i>
--------------------	--

Description

A [BiocNeighborParam](#) subclass for the jvecfor Java backend. Passing a JvecforParam object as the BNPARAM argument to [findKNN](#) or higher-level functions (e.g. `scan::buildSNNGraph`, `scater::runUMAP`) routes neighbor search through jvecfor's HNSW-DiskANN or VP-tree engine.

Usage

```

JvecforParam(
  type = "ann",
  distance = "Euclidean",
  M = 16L,
  ef.search = 0L,
  oversample.factor = 1,
  pq.subspaces = 0L,
  verbose = FALSE
)

## S4 method for signature 'JvecforParam'
show(object)

## S4 method for signature 'JvecforParam'
buildIndex(X, BNPARAM, transposed = FALSE, ...)

## S4 method for signature 'JvecforIndex'
findKnnFromIndex(
  BNINDEX,
  k,
  get.index = TRUE,
  get.distance = TRUE,
  num.threads = 1,
  subset = NULL,
  ...
)

```

Arguments

type	Character. "ann" (default) or "knn".
distance	Character. "Euclidean" (default) or "Cosine".
M	Integer. HNSW max connections per node. Default 16L.
ef.search	Integer. HNSW beam width (0 = auto). Default 0L.
oversample.factor	Numeric. Oversampling multiplier. Default 1.0.
pq.subspaces	Integer. PQ subspaces (0 = disabled). Default 0L.
verbose	Logical. Java progress logging. Default FALSE.
object	A JvecforParam object.
X	A numeric matrix (rows = observations, cols = features).
BNPARAM	A JvecforParam object.
transposed	Logical. If TRUE, X is features-by-obs and will be transposed. Default FALSE.
...	Ignored.
BNINDEX	A JvecforIndex object.
k	Integer. Number of nearest neighbors.
get.index	Logical. Return index matrix? Default TRUE.
get.distance	Logical. Return distance matrix? Default TRUE.

<code>num.threads</code>	Integer. Thread count. Default 1.
<code>subset</code>	Integer vector. Row indices to return results for. All rows are computed; this filters the output. Default NULL (all rows).

Value

A `JvecforParam` object.

A `JvecforIndex` object.

A named list with `index` (n-by-k integer matrix or NULL) and `distance` (n-by-k numeric matrix or NULL).

Methods (by generic)

- `show(JvecforParam)`: Print a summary of the parameter object.
- `buildIndex(JvecforParam)`: Build a `JvecforIndex` from a data matrix.
- `findKnnFromIndex(JvecforIndex)`: Find k-nearest neighbors using a `JvecforIndex`.

Functions

- `JvecforParam()`: Constructor for `JvecforParam` objects.

Slots

`type` Character. "ann" (HNSW-DiskANN, default) or "knn" (VP-tree exact).

`M` Integer. HNSW max connections per node. Default 16L.

`ef.search` Integer. HNSW beam width (0 = auto). Default 0L.

`oversample.factor` Numeric. Oversampling multiplier (≥ 1). Default 1.0.

`pq.subspaces` Integer. Product-quantization subspaces (0 = disabled). Default 0L.

`verbose` Logical. Enable Java progress logging. Default FALSE.

Supported distance metrics

"Euclidean" and "Cosine" (title-case, following `BiocNeighbors` convention). The `jvecfor`-specific "dot_product" metric is only available via `fastFindKNN` directly.

Limitations

- `queryKNN` is not supported. The Java backend performs self-KNN only (all points query against all points in a single JVM invocation).
- The index built by `buildIndex` stores the data matrix in R memory; the actual Java HNSW/VP-tree index is rebuilt each time `findKNN` is called.

See Also

`fastFindKNN` for the standalone function with full parameter control including `dot_product` metric.

Examples

```
library(BiocNeighbors)
p <- JvecforParam()
p

# Custom parameters
p2 <- JvecforParam(type = "knn", distance = "Cosine", M = 32L)

# Use with BiocNeighbors (requires Java >= 20):
# res <- findKNN(X, k = 10, BNPARAM = JvecforParam())
```

jvecfor_setup

*Install a Custom jvecfor JAR***Description**

Copies a custom jvecfor JAR to the user data directory (`tools::R_user_dir("jvecfor", "data")`). The bundled JAR in `inst/java/` is used by default; call `jvecfor_setup()` only to override it with a custom build. Alternatively, set `options(jvecfor.jar = "/path/to/jvecfor.jar")` for a session-level override without copying.

Usage

```
jvecfor_setup(jar_path = NULL)
```

Arguments

jar_path	Path to the jvecfor JAR. If NULL, auto-searches <code>java/jvecfor/target/jvecfor-*.jar</code> relative to the working directory (works when developing inside the source tree).
----------	--

Value

Invisibly returns the path to the installed JAR.

Examples

```
# Show where custom JARs are stored
tools::R_user_dir("jvecfor", "data")

# List bundled JARs
dir(system.file("java", package = "jvecfor"), pattern = "*.jar")
```

Index

BiocNeighborIndex, [7](#)
BiocNeighborParam, [7](#)
BiocParallelParam, [3](#), [5](#), [6](#)
bpparam, [3](#), [5](#), [6](#)
buildIndex, JvecforParam-method
 (JvecforParam-class), [7](#)

fastFindKNN, [2](#), [3](#), [5](#), [6](#), [9](#)
fastMakeKNNGraph, [2](#), [4](#)
fastMakeSNNGraph, [2](#), [6](#)
findKNN, [7](#)
findKnnFromIndex, JvecforIndex-method
 (JvecforParam-class), [7](#)

jvecfor (jvecfor-package), [2](#)
jvecfor-package, [2](#)
jvecfor_setup, [2](#), [10](#)
JvecforIndex, [8](#), [9](#)
JvecforIndex-class, [7](#)
JvecforParam, [2](#), [7](#)
JvecforParam (JvecforParam-class), [7](#)
JvecforParam-class, [7](#)

show, JvecforParam-method
 (JvecforParam-class), [7](#)