Package: jvecfor (via r-universe)

November 1, 2026

Type Package

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

Version 1.0.3

Backend 4.0.0-rc.8

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 >= 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 (>= 4.5.0)

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 (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs libglpk-dev default-jdk libxml2-dev

Repository Bioconductor 3.23

Date/Publication 2026-09-27 22:43:47 UTC

RemoteUrl https://github.com/bioc/jvecfor

RemoteRef RELEASE_3_23

RemoteSha efd76140cf479bda1f80dc60152c459b20b0fa5f

Contents

jvecfor-package 2
fastFindKNN 3
fastMakeKNNGraph 5
fastMakeSNNGraph 6
jvecfor_setup 7
JvecforIndex-class 8
JvecforParam-class 9

Index **12**

`jvecfor-package` *jvecfor: Fast K-Nearest Neighbor Search for Single-Cell Analysis*

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 >= 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

`X`	A numeric matrix, `data.frame`, or sparse matrix (`Matrix::dgCMatrix`) 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.
`k`	Integer. Number of nearest neighbors to find (excluding self). Default 15.
`type`	Character. `"ann"` for approximate (HNSW-DiskANN) or `"knn"` for exact (VP-tree). Default `"ann"`.
`metric`	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.

num.threads — Integer or NULL. Number of Java threads. If NULL, defaults to `BiocParallel::bpworkers(BPPARAM)`.

BPPARAM — A `BiocParallelParam` object controlling the thread count. Defaults to `bpparam()`.

ef.search — Integer. HNSW-DiskANN beam width override (0 = auto: `max(k+1, 3k)`). Only meaningful when `type = "ann"`. Default 0L.

M — 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.

oversample.factor — Numeric. Oversampling multiplier for the ANN beam width. When > 1.0, fetches `ceil(ef * oversample.factor)` 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: `ncol(X) / 2`. Reduces search time approximately 4-8x with minimal recall loss. Only meaningful when `type = "ann"`. Default 0L.

get.distance — Logical. Return distance matrix alongside index? Default TRUE.

verbose — Logical. Pass `--verbose` to the Java process, enabling HNSW-DiskANN build progress logging on stderr. Overrides the `jvecfor.verbose` global option when set explicitly. Default `getOption("jvecfor.verbose", FALSE)`.

Value

A named list:

index n x k integer matrix of 1-indexed neighbor indices.

distance n x 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

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

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

X	A numeric matrix, data.frame, or sparse matrix (Matrix::dgCMatrix) with rows = cells, cols = features/PCs.
k	Integer. Number of nearest neighbors. Default 15.
type	Character. "ann" or "knn". Default "ann".
metric	Character. "euclidean", "cosine", or "dot_product". Default "euclidean". See fastFindKNN for the dot_product restriction.

num.threads	Integer or NULL. Number of Java threads. If NULL, defaults to BiocParallel::bpworkers(BPPARAM).
BPPARAM	A BiocParallelParam object controlling the thread count. Defaults to bpparam().
ef.search	Integer. HNSW-DiskANN beam width override (0 = auto). Default 0L.
M	Integer. HNSW-DiskANN max connections per node. Default 16L.
oversample.factor	Numeric. Oversampling multiplier. Default 1.0.
pq.subspaces	Integer. PQ subspaces (0 = disabled). Default 0L.
verbose	Logical. Enable Java verbose logging. Default getOption("jvecfor.verbose", FALSE).
snn.type	Character. SNN weighting scheme passed to bluster::neighborsToSNNGraph: "rank", "jaccard", or "number". Default "rank".
...	Additional arguments forwarded to bluster::neighborsToSNNGraph.

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)
```

---

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 `java/jvecfor/target/jvecfor-*.jar` 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")
```

JvecforIndex-class *JvecforIndex: BiocNeighbors Index Class for jvecfor*

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 *JvecforParam: BiocNeighbors Parameter Class for jvecfor*

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. `scran::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.
num.threads	Integer. Thread count. Default 1.
subset	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())
```

Index

BiocNeighborIndex, *8*
BiocNeighborParam, *9*
BiocParallelParam, *4, 5*, *7*
bpparam, *4, 5*, *7*
buildIndex,JvecforParam-method
(JvecforParam-class), 9

fastFindKNN, *2*, 3, *5, 6*, *11*
fastMakeKNNGraph, *2*, 5
fastMakeSNNGraph, *2*, 6
findKNN, *8, 9*
findKnnFromIndex,JvecforIndex-method
(JvecforParam-class), 9

jvecfor (jvecfor-package), 2
jvecfor-package, 2
jvecfor_setup, *2*, 7
JvecforIndex, *10*
JvecforIndex-class, 8
JvecforParam, *2*, *8*
JvecforParam (JvecforParam-class), 9
JvecforParam-class, 9

show,JvecforParam-method
(JvecforParam-class), 9