Package ‘kdtools’

October 8, 2021

Type Package
Title Tools for Working with Multidimensional Data
Version 0.6.0
Description Provides various tools for working with multidimensional data in R and C++, including extremely fast nearest-neighbor- and range-queries without the overhead of linked tree nodes.
License MIT + file LICENSE
Encoding UTF-8
Imports Rcpp (>= 0.12.14)
LinkingTo Rcpp
Suggests testthat, knitr, markdown, rmarkdown, covr, ggplot2, tidytext, printr, scales, BH, sf
VignetteBuilder knitr
SystemRequirements A compiler that conforms to the C++17 standard
RoxygenNote 7.1.1
URL https://github.com/thk686/kdtools
BugReports https://github.com/thk686/kdtools/issues
Biarch true
NeedsCompilation yes
Author Timothy Keitt [aut, cre]
Maintainer Timothy Keitt <tkeitt@gmail.com>
Repository CRAN
Date/Publication 2021-10-08 08:30:02 UTC

R topics documented:

  has_cxx17 ................................................................. 2
  kd_lower_bound ......................................................... 2
  kd_nearest_neighbors ............................................... 4
has_cxx17

Description
Check if C++ 17 was available when building package

Usage
has_cxx17()

kd_lower_bound

Description
Search sorted data

Usage
kd_lower_bound(x, v)
kd_upper_bound(x, v)
k_range_query(x, l, u, ...)

## S3 method for class 'matrix'
k_range_query(x, l, u, cols = NULL, ...)

## S3 method for class 'arrayvec'
k_range_query(x, l, u, ...)

## S3 method for class 'data.frame'
k_range_query(x, l, u, cols = NULL, ...)

kd_rq_indices(x, l, u, ...)

## S3 method for class 'matrix'
kd_rq_indices(x, l, u, cols = NULL, ...)

kd_sort

lex_sort

matrix_to_tuples

print.arrayvec
kd_lower_bound

## S3 method for class 'arrayvec'
kd_rq_indices(x, l, u, ...)

## S3 method for class 'data.frame'
kd_rq_indices(x, l, u, cols = NULL, ...)

kd_binary_search(x, v)

## S3 method for class 'matrix'
kd_binary_search(x, v)

## S3 method for class 'arrayvec'
kd_binary_search(x, v)

Arguments

- **x**: an object sorted by `kd_sort`
- **v**: a vector specifying where to look
- **l**: lower left corner of search region
- **u**: upper right corner of search region
- **...**: ignored
- **cols**: integer or character vector or formula indicating columns

Value

- **kd_lower_bound**: a row of values (vector)
- **kd_upper_bound**: a row of values (vector)
- **kd_range_query**: a set of rows in the same format as the sorted input
- **kd_rq_indices**: a vector of integer indices specifying rows in the input
- **kd_binary_search**: a boolean

Examples

```r
if (has_cxx17()) {
  x = matrix(runif(200), 100)
  y = matrix_to_tuples(x)
  kd_sort(y, inplace = TRUE)
  y[kd_lower_bound(y, c(1/2, 1/2)),]
  y[kd_upper_bound(y, c(1/2, 1/2)),]
  kd_binary_search(y, c(1/2, 1/2))
  kd_range_query(y, c(1/3, 1/3), c(2/3, 2/3))
  kd_rq_indices(y, c(1/3, 1/3), c(2/3, 2/3))
}
```
**kd_nearest_neighbors**  
*Find nearest neighbors*

---

**Description**

Find nearest neighbors

**Usage**

kd_nearest_neighbors(x, v, n, ...)

## S3 method for class 'matrix'
kd_nearest_neighbors(x, v, n, cols = NULL, alpha = 0, ...)

## S3 method for class 'arrayvec'
kd_nearest_neighbors(x, v, n, ...)

## S3 method for class 'data.frame'
kd_nearest_neighbors(x, v, n, cols = NULL, w = NULL, ...)

kd_nn_indices(x, v, n, ...)

## S3 method for class 'arrayvec'
kdkd_nn_indices(x, v, n, distances = FALSE, ...)

## S3 method for class 'matrix'
kdkd_nn_indices(x, v, n, cols = NULL, distances = FALSE, alpha = 0, ...)

## S3 method for class 'data.frame'
kdkd_nn_indices(x, v, n, cols = NULL, w = NULL, distances = FALSE, ...)

kd_nearest_neighbor(x, v)

## S3 method for class 'matrix'
kdkd_nearest_neighbor(x, v)

## S3 method for class 'arrayvec'
kdkd_nearest_neighbor(x, v)

**Arguments**

- **x**: an object sorted by *kd_sort*
- **v**: a vector specifying where to look
- **n**: the number of neighbors to return
- **...**: ignored
- **cols**: integer or character vector or formula indicating columns
alpha approximate neighbors within (1 + alpha)

w distance weights

distances return distances as attribute if true

Value

kd_nearest_neighbors one or more rows from the sorted input
kd_nn_indices a vector of row indices indicating the result
kd_nearest_neighbor the row index of the neighbor

Examples

```r
if (has_cxx17()) {
  x = matrix(runif(200), 100)
  y = matrix_to_tuples(x)
  kd_sort(y, inplace = TRUE)
  y[kd_nearest_neighbor(y, c(1/2, 1/2)),]
  kd_nearest_neighbors(y, c(1/2, 1/2), 3)
  y[kd_nn_indices(y, c(1/2, 1/2), 5),]
}
```

---

kd_sort  

Sort multidimensional data

Description

Sort multidimensional data

Usage

kd_sort(x, ...)

## S3 method for class 'matrix'
kd_sort(x, cols = NULL, parallel = TRUE, ...)

## S3 method for class 'arrayvec'
kd_sort(x, inplace = FALSE, parallel = TRUE, ...)

## S3 method for class 'data.frame'
kd_sort(x, cols = NULL, parallel = TRUE, ...)

## S3 method for class 'sf'
kd_sort(x, cols = NULL, parallel = TRUE, ...)
kd_order(x, ...)

## S3 method for class 'matrix'
kd_order(x, cols = NULL, parallel = TRUE, ...)

## S3 method for class 'arrayvec'
kd_order(x, inplace = FALSE, parallel = TRUE, ...)

## S3 method for class 'data.frame'
kd_order(x, cols = NULL, parallel = TRUE, ...)

kd_is_sorted(x, ...)

Arguments

x  
a matrix or arrayvec object
...
  ignored
cols  
  integer or character vector or formula indicating columns
parallel  
  use multiple threads if true
inplace  
  sort as a side-effect if true

Details

The algorithm used is a divide-and-conquer quicksort variant that recursively partitions an range of
tuples using the median of each successive dimension. Ties are resolved by cycling over successive
dimensions. The result is an ordering of tuples matching their order if they were inserted into a
kd-tree.

kd_order returns permutation vector that will order the rows of the original matrix, exactly as
order. If inplace is true, then kd_order will also sort the arrayvec object as a side effect. This
can be more efficient when many subsequent queries are required.

kd_sort and kd_order have been extended to work directly on R native data.frame and matrix
types. All vector column types are supported (even lists of objects as long as equality and compari-
sion operators are defined). Additional, the user can specify a sequence of column indices that will
be used for sorting. These can be a subset of columns and given in any order.

Value

kd_sort  
  the table sorted in kd-tree order
kd_order  
  a permutation vector
kd_is_sorted  
  a boolean
See Also

arrayvec

Examples

```r
if (has_cxx17()) {
  z <- data.frame(real = runif(10), lgl = runif(10) > 0.5,
                  int = as.integer(rpois(10, 2)), char = sample(month.name, 10),
                  stringsAsFactors = FALSE)
  kd_sort(z)
  x <- matrix(runif(200), 100)
  y <- kd_sort(x)
  kd_is_sorted(y)
  kd_order(x)
  plot(y, type = "o", pch = 19, col = "steelblue", asp = 1)
}
```

---

**lex_sort**

*Sort a matrix into lexicographical order*

**Description**

Sort a matrix into lexicographical order.

**Usage**

```r
lex_sort(x, ...)
```

**Arguments**

- `x`: a matrix or arrayvec object
- `...`: other parameters

**Details**

Sorts a range of tuples into lexicographical order.

**Value**

the input type sorted

**Examples**

```r
if (has_cxx17()) {
  x = lex_sort(matrix(runif(200), 100))
  plot(x, type = "o", pch = 19, col = "steelblue", asp = 1)
}
```
matrix_to_tuples  
Convert a matrix to a vector of arrays

Description

Convert a matrix to a vector of arrays

Usage

matrix_to_tuples(x)

tuples_to_matrix(x)

Arguments

x  
object to be converted

Details

The algorithms in kdtools can accept either matrices or an arrayvec object. When a matrix is passed, it is converted to an arrayvec object internally and the results are converted back to a matrix. For optimal performance, pre-convert matrices.

Examples

```r
if (has_cxx17()) {
  x = matrix(1:10, 5)
  y = matrix_to_tuples(x)
  str(x)
  str(y)
  y[1:2, ]
}
```

print.arrayvec  
Support for C++ vector of arrays

Description

Support for C++ vector of arrays
print.arrayvec

Usage

## S3 method for class 'arrayvec'
print(x, ...)

## S3 method for class 'arrayvec'
dim(x)

## S3 method for class 'arrayvec'
as.matrix(x, ...)

## S3 method for class 'arrayvec'
as.data.frame(x, ...)

## S3 method for class 'arrayvec'
x[i, j, drop = TRUE]

## S3 method for class 'arrayvec'
x[[...]]

Arguments

x an arrayvec object

... other parameters

i row

j column

drop drop singleton dimensions if true

Details

Because kdtools is implemented in C++, it operates natively on a vector of arrays. An arrayvec object is a wrapper around a pointer to a vector of arrays. These functions provide some ability to manipulate the data as if it were a matrix.

Value

print.arrayvec the object invisibly
dim.arrayvec the rows and columns
as.matrix.arrayvec a matrix
as.data.frame.arrayvec a data frame
‘.[arrayvec’ a matrix or vector
‘[[.arrayvec’ a column vector
Index

arrayvec, 7, 8
arrayvec (print.arrayvec), 8
as.data.frame.arrayvec
  (print.arrayvec), 8
as.matrix.arrayvec (print.arrayvec), 8
dim.arrayvec (print.arrayvec), 8
has_cxx17, 2
kd_binary_search (kd_lower_bound), 2
kd_is_sorted (kd_sort), 5
kd_lower_bound, 2
kd_nearest_neighbor
  (kd_nearest_neighbors), 4
kd_nearest_neighbors, 4
kd_nn_indices (kd_nearest_neighbors), 4
kd_order (kd_sort), 5
kd_range_query (kd_lower_bound), 2
kd_rq_indices (kd_lower_bound), 2
kd_sort, 3, 4, 5
kd_upper_bound (kd_lower_bound), 2
lex_sort, 7
matrix_to_tuples, 8
order, 6
print.arrayvec, 8
tuples_to_matrix (matrix_to_tuples), 8