Package ‘kdtools’

April 24, 2019

Type Package
Title Tools for Working with Multidimensional Data
Version 0.4.1
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
LazyData true
Imports Rcpp (>= 0.12.14)
LinkingTo Rcpp, strider, BH
Suggests testthat, knitr, rmarkdown, covr, ggplot2, tidytext, printr,
scales
VignetteBuilder knitr
RoxygenNote 6.1.0
URL https://github.com/thk686/kdtools
BugReports https://github.com/thk686/kdtools/issues
NeedsCompilation yes
Author Timothy Keitt [aut, cre]
Maintainer Timothy Keitt <tkeitt@gmail.com>
Repository CRAN
Date/Publication 2019-04-23 22:50:03 UTC

R topics documented:

kn_lower_bound .................................................. 2
kd_nearest_neighbors ........................................... 3
kd_sort .......................................................... 3
lex_sort ........................................................ 4
matrix_to_tuples .............................................. 5
print.arrayvec ............................................... 6
kd_lower_bound

Index

| kd_lower_bound | Search sorted data |

Description

Search sorted data

Usage

kd_lower_bound(x, v)
kd_upper_bound(x, v)
kd_range_query(x, l, u)
kd_rq_indices(x, l, u)
kd_binary_search(x, v)

Arguments

x an object sorted by \texttt{kd_sort}
v a vector specifying where to look
l lower left corner of search region
u upper right corner of search region

Examples

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)
kd_nn_indices(x, v, n)
kd_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

Examples
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)),]
y[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, ...)
kd_order(x, ...)
kd_is_sorted(x)
Arguments

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

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.

Note

The matrix version will be slower because of data structure conversions.

See Also

arrayvec

Examples

```r
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
matrix_to_tuples

Details

Sorts a range of tuples into lexicographical order.

Examples

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

x = matrix(1:10, 5)
y = matrix_to_tuples(x)
str(x)
str(y)
y[1:2, ]
## Description

Support for C++ vector of arrays

## Usage

```r
## 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.
Index

[.arrayvec (print.arrayvec), 6
[[.arrayvec (print.arrayvec), 6
arrayvec, 4, 5
arrayvec (print.arrayvec), 6
as.data.frame.arrayvec
  (print.arrayvec), 6
as.matrix.arrayvec (print.arrayvec), 6
dim.arrayvec (print.arrayvec), 6
kd_binary_search (kd_lower_bound), 2
kd_is_sorted (kd_sort), 3
kd_lower_bound, 2
kd_nearest_neighbor
  (kd_nearest_neighbors), 3
kd_nearest_neighbors, 3
kd_nn_indices (kd_nearest_neighbors), 3
kd_order (kd_sort), 3
kd_range_query (kd_lower_bound), 2
kd_rq_indices (kd_lower_bound), 2
kd_sort, 2, 3, 3
kd_upper_bound (kd_lower_bound), 2

lex_sort, 4
matrix_to_tuples, 5
order, 4
print.arrayvec, 6
tuples_to_matrix (matrix_to_tuples), 5