

# Package ‘kdtools’

October 6, 2018

**Type** Package

**Title** Tools for Working with Multidimensional Data

**Version** 0.3.1

**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, print,  
scales

**VignetteBuilder** knitr

**RoxygenNote** 6.0.1

**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** 2018-04-26 21:16:32 UTC

## R topics documented:

kd_lower_bound . . . . .	2
kd_nearest_neighbors . . . . .	2
kd_sort . . . . .	3
lex_sort . . . . .	4
matrix_to_tuples . . . . .	4
print.arrayvec . . . . .	5
<b>Index</b>	<b>6</b>

---

kd_lower_bound	<i>Search sorted data</i>
----------------	---------------------------

---

**Description**

Search sorted data

**Usage**

kd\_lower\_bound(x, v)

kd\_upper\_bound(x, v)

kd\_range\_query(x, l, u)

kd\_binary\_search(x, v)

**Arguments**

x	an object sorted by <a href="#">kd_sort</a>
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_nearest_neighbors	<i>Find nearest neighbors</i>
----------------------	-------------------------------

---

**Description**

Find nearest neighbors

**Usage**

kd\_nearest\_neighbors(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)),]
kd_nearest_neighbors(y, c(1/2, 1/2), 3)
```

---

kd_sort	<i>Sort multidimensional data</i>
---------	-----------------------------------

---

**Description**

Sort multidimensional data

**Usage**

```
kd_sort(x, ...)
```

```
kd_is_sorted(x)
```

**Arguments**

x                    a matrix or arrayvec object  
...                  other arguments

**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.

**Note**

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

**See Also**

[arrayvec](#)

**Examples**

```
x = kd_sort(matrix(runif(200), 100))
kd_is_sorted(x)
plot(x, type = "o", pch = 19, col = "steelblue", asp = 1)
```

---

lex_sort	<i>Sort a matrix into lexicographical order</i>
----------	---

---

**Description**

Sort a matrix into lexicographical order

**Usage**

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

**Arguments**

x	a matrix or arrayvec object
...	other parameters

**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	<i>Convert a matrix to a vector of arrays</i>
------------------	---

---

**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, ]
```

---

print.arrayvec

*Support for C++ vector of arrays*


---

**Description**

Support for C++ vector of arrays

**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.

# Index

`[.arrayvec (print.arrayvec), 5`  
`[ [.arrayvec (print.arrayvec), 5`  
  
`arrayvec, 3, 4`  
`arrayvec (print.arrayvec), 5`  
`as.data.frame.arrayvec`  
    `(print.arrayvec), 5`  
`as.matrix.arrayvec (print.arrayvec), 5`  
  
`dim.arrayvec (print.arrayvec), 5`  
  
`kd_binary_search (kd_lower_bound), 2`  
`kd_is_sorted (kd_sort), 3`  
`kd_lower_bound, 2`  
`kd_nearest_neighbor`  
    `(kd_nearest_neighbors), 2`  
`kd_nearest_neighbors, 2`  
`kd_range_query (kd_lower_bound), 2`  
`kd_sort, 2, 3, 3`  
`kd_upper_bound (kd_lower_bound), 2`  
  
`lex_sort, 4`  
  
`matrix_to_tuples, 4`  
  
`print.arrayvec, 5`  
  
`tuples_to_matrix (matrix_to_tuples), 4`