Package ‘SearchTrees’

October 12, 2022

Type Package
Title Spatial Search Trees
Version 0.5.5
Date 2022-10-03
Author Gabriel Becker
Maintainer Gabriel Becker <gabembecker@gmail.com>
Description The QuadTree data structure is useful for fast,
neighborhood-restricted lookups. We use it to implement fast k-Nearest
Neighbor and Rectangular range lookups in 2 dimensions. The
primary target is high performance interactive graphics.
Depends methods
License LGPL
LazyLoad yes
URL https://github.com/gmbecker/SearchTrees
BugReports https://github.com/gmbecker/SearchTrees/issues
NeedsCompilation yes
Repository CRAN
Date/Publication 2022-10-03 22:20:06 UTC

R topics documented:

createTree ......................................................... 2
knnLookup .......................................................... 3
knnLookup-methods ................................................. 4
QuadTree-class ...................................................... 5
rectLookup .......................................................... 6
rectLookup-methods ............................................... 7
SearchTree-class .................................................. 7

Index 9
createTree

Create a Search Tree Index

Description

Create a search tree from the supplied data for use in during future lookups.

Usage

createTree(data, treeType = "quad", dataType = "point",
            columns = if (dataType=="point") 1:2 else 1:4, ...)

Arguments

data: data.frame or matrix. Data to be indexed.
treeType: Character. Indicates type of index tree to be created. Currently only "quad"
          (quad trees) is supported.
dataType: Character. Indicates type of data being indexed. Currently "point", and "rect"
          are supported corresponding to points and rectangles, respectively. Defaults to
          "point".
columns: Numeric. Indicates columns in data the information to be indexed can be found.
          Length depends on value of dataType (2 for "point" and 4 for "rect"). Defaults
to columns 1 and 2. See Details.
...

Details

Data frame or matrix. Data to be indexed.

Usage

createTree(data, treeType = "quad", dataType = "point",
            columns = if (dataType=="point") 1:2 else 1:4, ...)

Arguments

data: data.frame or matrix. Data to be indexed.
treeType: Character. Indicates type of index tree to be created. Currently only "quad"
          (quad trees) is supported.
dataType: Character. Indicates type of data being indexed. Currently "point", and "rect"
          are supported corresponding to points and rectangles, respectively. Defaults to
          "point".
columns: Numeric. Indicates columns in data the information to be indexed can be found.
          Length depends on value of dataType (2 for "point" and 4 for "rect"). Defaults
to columns 1 and 2. See Details.
...

Details

For a point based tree, the two columns specified in columns represent the x and y values of the
points.
For a rectangle based tree, four columns must be specified. These columns represent the x and y
coordinates of point 1 and the x and y coordinates of point 2, in that order (where point 1 and point
2 specify the rectangle to be stored).

Value

The class of the returned object depends on the tree type created, though all will inherit from the
SearchTree S4 class and have the following slots:

ref: An external pointer to the C level data structure.
numNodes: Total number of nodes comprising the tree.
dataNodes: Number of nodes which store at least one data point.
**knnLookup**

- **maxDepth**: Maximum depth of the tree.
- **maxBucket**: Maximum number of data points stored in a single node.
- **totalData**: Number of items indexed in the tree.
- **dataType**: Type of objects stored in the tree.

**Author(s)**

Gabriel Becker

**References**


**See Also**

SearchTree linkS4Class{QuadTree}

**Examples**

```r
x = rnorm(100)
y = rnorm(100)
dat = cbind(x,y)
tree = createTree(dat)
```

---

**knnLookup**  
Perform k-Nearest Neighbors Lookup Using a Search Tree

**Description**

This function performs fast k-Nearest Neighbors lookup on a SearchTree object

**Usage**

```r
knnLookup(tree, newx, newy, newdat, columns = 1:2, k = 5)
```

**Arguments**

- **tree**: An object which inherits from the SearchTree S4 class.
- **newx**: Numeric. Vector of x values for the points to look up neighbors for.
- **newy**: Numeric. Vector of x values for the points to look up neighbors for.
- **newdat**: Matrix or data.frame. Data containing x and y values of the points to look up neighbors for. Ignored if x and y are specified.
- **columns**: Numeric. Columns x and y values can be found in within newdat.
- **k**: Numeric. Number of neighbors to find for each point.
Value

The return value is an integer matrix indicating the indices in the original data used to create the tree where the nearest neighbors were found. Row indicates the indice of the new point, while column indicates the order of the k neighbors.

Note

No defined order is specified for exact ties in distance.

Author(s)

Gabriel Becker

See Also

createTree rectLookup

Examples

```r
x = rnorm(100)
y = rnorm(100)
tree = createTree(cbind(x,y))
newx = c(0, .5)
newy = c(.5, 0)
inds = knnLookup(tree, newx, newy, k=7)

ch = rep(1, times=100)
ch[inds[1:7]] = 3
ch[inds[8:14]] = 5
cls = rep("black", times=100)
cls[inds[1:7]] = "red"
cls[inds[8:14]] = "blue"

plot(x,y, pch=ch, col = cls)
abline(v=newx[1], h = newy[1] , col="red")
abline(v=newx[2], h = newy[2], col = "blue")
```

---

**knnLookup-methods**

Methods

```
signature(tree = "QuadTree")
```
QuadTree-class

Class "QuadTree"

Description
A class representing a Quad Tree object for storing 2 dimensional points for efficient rectangular range and knn lookup.

Objects from the Class
Objects can be created by calls of the form new("QuadTree", ...).

Slots
ref: Object of class "externalptr" Pointer to the internal representation of the tree
numNodes: Object of class "integer" Number of nodes in the tree
dataNodes: Object of class "integer" Number of nodes in the tree which are storing data
maxDepth: Object of class "integer" Maximum depth of the tree.
maxBucket: Object of class "integer" Maximum number of data points which are stored at a single node
totalData: Object of class "integer" Number of objects stored in the tree
dataType: Object of class "character" Indicates type of data stored in the tree.

Extends
Class "SearchTree", directly.

Methods
knnLookup signature(tree = "QuadTree"): ...
rectLookup signature(tree = "QuadTree"): ...

Note
When using createIndex to create a quadTree, only two columns of the matrix/data.frame passed to the function will be used to create the tree. See the columns argument in createTree

Author(s)
Gabriel Becker

See Also
createTree

Examples
showClass("QuadTree")
rectLookup

Perform Rectangular Lookup in 2d Space

Description

Determine which objects, stored in a SearchTrees indexing object, fall within a given rectangle in two-dimensional space.

Usage

rectLookup(tree, ptOne, ptTwo, xlims, ylims)

Arguments

tree
- SearchTree. A SearchTree object to perform the lookup on.

ptOne
- Numeric. A numeric of length two indicating x and y values for one corner of the rectangle.

ptTwo
- Numeric. A numeric of length two indicating x and y values for the corner of the rectangle opposite to ptOne

xlims
- Numeric. A numeric vector indicating the minimum and maximum x value for the rectangle. Overrides ptOne and ptTwo

ylims
- Numeric. A numeric vector indicating the minimum and maximum y value for the rectangle. Overrides ptOne and ptTwo

Details

In the case of lookup for rectangular objects, any rectangle which overlaps the query rectangle will be returned.

Value

A numeric vector indicating the indices of the object (in the order they were in when the SearchTree object was created) which fall (at least partially) within the rectangular query.

Author(s)

Gabriel Becker

See Also

QuadTree knnLookup
Examples

```r
x = rnorm(100)
y = rnorm(100)
x2 = x + runif(100, .5, 2)
y2 = y + runif(100, .5, 2)
dat2 = cbind(x, y, x2, y2)
tree2 = createTree(dat2, dataType="rect", columns= 1:4)
inrect = rectLookup(tree2, xlim = c(0,1), ylim=c(0, 1))
col = rgb(0, 1, 0, alpha=.5)
plot(x, y2, col="white")
rect(x[inrect], y[inrect], x2[inrect], y2[inrect], col=col)
rect(0, 0, 1, 1, col="blue", lwd=3)
```

Description

Methods for function `rectLookup` in package `SearchTrees`

Methods

```
signature(tree = "QuadTree")
```

SearchTree-class

Class "SearchTree"

Description

A virtual class representing a search tree for storing geometric points in a manner designed for efficient lookup.

Objects from the Class

This is a virtual class so objects of class `SearchTree` cannot be created directly. No methods defined with class "SearchTree" in the signature.

Slots

- `ref`: Object of class "externalPtr" Pointer to the internal representation of the tree.
- `numNodes`: Object of class "integer" Number of nodes in the tree.
- `dataNodes`: Object of class "integer" Number of nodes in the tree which are storing data.
- `maxDepth`: Object of class "integer" Maximum depth of the tree.
- `maxBucket`: Object of class "integer" Maximum number of data points stored in a single node.
- `totalData`: Object of class "integer" Number of data objects stored in the tree.
- `dataType`: Object of class "character" Indicates type of data stored in the tree.
Methods

knnLookup, rectLookup

Author(s)

Gabriel Becker

See Also

QuadTree createTree
Index

* classes
  QuadTree-class, 5
  SearchTree-class, 7
* indexing
  createTree, 2
* knn
  knnLookup, 3
* lookup
  knnLookup, 3
  rectLookup, 6
  SearchTree-class, 7
* methods
  knnLookup-methods, 4
  rectLookup-methods, 7
* neighbors
  knnLookup, 3
* quadtree
  createTree, 2
* query
  rectLookup, 6
* rectangular
  rectLookup, 6

createTree, 2, 4, 5, 8

knnLookup, 3, 6
knnLookup, QuadTree-method (knnLookup), 3
knnLookup, QuadTree-method (knnLookup-methods), 4
knnLookup-methods, 4

QuadTree, 6, 8
QuadTree-class, 5

rectLookup, 4, 6
rectLookup, QuadTree-method (rectLookup), 6
rectLookup, QuadTree-method (rectLookup-methods), 7
rectLookup-methods, 7
SearchTree, 3, 5
SearchTree-class, 7