Package 'rIntervalTree'

October 14, 2022

Type Package

Title An Interval Tree Tool for Real Numbers

Version 0.1.0

Author Shuye Pu [aut, cre]

Maintainer Shuye Pu <shuye2009@gmail.com>

Description This tool can be used to build binary interval trees using real number inputs.
The tree supports queries of intervals overlapping a single number or an interval (start, end).
Intervals with same bounds but different names are treated as distinct intervals.
Insertion of intervals is also allowed. Deletion of intervals is not implemented at this point.
See Mark de Berg, Otfried Cheong, Marc van Kreveld, Mark Overmars (2008). Computational Geometry: Algorithms and Applications, for a reference.

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Collate 'Interval.R' 'IntervalTree.R'

Imports methods

NeedsCompilation no

Repository CRAN

Date/Publication 2019-04-23 09:20:03 UTC

R topics documented:

`buildTree` *buildTree*

Description

Method for building a binary interval tree given an IntervalTree object with a defined data slot but an undefined root slot. This method is a wrapper function of the treeFromInterval function. In the first step, the dataframe in the data slot is converted into a list of Interval objects. Then, the treeFromInterval function is called to construct an interval tree using the list as an input, and the root of the resulting interval tree is assigned to the root slot of the IntervalTree object. This method is called implicitly when an IntervalTree object is initialized with a non-empty dataframe.

Usage

```
buildTree(theObject)
```

Arguments

`theObject`	an IntervalTree object containing a non-empty dataframe.

Value

an IntervalTree object, with the root being an recursive list of Intervals.

Examples

```
m_ranges <- data.frame(c("A", "B", "C", "D"), c(1,2,3,4), c(5,4,6,10))
I <- new("IntervalTree")
I@data <- m_ranges
m_interval_tree <- buildTree(I)
## buildTree is called implicitly
II <- IntervalTree(data=m_ranges, root=list())
## buildTree is called implicitly
m_interval_tree <- new("IntervalTree", data=m_ranges, root=list())
```

buildTree,IntervalTree-method *buildTree*

Description

Method for building a binary interval tree given an IntervalTree object with a defined data slot but an undefined root slot. This method is a wrapper function of the treeFromInterval function. In the first step, the dataframe in the data slot is converted into a list of Interval objects. Then, the treeFromInterval function is called to construct an interval tree using the list as an input, and the root of the resulting interval tree is assigned to the root slot of the IntervalTree object. This method is called implicitly when an IntervalTree object is initialized with a nonempty dataframe.

Usage

```
## S4 method for signature 'IntervalTree'
buildTree(theObject)
```

Arguments

theObject	an IntervalTree object containing a non-empty dataframe.

Value

an IntervalTree object, with the root being an recursive list of Intervals.

Examples

```
m_ranges <- data.frame(c("A", "B", "C", "D"), c(1,2,3,4), c(5,4,6,10))
I <- new("IntervalTree")
I@data <- m_ranges
m_interval_tree <- buildTree(I)
## buildTree is called implicitly
II <- IntervalTree(data=m_ranges, root=list())
## buildTree is called implicitly
m_interval_tree <- new("IntervalTree", data=m_ranges, root=list())
```

collectIntervals *collectIntervals*

Description

Method for enumerating all intervals in a interval tree (a list object).

Usage

```
collectIntervals(aTree)
```

Arguments

aTree	a recursive list storing Interval object

Value

a flattened list of Interval object

Examples

```
i1 <- new("Interval", start=1.1, end=1.2, key="dummy1")
i2 <- new("Interval", start=-1.1, end=1.2, key="dummy2")
i3 <- new("Interval", start=-10.1, end=-1.2, key="dummy3")
i4 <- new("Interval", start=-1.1, end=1.2, key="dummy4")
i5 <- new("Interval", start=-10, end=2, key="dummy5")
i6 <- new("Interval", start=-8, end=-5, key="dummy6")
myList <- list(i1, i2, i3, i4, i5, i6)
atree <- treeFromInterval(myList)
collectIntervals(atree)
collectIntervals(list())
```

---

compounds *Compounds with mass ranges*

Description

A dataset containing the the mass ranges of chemical compounds with a 10 ppm tolerance. The variables are as follows.

Usage

```
data(compounds)
```

Format

A data frame with 23 rows and 3 variables

Details

- name. name of the compound
- low. low bound of the mass
- high. high bound of the mass

insert *insert*

Description

Method for inserting an Interval into an interval tree (a recursive list). The structure of the final tree is invariant of the order of insertion. Tree balancing is not implemented, therefore, the resulting tree may not be balanced.

Usage

```
insert(aTree, anInterval)
```

Arguments

aTree	an interval tree (a list object)
anInterval	an Interval object

Value

a list object, representing a binary interval tree

Examples

```
i1 <- new("Interval", start=1.1,end=1.2, key="dummy1")
i2 <- new("Interval", start=-1.1,end=1.2, key="dummy2")
i3 <- new("Interval", start=-10.1,end=-1.2, key="dummy3")
i4 <- new("Interval", start=-1.1,end=1.2, key="dummy4")
i5 <- new("Interval", start=-10,end=2, key="dummy5")
i6 <- new("Interval", start=-8,end=-5, key="dummy6")
i7 <- new("Interval", start=80,end=100, key="dummy7")
i8 <- new("Interval", start=-80,end=-15, key="dummy8")

atree <- list()
atree <- insert(atree, i1)
atree <- insert(atree, i2)
atree <- insert(atree, i3)
atree <- insert(atree, i4)
atree <- insert(atree, i5)
atree <- insert(atree, i6)
atree <- insert(atree, i7)
atree <- insert(atree, i8)

intersectInterval(atree, 85)
intersectInterval(atree, 0)
intersectInterval(atree, c(-70, -9))
## Not run:
intersectInterval(atree, c(80,0)) ## generate an error

## End(Not run)
```

`insertInterval` *insertInterval*

Description

Method for inserting an interval into an IntervalTree object. Given an ordered pair of numbers denoting the start and end of an interval, the interval is first converted into an Interval object, then the Interval object is inserted by calling the insert() function.

Usage

```
insertInterval(theObject, anInterval)
```

Arguments

`theObject`	an IntervalTree object
`anInterval`	an interval in the form of (name, start, end).

Value

an IntervalTree object.

Examples

```
m_ranges <- data.frame(c("A", "B", "C", "D"), c(-1.1,2,3,4), c(5,4,6,10))
m_interval_tree <- new("IntervalTree", data=m_ranges, root=list())
m_interval_tree <- insertInterval(m_interval_tree, c("testInterval1", 2, 5))
res <- insertInterval(m_interval_tree, c("a",2.5,7))
```

`insertInterval,IntervalTree,character-method` *insertInterval*

Description

Method for inserting an interval into an IntervalTree object. Given an ordered pair of numbers denoting the start and end of an interval, the interval is first converted into an Interval object, then the Interval object is inserted by calling the insert() function.

Usage

```
## S4 method for signature 'IntervalTree,character'
insertInterval(theObject, anInterval)
```

Arguments

theObject	an IntervalTree object
anInterval	an interval in the form of (name, start, end).

Value

an IntervalTree object.

Examples

```
m_ranges <- data.frame(c("A", "B", "C", "D"), c(-1.1,2,3,4), c(5,4,6,10))
m_interval_tree <- new("IntervalTree", data=m_ranges, root=list())
m_interval_tree <- insertInterval(m_interval_tree, c("testInterval2", 200, 500))
res <- insertInterval(m_interval_tree, c("a",-25,7))
```

---

intersectInterval *intersectInterval*

Description

Method for searching the interval tree. Given a single number or an ordered pair of numbers denoting the start and end of an interval, all intervals that overlapping the query interval in the interval tree will be retrieved.

Usage

```
intersectInterval(aTree, someNumbers)
```

Arguments

aTree	a list object representing an interval tree
someNumbers	a vector of one or two numbers to test for overlap. If two numbers are provided, they are treated as an interval (start, end).

Value

a list of vectors. Each vector contains (name, start, end) of an interval

Examples

```
i1 <- new("Interval", start=1.1,end=1.2, key="dummy1")
i2 <- new("Interval", start=-1.1,end=1.2, key="dummy2")
i3 <- new("Interval", start=-10.1,end=-1.2, key="dummy3")
i4 <- new("Interval", start=-1.1,end=1.2, key="dummy4")
i5 <- new("Interval", start=-10,end=2, key="dummy5")
i6 <- new("Interval", start=-8,end=-5, key="dummy6")

myList <- list(i1, i2, i3, i4, i5, i6)
```

```
atree <- treeFromInterval(myList)
## Not run:
intersectInterval(atree, c(-16, -26)) # generate an error

## End(Not run)
intersectInterval(atree, c(1, 5))
intersectInterval(atree, c(-12, 15))
intersectInterval(atree, 0)
```

`Interval-class` *Interval*

Description

An S4 class to represent a named numeric interval.

Slots

`start` the lower end of the interval

`end` the higher end of the interval

`key` the name of the interval

`IntervalTree-class` *IntervalTree*

Description

A S4 class to represent interval tree.

Slots

`data` a dataframe providing the intervals to be stored in the interval tree. The columns are key, start and end of intervals

`root` the root list of the interval tree built upon the data

isOverlap *isOverlap*

Description

Method for checking if an interval is overlapping with a single number (start = end) or a pair of numbers (start < end). A pair of intervals (start1, end1) and (start2, end2) are overlapping if (end2 >= start1 and start2 <= end1).

Usage

```
isOverlap(theObject, someNumbers)
```

Arguments

theObject	an Interval object
someNumbers	a vector of one or two numbers to test overlap. If two numbers are provided, they are treated as an interval (start, end)

Value

a logical value TRUE or FALSE

Examples

```
i1 <- new("Interval", start=1.1, end=1.2, key="dummy")
isOverlap(i1, c(1.0, 1.5))
isOverlap(i1, 1.0)
## Not run:
isOverlap(i1, c(2.0, 1.5))  # generate an error
isOverlap(i1, c(1.0, 1.5, 2))  # generate an error

## End(Not run)
```

isOverlap,Interval,numeric-method *isOverlap*

Description

Method for checking if an interval is overlapping with a single number (start = end) or an ordered pair of numbers (start < end). Two intervals (start1, end1) and (start2, end2) are overlapping if (end2 >= start1 and start2 <= end1).

Usage

```
## S4 method for signature 'Interval,numeric'
isOverlap(theObject, someNumbers)
```

Arguments

theObject	an Interval object
someNumbers	a vector of one or two numbers to test overlap. If two numbers are provided, they are treated as an interval (start, end)

Value

a logical value TRUE or FALSE

Examples

```
i1 <- new("Interval", start=1.1, end=1.2, key="dummy")
isOverlap(i1, c(1.0, 1.5))
isOverlap(i1, 1.0)
## Not run:
isOverlap(i1, c(2.0, 1.5))  # generate an error
isOverlap(i1, c(1.0, 1.5, 2))  # generate an error

## End(Not run)
```

---

overlapQuery *overlapQuery*

Description

Method for searching an IntervalTree object. Given a number or an ordered pair of numbers denoting the start and end of an interval, all intervals that overlapping the query interval in the IntervalTree object will be retrieved.

Usage

```
overlapQuery(theObject, anInterval)
```

Arguments

theObject	an IntervalTree object
anInterval	a vector of one or two numbers to check overlap, if two numbers are provided, they are treated as an interval (start, end).

Value

a list of vectors. Each vector contains information about an interval (name, start, end).

Examples

```
m_ranges <- data.frame(c("A", "B", "C", "D"), c(-1.1,2,3,4), c(5,4,6,10))

m_interval_tree <- new("IntervalTree", data=m_ranges, root=list())
overlapQuery(m_interval_tree, 4)
res <- overlapQuery(m_interval_tree, c(2.5,7))
res
```

---

overlapQuery,IntervalTree,numeric-method
overlapQuery

Description

Method for searching an IntervalTree object. Given a number or an ordered pair of numbers denoting the start and end of an interval, all intervals that overlapping the query interval in the IntervalTree object will be retrieved.

Usage

```
## S4 method for signature 'IntervalTree,numeric'
overlapQuery(theObject, anInterval)
```

Arguments

theObject	an IntervalTree object
anInterval	a vector of one or two numbers to check overlap, if two numbers are provided, they are treated as an interval (start, end).

Value

a list of vectors. Each vector contains information about an interval (name, start, end).

Examples

```
m_ranges <- data.frame(c("A","B","C","D","E","F"), c(-1.1,2,3,4,20,200), c(5,4,6,10,21.2,400))

m_interval_tree <- new("IntervalTree", data=m_ranges, root=list())
overlapQuery(m_interval_tree, 4)
res <- overlapQuery(m_interval_tree, c(2.5,7))
res
```

treeFromInterval *treeFromInterval*

Description

Method for constructing interval tree for a list of Interval objects. A node in the tree is a list object. As the leftChild and rightChild of each node are nodes themselves, a binary interval tree stored in a recursive list will be produced if this function is executed successfully.

Usage

```
treeFromInterval(interval_list)
```

Arguments

interval_list	a list of Interval objects

Value

a list object representing a binary interval tree

Examples

```
i1 <- new("Interval", start=1.1,end=1.2, key="dummy1")
i2 <- new("Interval", start=-1.1,end=1.2, key="dummy2")
i3 <- new("Interval", start=-10.1,end=-1.2, key="dummy3")
i4 <- new("Interval", start=-1.1,end=1.2, key="dummy4")
i5 <- new("Interval", start=-10,end=2, key="dummy5")
i6 <- new("Interval", start=-8,end=-5, key="dummy6")
myList <- list(i1, i2, i3, i4, i5, i6)
atree <- treeFromInterval(myList)
```

Index