Package ‘roverlaps’

May 16, 2018

Type Package

Title Fast, Memory-Efficient Genomic Range Overlaps

Version 0.1.1

Date 2018-05-13

Maintainer Jeremiah Wala <jwala@broadinstitute.org>

Description Range overlaps between different sets of genomic intervals can be memory intensive and slow for huge (1M+) interval queries. This package implements fast, memory-efficient interval tree overlaps in C++ and provides flexibility for users to choose only what they need, thereby improving memory performance.

License MIT + file LICENSE

SystemRequirements A C++11 compiler. Version 4.6.* of g++ (as currently in Rtools) is insufficient; versions 4.8.*, 4.9.* or later will be fine.

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Imports Rcpp (>= 0.12.16), GenomicRanges, data.table, S4Vectors, methods

LinkingTo Rcpp

Suggests testthat

R topics documented:

`rcovered` *Return logical of whether a query region intersects subject*

Description

Determines which members of a query set of regions has any overlap with a set of subject intervals. The query does not have to be fully contained within the subject to be counted as `TRUE` overlap.

Usage

```
rcovered(query, subject, verbose = FALSE)
```

Arguments

`query`	Set of query intervals to evaluate for membership in `subject`
`subject`	Set of subject intervals to check `query` against
`verbose`	Set the verbosity `[FALSE]`

Value

Logical vector of same length as `query`, `TRUE` if interval overlaps subject

Examples

```
library(data.table)
o1 <- data.table(seqnames=factor(c("1","2")),start=1,end=5)
o2 <- data.table(seqnames=factor(c("1","1")),start=3,end=5)

## useful for subsetting in one line
o1 <- o1[rcovered(o1,o2)]

## output
#    seqnames start end
# 1:        1     1   5
```

`roverlaps` *Fast overlaps of* `data.table` *or* `GRanges`

Description

Performs interval overlaps between two genomic ranges, returning the intersecting set of ranges and the indicies of the query and subject which created the interval.

Usage

```
roverlaps(query, subject, verbose = FALSE, index_only = FALSE)
```

Arguments

`query`	Query ranges as a `data.table` with mandatory fields `seqnames` and `start`
`subject`	Subject ranges as a `data.table` with mandatory fields `seqnames` and `start`
`verbose`	Increase the verbosity `[FALSE]`
`index_only`	Return only the indicies ('query.id' and 'subject.id') `[FALSE]`

Value

data.table ('seqnames', 'start', 'end', 'query.id', 'subject.id') of overlaps

Note

Positions in ranges are inclusive. Example: chr1:2-5 (query) and chr1:5-7 (subject) will create an overlap with value chr1:5-5.

Examples

```
library(data.table)
set.seed(42)
C=10000
sn1 <- factor(c(1:22, "X")[sample(seq(23),C, replace=TRUE)])
s1  <- sample(seq(100000), C, replace=TRUE)
o1 <- data.table(seqnames=sn1, start=s1)
o1[, end := start + 100]
sn2 <- factor(c(1:22, "X")[sample(seq(23),C, replace=TRUE)])
s2  <- sample(seq(100000), C, replace=TRUE)
o2 <- data.table(seqnames=sn2, start=s2)
o2[, end := start + 100]
o <- roverlaps(o1,o2)

## output
# seqnames start   end query.id subject.id
#        7 83405 83451        3       3315
#        7 83405 83463        3       3148
#        7 83485 83505        3       1022

oi <- roverlaps(o1, o2, index_only=TRUE)

## output
#    query.id subject.id
#           3       3315
#           3       3148
#           3       1022
```

Index