Package 'IntervalSurgeon'

May 7, 2026

Type Package

Title Operating on Integer-Bounded Intervals

Encoding UTF-8

Version 1.3

Date 2024-02-20

Author Daniel Greene

Maintainer Daniel Greene <dg333@cam.ac.uk>

Description Manipulate integer-bounded intervals including finding overlaps, piling and merging.

License GPL (>= 2)

Imports Rcpp (>= 0.12.4)

LinkingTo Rcpp

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.3.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-02-20 22:30:02 UTC

Contents

`IntervalSurgeon-package`
Operating on Integer-Bounded Intervals

Description

Manipulate integer-bounded intervals including finding overlaps, piling and merging.

Details

The DESCRIPTION file:

Package:	IntervalSurgeon
Type:	Package
Title:	Operating on Integer-Bounded Intervals
Encoding:	UTF-8
Version:	1.3
Date:	2024-02-20
Author:	Daniel Greene
Maintainer:	Daniel Greene <dg333@cam.ac.uk>
Description:	Manipulate integer-bounded intervals including finding overlaps, piling and merging.
License:	GPL (>= 2)
Imports:	Rcpp (>= 0.12.4)
LinkingTo:	Rcpp
Suggests:	knitr, rmarkdown
VignetteBuilder:	knitr
RoxygenNote:	7.3.1

Index of help topics:

sections	Get the sections from a set of interval breaks
stitch	Stich together touching intervals and remove empty intervals

IntervalSurgeon presents functions for manipulating integer-bounded sets of intervals. Sets of intervals are represented by two-column matrices, where inclusive start points are stored in the first column, and exclusive end points in the second. A central concept in the package is the 'sections' of a set of intervals x: the non-overlapping, completely-covering set of intervals on the range of x, formed by making intervals between the consecutive sorted start/end points of the intevals in x. The function sections returns such a set of intervals given an input set.

Author(s)

Daniel Greene

Maintainer: Daniel Greene <dg333@cam.ac.uk>

---

annotate *Annotate one set of intervals with the names of those which intersect with the other*

Description

Create a list of vectors of indices/names of intervals/points in annotation (if annotation is a two-column matrix/vector respectively) which intersect with each interval/point in x (if x is a two-column matrix/vector respectively).

Usage

```
annotate(x, annotation)
```

Arguments

x	Integer matrix of two columns, the first column giving the (inclusive) start points of intervals and the second column giving the corresponding (exclusive) end points, or, an integer vector specifying the location of points.
annotation	Matrix specifying intervals or vector specifying points with which to annotate x.

Value

List of vectors of indices of overlapping intervals/points.

Examples

```
annotate(rbind(A=c(1, 100), B=c(50, 100)), rbind(a=c(1, 2), b=c(49, 51), c=c(50, 200)))
annotate(rbind(A=c(1, 100), B=c(50, 100)), c(a=1, b=49, c=51, d=100))
```

breaks *Get break points for set of intervals*

Description

Get the sorted set start points and end points for a set of intervals specified as an integer matrix.

Usage

```
breaks(x)
```

Arguments

x	Integer matrix of two columns, the first column giving the (inclusive) start points of intervals and the second column giving the corresponding (exclusive) end points.

Value

Ordered integer vector of unique interval start/end points.

Examples

```
breaks(cbind(2*1:5, 3*1:5))
```

depth *Depth of piled intervals*

Description

Get the depth of piled intervals for each section in the sections of x (see sections).

Usage

```
depth(x, include_intervals = FALSE)
```

Arguments

x	Integer matrix of two columns, the first column giving the (inclusive) start points of intervals and the second column giving the corresponding (exclusive) end points.
include_intervals	Logical value determining whether the function should return a vector of depths at each 'section' in the range of x (see sections), or a list with properties intervals and depths specifying the intervals of the sections and the corresponding depths respectively.

Value

Integer vector giving depth of piled intervals from `x` (within each sub-interval) or list containing a property `"intervals"`, a matrix of sections, and property `"depths"`, giving the corresponding pile depths.

Examples

```
depth(cbind(1:10, 11:20))
```

---

detached_sorted_nonempty

Check intervals are detached, sorted and non-empty.

Description

Check that `x` is an integer matrix specifying intervals, that the specified intervals are detached (i.e. non-overlapping/disjoint and non-touching) and that it is sorted (given that the intervals are detached, sorting by start position gives a unique result), and that the start points are greater than the end points (i.e. that they are non-empty/the lengths of all intervals is greater than zero).

Usage

```
detached_sorted_nonempty(x)
```

Arguments

`x`	Integer matrix of two columns, the first column giving the (inclusive) start points of intervals and the second column giving the corresponding (exclusive) end points.

Value

Boolean value.

Examples

```
detached_sorted_nonempty(cbind(1:2, 2:3))
detached_sorted_nonempty(cbind(c(1, 3), c(2, 4)))
detached_sorted_nonempty(cbind(1, 1))
```

flatten *Flatten a set of intervals*

Description

For a given set of intervals compute the set of intervals where there is overlap with at least one from the given. The resulting intervals are sorted and detached.

Usage

```
flatten(x)
```

Arguments

x	Integer matrix of two columns, the first column giving the (inclusive) start points of intervals and the second column giving the corresponding (exclusive) end points.

Value

Intervals represented by integer matrix of two columns.

Examples

```
flatten(rbind(c(1, 3), c(2, 4), c(5, 6)))
```

intersected *Determine whether each interval in a given set are intersected/covered by intervals in another set*

Description

Compute a logical vector indicating whether corresponding intervals specified by x overlap (intersected)/are covered by (covered) those in by_intervals.

Usage

```
intersected(x, by_intervals)

covered(x, by_intervals)
```

Arguments

x	Integer matrix of two columns, the first column giving the (inclusive) start points of intervals and the second column giving the corresponding (exclusive) end points, or, an integer vector specifying the location of points.
by_intervals	Matrix specifying intervals to test for intersection of.

Value

Logical vector with elements corresponding to rows of x.

Examples

```
intersected(rbind(c(1, 2), c(49, 51), c(50, 200)), rbind(c(50, 100)))
covered(rbind(c(1, 10), c(49, 51), c(50, 200)), rbind(c(2, 60)))
```

---

join *Get all overlapping tuples of intervals from multiple sets*

Description

Get matrix specifying overlapping tuples of intervals from multiple sets. Each row specifies an overlapping tuple. The nth element in a row contains the row index of the interval in the nth set of intervals passed to the function. Depending on the value of the output argument, there may two additional columns giving the start and end coordinates of the overlap (the default: output="intervals", no extra columns (output="indices") or one additional column giving the row index of the 'section' of the complete set of intervals (output="sections", see sections).

Usage

```
join(..., output = "intervals")
```

Arguments

...	Integer matrices of two columns, the first column giving the (inclusive) start points of intervals and the second column giving the corresponding (exclusive) end points.
output	Character value, one of "intervals", "indices" and "sections".

Value

Integer matrix.

Examples

```
join(rbind(c(1, 100), c(50, 100)), rbind(c(1, 2), c(49, 51), c(50, 200)))
```

overlaps *Compute overlaps of two sets of detached and sorted intervals*

Description

Find intervals satisfying particular conditions, including corresponding base R functions `intersect` (i.e. find intersections of intervals), `union` (i.e. unions of intervals) and `setdiff` (i.e. finding intervals which are contained in one set of intervals but not another).

Usage

```
overlaps(x, y, check = TRUE, in_x = TRUE, in_y = TRUE, op = "and")

intersects(x, y, ...)

unions(x, y, ...)

setdiffs(x, y, ...)
```

Arguments

x	Integer matrix of two columns, the first column giving the (inclusive) start points of intervals and the second column giving the corresponding (exclusive) end points.
y	Same as x.
check	Boolean value determining whether to check that the intervals specified in arguments x and y are sorted and non-overlapping (uses function detached_sorted_nonempty). Defaults to TRUE, but setting to FALSE may allow faster execution.
in_x	Boolean value determining whether to flag TRUE on intervals contained in x.
in_y	Boolean value determining whether to flag TRUE on intervals contained in y.
op	Character value specifying operator used to combine flags for each interval, either "and" or "or".
...	Additional arguments to be passed to overlaps.

Value

Intervals represented by integer matrix of two columns.

Examples

```
intersects(cbind(1, 3), cbind(2, 4))
setdiffs(cbind(1, 3), cbind(2, 4))
unions(cbind(1, 3), cbind(2, 4))
```

pile *Get IDs of intervals covering each sub-interval*

Description

Get the intervals overlapping each section as a list.

Usage

```
pile(x, interval_names = rownames(x), output = "list")
```

Arguments

x	Integer matrix of two columns, the first column giving the (inclusive) start points of intervals and the second column giving the corresponding (exclusive) end points.
interval_names	Character vector of names for each interval, not necessarily unique. If they are not unique, one might wish to lapply unique to the list of members for each sub-interval returned by this function. Defaults to the rownames of x.
output	Character value either "list" or "vector" determining whether a named list of interval index/name vectors or flat vector of members (corresponding to the output of depth) is returned.

Value

See notes on output parameter.

Examples

```
pile(cbind(1:10, 11:20))
```

proportion_overlap *Calculate proportion overlapping of intersecting intervals*

Description

Proportion overlapping is calculated as the size of the intersection of intervals, divided by the size of the union.

Usage

```
proportion_overlap(...)
```

Arguments

...	Interval matrices (passed to join).

Value

`data.frame` containing integer columns corresponding to indices of intervals within the input matrices and a final numeric column called `proportion_overlap` containing the fraction of the size of the intersection within the union.

Examples

```
proportion_overlap(rbind(c(1, 2), c(49, 51), c(50, 200)), rbind(c(50, 100)))
```

---

sections *Get the sections from a set of interval breaks*

Description

Given a set of interval breaks (see `breaks`), generate a new set of intervals, the 'sections', which partitions the full range of the given set, with an interval between every 'break' (i.e. start/end point) in the given set.

Usage

```
sections(x)
```

Arguments

`x`	Sorted integer vector.

Value

Intervals represented by integer matrix of two columns.

Examples

```
sections(1:10)
```

---

stitch *Stich together touching intervals and remove empty intervals*

Description

Given an integer matrix specifying disjoint intervals sorted by start position, merge intervals with matching start and ends, and remove intervals of length zero.

Usage

```
stitch(x)
```

Arguments

x	Integer matrix of two columns, the first column giving the (inclusive) start points of intervals and the second column giving the corresponding (exclusive) end points.

Value

Intervals represented by integer matrix of two columns.

Examples

```
stitch(cbind(1:2, 2:3))
```

Index