cluster_interval

Cluster ranges which are implemented as 2 equal-length numeric vectors.

Description

Cluster ranges which are implemented as 2 equal-length numeric vectors.

Usage

cluster_interval(starts, ends, max_distance = 0L)

Arguments

starts A numeric vector that defines the starts of each interval
ends A numeric vector that defines the ends of each interval
max_distance The maximum distance up to which intervals are still considered to be the same cluster. Default: 0.

Examples

starts <- c(50, 100, 120)
ends <- c(75, 130, 150)
j <- cluster_interval(starts, ends)
j == c(0,1,1)
**genome_cluster**

**Intersect data frames based on chromosome, start and end.**

**Description**

Intersect data frames based on chromosome, start and end.

**Usage**

```r
genie_cluster(x, by = NULL, max_distance = 0,
              cluster_column_name = "cluster_id")
```

**Arguments**

- `x`: A dataframe.
- `by`: A character vector with 3 entries which are the chromosome, start and end column. For example: `by=c("chr", "start", "end")`
- `max_distance`: The maximum distance up to which intervals are still considered to be the same cluster. Default: 0.
- `cluster_column_name`: A string that is used as the new column name.

**Value**

The dataframe with the additional column of the cluster.

**Examples**

```r
library(dplyr)
x1 <- data.frame(id = 1:4, ble=letters[1:4],
                  chromosome = c("chr1", "chr1", "chr2", "chr1"),
                  start = c(100, 120, 300, 260),
                  end = c(150, 250, 350, 450))
genome_cluster(x1, by=c("chromosome", "start", "end"))
genome_cluster(x1, by=c("chromosome", "start", "end"), max_distance=10)
```
genome_complement  Calculates the complement to the intervals covered by the intervals in a data frame. It can optionally take a chromosome_size data frame that contains 2 or 3 columns, the first the names of chromosome and in case there are 2 columns the size or first the start index and lastly the end index on the chromosome.

Description

Calculates the complement to the intervals covered by the intervals in a data frame. It can optionally take a chromosome_size data frame that contains 2 or 3 columns, the first the names of chromosome and in case there are 2 columns the size or first the start index and lastly the end index on the chromosome.

Usage

```r
genome_complement(x, chromosome_size = NULL, by = NULL)
```

Arguments

- **x**
  A data frame for which the complement is calculated

- **chromosome_size**
  A dataframe with at least 2 columns that contains first the chromosome name and then the size of that chromosome. Can be NULL in which case the largest value per chromosome from x is used.

- **by**
  A character vector with 3 entries which are the chromosome, start and end column. For example: by=c("chr", "start", "end")

Examples

```r
library(dplyr)
x1 <- data.frame(id = 1:4, bla=letters[1:4],
  chromosome = c("chr1", "chr1", "chr2", "chr1"),
  start = c(100, 200, 300, 400),
  end = c(150, 250, 350, 450))
genome_complement(x1, by=c("chromosome", "start", "end"))
```
**Description**

Intersect data frames based on chromosome, start and end.

**Usage**

```r
genie_intersect(x, y, by = NULL, mode = "both")
```

**Arguments**

- **x**: A dataframe.
- **y**: A dataframe.
- **by**: A character vector with 3 entries which are used to match the chromosome, start and end column. For example: `by=c("Chromosome"="chr", "Start"="start", "End"="end")`
- **mode**: One of "both", "left", "right" or "anti".

**Value**

The intersected dataframe of x and y with the new boundaries.

**Examples**

```r
library(dplyr)
x1 <- data.frame(id = 1:4, bla=letters[1:4],
  chromosome = c("chr1", "chr1", "chr2", "chr2"),
  start = c(100, 200, 300, 400),
  end = c(150, 250, 350, 450))
x2 <- data.frame(id = 1:4, BLA=LETTERS[1:4],
  chromosome = c("chr1", "chr2", "chr2", "chr1"),
  start = c(140, 210, 400, 300),
  end = c(160, 240, 415, 320))
j <- genome_intersect(x1, x2, by=c("chromosome", "start", "end"), mode="both")
print(j)
```
**genome_join_closest**  

*Join intervals on chromosomes in data frames, to the closest partner*

**Description**

Join intervals on chromosomes in data frames, to the closest partner

**Usage**

```r
genie_join_closest(x, y, by = NULL, mode = "inner",  
distance_column_name = NULL, max_distance = Inf, select = "all")

genie_inner_join_closest(x, y, by = NULL, ...)

genie_left_join_closest(x, y, by = NULL, ...)

genie_right_join_closest(x, y, by = NULL, ...)

genie_full_join_closest(x, y, by = NULL, ...)

genie_semi_join_closest(x, y, by = NULL, ...)

genie_anti_join_closest(x, y, by = NULL, ...)
```

**Arguments**

- **x**  A dataframe.
- **y**  A dataframe.
- **by**  A character vector with 3 entries which are used to match the chromosome, start and end column. For example: `by=c("Chromosome"="chr", "Start"="start", "End"="end")`
- **mode**  One of "inner", "full", "left", "right", "semi" or "anti".
- **distance_column_name**  A string that is used as the new column name with the distance. If NULL no new column is added.
- **max_distance**  The maximum distance that is allowed to join 2 entries.
- **select**  A string that is passed on to `IRanges::distanceToNearest`, can either be all which means that in case that multiple intervals have the same distance all are reported, or arbitrary which means in that case one would be chosen at random.
- **...**  Additional arguments parsed on to `genome_join_closest`.

**Value**

The joined dataframe of `x` and `y`. 
Examples

library(dplyr)

x1 <- data.frame(id = 1:4, bla=letters[1:4],
              chromosome = c("chr1", "chr1", "chr2", "chr2"),
              start = c(100, 200, 300, 400),
              end = c(150, 250, 350, 450))

x2 <- data.frame(id = 1:4, BLA=LETTERS[1:4],
              chromosome = c("chr1", "chr2", "chr2", "chr1"),
              start = c(140, 210, 400, 300),
              end = c(160, 240, 415, 320))

j <- genome_intersect(x1, x2, by=c("chromosome", "start", "end"), mode="both")
print(j)

---

**genome_intersect**

Subtract one data frame from another based on chromosome, start and end.

**Arguments**

- **x** A dataframe.
- **y** A dataframe.
- **by** A character vector with 3 entries which are used to match the chromosome, start and end column. For example: `by=c("Chromosome"="chr", "Start"="start", "End"="end")`

**Value**

The subtracted dataframe of x and y with the new boundaries.

**Usage**

`genome_intersect(x, y, by = NULL)`

**Examples**

library(dplyr)

x1 <- data.frame(id = 1:4, bla=letters[1:4],
              chromosome = c("chr1", "chr1", "chr2", "chr2"),
              start = c(100, 200, 300, 400),
              end = c(150, 250, 350, 450))

x2 <- data.frame(id = 1:4, BLA=LETTERS[1:4],
              chromosome = c("chr1", "chr2", "chr2", "chr1"),
              start = c(140, 210, 400, 300),
              end = c(160, 240, 415, 320))

j <- genome_intersect(x1, x2, by=c("chromosome", "start", "end"), mode="both")
print(j)
x2 <- data.frame(id = 1:4, BLA = LETTERS[1:4],
                 chromosome = c("chr1", "chr2", "chr1", "chr1"),
                 start = c(120, 210, 300, 400),
                 end = c(125, 240, 320, 415))

j <- genome_subtract(x1, x2, by = c("chromosome", "start", "end"))
print(j)
Index

cluster_interval, 2

gene_anti_join_closest
  (genome_join_closest), 6

gene_cluster, 3

gene_complement, 4

gene_full_join_closest
  (genome_join_closest), 6

gene_inner_join_closest
  (genome_join_closest), 6

gene_intersect, 5

gene_join_closest, 6

gene_left_join_closest
  (genome_join_closest), 6

gene_right_join_closest
  (genome_join_closest), 6

gene_semi_join_closest
  (genome_join_closest), 6

gene_subtract, 7