Package ‘tidygenomics’

August 8, 2019

Type Package

Title Tidy Verbs for Dealing with Genomic Data Frames

Version 0.1.2

Description Handle genomic data within data frames just as you would with 'GRanges'. This packages provides method to deal with genomic intervals the `"tidy-way"` which makes it simpler to integrate in the the general data munging process. The API is inspired by the popular `bedtools` and the `genome_join()` method from the `fuzzyjoin` package.

URL https://github.com/const-ae/tidygenomics

License GPL-3

Encoding UTF-8

LazyData true

Imports dplyr, rlang, purrr, tidyr, fuzzyjoin (>= 0.1.3), IRanges, Rcpp

Suggests testthat, knitr, rmarkdown

RoxygenNote 6.1.1

LinkingTo Rcpp

VignetteBuilder knitr

NeedsCompilation yes

Author Constantin Ahlmann-Eltze [aut, cre] (<https://orcid.org/0000-0002-3762-068X>), Stan Developers [cph] (Code from the Stan Math library is reused in 'cluster_interval.cpp'), David Robinson [cph] (Code from the fuzzyjoin package is reused)

Maintainer Constantin Ahlmann-Eltze <artjom31415@googlemail.com>

Repository CRAN

Date/Publication 2019-08-08 11:50:02 UTC
**R topics documented:**

- `cluster_interval` ................................................. 2
- `genome_cluster` .................................................. 3
- `genome_complement` .............................................. 4
- `genome_intersect` ................................................ 5
- `genome_join_closest` ............................................ 6
- `genome_subtract` .................................................. 7

---

**Description**

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

**Usage**

```r
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**

```r
starts <- c(50, 100, 120)
ends <- c(75, 130, 150)
j <- cluster_interval(starts, ends)
j == c(0,1,1)
```
Intersect data frames based on chromosome, start and end.

Usage

```r
genome_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, bla=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

gene_complement(x, chromosome_size = NULL, by = NULL)

Arguments

x
A data frame for which the complement is calculated

chromosome_size
A data frame 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

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"))
genome_intersect

Intersect data frames based on chromosome, start and end.

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

Usage
genome_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
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
genome_join_closest(x, y, by = NULL, mode = "inner",
                distance_column_name = NULL, max_distance = Inf, select = "all")
```

```r
genome_inner_join_closest(x, y, by = NULL, ...)
```

```r
genome_left_join_closest(x, y, by = NULL, ...)
```

```r
genome_right_join_closest(x, y, by = NULL, ...)
```

```r
genome_full_join_closest(x, y, by = NULL, ...)
```

```r
genome_semi_join_closest(x, y, by = NULL, ...)
```

```r
genome_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)

gene_subtract

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

Description

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

Usage

gene_subtract(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")

Value

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

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", "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

genome_anti_join_closest
  (genome_join_closest), 6

genome_cluster, 3

genome_complement, 4

genome_full_join_closest
  (genome_join_closest), 6

genome_inner_join_closest
  (genome_join_closest), 6

genome_intersect, 5

genome_join_closest, 6

genome_left_join_closest
  (genome_join_closest), 6

genome_right_join_closest
  (genome_join_closest), 6

genome_semi_join_closest
  (genome_join_closest), 6

genome_subtract, 7