Package ‘GenomicInteractions’

April 23, 2016

Type Package

Title R package for handling genomic interaction data

URL https://github.com/ComputationalRegulatoryGenomicsICL/GenomicInteractions/

Version 1.4.2

Date 2015-07-20

Author Harmston, N., Ing-Simmons, E., Perry, M., Baresic, A., Lenhard, B.

Maintainer Malcolm Perry <mgperry32@gmail.com>, Liz Ing-Simmons <liz.ing-simmons12@csc.mrc.ac.uk>

Imports Rsamtools, GenomicRanges, IRanges, BiocGenerics (>= 0.15.3), data.table, stringr, GenomeInfoDb, ggplot2, grid, gridExtra, methods, igraph, S4Vectors, dplyr, Gviz

Suggests knitr, BiocStyle, testthat

VignetteBuilder knitr

Description R package for handling Genomic interaction data, such as ChIA-PET/Hi-C, annotating genomic features with interaction information and producing various plots / statistics.

biocViews Software,Infrastructure,DataImport,DataRepresentation,HiC

License GPL-3

Depends R (>= 2.10)

NeedsCompilation no

R topics documented:

`GenomicInteractions-package` *A package for looking at genomic interaction data.*

Description

A package for looking at genomic interaction data.

`annotateAnchors` *Annotate anchors*

Description

This function directly annotates a single set of anchors using the GRanges elementMetadata.

Usage

```
annotateAnchors(GIObject, oneOrTwo, name, dat)


  ## S4 method for signature 'GenomicInteractions,numeric,character,vector'
annotateAnchors(GIObject,
  oneOrTwo, name, dat)
```

Arguments

`GIObject`	A GenomicInteractions object
`oneOrTwo`	An integer indicating which anchor to annotate
`name`	Character. Will be used as a column name for the elementMetadata of the annotated anchor.
`dat`	Vector of the same length as the GenomicInteractions object, containing data with which to annotate the object.

Value

invisible(1)

annotateInteractions *Annotate the interactions in a GenomicInteractions object*

Description

This function will annotate both anchors with a list of named GRanges objects. Each metadata column is labeled "name.id" and contains the id of the genomic interval(s) it overlaps. Anonymous lists will be given names "FEATURE#.id" where # is the position in the list.

Usage

```
annotateInteractions(GIObject, annotations)

## S4 method for signature 'GenomicInteractions,list'
annotateInteractions(GIObject, annotations)
```

Arguments

`GIObject`	A GenomicInteractions object to be annotated
`annotations`	A list containing GRanges (or GRangesList) objects with which to annotate the GenomicInteractions object.

Details

For each anchor a "node.class" metadata column will also be added, containing the name of the list element which was *first* annotated to each range. Ranges with no overlaps will be classified as "distal". The identifiers for each individual feature/annotation are taken from either the name of the list item in the case of a GRangesList or from either the names of a the provided GRanges or an id column in its associated metadata.

Value

invisible(1)

Examples

```
library("GenomicRanges")
data(hic_example_data)
data(mm9_refseq_promoters)
mm9_refseq_grl = split(mm9_refseq_promoters, mm9_refseq_promoters$id)
annotateInteractions(hic_example_data, list(promoter=mm9_refseq_grl))
```

availableDisplayPars — *The default display parameters for a track object class can be queries using the availableDisplayPars function.*

Description

The default display parameters for a track object class can be queries using the availableDisplayPars function.

Usage

```
availableDisplayPars(class)
```

Arguments

`class`	A valid track object class name, or the object itself, in which case the class is derived directly from it. This function provides the same functionality as `Gviz::availableDisplayPars` and allows the user to display the default display parameters for the `InteractionTrack` class. If the class of the track is not an `InteractionTrack` then the function calls the availableDisplayPars method in Gviz.

Value

returns a list of the default display parameters.

c — *Combine GenomicInteractions Methods*

Description

This method will fail if the seqlengths of the objects to be combined do not match. If some chromosomes appear in one set of seqinfo but not the other, the seqinfo will be merged.

Usage

```
## S4 method for signature 'GenomicInteractions'
c(x, ..., ignore.mcols = FALSE,
  recursive = FALSE)
```

Arguments

`x,...`	GenomicInteractions objects to be concatenated
`ignore.mcols`	Logical, default FALSE, remove mcols in combined object.
`recursive`	Not supported

Value

A GenomicInteractions object.

`calculateDistances` *Calculate interaction distances*

Description

This function takes a GenomicInteractions object and calculates the distances between the anchors according to the value of `method`. The distances returned follow the same convention as distance(x, y) in GenomicRanges where the distance between adjacent regions is 0. Note that if anchors are overlapping this method will print a warning and return the distance as 0.

Usage

```
calculateDistances(GIObject, method = "midpoint", floor = TRUE)

## S4 method for signature 'GenomicInteractions'
calculateDistances(GIObject,
  method = "midpoint", floor = TRUE)
```

Arguments

`GIObject`	A GenomicInteractions object
`method`	Character vector indicating how to calculate distances, must be one of 'midpoint', 'outer', 'inner'.
`floor`	A logical specifying whether to round down distances to nearest base pair or not. Default TRUE.

Value

An vector containing the distances between anchors/GRanges, NA if on different chromosomes, rounded down to the nearest bp.

Examples

```
library(GenomicRanges)

anchor.one = GRanges(c("chr1", "chr1", "chr1", "chr1"), IRanges(c(10, 20, 30, 20), width=5))
anchor.two = GRanges(c("chr1", "chr1", "chr1", "chr2"), IRanges(c(100, 200, 300, 50), width=5))
interaction_counts = sample(1:10, 4)
test <- GenomicInteractions(anchor.one, anchor.two, experiment_name="test",
                            description="this is a test", counts=interaction_counts)
calculateDistances(test, method="midpoint")
```

`capitalize` *Capitalize first letter of string*

Description

This function will capitalize the first letter of each string in a character vector, and lowercase following letters.

Usage

```
capitalize(x)
```

Arguments

`x`	A character vector

Value

a string with the first letter capitalised

`categoriseInteractions` *Get the numbers of interaction types existing in your data*

Description

Get the numbers of interaction types existing in your data

Usage

```
categoriseInteractions(GIObject, node.classes = NULL, viewpoints = NULL)
```

Arguments

`GIObject`	A GenomicInteractions object
`node.classes`	Optional. All node.classes to include in the analysis. Default: all node classes.
`viewpoints`	Optional. If set will only consider interactions where at least one anchor is of this node class. Default: all classes in node.classes.

Value

A data.frame.

Examples

```
data(hic_example_data)
categoriseInteractions(hic_example_data)
```

`countsBetweenAnchors` *Summarise Interactions between defined anchors*

Description

Calculate the number of of paired-end reads mapping between a defined set of anchors. This function will ignore counts present in the input data.

Usage

```
countsBetweenAnchors(x, y, ...)

## S4 method for signature 'GenomicInteractions,GRanges'
countsBetweenAnchors(x, y,
  ignore_overlaps = FALSE, ...)
```

Arguments

`x`	A GenomicInteractions object
`y`	A GenomicRanges object
`...`	Extra parameters to pass to findOverlaps
`ignore_overlaps`	Allow overlapping anchors. Use this when you have overlapping anchors but be careful with multi-mapping. The "within" option can help with this.

Value

A GenomicInteractions object with annotated counts between anchors

`duplicated,GenomicInteractions-method` *duplicated, GenomicInteractions-method*

Description

Finds duplicated interactions in a GenomicInteractions object.

Usage

```
## S4 method for signature 'GenomicInteractions'
duplicated(x, fromLast = FALSE,
  dropMetadata = FALSE)
```

Arguments

`x`	A GenomicInteractions object
`fromLast`	Whether to identify duplicates starting from last item in the Genomicinteractions object or not. Default: FALSE.
`dropMetadata`	Logical, default FALSE. Whether to drop interaction mcols when considering unique interactions.

Details

Uniqueness is based on anchor positions and metadata, interaction counts, and interaction metadata.

Value

A vector containing indices of duplicated interactions

`export.bed12` *Export interactions in BED12 format.*

Description

Export interactions in BED12 format.

Usage

```
export.bed12(GIObject, fn = NULL, score = "counts", drop.trans = c(FALSE,
  TRUE))

## S4 method for signature 'GenomicInteractions'
export.bed12(GIObject, fn = NULL,
  score = "counts", drop.trans = c(FALSE, TRUE))
```

Arguments

`GIObject`	A GenomicInteractions object.
`fn`	A filename to write the object to
`score`	Which metadata column to export as score
`drop.trans`	Logical indicating whether to drop trans interactions. Exports a GenomicInteractions object to BED12 format, and writes to a specified file. If filename is not specified, then a data.frame containing the information is returned. Bed12 files provide a method for visualising interactions, it is not a good format for storing all of the data associated with an interaction dataset, particularly for trans-chromosomal interactions, which can only be stored in the bed12 names field.

Value

invisible(1) if outputting to file or a data.frame containing all of the corresponding information

---

`export.bedpe` *Export interactions in BED Paired-End format.*

Description

#' Exports a GenomicInteractions object to BED-PE format, and writes to a specified file. If filename is not specified, then a data.frame containing the information is returned. The value of the score parameter defines which field is used to populate the score field.

Usage

```
export.bedpe(GIObject, fn = NULL, score = "counts")

## S4 method for signature 'GenomicInteractions'
export.bedpe(GIObject, fn = NULL,
  score = "counts")
```

Arguments

`GIObject`	A GenomicInteractions object.
`fn`	A filename to write the interactions data to
`score`	Which metadata column to use as score

Value

invisible(1) if outputting to file or a data.frame containing all of the corresponding information

---

`export.chiasig` *Export interactions in a BEDPE-like format for use with ChiaSig*

Description

Exports a GenomicInteractions object to BEDPE like format, (anchor specifications and a column for reads connecting them) and writes to a specified file. If filename is not specified, then a data.frame containing the information is returned. The value of the score parameter defines which field is used to populate the score field.

Usage

```
export.chiasig(GIObject, fn = NULL, score = "counts")

## S4 method for signature 'GenomicInteractions'
export.chiasig(GIObject, fn = NULL,
  score = "counts")
```

Arguments

`GIObject`	A GenomicInteractions object.
`fn`	A filename to write the interactions data to
`score`	Which metadata column to use as the score: counts or normalised

Value

invisible(1) if outputting to file or a data.frame containing all of the corresponding information

---

`export.igraph` *Export interactions to an igraph object.*

Description

Exports a GenomicInteractions object to graph.data.frame for use by igraph package. This uses unique anchors as nodes and generates edges between them. For the resulting graph to be easily interpretable, anchors should be non-overlapping. This should already be the case for HiC data (either binned or restriction fragments), however ChIA-PET data can contain overlapping anchors, which may need to be reduced to non-overlapping regions before graph export.

Usage

```
export.igraph(GIObject)

## S4 method for signature 'GenomicInteractions'
export.igraph(GIObject)
```

Arguments

`GIObject`	A GenomicInteractions object.

Value

a graph.data.frame representation of the GenomicInteractions object

findOverlaps *Find overlaps between GRanges and GenomicInteractions objects*

Description

When called with a GRanges and a GenomicInteractions object, this function calls findOverlaps separately on each anchor and returns a list. countOverlaps and overlapsAny return a list of integer vectors and logical vectors respectively.

Usage

```
## S4 method for signature 'GenomicInteractions,GRanges'
findOverlaps(query, subject,
  maxgap = 0L, minoverlap = 1L, type = c("any", "start", "end", "within",
  "equal"), select = c("all", "first", "last", "arbitrary"),
  ignore.strand = FALSE)

## S4 method for signature 'GRanges,GenomicInteractions'
findOverlaps(query, subject,
  maxgap = 0L, minoverlap = 1L, type = c("any", "start", "end", "within",
  "equal"), select = c("all", "first", "last", "arbitrary"),
  ignore.strand = FALSE)

## S4 method for signature 'GenomicInteractions,GRanges'
countOverlaps(query, subject,
  maxgap = 0L, minoverlap = 1L, select = c("all", "first", "last",
  "arbitrary"), type = c("any", "start", "end", "within", "equal"),
  ignore.strand = FALSE)

## S4 method for signature 'GRanges,GenomicInteractions'
countOverlaps(query, subject,
  maxgap = 0L, minoverlap = 1L, type = c("any", "start", "end", "within",
  "equal"), select = c("all", "first", "last", "arbitrary"),
  ignore.strand = FALSE)

## S4 method for signature 'GenomicInteractions,GRanges'
overlapsAny(query, subject,
  maxgap = 0L, minoverlap = 1L, type = c("any", "start", "end", "within",
  "equal"), ignore.strand = FALSE)

## S4 method for signature 'GRanges,GenomicInteractions'
overlapsAny(query, subject,
  maxgap = 0L, minoverlap = 1L, type = c("any", "start", "end", "within",
  "equal"), ignore.strand = FALSE)

## S4 method for signature 'GenomicInteractions,GenomicInteractions'
findOverlaps(query, subject)
```

Arguments

`query`	GenomicInteractions or GRanges
`subject`	GRanges or GenomicInteractions
`maxgap,minoverlap,type,select,ignore.strand`	See 'findOverlaps' in the IRanges package.

Details

When

See 'findOverlaps' in the GenomicRanges package for detailed documentation for this function.

Value

A Hits object or a list containing Hits objects for both anchors.

`GenomicInteractions` *Function to create a GenomicInteraction object*

Description

Create GenomicInteraction objects from two GRanges ojects.

Usage

```
GenomicInteractions(anchor_one = GRanges(), anchor_two = GRanges(),
  counts = integer(), experiment_name = NULL, description = NULL, ...)
```

Arguments

`anchor_one,anchor_two`	GRanges objects.
`counts`	An integer vector, defaults to 1.
`experiment_name`	Experiment name.
`description`	Description of experiment.
`...`	Additional data to be added to mcols

Value

a GenomicInteractions object

Examples

```
library(GenomicRanges)

anchor.one = GRanges(c("chr1", "chr1", "chr1", "chr1"), IRanges(c(10, 20, 30, 20), width=5))
anchor.two = GRanges(c("chr1", "chr1", "chr1", "chr2"), IRanges(c(100, 200, 300, 50), width=5))
interaction_counts = sample(1:10, 4)
test <- GenomicInteractions(anchor.one, anchor.two, experiment_name="test",
                            description="this is a test", counts=interaction_counts)
```

`GenomicInteractions-class` *A S4 class to represent interactions between genomic regions.*

Description

A S4 class to represent interactions between genomic regions.

Slots

`metadata` List, defaults to "experiment_name" and "description", inherited from S4Vectors::Vector

`anchor_one,anchor_two` GRanges. Set of anchors of interactions.

`counts` integer vector, contains raw counts

`elementMetadata` DataFrame

This class is used to store information on which genomic regions are interacting with each other. Objects of this class contain information of the genomic coordinates of the interacting regions and the strength of these interactions, and associated metadata such as the name of the dataset and a brief description of the dataset. Interacting regions are stored as a pair of GenomicRanges: each set of anchor regions is stored as a separate GenomicRanges object, accessed by `getAnchorOne` and `getAnchorTwo`.

Examples

```
showClass("GenomicInteractions")
library(GenomicRanges)

anchor.one = GRanges(c("chr1", "chr1", "chr1", "chr1"), IRanges(c(10, 20, 30, 20), width=5))
anchor.two = GRanges(c("chr1", "chr1", "chr1", "chr2"), IRanges(c(100, 200, 300, 50), width=5))
interaction_counts = sample(1:10, 4)
test <- GenomicInteractions(anchor.one, anchor.two, experiment_name="test",
                            description="this is a test", counts=interaction_counts)
```

`getters` *Functions to access data held in a GenomicInteractions object.*

Description

Use these functions to access data stored in each of the slots of a GenomicInteractions object.

Usage

```
name(GIObject)

description(GIObject)

anchorOne(GIObject)

anchorTwo(GIObject)

interactionCounts(GIObject)

annotationFeatures(GIObject)

## S4 method for signature 'GenomicInteractions'
name(GIObject)

## S4 method for signature 'GenomicInteractions'
description(GIObject)

## S4 method for signature 'GenomicInteractions'
anchorOne(GIObject)

## S4 method for signature 'GenomicInteractions'
anchorTwo(GIObject)

## S4 method for signature 'GenomicInteractions'
interactionCounts(GIObject)

## S4 method for signature 'GenomicInteractions'
annotationFeatures(GIObject)
```

Arguments

`GIObject`	A GenomicInteractions object

Value

For 'anchorOne' and 'anchorTwo', a GRanges. For 'interactionCounts', a numeric vector with counts for each interaction in the object. For 'description' and 'name', a character vector with length

1. For 'annotationFeatures', a character vector of features with which the object was previously annotated, or 'NA' if the object is unannotated.

Examples

```
library(GenomicRanges)

anchor.one = GRanges(c("chr1", "chr1", "chr1", "chr1"), IRanges(c(10, 20, 30, 20), width=5))
anchor.two = GRanges(c("chr1", "chr1", "chr1", "chr2"), IRanges(c(100, 200, 300, 50), width=5))
interaction_counts = sample(1:10, 4)
test <- GenomicInteractions(anchor.one, anchor.two, experiment_name="test",
                            description="this is a test", counts=interaction_counts)

name(test)
description(test)
anchorOne(test)
anchorTwo(test)
interactionCounts(test)
```

---

get_binom_ligation_threshold

get self ligation threshold with binomial test

Description

This function calculates a self ligation threshold according to a method based on that of Heidari et al., Genome Research, 2014. Briefly, paired reads are divided into in evenly spaced bins. For each bin, the number of reads that are aligned to opposite strand vs to the same strand is calculated. A binomial test is used to test if this is significantly different from the 50:50 ratio expected by chance if all reads are real interactions.

Usage

```
get_binom_ligation_threshold(GIObject, max.distance = 20000, bin.size = 500,
  p.cutoff = 0.05, adjust = "fdr", plot = TRUE)
```

Arguments

`GIObject`	a GenomicInteractions object of paired end reads
`max.distance`	The maximum distance to consider between reads. Reads further apart than this distance should be very unlikely to be self ligations.
`bin.size`	Bin size in base pairs.
`p.cutoff`	P value cut off for a significant difference from 50:50. Default: 0.05
`adjust`	Method to use to adjust p values. Default: fdr. See 'help(p.adjust)' for accepted values. Can also be NA for no adjustment.
`plot`	TRUE by default. Whether to plot the percentage of reads on opposite strands vs difference and the binomial test p value vs distance.

Value

The cutoff in base pairs below which an interaction is likely to be a self ligation.

`get_self_ligation_threshold`
Get self ligation threshold with SD method from Heidari et al

---

Description

This function calculates a self ligation threshold according to the method published in Heidari et al., Genome Research, 2014. Briefly, paired reads are divided into in evenly sized bins. For each bin, the log2 ratio of reads that are aligned to opposite strand vs to the same strand is calculated. Twice the standard deviation of this ratio at high distances is used a cutoff to determine which bins are likely to contain mostly self-liagted reads.

Usage

```
get_self_ligation_threshold(GIObject, bins = 100, distance_th = 4e+05,
  plot = TRUE)
```

Arguments

`GIObject`	a GenomicInteractions object of paired end reads
`bins`	Number of evenly sized bins to use.
`distance_th`	The threshold, in base pairs, to use as a cutoff to pick which bins to use to determine the standard deviation.
`plot`	TRUE by default. Whether to plot the log2ratio of opposite to same strand reads vs distance.

Value

The cutoff in base pairs below which an interaction is likely to be a self ligation.

`hg19.refseq.transcripts`
Human Refseq transcripts from chr 17-18

Description

This dataset contains a subset of the transcripts from the Refseq annotation for mouse genome build hg19 See the ChIA-PET analysis vignette (`vignettes(GenomicInteractions)`) for more information on how this dataset was created.

Usage

```
data(hg19.refseq.transcripts)
```

Format

A GRanges object with length 2441.

Value

A GRanges object.

`hic_example_data` *Example HiC dataset*

Description

This dataset contains HiC data from Seitan et al. 2013. The data was analysed using HOMER (Heinz et al. 2010) at a resolution of 100kb to find significant interactions. This example dataset has been filtered to retain only interactions on chromosomes 14 and 15 with a FDR < 0.1. The data has also been annotated for overlaps with Refseq promoters. See the HiC analysis vignette (`vignettes(GenomicInteractions)`) for more information on how this dataset was created.

Usage

```
data(hic_example_data)
```

Format

A GenomicInteractions object with length 8171.

Value

GenomicInteractions object

References

Seitan, V. C. et al. Cohesin-based chromatin interactions enable regulated gene expression within pre-existing architectural compartments. Genome Res. 23, 2066-77 (2013).

Heinz S, Benner C, Spann N, Bertolino E et al. Simple Combinations of Lineage-Determining Transcription Factors Prime cis-Regulatory Elements Required for Macrophage and B Cell Identities. Mol Cell 2010 May 28;38(4):576-589.

`InteractionTrack` *Constructor to create an InteractionTrack object*

Description

Create InteractionTrack object from an GenomicInteractions object to visualise a specified chromosome.

Usage

```
InteractionTrack(x, chromosome = "", name = NULL, start = NULL,
  end = NULL)
```

Arguments

`x`	A GenomicInteractions object
`chromosome`	specify which chromosome to hold information on - can be null
`name`	specify the name of the track - if null takes it to be the name of the GenomicInteractions passed
`start`	specify which start location to hold information on - can be null
`end`	specify which end location to hold information on - can be null

Value

an InteractionTrack object

Examples

```
library(Gviz)

anchor.one = GRanges(c("chr1", "chr1", "chr1", "chr1"), IRanges(c(10, 20, 30, 20), width=5))
anchor.two = GRanges(c("chr1", "chr1", "chr1", "chr2"), IRanges(c(100, 200, 300, 50), width=5))
interaction_counts = sample(1:10, 4)
test <- GenomicInteractions(anchor.one, anchor.two, experiment_name="test",
                            description="this is a test", counts=interaction_counts)
interactions.track = InteractionTrack(name="Test", test, chromosome="chr1")
plotTracks(list(interactions.track), chromosome="chr1", from=0, to=500)
```

InteractionTrack-class

A class to hold chromatin interaction data for a specific genomic region.

Description

A class to hold chromatin interaction data for a specific genomic region.

Slots

`plottingFunction` function

`variables` list

`chromosome` chromosome

`stacking` character

InteractionTrack is a specific Gviz-derived class for enabling the visualisation of chromatin interaction data. The InteractionTrack class allows interactions on a specified chromosome to be visualised by examining interactions between anchors as bezier curves. The object is instantiated and used in a similar fashion to standard Gviz tracks and plotted using the `plotTracks`.

Several additional display parameters (i.e. `displayPars(foo)=list(...)` are defined for this class, including `plot.anchors` which can be used to specify whether anchors are to be drawn. `col.anchors.line` which can be used to alter the colour of border of these anchor elements and `col.anchors.fill` can be used to alter the fill colour of these elements. The value of `plot.outside` determines whether or not interactions which span outside of the window are to be plotted, and `col.outside` defines the colour of these interactions. Similarly `plot.trans` determines whether trans-interactions are plotted and `col.trans` specifies the colour of trans-interactions. By default, the height of an arc representing an interaction is proportional to the number of reads/counts supporting that interaction. Instead of using the counts to define this, the height can be set to be proportional to either `fdr` or `p.value` using the `interaction.measure` display parameter. By changing the `interaction.dimension` to width, the line widths of each arc now represent the statistic supporting them. The heights of the arcs can be made to be proportional to log10 of the supporting statistic by changing `interaction.dimension.transform` to log. `col.interactions` sets the colour of arcs representing interactions within the region of interest. It is possible to colour the arcs by the type of interaction they are involved in (i.e. promoter-promoter interactions etc) by setting the `col.interactions.types` display parameter to be a named vector of colours, where the name corresponds to the type of interaction. This is applicable to anchors regions through the use of the `col.anchors.line.node.class` and `col.anchors.fill.node.class` parameters.

is.pp *Interaction Type Helpers*

Description

Functions to classify interactions within GenomicInteractions objects.

- "isInteractionType" takes two character arguments which are annotated node classes and returns interactions between them.
- "is.pp", "is.pd" etc. are bindings for common annotations:

  p promoter

  d distal

  t terminator
- "is.trans" & "is.cis" select trans-chromosomal and intra-chromosomal interactions, respectively

Usage

```
is.pp(GIObject)

is.pd(GIObject)

is.pt(GIObject)

is.dd(GIObject)

is.dt(GIObject)

is.tt(GIObject)

isInteractionType(GIObject, x, y)

is.trans(GIObject)

is.cis(GIObject)

## S4 method for signature 'GenomicInteractions'
is.pp(GIObject)

## S4 method for signature 'GenomicInteractions'
is.pd(GIObject)

## S4 method for signature 'GenomicInteractions'
is.pt(GIObject)

## S4 method for signature 'GenomicInteractions'
```

```
is.dd(GIObject)

## S4 method for signature 'GenomicInteractions'
is.dt(GIObject)

## S4 method for signature 'GenomicInteractions'
is.tt(GIObject)

## S4 method for signature 'GenomicInteractions'
isInteractionType(GIObject, x, y)

## S4 method for signature 'GenomicInteractions'
is.trans(GIObject)

## S4 method for signature 'GenomicInteractions'
is.cis(GIObject)
```

Arguments

`GIObject`	A GenomicInteractions object
`x,y`	Names of annotated node classes

Value

A logical vector

`length,GenomicInteractions-method`

Get the length of a GenomicInteractions GIObject

Description

Get the length of a GenomicInteractions GIObject

Usage

```
## S4 method for signature 'GenomicInteractions'
length(x)
```

Arguments

`x`	GenomicInteractions GIObject

Value

A numeric vector containing the length of the GIObject

`makeGenomicInteractionsFromFile`
Function to create GenomicInteraction objects from a file

Description

Function to create GenomicInteraction objects from a variety of files. The resulting objects contain information on which genomic regions are interacting with each other, and the number of counts supporting each interaction. It is also possible to store information on associated p-values and false-discovery rates (FDR). It is possible to create GenomicInteractions objects for various datasets including Hi-C and ChIA-PET. It is possible to read interactions from a variety of files including BAM files, bed files (BED12 and BEDPE) and from the output from standard processing pipelines, such as HOMER and ChIA-PET tool. GenomicInteractions objects can also be created using calls of the form `new("GenomicInteractions", ...)`. For hiclib, it expects the directory in which the files extracted using h5dictToTxt.py from the hdf5 file are located, where as for all of the other file types it expects the full filename. Note that recent versions of hiclib (2015-) cannot export the required data and so this function will only work with older files.

Usage

```
makeGenomicInteractionsFromFile(fn, type, experiment_name = "",
  description = "", chr_names = NULL)
```

Arguments

`fn`	Filename or, if type="hiclib", folder
`type`	One of "chiapet.tool", "bed12", "bedpe", "hiclib", "homer", "bam", "two.bams".
`experiment_name`	Experiment name.
`description`	Description of experiment.
`chr_names`	a vector of chromosome names in order, required for re-naming chromosomes for hiclib import

Value

a GenomicInteractions object

Examples

```
k562.rep1 = makeGenomicInteractionsFromFile(file.path(system.file(package="GenomicInteractions"), "extdata", "k
            type="chiapet.tool", experiment_name="k562", description="k562 pol2 8wg16")

k562.rep1
```

`mm9_refseq_promoters` *Mouse Refseq promoters from chr 14-15*

Description

This dataset contains a subset of the promoters from the Refseq annotation for mouse genome build mm9. See the HiC analysis vignette (`vignettes(GenomicInteractions)`) for more information on how this dataset was created.

Usage

```
data(mm9_refseq_promoters)
```

Format

A GRanges object with length 2441.

Value

A GRanges object.

`plotAvgViewpoint` *Plot coverage around a set of virtual 4C viewpoints*

Description

Plots summarised coverage of interactions around a set of viewpoints, e.g. promoters. This function requires the output of ‘viewPoint()‘ as input.

Usage

```
plotAvgViewpoint(x, left_dist = 1e+05, right_dist = 1e+05,
  ylab = "Average signal", xlab = "Relative position", fix = "center",
  ...)
```

Arguments

`x`	A GenomicInteractions object which is output from viewPoint
`left_dist`	Distance ’left’ of interactions to consider, in bp.
`right_dist`	Distance ’right’ of interactions to consider, in bp.
`ylab`	Y axis label.
`xlab`	X axis label.
`fix`	One of "center", "start", "end". Passed to ‘resize‘. Interaction distances are calculated relative to this part of the bait.
`...`	additional arguments to plot

Value

Coverage that is plotted (invisibly)

---

plotCisTrans — *Plots the percentages of cis and trans interactions for a GenomicInteractions object as a donut plot.*

Description

Plots the percentages of cis and trans interactions for a GenomicInteractions object as a donut plot.

Usage

```
plotCisTrans(GIObject)
```

Arguments

`GIObject`	A GenomicInteractions object

Value

A ggplot2 plot

Examples

```
data(hic_example_data)
plotCisTrans(hic_example_data)
```

---

plotCounts — *Plot a bar chart of the number of interactions supported by different numbers of reads in your data.*

Description

Plot a bar chart of the number of interactions supported by different numbers of reads in your data.

Usage

```
plotCounts(GIObject, normalise = FALSE, cut = 10)
```

Arguments

`GIObject`	A GenomicInteractions object.
`normalise`	Logical. If TRUE, plots proportion of total reads instead of count.
`cut`	Numeric, can be NULL. Default: 10. All interactions with counts > cut are consolidated into a single category.

Value

A ggplot2 plot

Examples

```
data(hic_example_data)
plotCounts(hic_example_data)
plotCounts(hic_example_data, normalise=TRUE)
```

---

plotDists *Plots a histogram of interaction distances for a GenomicInteractions Object*

Description

Plots a histogram of interaction distances for a GenomicInteractions Object

Usage

```
plotDists(GIObject, breaks = c(0, 1000, 5000, 10000, 50000, 1e+05, 5e+05,
  1e+06, 2e+06), method = "midpoint")
```

Arguments

`GIObject`	A GenomicInteractions object
`breaks`	A numeric vector of breaks for the histogram
`method`	Method used for distance between anchors. Passed to calculateDistances. One of "midpoint", "inner", or "outer".

Value

A ggplot2 plot

Examples

```
data(hic_example_data)
plotDists(hic_example_data)
```

plotInteractionAnnotations

Plot a donut plot of interaction types for an annotated GenomicInteractions object

Description

Plot a donut plot of interaction types for an annotated GenomicInteractions object

Usage

```
plotInteractionAnnotations(GIObject, node.classes = NULL, viewpoints = NULL,
  other = 0, keep.order = FALSE, legend = FALSE)
```

Arguments

`GIObject`	A GenomicInteractions object
`node.classes`	Optional. All node.classes to include in the analysis. Default: all node classes.
`viewpoints`	Optional. If set will only consider interactions where at least one anchor is of this node class. Default: all classes in node.classes.
`other`	Optional. Interaction types making up fewer than "other" percent of the total interactions will be consolidated into a single "other" category.
`keep.order`	Optional. Logical. Keep original order of node.classes for plotting or not. Default: FALSE, alphabetical order.
`legend`	Optional. Logical. If TRUE, legend is plotted to right of donut plot. If FALSE, donut plot is annotated with category names.

Value

A ggplot2 plot

Examples

```
data(hic_example_data)
plotInteractionAnnotations(hic_example_data)
```

plotSummaryStats *Plot summary statistics for a GenomicInteractions object*

Description

Makes summary plots of the counts, interaction distances, interaction annotations, and percentage of cis and trans interactions for a GenomicInteractions object using ‘plotCounts‘, ‘plotDists‘, ‘plotCisTrans‘, and ‘plotInteractionAnnotations‘.

Usage

```
plotSummaryStats(GIObject, other = 5, cut = 10)
```

Arguments

GIObject	A GenomicInteractions object
other	Default 5. Passed to plotInteractionAnnotations. Interaction types making up fewer than "other" percent of the total interactions will be consolidated into a single "other" category.
cut	Default 10. Passed to plotCounts.All interactions with counts > cut are consolidated into a single category.

Value

invisible(1)

Examples

```
data(hic_example_data)
plotSummaryStats(hic_example_data)
```

plotViewpoint *Plot coverage around a virtual 4C viewpoint*

Description

Plots coverage of interactions around a given viewpoint. This function requires the output of ‘viewPoint()‘ as input. You should additionally specify the total region you wish to plot.

Usage

```
plotViewpoint(x, region, ylab = "Signal", xlab = NULL, ...)
```

Arguments

`x`	a GenomicInteractions object which is output from viewPoint
`region`	The genomic region to plot
`ylab`	Y axis label.
`xlab`	X axis label. By default this is the chromosome of the region that is being plotted.
`...`	additional arguments to plot

Value

Coverage that is plotted (invisibly)

`print,GenomicInteractions-method`
Print function for GenomicInteractions

Description

Print function for GenomicInteractions

Usage

```
## S4 method for signature 'GenomicInteractions'
print(x)
```

Arguments

`x`	GenomicInteractionsObject

Value

invisible(1)

removeDups *Remove all but one occurences of a duplicated interaction*

Description

Removes all but the first occurence of a duplicated interaction (defined as having identical coordinates for both anchors). N.B. this does not summarise the total counts of all the duplicates. It is designed for removing potential PCR duplicates after reading in .bam files.

Usage

```
removeDups(GIObject)
```

Arguments

`GIObject`	A GenomicInteractions object.

Value

A GenomicInteractions object that is a subset of the input object.

resetAnnotations *Reset annotations made to a GenomicInteractions object*

Description

This function removes all annotations from a GenomicInteractions object by deleting all of the metadata columns associated with both anchors.

Usage

```
resetAnnotations(GIObject)

## S4 method for signature 'GenomicInteractions'
resetAnnotations(GIObject)
```

Arguments

`GIObject`	An annotated GenomicInteractions object

Value

invisible(1)

sameStrand — *Tests whether anchors have the same strand.*

Description

This is designed for processing .bam files.

Usage

```
sameStrand(GIObject)
```

Arguments

`GIObject`	A GenomicInteractions object

Value

A logical vector denoting with TRUE if both anchors of an interaction are on the same strand and FALSE otherwise.

seqinfo,GenomicInteractions-method — *Acessing/modifying sequence information for a GenomicInteractions object*

Description

Allows access/modification of seqinfo for GenomicInteractions objcets. When used with "force=True", interactions with either (or both) anchors on invalid chromosomes will be removed.

Usage

```
## S4 method for signature 'GenomicInteractions'
seqinfo(x)

## S4 replacement method for signature 'GenomicInteractions'
seqinfo(x, new2old = NULL, force = FALSE) <-
  value
```

Arguments

`x`	A GenomicInteractions object
`new2old`	Mapping between new and old seqnames. See ?seqinfo in GenomeInfoDb for details.
`force`	A logical indicating whether or not to drop invalid levels.
`value`	A replacement seqinfo object

Details

For more information see ?seqinfo in the GenomeInfoDb package.

Value

A seqinfo object,

---

setters *Functions to set data held in a GenomicInteractions object.*

Description

Use these functions to set data stored in each of the slots of a GenomicInteractions object.

Usage

```
name(GIObject) <- value

description(GIObject) <- value

interactionCounts(GIObject) <- value

## S4 replacement method for signature 'GenomicInteractions'
name(GIObject) <- value

## S4 replacement method for signature 'GenomicInteractions'
description(GIObject) <- value

## S4 replacement method for signature 'GenomicInteractions'
interactionCounts(GIObject) <- value
```

Arguments

`GIObject`	A GenomicInteractions object
`value`	A vector to replace a slot in the object

Value

GenomicInteractions object

Examples

```
library(GenomicRanges)

anchor.one = GRanges(c("chr1", "chr1", "chr1", "chr1"), IRanges(c(10, 20, 30, 20), width=5))
anchor.two = GRanges(c("chr1", "chr1", "chr1", "chr2"), IRanges(c(100, 200, 300, 50), width=5))
interaction_counts = sample(1:10, 4)
```

```
test <- GenomicInteractions(anchor.one, anchor.two, experiment_name="test",
                            description="this is a test", counts=interaction_counts)

name(test) <- "Mouse test"
name(test)

description(test) <- "This is a test using the mouse genome"
description(test)

interactionCounts(test) <- c(2,3,8,5)
interactionCounts(test)
```

`show,GenomicInteractions-method` *Representation function for GenomicInteractions*

Description

Representation function for GenomicInteractions

Usage

```
## S4 method for signature 'GenomicInteractions'
show(object)
```

Arguments

`object`	A GenomicInteractionsObject

Value

invisible(1)

`sort,GenomicInteractions-method` *Sort GenomicInteractions Object*

Description

This method will sort a GenomicInteractions object by first arranging all interactions start on the lower-ordered anchor; for trans-chromosomal interactions this is the anchor on the lower ordered chromomsome (defined by the seqlevels factor); and then by the position of the first anchor. See ?"GenomicRanges-comparison" for ordering rules.

Usage

```
## S4 method for signature 'GenomicInteractions'
sort(x, decreasing = FALSE,
  order.interactions = TRUE)
```

Arguments

`x`	GenomicInteractions Object
`decreasing`	A logical indicating sort order
`order.interactions`	A logical indicating if interactions should be reordered, or only rearranged so that start(anchorOne) < start(anchorTwo)

Value

A sorted GenomicInteractions object

`subsetByFeatures` *Subset a GenomicInteractions object by features*

Description

Subsets interactions for which at least one of the anchors overlaps with a given GRanges object. Alternatively, subsets interactions based on annotated feature IDs for a particular feature.

Usage

```
subsetByFeatures(GIObject, features, feature.class = NULL)

## S4 method for signature 'GenomicInteractions,GRanges,missing'
subsetByFeatures(GIObject,
  features, feature.class = NULL)

## S4 method for signature 'GenomicInteractions,GRangesList,missing'
subsetByFeatures(GIObject,
  features, feature.class = NULL)

## S4 method for signature 'GenomicInteractions,character,character'
subsetByFeatures(GIObject,
  features, feature.class = NULL)
```

Arguments

`GIObject`	A GenomicInteractions object
`features`	A GRanges or GRangesList object, or a character vector containing IDs of annotated features, e.g. promoter IDs.
`feature.class`	If ‘features’ is a character vector, the corresponding feature name, e.g. "promoter".

Value

a subsetted GenomicInteractions object

`sum,GenomicInteractions-method`
Return the total number of interactions in a GenomicInteractions GIObject

---

Description

Return the total number of interactions in a GenomicInteractions GIObject

Usage

```
## S4 method for signature 'GenomicInteractions'
sum(x)
```

Arguments

`x`	GenomicInteractions GIObject

Value

The sum of the counts in GIObject

`summariseByFeaturePairs`
Summarise the number of interactions between two sets of features.

Description

This function will calculate the number of observed interactions between two sets of features provided by the end-user. This allows the summarisation of the number of features of a specific type a particular region is involved in and how many interactions exist between them.

Usage

```
summariseByFeaturePairs(GIObject, features.one, feature.name.one, features.two,
  feature.name.two)

## S4 method for signature 'GenomicInteractions'
summariseByFeaturePairs(GIObject, features.one,
  feature.name.one, features.two, feature.name.two)
```

Arguments

`GIObject`	An annotated GenomicInteractions object
`features.one`	A GRanges object containing the feature set of interest
`feature.name.one`	The name of the first feature set of interest
`features.two`	A GRanges object containing the second feature set of interest
`feature.name.two`	The name of the second feature set of interest

Value

A data frame with one line for each range in 'features'

`summariseByFeatures` *Summary statistics of interactions for a given feature set*

Description

This function will calculate summary statistics for each element in the given feature set, including the number of interactions (the sum of all interaction counts), number of unique interactions and number of trans- (interchromosomal) interations. It also returns some statistics for the distances of interactions for all interactions of the feature, and for the different interaction types e.g. promoter-distal.

Usage

```
summariseByFeatures(GIObject, features, feature.name,
  distance.method = "midpoint", annotate.self = FALSE)

## S4 method for signature 'GenomicInteractions'
summariseByFeatures(GIObject, features,
  feature.name, distance.method = "midpoint", annotate.self = FALSE)
```

Arguments

`GIObject`	An annotated GenomicInteractions object
`features`	A GRanges object containing the feature set
`feature.name`	The name of the feature set
`distance.method`	Method for calculating distances between anchors, see ?calculateDistances
`annotate.self`	Logical. Indicates whether to annotate self interactions, i.e. where a feature in 'features' overlaps both anchors of an interaction. Default: FALSE.

Value

A data frame with one line for each range in 'features'

trim,GenomicInteractions-method

Trim a GenomicInteractions object

Description

This will remove any interactions with an anchor falling outside of the seqlengths in a GenomicInteractions object, and trim ranges which cross the ends of chromosomes.

Usage

```
## S4 method for signature 'GenomicInteractions'
trim(x, minAnchorSize = 1, ...)
```

Arguments

`x`	A GenomicInteractions object
`minAnchorSize`	The minimum size anchor to allow when trimming ranges.
`...`	any additional arguments to trim

Value

A trimmed GenomicInteractions object

unique,GenomicInteractions-method

unique, GenomicInteractions-method

Description

Finds unique interactions in a GenomicInteractions object.

Usage

```
## S4 method for signature 'GenomicInteractions'
unique(x, dropMetadata = FALSE)
```

Arguments

`x`	GenomicInteractionsObject
`dropMetadata`	Logical, default FALSE. Whether to drop interaction mcols when considering unique interactions.

Details

Uniqueness is based on anchor positions and metadata, interaction counts, and interaction metadata (unless dropMetadata is TRUE)

Value

A GenomicInteractions object

Examples

```
library(GenomicInteractions)

data(hic_example_data)
unique(hic_example_data[c(1:4, 1:5)])
```

---

viewPoint *Virtual 4C viewpoint*

Description

This function creates a GenomicInteractions object representing interactions originating at a given viewpoint ("bait"), or set of viewpoints. This is similar to the idea of a virtual 4C experiment where you are interested in interactions with a specific region.

Usage

```
viewPoint(x, bait, region = NULL, ...)
```

Arguments

`x`	A GenomicInteractions object.
`bait`	A GRanges object describing bait regions.
`region`	If present, a GenomicInteractions object specifying the region to look for bait interactions in.
`...`	additional arguments to findoverlaps

Details

The object returned has the "bait" as anchor one, and the interacting regions as anchor two. By default this is genome wide. If you only want to consider interactions within a certain distance around the bait, you can specify a region to consider.

Multiple baits can be given, e.g. to find all interactions around promoters.

You may want to visualise the resulting interactions in a genome browser - you can do this by creating coverage over anchor two of the object and exporting as a wig or bedgraph file.

Value

A GenomicInteractions object.

Examples

```
## Not run:
data(hic_data)
library(GenomicRanges)
pos <- GRanges(seqnames="chr5", ranges=IRanges(start=115938063, end=115941352))
region <- GRanges(seqnames="chr5", ranges=IRanges(start=115838063, end=116041352))
viewPoint(hic_data, pos, region)

## End(Not run)
```

[*Standard subsetting methods for GenomicInteractions objects*

Description

Standard subsetting methods for GenomicInteractions objects

Usage

```
## S4 method for signature 'GenomicInteractions,ANY,ANY'
x[i, j, drop]
```

Arguments

x	A genomicInteractions object
i	A numeric, logical or Rle vector
j	A numeric or logical vector
drop	Logical. If TRUE, result is coerced to lowest possible dimension.

Value

A GenomicInteractions object containing only the features specified by ‘i‘.

$,GenomicInteractions-method

Quick access to GenomicInteractions metadata columns

Description

Quick access to GenomicInteractions metadata columns

Usage

```
## S4 method for signature 'GenomicInteractions'
x$name

## S4 replacement method for signature 'GenomicInteractions'
x$name <- value
```

Arguments

x	A GenomicInteractions object
name	Column to select
value	Replacement value

Value

A vector containing the contents of the column

Index