Package ‘GenomicInteractions’

March 28, 2017

Type Package

Title R package for handling genomic interaction data

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

Version 1.8.1

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, rtracklayer, GenomicRanges, IRanges, BiocGenerics (>= 0.15.3), data.table, stringr, GenomeInfoDb, ggplot2, grid, gridExtra, methods, igraph, S4Vectors, dplyr, Gviz, Biobase, graphics, stats, utils

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 (>= 3.3), InteractionSet

RoxygenNote 5.0.1

NeedsCompilation no

R topics documented:

GenomicInteractions-package .................................................. 2
annotateAnchors ................................................................. 3
annotateInteractions .............................................................. 3
annotateRegions ................................................................. 4
asBED, GInteractions-method .................................................. 5
availableDisplayPars ............................................................ 6
calculateDistances ............................................................... 6
categoriseInteractions ......................................................... 7
GenomicInteractions-package

GenomicInteractions: A package for analysing chromosome conformation

Description

GenomicInteractions: A package for analysing chromosome conformation

Examples

library(GenomicInteractions)
annotateAnchors

Annotate anchors - DEPRECATED

Description

This function is deprecated and will be removed in the next release of Bioconductor. Use 'annotateRegions' instead.

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 GInteractions 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 'GInteractions,list'
annotateInteractions(GIObject, annotations)
**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIObjec</td>
<td>A GInteractions object to be annotated</td>
</tr>
<tr>
<td>annotations</td>
<td>A list containing GRanges (or GRangesList) objects with which to annotate the GInteractions object.</td>
</tr>
</tbody>
</table>

**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**

```r
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))
```

**Description**

Use this function to add metadata parallel to the ‘regions’ slot of a GenomicInteractions or GInteractions object.

**Usage**

```r
annotateRegions(GIObject, name, dat)
```

## S4 method for signature 'GInteractions,character,vector'

```r
annotateRegions(GIObject, name, dat)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIObjec</td>
<td>A GenomicInteractions or GInteractions object</td>
</tr>
<tr>
<td>name</td>
<td>Character. Will be used as a column name.</td>
</tr>
<tr>
<td>dat</td>
<td>Vector of the same length as the GInteractions object, containing data with which to annotate the object.</td>
</tr>
</tbody>
</table>

**Value**
invisible(1)
Examples

data(hic_example_data)
chip <- runif(n = length(regions(hic_example_data)), max = 1000)
annotateRegions(hic_example_data, "chip", chip)

asBED,GInteractions-method

Coerce to BED structure

Description

Coerce the structure of an object to one following BED-like conventions, i.e., with columns for blocks and thick regions.

Usage

## S4 method for signature 'GInteractions'
asBED(x, keep.mcols = FALSE, score = "score")

Arguments

x
  Generally, a tabular object to structure as BED

keep.mcols
  logical whether to keep non-BED12 columns in final output (may cause problems with some parsers).

score
  character, which field to export as "score" in BED12. Defaults to "auto" which will choose score, then counts, if present, or fill column with zeros.

...
  Arguments to pass to methods

  The exact behavior depends on the class of 'object'.

  'GRangesList' This treats 'object' as if it were a list of transcripts, i.e., each element contains the exons of a transcript. The 'blockStarts' and 'blockSizes' columns are derived from the ranges in each element. Also, add 'name' column from 'names(object)'.

Value

A 'GRanges', with the metadata columns 'name', 'blockStarts' and 'blockSizes' added.

Examples

data(hic_example_data)
asBED(hic_example_data)
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 a InteractionTrack then the function calls the availableDisplayPars method in Gviz.

Value

returns a list of the default display parameters.

Examples

availableDisplayPars("InteractionTrack")

calculateDistances

Calculate interaction distances

Description

This function takes a GInteractions 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 'GInteractions'
calculateDistances(GIObject, method = "midpoint", floor = TRUE)
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 GInteractions 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**

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

---

**countsBetweenAnchors**

*Summarise interactions between defined anchors*

**Description**

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

**Usage**

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

---

**Arguments**

- `x`: A GInteractions 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 GInteractions object with annotated counts between anchors

---

**export.bed12**

*Export interactions in BED12 format.*

**Description**

Export interactions in BED12 format.

**Usage**

```r
export.bed12(GIObject, fn = NULL, score = "counts")
```

---

**Arguments**

- `GIObject`: A GInteractions object
- `fn`: Output file name
- `score`: A character string specifying which score to use (default = "counts")
export.bedpe

Arguments

GIObject A GInteractions object.
fn A filename to write the object to
score Which metadata column to export as score

Exports a GInteractions 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

Examples

data(hic_example_data)
export.bed12(hic_example_data, fn = tempfile(), score = "counts")

export.bedpe

Export interactions in BED Paired-End format.

Description

Exports a GInteractions 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 'GInteractions'
export.bedpe(GIObject, fn = NULL, score = "counts")

Arguments

GIObject A GInteractions 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

Examples

data(hic_example_data)
export.bedpe(hic_example_data, fn = tempfile(), score = "counts")
**export.chiasig**

Export interactions in a BEDPE-like format for use with ChiaSig

**Description**

Exports a GInteractions 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**

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

**Arguments**

- **GIObject**: A GInteractions 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

**Examples**

```r
data(hic_example_data)
eexport.chiasig(hic_example_data, fn = tempfile(), score = "counts")
```

---

**export.igraph**

Export interactions to an igraph object.

**Description**

Exports a GInteractions 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**

```r
eexport.igraph(GIObject)
```

```r
## S4 method for signature 'GInteractions'
eexport.igraph(GIObject)
```
**GenomicInteractions**

**Arguments**

- *GIObject*: A GIInteractions object.

**Value**

- A graph.data.frame representation of the GIInteractions object.

**Examples**

```r
data(hic_example_data)
ig <- export.igraph(hic_example_data)
```

---

**GenomicInteractions** *Function to create a GenomicInteractions object*

**Description**

Create GenomicInteractions objects from two GRanges objects.

**Usage**

```r
GenomicInteractions(anchor1, anchor2, counts, ...)
```

- ## S4 method for signature 'GRanges,GRanges,numeric'
  ```r
  GenomicInteractions(.anchor1, anchor2, counts, ...)
  ```

- ## S4 method for signature 'GIInteractions,ANY,ANY'
  ```r
  GenomicInteractions(anchor1)  
  ```

- ## S4 method for signature 'GIInteractions,numeric,ANY'
  ```r
  GenomicInteractions( anchor1, anchor2)  
  ```

- ## S4 method for signature 'numeric,numeric,GRanges'
  ```r
  GenomicInteractions(anchor1, anchor2, counts, ...)
  ```

- ## S4 method for signature 'GRanges,GRanges,GenomicRangesORmissing'
  ```r
  GenomicInteractions( anchor1, anchor2, counts, ...)
  ```

- ## S4 method for signature 'missing,missing,GenomicRangesORmissing'
  ```r
  GenomicInteractions( anchor1, anchor2, counts, ...)
  ```

- ## S4 method for signature 'ANY,ANY,ANY'
  ```r
  GenomicInteractions( anchor1, anchor2, counts, ...)
  ```
Arguments

anchor1, anchor2
  GRanges objects.

counts
  An integer vector, defaults to 1.

... 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, counts=interaction_counts)

GenomicInteractions-class

A S4 class to represent interactions between genomic regions.

Description

@slot metadata List, defaults to "experiment_name" and "description", inherited from S4Vectors::Vector
@slot anchor_one, anchor_two GRanges. Set of anchors of interactions. @slot counts integer vector, contains raw counts @slot 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, 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)

anchorOne(GIObject)

anchorTwo(GIObject)

interactionCounts(GIObject)

annotationFeatures(GIObject)

Arguments

GIObject  A GenInteractions object

object  Object, possibly derived from class eSet-class.

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

```r
library(GenomicRanges)

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

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

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

Arguments

- `GIObject`: a `GInteractions` 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 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 as a cutoff to determine which bins are likely to contain mostly self-ligated reads.

Usage

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

Arguments

GIObject a GInteractions 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 log2 ratio 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.
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

```r
data(hic_example_data)
```

Format

A GenomicInteractions object with length 8171.

Value

GenomicInteractions object

References


InteractionTrack

Constructor to create an InteractionTrack object

Description

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

Usage

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

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

```r
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

@slot `plottingFunction` function @slot `variables` list @slot `chromosome` chromosome @slot `stacking` character

Details

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 GIInteractions 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**

```r
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 'GIInteractions'
is.pp(GIObject)
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

```r
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

```r
k562.rep1 <- makeGenomicInteractionsFromFile(
  system.file(package="GenomicInteractions", "extdata", "k562.rep1.cluster.pet3+.txt"),
  type="chiapet.tool", experiment_name="k562", description="k562 pol2 8wg16")
```

```r
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

```r
data(mm9_refseq_promoters)
```
**plotAvgViewpoint**

**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**

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

**Arguments**

- **x**: A GInteractions 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)

**Examples**

```r
data(hic_example_data)
library(GenomicRanges)
pos <- GRanges(seqnames="chr15", ranges=IRanges(start=59477709, end=59482708))
region <- GRanges(seqnames="chr15", ranges=IRanges(start=58980209, end=59980208))
vp <- viewPoint(hic_example_data, pos, region)
plotAvgViewpoint(vp, left_dist = 1000000, right_dist = 100000)
```
plotCisTrans

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

Description

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

Usage

plotCisTrans(GIObject)

Arguments

GIObject

A GInteractions 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 GInteractions 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

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

Description

Plots a histogram of interaction distances for a GInteractions Object

Usage

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

Arguments

- **GIObject**: A GInteractions 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

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

plotInteractionAnnotations

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

Description

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

Usage

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

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIObjec</td>
<td>A GInteractions object</td>
</tr>
<tr>
<td>node.classes</td>
<td>Optional. All node.classes to include in the analysis. Default: all node classes.</td>
</tr>
<tr>
<td>viewpoints</td>
<td>Optional. If set will only consider interactions where at least one anchor is of this node class. Default: all classes in node.classes.</td>
</tr>
<tr>
<td>other</td>
<td>Optional. Interaction types making up fewer than &quot;other&quot; percent of the total interactions will be consolidated into a single &quot;other&quot; category.</td>
</tr>
<tr>
<td>keep.order</td>
<td>Optional. Logical. Keep original order of node.classes for plotting or not. Default: FALSE, alphabetical order.</td>
</tr>
<tr>
<td>legend</td>
<td>Optional. Logical. If TRUE, legend is plotted to right of donut plot. If FALSE, donut plot is annotated with category names.</td>
</tr>
</tbody>
</table>

**Value**

A ggplot2 plot

**Examples**

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

**Description**

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

**Usage**

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

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

---

**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**

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

**Arguments**

- `x` a GIInteractions 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)

**Examples**

```r
data(hic_example_data)
library(GenomicRanges)
pos <- GRanges(seqnames="chr15", ranges=IRanges(start=59477709, end=59482708))
region <- GRanges(seqnames="chr15", ranges=IRanges(start=58980209, end=59980208))
vp <- viewPoint(hic_example_data, pos, region)
plotViewpoint(vp, region)
```
**removeDups**

*Remove all but one occurrence of a duplicated interaction*

**Description**

Removes all but the first occurrence 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**

```r
removeDups(GIObject)
```

**Arguments**

- `GIObject` A GInteractions object.

**Value**

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

---

**resetAnnotations**

*Reset annotations made to a GInteractions object*

**Description**

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

**Usage**

```r
resetAnnotations(GIObject)
```

**Arguments**

- `GIObject` An annotated GInteractions object.

**Value**

`invisible(1)`

**Examples**

```r
data(hic_example_data)
resetAnnotations(hic_example_data)
```
sameStrand

Tests whether anchors have the same strand.

Description
This is designed for processing .bam files.

Usage
sameStrand(GIObject)

Arguments
GIObject A GInteractions object

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

setters
Functions to set data held in a GInteractions object.

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

Usage
name(GIObject) <- value
interactionCounts(GIObject) <- value

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

## S4 replacement method for signature 'GInteractions,ANY'
description(object) <- value

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

Arguments
GIObject A GenomicInteractions object
value A vector to replace a slot in the object
object Object, possibly derived from class eSet-class.
subsetByFeatures

Value

GenomicInteractions object

Examples

```r
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"

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

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

interactionCounts(test)
```

subsetByFeatures  Subset a GInteractions 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 'GInteractions,GRanges,missing'
subsetByFeatures(GIObject, features, feature.class = NULL)
## S4 method for signature 'GInteractions,GRangesList,missing'
subsetByFeatures(GIObject, features, feature.class = NULL)
## S4 method for signature 'GInteractions,character,character'
subsetByFeatures(GIObject, features, feature.class = NULL)
```

Arguments

- **GIObject**: A GInteractions 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 GInteractions object

Examples

data("hic_example_data")
data("mm9_refseq_promoters")
annotateInteractions(hic_example_data, list(promoter = mm9_refseq_promoters))
ids <- names(mm9_refseq_promoters[1:10])
subsetByFeatures(hic_example_data, ids, "promoter")

sum,GInteractions-method

Return the total number of interactions in a GInteractions GIObject

Description

Return the total number of interactions in a GInteractions GIObject

Usage

## S4 method for signature 'GInteractions'
sum(x)

Arguments

x GInteractions 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 'GIInteractions'
summariseByFeaturePairs(GIObject, features.one, feature.name.one, features.two, feature.name.two)

Arguments

- **GIObject**: An annotated GIInteractions 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'

Examples

data("hic_example_data")
data("mm9.refseq_promoters")
data("thymus_enhancers")
annotateInteractions(hic_example_data, list(promoter = mm9.refseq_promoters, enhancer = thymus_enh))
# can be slow so subset of features used for examples
p <- unique(unlist(head(regions(hic_example_data)$promoter.id)))[!is.na(p)]
e <- unique(unlist(head(regions(hic_example_data)$enhancer.id)))[!is.na(e)]
ep_summary <- summariseByFeaturePairs(hic_example_data, p, "promoter", e, "enhancer")

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) interactions. 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

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

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

Arguments

- `GIObject` An annotated GInteractions 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`

Examples

```r
data("hic_example_data")
data("mm9_refseq_promoters")
annotateInteractions(hic_example_data, list(promoter = mm9_refseq_promoters))
summariseByFeatures(hic_example_data, mm9_refseq_promoters[1:10], "promoter")
```

---

thymus_enh  Putative enhancers from mouse thymus data

Description

This dataset contains a set of mouse thymus enhancers derived from ChIP-seq data from mouse thymus, as described in Shen et al. 2012. See the HiC analysis vignette for more details. (vignettes(GenomicInteractions))

Usage

```r
data("thymus_enhancers")
```

Format

A GRanges object

Value

A GRanges object

References

**updateObject, GenomicInteractions-method**

*updateObject method for GenomicInteractions 1.3.7 and earlier*

---

**Description**

updateObject method for GenomicInteractions 1.3.7 and earlier

**Usage**

```r
## S4 method for signature 'GenomicInteractions'
updateObject(object, ..., verbose = FALSE)
```

**Arguments**

- `object` Object to be updated for `updateObject` and `updateObjectFromSlots`. Object for slot information to be extracted from for `getObjectSlots`.
- `...` Additional arguments, for use in specific `updateObject` methods.
- `verbose` TRUE or FALSE, indicating whether information about the update should be reported. Use `message` to report this information.

**Value**

A GenomicInteractions object

---

**viewPoint**

*Virtual 4C viewpoint*

---

**Description**

This function creates a GInteractions 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**

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

**Arguments**

- `x` A GInteractions object.
- `bait` A GRanges object describing bait regions.
- `region` If present, a GRanges 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 GInteractions object.

Examples

data(hic_example_data)
library(GenomicRanges)
pos <- GRanges(seqnames="chr15", ranges=IRanges(start=59477709, end=59482708))
region <- GRanges(seqnames="chr15", ranges=IRanges(start=58980209, end=59980208))
vp <- viewPoint(hic_example_data, pos, region)
Index

*Topic datasets
  hg19.refseq.transcripts, 15
  hic_example_data, 16
  mm9.refseq.promoters, 20
  thymus_enh, 31
anchorOne (getters), 13
anchorOne,GInteractions-method (getters), 13
anchorTwo (getters), 13
anchorTwo,GInteractions-method (getters), 13
annotateAnchors, 3
annotateAnchors,GInteractions,numerics,character (annotateAnchors), 3
annotateInteractions, 3
annotateInteractions,GInteractions,list-method (annotateInteractions), 3
annotateRegions, 4
annotateRegions,GInteractions,character,vector (annotateRegions), 4
annotationFeatures (getters), 13
annotationFeatures,GInteractions-method (annotationFeatures), 13
asBED,GInteractions-method, 5
availableDisplayPars, 6
calculateDistances, 6
calculateDistances,GInteractions-method (calculateDistances), 6
categoriseInteractions, 7
countsBetweenAnchors, 8
countsBetweenAnchors,GInteractions,GRanges-method (countsBetweenAnchors), 8
description,GInteractions-method (getters), 13
description<-,GInteractions,ANY-method (setters), 27
export.bed12, 8
export.bed12,GInteractions-method (export.bed12), 8
export.bedpe, 9
export.bedpe,GInteractions-method (export.bedpe), 9
export.chiasig, 10
export.chiasig,GInteractions-method (export.chiasig), 10
export.igraph, 10
export.igraph,GInteractions-method (export.igraph), 10
GenomicInteractions, 11
GenomicInteractions,ANY,ANY,ANY-method (GenomicInteractions), 11
GenomicInteractions,GInteractions,ANY,ANY-method (GenomicInteractions), 11
GenomicInteractions,GInteractions,numerics,ANY-method (GenomicInteractions), 11
GenomicInteractions,GInteractions,numeric,ANY-method (GenomicInteractions), 11
GenomicInteractions,GInteractions,numerics,numeric-method (GenomicInteractions), 11
GenomicInteractions,GInteractions,numeric,numerics-method (GenomicInteractions), 11
GenomicInteractions-class, 12
GenomicInteractions-package, 2
get_binom_ligation_threshold, 14
get_self_ligation_threshold, 15
getters, 13
hg19.refseq.transcripts, 15
hic_example_data, 16
interactionCounts (getters), 13
interactionCounts,GInteractions-method (getters), 13
interactionCounts<-,GInteractions-method (setters), 27
interactionCounts<-,GInteractions-method (setters), 27
InteractionHelpers (is.pp), 18
InteractionTrack, 16
InteractionTrack-class, 17
is.cis (is.pp), 18
is.cis,GInteractions-method (is.pp), 18
INDEX

is.dd(is.pp), 18
is.dd,GInteractions-method(is.pp), 18
is.dt(is.pp), 18
is.dt,GInteractions-method(is.pp), 18
is.pd(is.pp), 18
is.pd,GInteractions-method(is.pp), 18
is.pp, 18
is.pp,GInteractions-method(is.pp), 18
is.pt(is.pp), 18
is.pt,GInteractions-method(is.pp), 18
is.trans(is.pp), 18
is.trans,GInteractions-method(is.pp), 18
is.tt(is.pp), 18
is.tt,GInteractions-method(is.pp), 18
isInteractionType(is.pp), 18
isInteractionType,GInteractions-method(is.pp), 18
makeGenomicInteractionsFromFile, 19
message, 32
mm9_refseq_promoters, 20

name(getters), 13
name,GInteractions-method(getters), 13
name<-(setters), 27
name<-,GInteractions-method(setters), 27

plotAvgViewpoint, 21
plotCisTrans, 22
plotCounts, 22
plotDists, 23
plotInteractionAnnotations, 23
plotSummaryStats, 24
plotViewpoint, 25
removeDups, 26
resetAnnotations, 26
resetAnnotations,GInteractions-method(resetAnnotations), 26

sameStrand, 27
setters, 27
subsetByFeatures, 28
subsetByFeatures,GInteractions,character,character-method(subsetByFeatures), 28
subsetByFeatures,GInteractions,GRanges,missing-method(subsetByFeatures), 28
subsetByFeatures,GInteractions,GRangesList,missing-method(subsetByFeatures), 28
sum,GInteractions-method, 29
summariseByFeaturePairs, 29

summariseByFeaturePairs,GInteractions-method(summariseByFeaturePairs), 29
summariseByFeatures, 30
summariseByFeatures,GInteractions-method(summariseByFeatures), 30
thymus_enh, 31
updateObject,GInteractions-method, 32

viewPoint, 32