Package: GenomicInteractions (via r-universe)

November 1, 2026

Type Package

Title Utilities for handling genomic interaction data

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

Version 1.46.0

Date 2025-07-22

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

Maintainer Liz Ing-Simmons <liz.ingsimmons@gmail.com>

Imports Rsamtools, rtracklayer, GenomicRanges (>= 1.29.6), IRanges, BiocGenerics (>= 0.15.3), data.table, stringr, Seqinfo, ggplot2, grid, gridExtra, methods, igraph, S4Vectors (>= 0.13.13), dplyr, Gviz, Biobase, graphics, stats, utils, grDevices

Suggests knitr, rmarkdown, BiocStyle, testthat

VignetteBuilder knitr

Description Utilities for handling genomic interaction data such as ChIA-PET or Hi-C, annotating genomic features with interaction information, and producing plots and summary statistics.

biocViews Software,Infrastructure,DataImport,DataRepresentation,HiC

License GPL-3

Depends R (>= 3.5), InteractionSet

RoxygenNote 7.1.2

Config/pak/sysreqs cmake libglpk-dev make libbz2-dev libicu-dev libjpeg-dev liblzma-dev libpng-dev libuv1-dev libxml2-dev libssl-dev xz-utils zlib1g-dev

Repository Bioconductor 3.23

Date/Publication 2026-04-28 12:39:48 UTC

RemoteUrl https://github.com/bioc/GenomicInteractions

RemoteRef RELEASE_3_23

RemoteSha 35b0cfd48274486e5b684e79e61d30efe62746e5

Contents

GenomicInteractions-package . . . 3
.importHicLib . . . 3
.importHomer . . . 4
.processChiapetName . . . 4
.readBam . . . 5
.readTwoBams . . . 5
.validateInput . . . 6
annotateInteractions . . . 6
annotateRegions . . . 7
asBED,GInteractions-method . . . 8
availableDisplayPars . . . 9
calculateDistances . . . 9
categoriseInteractions . . . 10
countsBetweenAnchors . . . 11
export.bed12 . . . 12
export.bedpe . . . 12
export.chiasig . . . 13
export.igraph . . . 14
GenomicInteractions . . . 15
GenomicInteractions-class . . . 16
get_binom_ligation_threshold . . . 16
get_self_ligation_threshold . . . 17
getters . . . 18
hg19.refseq.transcripts . . . 19
hic_example_data . . . 20
InteractionTrack . . . 20
InteractionTrack-class . . . 21
is.pp . . . 23
makeGenomicInteractionsFromFile . . . 25
mm9_refseq_promoters . . . 26
plotAvgViewpoint . . . 26
plotCisTrans . . . 27
plotCounts . . . 28
plotDists . . . 29
plotInteractionAnnotations . . . 29
plotSummaryStats . . . 30
plotViewpoint . . . 31
removeDups . . . 32
resetAnnotations . . . 32
sameStrand . . . 33
setters . . . 33
subsetByFeatures . . . 35
sum,GInteractions-method . . . 36
summariseByFeaturePairs . . . 36
summariseByFeatures . . . 37
thymus_enh . . . 38

updateObject,GenomicInteractions-method 39
viewPoint 40

Index **41**

`GenomicInteractions-package` *GenomicInteractions*

Description

Utilities for handling genomic interaction data such as ChIA-PET or Hi-C, annotating genomic features with interaction information, and producing plots and summary statistics.

`.importHicLib` *Function to read in processed Hi-C interaction data generated by hiclib*

Description

This function reads in the interaction data processed by [hiclib](https://bitbucket.org/mirnylab/hiclib). The data is extracted from hiclib and exported from its internal HDF5 format to text using h5dictToTxt.py. This function reads in the relevant files present in the directory and generates a data.table containing all of the information. It assumes that there are files with the following names in the directory: fragids1, chrms1, mids1, fraglens1, fragids2, chrms2, mids2, fraglens2, distances.

Usage

```
.importHicLib(dir, chr_names)
```

Arguments

`dir`	directory containing the output files generated by hiclib, extracted using h5dictToTxt.py
`chr_names`	ordered chromosome names (for renaming)

Value

data.table containing information on the individual interactions

---

`.importHomer` *Function to read in processed Hi-C interaction data generated by HOMER*

Description

This function reads in the interaction data outputted by HOMER (http://homer.salk.edu/homer/interactions/).

Usage

```
.importHomer(fn)
```

Arguments

`fn`	location of data exported by HOMER

Value

a data frame containing the relevant information

`.processChiapetName` *Function to process names relating to interactions stored in bed12 formats.*

Description

In bed(n) formats interaction data cannot be stored directly due to the inability of the format to handle trans-interactions. It is only capable of storing cis-interactions through the use of blockStarts and blockEnds. As such the information is typically stored as a string of the format chr1:start1..end1-chr2:start2..end2,N (e.g. \'chr1:3268539..3269061-chr10:103124111..103124631,4\'.). This function takes a vector of these strings and converts them to a data.frame.

Usage

```
.processChiapetName(x)
```

Arguments

`x`	a vector of names stored in the bed file

Value

a data.frame containing all of the processed information

.readBam *Function to read in interaction-data stored in a BAM file*

Description

Reads in interactions stored in a BAM file. Assumes that each interaction is represented by pair of PETs with the same qname. The function reads in and determines which anchor is which by examining the isFirstMateRead and isSecondMateRead fields in the BAM file.

Usage

```
.readBam(fn)
```

Arguments

fn	name of BAM file containing interaction information

Value

list of GRanges - storing the anchor information for each interaction

.readTwoBams *Function to read in interaction-data stored in a pair of BAM files*

Description

Reads in interactions stored in a a pair of BAM files, e.g. from independent alignment of paired-end reads. Assumes that each interaction is represented by pair of PETs with the same qname (this may not always be true. Depending on data origin read qnames may end in '/1' or '/2' to denote first or second read in the pair). The function reads in files, removes unpaired reads, and pairs reads based on macthing qnames.

Usage

```
.readTwoBams(fn)
```

Arguments

fn	Character vector of two BAM files with aligned reads.

Value

list of two GRanges, storing the anchor information for each interaction

.validateInput *Function to validate tabular input*

Description

Check for columns in a data.frame.

Usage

```
.validateInput(x, h)
```

Arguments

x	A data frame.
h	A character vector containing expected headings

Value

list of two GRanges, storing the anchor information for each interaction

annotateInteractions *Annotate the interactions in a GInteractions object*

Description

This function annotates the regions of a GInteractions object according to their overlaps with 'annotations', a list of named GRanges (or GRangesList) objects.

Usage

```
annotateInteractions(GIObject, annotations, id.col = NULL)

## S4 method for signature 'GInteractions,list'
annotateInteractions(GIObject, annotations, id.col = NULL)
```

Arguments

GIObject	A GInteractions object to be annotated
annotations	A list containing GRanges (or GRangesList) objects with which to annotate the GInteractions object.
id.col	Metadata column of GRanges objects to use as feature ID. Default: NULL.

Details

Metadata columns will be added to the regions of the GInteractions object, named according to the names of 'annotations' and containing the id(s) of the corresponding overlapping genomic interval(s). If 'annotations' is not named, the metadata columns will be named 'FEATURE#.id' where # is the position in the list.

IDs for features in each element of annotations will be extracted according to the following rules: if the annotation features are a GRanges object and a metadata column is specified using 'id.col', this column will be used as feature IDs. Otherwise, if the annotation features are named, their names will be used. If neither of these are present, they will be given numeric IDs.

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'.

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)
# This adds a `promoter.id` metadata column to regions(hic_example_data)
# containing IDs of overlapping promoters for each region, taken from
# names(mm9_refseq_grl)
annotateInteractions(hic_example_data, list(promoter=mm9_refseq_grl))
```

---

annotateRegions *Annotate regions*

Description

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

Usage

```
annotateRegions(GIObject, name, dat)

## S4 method for signature 'GInteractions,character,vector'
annotateRegions(GIObject, name, dat)
```

Arguments

GIObject	A GenomicInteractions or GInteractions object
name	Character. Will be used as a column name.
dat	Vector of the same length as the GInteractions object, containing data with which to annotate the object.

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, see asBED for details.

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 an `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)
```

Arguments

GIObject	A GInteractions 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')
```

---

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

```
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))
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 'GInteractions,GRanges'
countsBetweenAnchors(x, y, ignore_overlaps = FALSE, ...)
```

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

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

## S4 method for signature 'GInteractions'
export.bed12(GIObject, fn = NULL, score = "counts")
```

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

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

## S4 method for signature 'GInteractions'
export.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

```
data(hic_example_data)
export.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

```
export.igraph(GIObject)

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

Arguments

`GIObject`	A GInteractions object.

Value

a graph.data.frame representation of the GInteractions object

Examples

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

GenomicInteractions *Function to create a GenomicInteractions object*

Description

Create GenomicInteractions objects from two GRanges ojects.

Usage

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

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

## S4 method for signature 'GInteractions,ANY,ANY'
GenomicInteractions(anchor1)

## S4 method for signature 'GInteractions,numeric,ANY'
GenomicInteractions(anchor1, anchor2)

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

## S4 method for signature 'GRanges,GRanges,GenomicRanges_OR_missing'
GenomicInteractions(anchor1, anchor2, counts, ...)

## S4 method for signature 'missing,missing,GenomicRanges_OR_missing'
GenomicInteractions(anchor1, anchor2, counts, ...)

## S4 method for signature 'ANY,ANY,ANY'
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)
```

`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 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 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 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 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.

---

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)

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

## S4 method for signature 'GInteractions'
description(object)

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

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

## S4 method for signature 'GInteractions'
interactionCounts(GIObject)
```

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

Arguments

GIObject	A Gnteractions 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

```
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)

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

---

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.

A GRanges object

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

@slot giobject GenomicInteractions object from which the object was created

@slot variables list of chromosome, start, and end. Start and end can be NULL

@slot chromosome chromosome defined for the object

@slot stacking character

@slot dp DisplayPars for the object, access with ‘availableDisplayPars()‘

@slot name Object name

@slot imageMap NULL

Usage

```
## S4 method for signature 'InteractionTrack'
start(x)

## S4 method for signature 'InteractionTrack'
end(x)

## S4 method for signature 'InteractionTrack'
chromosome(GdObject)

## S4 method for signature 'InteractionTrack'
subset(x, from, to, chromosome, ...)
```

Arguments

x	An InteractionTrack object
GdObject	An InteractionTrack object
from	Integer start coordinate of subset region
to	Integer end coordinate of subset region
chromosome	Chromosome of subset region
...	additional arguments are ignored.

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.interaction.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.

Functions

- `start,InteractionTrack-method`: Extract stored start position for object or, if that is NULL, minimum of region starts.
- `end,InteractionTrack-method`: Extract stored end position for object or, if that is NULL, maximum of region ends.
- `chromosome,InteractionTrack-method`: Extract stored chromosome of object
- `subset,InteractionTrack-method`: Subset the object to only contain interactions within the specified genomic region

---

`is.pp` *Interaction Type Helpers*

Description

Functions to classify interactions within GInteractions 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 'GInteractions'
is.pp(GIObject)

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

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

## S4 method for signature 'GInteractions'
is.dd(GIObject)

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

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

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

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

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

Arguments

GIObject	A GInteractions object
x, y	Names of annotated node classes

Value

A logical vector

Examples

```
data(hic_example_data)
table(is.cis(hic_example_data))
sum(interactionCounts(hic_example_data))
```

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. Hiclib support will be removed in the next version of GenomicInteractions.

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(
     system.file(package='GenomicInteractions', 'extdata', 'k562.rep1.cluster.pet3+.txt'),
       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.

A GRanges object

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 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

```
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

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

`plotDists` *Plots a histogram of interaction distances for a GInteractions Object*

Description

Plots a histogram of interaction distances for a GInteractions Object

Usage

```
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

```
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

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

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.
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

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

---

plotSummaryStats *Plot summary statistics for a GInteractions object*

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)
```

Arguments

GIObject	A GInteractions 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 GInteractions 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

```
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 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 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

```
resetAnnotations(GIObject)

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

Arguments

GIObject	An annotated GInteractions object

Value

invisible(1)

Examples

```
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.

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)
```

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')
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 'GInteractions'
summariseByFeaturePairs(
  GIObject,
  features.one,
```

```
  feature.name.one,
  features.two,
  feature.name.two
)
```

Arguments

GIObject	An annotated GInteractions 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)$promoter.id)))
e <- unique(unlist(head(regions(hic_example_data)$enhancer.id)))
p <- p[!is.na(p)]
p <- mm9_refseq_promoters[p]
e <- e[!is.na(e)]
e <- thymus_enh[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) 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 '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

```
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

```
data('thymus_enhancers')
```

Format

A GRanges object

Value

A GRanges object

References

Shen, Y et al. A map of cis-regulatory sequences in the mouse genome. Nature (2012).

---

updateObject,GenomicInteractions-method

updateObject method for GenomicInteractions 1.3.7 and earlier

---

Description

updateObject method for GenomicInteractions 1.3.7 and earlier

Usage

```
## 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

```
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

∗ **datasets**
    hg19.refseq.transcripts, 19
    hic_example_data, 20
    mm9_refseq_promoters, 26
    thymus_enh, 38
.importHicLib, 3
.importHomer, 4
.processChiapetName, 4
.readBam, 5
.readTwoBams, 5
.validateInput, 6

anchorOne *(*getters*)*, 18
anchorOne,GInteractions-method
        *(*getters*)*, 18
anchorTwo *(*getters*)*, 18
anchorTwo,GInteractions-method
        *(*getters*)*, 18
annotateInteractions, 6
annotateInteractions,GInteractions,list-method
        *(*annotateInteractions*)*, 6
annotateRegions, 7
annotateRegions,GInteractions,character,vector-method
        *(*annotateRegions*)*, 7
annotationFeatures *(*getters*)*, 18
annotationFeatures,GInteractions-method
        *(*getters*)*, 18
asBED, *8*
asBED,GInteractions-method, 8
availableDisplayPars, 9

calculateDistances, 9
calculateDistances,GInteractions-method
        *(*calculateDistances*)*, 9
categoriseInteractions, 10
chromosome,InteractionTrack-method
        *(*InteractionTrack-class*)*, 21
countsBetweenAnchors, 11
countsBetweenAnchors,GInteractions,GRanges-method
        *(*countsBetweenAnchors*)*, 11

description,GInteractions-method
        *(*getters*)*, 18
description<-,GInteractions,ANY-method
        *(*setters*)*, 33

end,InteractionTrack-method
        *(*InteractionTrack-class*)*, 21
export.bed12, 12
export.bed12,GInteractions-method
        *(*export.bed12*)*, 12
export.bedpe, 12
export.bedpe,GInteractions-method
        *(*export.bedpe*)*, 12
export.chiasig, 13
export.chiasig,GInteractions-method
        *(*export.chiasig*)*, 13
export.igraph, 14
export.igraph,GInteractions-method
        *(*export.igraph*)*, 14

GenomicInteractions, 15
GenomicInteractions,ANY,ANY,ANY-method
        *(*GenomicInteractions*)*, 15
GenomicInteractions,GInteractions,ANY,ANY-method
        *(*GenomicInteractions*)*, 15
GenomicInteractions,GInteractions,numeric,ANY-method
        *(*GenomicInteractions*)*, 15
GenomicInteractions,GRanges,GRanges,GenomicRanges_OR_missi
        *(*GenomicInteractions*)*, 15
GenomicInteractions,GRanges,GRanges,numeric-method
        *(*GenomicInteractions*)*, 15
GenomicInteractions,missing,missing,GenomicRanges_OR_missi
        *(*GenomicInteractions*)*, 15
GenomicInteractions,numeric,numeric,GRanges-method
        *(*GenomicInteractions*)*, 15
GenomicInteractions-class, 16
GenomicInteractions-package, 3
get_binom_ligation_threshold, 16
get_self_ligation_threshold, 17
getters, 18

hg19.refseq.transcripts, 19
hic_example_data, 20

interactionCounts (getters), 18
interactionCounts,GInteractions-method (getters), 18
interactionCounts<- (setters), 33
interactionCounts<-,GInteractions-method (setters), 33
InteractionHelpers (is.pp), 23
InteractionTrack, 20
InteractionTrack-class, 21
is.cis (is.pp), 23
is.cis,GInteractions-method (is.pp), 23
is.dd (is.pp), 23
is.dd,GInteractions-method (is.pp), 23
is.dt (is.pp), 23
is.dt,GInteractions-method (is.pp), 23
is.pd (is.pp), 23
is.pd,GInteractions-method (is.pp), 23
is.pp, 23
is.pp,GInteractions-method (is.pp), 23
is.pt (is.pp), 23
is.pt,GInteractions-method (is.pp), 23
is.trans (is.pp), 23
is.trans,GInteractions-method (is.pp), 23
is.tt (is.pp), 23
is.tt,GInteractions-method (is.pp), 23
isInteractionType (is.pp), 23
isInteractionType,GInteractions-method (is.pp), 23

makeGenomicInteractionsFromFile, 25
message, *39*
mm9_refseq_promoters, 26

name (getters), 18
name,GInteractions-method (getters), 18
name<- (setters), 33
name<-,GInteractions-method (setters), 33

plotAvgViewpoint, 26
plotCisTrans, 27
plotCounts, 28
plotDists, 29
plotInteractionAnnotations, 29
plotSummaryStats, 30
plotViewpoint, 31

removeDups, 32
resetAnnotations, 32
resetAnnotations,GInteractions-method (resetAnnotations), 32

sameStrand, 33
setters, 33
start,InteractionTrack-method (InteractionTrack-class), 21
subset,InteractionTrack-method (InteractionTrack-class), 21
subsetByFeatures, 35
subsetByFeatures,GInteractions,character,character-method (subsetByFeatures), 35
subsetByFeatures,GInteractions,GRanges,missing-method (subsetByFeatures), 35
subsetByFeatures,GInteractions,GRangesList,missing-method (subsetByFeatures), 35
sum,GInteractions-method, 36
summariseByFeaturePairs, 36
summariseByFeaturePairs,GInteractions-method (summariseByFeaturePairs), 36
summariseByFeatures, 37
summariseByFeatures,GInteractions-method (summariseByFeatures), 37

thymus_enh, 38

updateObject,GenomicInteractions-method, 39

viewPoint, 40