

# Package ‘plyinteractions’

December 19, 2024

**Title** Extending tidy verbs to genomic interactions

**Description** Operate on `GInteractions` objects as tabular data using `dplyr`-like verbs. The functions and methods in `plyinteractions` provide a grammatical approach to manipulate `GInteractions`, to facilitate their integration in genomic analysis workflows.

**Version** 1.5.0

**Date** 2023-08-21

**License** Artistic-2.0

**URL** <https://github.com/js2264/plyinteractions>

**BugReports** <https://github.com/js2264/plyinteractions/issues>

**biocViews** Software, Infrastructure

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**Imports** InteractionSet, GenomeInfoDb, BiocGenerics, GenomicRanges, plyranges, IRanges, S4Vectors, rlang, dplyr, tibble, tidyselect, methods, utils

**Suggests** tidyverse, BSgenome.Mmusculus.UCSC.mm10, Biostrings, BiocParallel, scales, HiContactsData, rtracklayer, BiocStyle, covr, knitr, rmarkdown, sessioninfo, testthat (>= 3.0.0), RefManageR

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**RoxygenNote** 7.3.1

**Collate** 'AllGenerics.R' 'AllClasses.R' 'add\_pairdist.R' 'anchor.R' 'annotate.R' 'arrange.R' 'count-overlaps.R' 'count.R' 'data.R' 'export.R' 'filter-overlaps.R' 'filter.R' 'find-overlaps.R' 'flank.R' 'ginteractions-construct.R' 'ginteractions-env.R' 'ginteractions-getters.R' 'ginteractions-scoping.R' 'ginteractions-setters.R' 'tbl\_vars.R' 'group\_data.R' 'group\_by.R' 'internals.R' 'join-overlap-left.R' 'methods-AnchoredPinnedGInteractions.R'

```
'methods-DelegatingGInteractions.R'
'methods-GroupedGInteractions.R'
'methods-PinnedGInteractions.R' 'methods-show.R' 'mutate.R'
'pair-granges.R' 'pin.R' 'plyinteractions.R'
'reexports-dplyr.R' 'reexports-plyranges.R' 'reexports.R'
'rename.R' 'replace-anchors.R' 'select.R' 'shift.R' 'slice.R'
'stretch.R' 'summarize.R'
```

**LazyData** false

**Depends** R (>= 4.3.0)

**git\_url** <https://git.bioconductor.org/packages/plyinteractions>

**git\_branch** devel

**git\_last\_commit** ac2f117

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.21

**Date/Publication** 2024-12-18

**Author** Jacques Serizay [aut, cre]

**Maintainer** Jacques Serizay <jacquesserizay@gmail.com>

## Contents

plyinteractions-package . . . . .	3
add_pairdist . . . . .	4
anchors1 . . . . .	5
annotate . . . . .	7
as_ginteractions . . . . .	9
ce10_ARCC . . . . .	11
ce10_REs . . . . .	12
delegating-ginteractions-methods . . . . .	12
dplyr-arrange . . . . .	14
dplyr-count . . . . .	15
dplyr-filter . . . . .	16
dplyr-group_by . . . . .	17
dplyr-mutate . . . . .	19
dplyr-rename . . . . .	21
dplyr-select . . . . .	22
dplyr-slice . . . . .	23
dplyr-summarize . . . . .	24
ginteractions-anchor . . . . .	25
ginteractions-count-overlaps . . . . .	27
ginteractions-export . . . . .	28
ginteractions-filter-overlaps . . . . .	29
ginteractions-find-overlaps . . . . .	31
ginteractions-join-overlap-left . . . . .	33
GM12878_HiCCUPS . . . . .	35
group-group_data . . . . .	35

mutate . . . . .	37
pair_granges . . . . .	39
pin . . . . .	39
plyranges-flank . . . . .	42
plyranges-shift . . . . .	44
plyranges-stretch . . . . .	45
replace_anchors . . . . .	46
set_seqnames1 . . . . .	48

<b>Index</b>	<b>51</b>
--------------	-----------

---

plyinteractions-package

*plyinteractions: a grammar of data manipulation for genomic interactions*

---

## Description

plyinteractions is a dplyr-like API to the GInteractions infrastructure in Bioconductor.

## Details

plyinteractions provides a consistent interface for importing and wrangling genomic interactions from a variety of sources. The package defines a grammar of genomic interactions manipulation through a set of verbs. These verbs can be used to construct human-readable analysis pipelines based on GInteractions.

- Group genomic interactions with `group_by`;
- Summarize grouped genomic interactions with `summarize`;
- Tally/count grouped genomic interactions with `tally` and `count`;
- Modify genomic interactions with `mutate`;
- Subset genomic interactions with `filter` using `<data-masking>` and logical expressions;
- Pick out any columns from the associated metadata with `select` using `<tidy-select>` arguments;
- Subset using indices with `slice`;
- Order genomic interactions with `arrange` using categorical/numerical variables.

For more details on the features of plyinteractions, read the vignette: `browseVignettes(package = "plyinteractions")`

## Author(s)

**Maintainer:** Jacques Serizay <jacquesserizay@gmail.com>

**See Also**

Useful links:

- <https://github.com/js2264/plyinteractions>
- Report bugs at <https://github.com/js2264/plyinteractions/issues>

---

add_pairedist	<i>Appends distance between interaction anchors</i>
---------------	---

---

**Description**

Appends distance between interaction anchors, using `InteractionSet::pairedist`

**Usage**

```
add_pairedist(x, type = "mid", colname = "pairedist")
```

**Arguments**

x	The query <code>GInteractions</code>
type	A character string specifying the type of distance to compute. Can take values of "mid", "gap", "span", "diag" or "intra".
colname	name of column to hold pair distance values

**Value**

The `GInteractions` with an additional column containing the distance between each pair of anchors.

**Examples**

```
gi <- read.table(text = "
chr1 100 200 chr1 5000 5100 bedpe_example1 30 + -
chr1 1000 5000 chr2 3000 3800 bedpe_example2 100 + -",
col.names = c(
  "seqnames1", "start1", "end1",
  "seqnames2", "start2", "end2", "name", "score", "strand1", "strand2")
) |> as_ginteractions()

add_pairedist(gi)
```

---

anchors1	<i>Enhanced GInteractions getters</i>
----------	---------------------------------------

---

**Description**

Enhanced GInteractions getters

**Usage**

anchors1(x)

anchors2(x)

seqnames1(x)

seqnames2(x)

start1(x)

start2(x)

end1(x)

end2(x)

width1(x)

width2(x)

strand1(x)

strand2(x)

ranges1(x)

ranges2(x)

## S4 method for signature 'GInteractions'  
x\$name

## S4 method for signature 'GInteractions'  
anchors1(x)

## S4 method for signature 'GInteractions'  
anchors2(x)

## S4 method for signature 'GInteractions'

```
seqnames1(x)

## S4 method for signature 'GInteractions'
seqnames2(x)

## S4 method for signature 'GInteractions'
start1(x)

## S4 method for signature 'GInteractions'
start2(x)

## S4 method for signature 'GInteractions'
end1(x)

## S4 method for signature 'GInteractions'
end2(x)

## S4 method for signature 'GInteractions'
width1(x)

## S4 method for signature 'GInteractions'
width2(x)

## S4 method for signature 'GInteractions'
strand1(x)

## S4 method for signature 'GInteractions'
strand2(x)

## S4 method for signature 'GInteractions'
ranges1(x)

## S4 method for signature 'GInteractions'
ranges2(x)
```

### Arguments

x	a GInteractions object
name	The pattern or name of a column stored in the GInteractions metadata (mcols).

### Value

One of the core GInteractions fields (e.g. seqnames1, start1, ...) or one of the metadata columns when using \$. Note that auto-completion works with \$.

### Examples

```
gi <- data.frame(
  seqnames1 = 'chr1', start1 = 1, end1 = 10,
```

```

    seqnames2 = 'chr1', start2 = 2, end2 = 20
) |> as_ginteractions() |> mutate(type = 'cis')
anchors1(gi)
anchors2(gi)
seqnames1(gi)
seqnames2(gi)
start1(gi)
start2(gi)
end1(gi)
end2(gi)
width1(gi)
width2(gi)
ranges1(gi)
ranges2(gi)
strand1(gi)
strand2(gi)
gi$type

```

---

 annotate

*Annotate both anchors of a GInteractions*


---

### Description

For each interaction in a `GInteractions` object, `annotate` returns the pairs of annotations from the `GRanges` object it overlaps with.

### Usage

```
annotate(x, y, by)
```

```
annotate_directed(x, y, by)
```

```
## S4 method for signature 'GInteractions,GRanges,character'
annotate(x, y, by)
```

```
## S4 method for signature 'GInteractions,GRanges,character'
annotate_directed(x, y, by)
```

### Arguments

<code>x</code>	a <code>GInteractions</code> object
<code>y</code>	a <code>GRanges</code> object to extract annotations from
<code>by</code>	Column name from <code>y</code> to use to extract annotations

### Value

a `GInteractions` object with two extra metadata columns named `by.1` and `by.2`.

## Examples

```
#####
# 1. Basic example
#####

gi <- read.table(text = "
  chr1 11 20 - chr1 21 30 +
  chr1 21 30 + chr2 51 60 +",
  col.names = c(
    "seqnames1", "start1", "end1", "strand1",
    "seqnames2", "start2", "end2", "strand2"
  )
) |> as_ginteractions()

gr <- GenomicRanges::GRanges(c("chr1:20-30:+", "chr2:55-65:+")) |>
  plyranges::mutate(id = 1:2)

annotate(gi, gr, by = 'id')

annotate_directed(gi, gr, by = 'id')

#####
# 2. Match loops with tiled genomic bins
#####

data(GM12878_HiCCUPS)
loops <- GM12878_HiCCUPS |>
  pin_by('first') |>
  anchor_center() |>
  mutate(width1 = 500) |>
  pin_by('second') |>
  anchor_center() |>
  mutate(width2 = 500)

genomic_bins <- GenomeInfoDb::getChromInfoFromUCSC(
  'hg19', assembled.molecules.only = TRUE, as.Seqinfo = TRUE
) |>
  GenomicRanges::tileGenome(tilewidth = 10000) |>
  unlist() |>
  plyranges::mutate(binID = seq_len(plyranges::n()))

annotate(loops, genomic_bins, by = 'binID') |>
  select(starts_with('binID'))

#####
# 3. Annotate interactions by a set of regulatory elements
#####

data(ce10_ARCC)
data(ce10_REs)
annotate(ce10_ARCC, ce10_REs, by = 'annot') |>
  count(annot.1, annot.2) |>
```



```
as.data.frame() |>
dplyr::arrange(desc(n))
```

---

as_ginteractions	<i>Construct a GInteractions object from a tibble, DataFrame or data.frame</i>
------------------	--

---

### Description

The `as_ginteractions` function looks for column names in `.data` called `seqnames{1,2}`, `start{1,2}`, `end{1,2}`, and `strand{1,2}` in order to construct a `GInteractions` object. By default other columns in `.data` are placed into the `mcols` (metadata columns) slot of the returned object.

### Usage

```
as_ginteractions(
  .data,
  ...,
  keep.extra.columns = TRUE,
  starts.in.df.are.0based = FALSE
)
```

### Arguments

<code>.data</code>	A <code>data.frame()</code> , <code>DataFrame()</code> or <code>tibble()</code> to construct a <code>GInteractions</code> object from.
<code>...</code>	Optional named arguments specifying which the columns in <code>.data</code> contain the core components a <code>GInteractions</code> object.
<code>keep.extra.columns</code>	TRUE or FALSE (the default). If TRUE, the columns in <code>df</code> that are not used to form the genomic ranges of the returned <code>GRanges</code> object are then returned as metadata columns on the object. Otherwise, they are ignored.
<code>starts.in.df.are.0based</code>	TRUE or FALSE (the default). If TRUE, then the start positions of the genomic ranges in <code>df</code> are considered to be 0-based and are converted to 1-based in the returned <code>GRanges</code> object.

### Value

a `GInteractions` object.

### See Also

[InteractionSet::GInteractions\(\)](#)

## Examples

```
#####
# 1. GInteractions from bedpe files imported into a data.frame
#####

bedpe <- read.table(text = "
chr1 100 200 chr1 5000 5100 bedpe_example1 30 + -
chr1 1000 5000 chr1 3000 3800 bedpe_example2 100 + -",
col.names = c(
  "chrom1", "start1", "end1",
  "chrom2", "start2", "end2", "name", "score", "strand1", "strand2"))
bedpe |>
  as_ginteractions(seqnames1 = chrom1, seqnames2 = chrom2)

#####
# 2. GInteractions from standard pairs files imported into a data.frame
#####

# Note how the pairs are 0-based and no "end" field is provided
# (the standard pairs file format does not have "end" fields)
# We can provide width1 and width2 to fix this problem.

pairs <- read.table(text = "
pair1 chr1 10000 chr1 20000 + +
pair2 chr1 50000 chr1 70000 + +
pair3 chr1 60000 chr2 10000 + +
pair4 chr1 30000 chr3 40000 + -",
col.names = c(
  "pairID", "chr1", "pos1", "chr2", "pos2", "strand1", "strand2")
)
pairs |>
  as_ginteractions(
    seqnames1 = chr1, start1 = pos1, width1 = 1000,
    seqnames2 = chr2, start2 = pos2, width2 = 1000,
    starts.in.df.are.0based = TRUE
  )

#####
# 3. GInteractions from data.frame with extra fields
#####

df <- read.table(text = "
chr1 100 200 chr1 5000 5100
chr1 1000 5000 chr1 3000 3800",
col.names = c("chr1", "start1", "end1", "chr2", "start2", "end2"))
df |>
  as_ginteractions(seqnames1 = chr1, seqnames2 = chr2)

df <- read.table(text = "
chr1 100 200 chr1 5000 5100
chr1 1000 5000 chr1 3000 3800",
col.names = c("chr1", "start1", "end1", "chr2", "start2", "end2"))
```

```

df |>
  as_ginteractions(
    seqnames1 = chr1, seqnames2 = chr2, strand1 = '+', strand2 = '-'
  )

data.frame(type = "cis", count = 3) |>
  as_ginteractions(
    seqnames1 = 'chr1', start1 = 1, end1 = 10,
    seqnames2 = 'chr1', start2 = 40, end2 = 50
  )

#####
# 4. GInteractions from a real like pairs files
#####

pairsf <- system.file('extdata', 'pairs.gz', package = 'plyinteractions')
pairs <- read.table(pairsf, comment.char = '#', header = FALSE)
head(pairs)
pairs |>
  as_ginteractions(
    seqnames1 = V2, start1 = V3, width1 = 1, strand1 = V6,
    seqnames2 = V4, start2 = V5, width2 = 1, strand2 = V7,
    starts.in.df.are.0based = TRUE
  )

```

ce10\_ARCC

*Interactions identified in L3 C. elegans by ARC-C***Description**

Supplemental Table 2 obtained from Genome Biology online publication.

Huang N, Seow WQ, Appert A, Dong Y, Stempor P and Ahringer J Accessible Region Conformation Capture (ARC-C) gives high-resolution insights into genome architecture and regulation. *Genome Res* 2022 Feb;32(2):357-366. PMID: 34933938

**Usage**

```
ce10_ARCC
```

**Format**

An object of class GInteractions of length 14992.

**Value**

A GInteractions object

**Source**

<https://genome.cshlp.org/content/early/2021/12/21/gr.275669.121>

---

ce10\_REs

*Annotated regulatory elements in C. elegans*

---

**Description**

Figure 2 - Source data 1 obtained from eLife online publication.

Jänes J, Dong Y, Schoof M, Serizay J, Appert A, Cerrato C, Woodbury C, Chen R, Gemma C, Huang N, Kissiov D, Stempor P, Steward A, Zeiser E, Sauer S and Ahringer J Chromatin accessibility dynamics across *C. elegans* development and ageing. *Elife* 2018 Oct 26;7. PMID: 30362940

**Usage**

ce10\_REs

**Format**

An object of class GRanges of length 42245.

**Value**

A GRanges object

**Source**

<https://genome.cshlp.org/content/early/2021/12/21/gr.275669.121>

---

delegating-ginteractions-methods

*Methods for DelegatingGInteractions objects*

---

**Description**

Methods for DelegatingGInteractions objects

**Usage**

```
## S4 method for signature 'DelegatingGInteractions'  
anchors1(x)
```

```
## S4 method for signature 'DelegatingGInteractions'  
ranges1(x)
```

```
## S4 method for signature 'DelegatingGInteractions'  
seqnames1(x)
```

```
## S4 method for signature 'DelegatingGInteractions'  
start1(x)  
  
## S4 method for signature 'DelegatingGInteractions'  
end1(x)  
  
## S4 method for signature 'DelegatingGInteractions'  
width1(x)  
  
## S4 method for signature 'DelegatingGInteractions'  
strand1(x)  
  
## S4 method for signature 'DelegatingGInteractions'  
anchors2(x)  
  
## S4 method for signature 'DelegatingGInteractions'  
ranges2(x)  
  
## S4 method for signature 'DelegatingGInteractions'  
seqnames2(x)  
  
## S4 method for signature 'DelegatingGInteractions'  
start2(x)  
  
## S4 method for signature 'DelegatingGInteractions'  
end2(x)  
  
## S4 method for signature 'DelegatingGInteractions'  
width2(x)  
  
## S4 method for signature 'DelegatingGInteractions'  
strand2(x)  
  
## S4 method for signature 'DelegatingGInteractions'  
anchors(x)  
  
## S4 method for signature 'DelegatingGInteractions'  
regions(x)  
  
## S4 method for signature 'DelegatingGInteractions'  
seqinfo(x)  
  
## S4 method for signature 'DelegatingGInteractions'  
mcols(x)  
  
## S4 method for signature 'DelegatingGInteractions'  
show(object)
```

**Value**

One of the core GInteractions fields (e.g. seqnames1, start1, ...)

---

dplyr-arrange	<i>Arrange a GInteractions by a column</i>
---------------	--

---

**Description**

Arrange a GInteractions by a column

**Usage**

```
## S3 method for class 'GInteractions'
arrange(.data, ...)
```

**Arguments**

.data	a GInteractions object
...	Variables, or functions of variables. Use <code>dplyr::desc()</code> to sort a variable in descending order.

**Value**

a GInteractions object.

**Examples**

```
gi <- read.table(text = "
chr1 1 10 chr1 1 10
chr1 2 10 chr2 1 10
chr3 3 10 chr3 1 10
chr4 4 10 chr4 1 10
chr5 5 10 chr5 1 10",
col.names = c(
  "seqnames1", "start1", "end1",
  "seqnames2", "start2", "end2")
) |>
  as_ginteractions() |>
  mutate(cis = seqnames1 == seqnames2, score = runif(5)*100, gc = runif(5))
gi

#####
# 1. Arrange GInteractions by a numerical column
#####

gi |> arrange(gc)

#####
# 2. Arrange GInteractions by a logical column
```

```
#####
gi |> arrange(cis)

#####
# 3. Arrange GInteractions by a factor
#####

gi |>
  mutate(rep = factor(c("rep1", "rep2", "rep1", "rep2", "rep1"))) |>
  arrange(rep)

#####
# 4. Combine sorting variables
#####

gi |>
  mutate(rep = factor(c("rep1", "rep2", "rep1", "rep2", "rep1"))) |>
  arrange(dplyr::desc(rep), score)
```

---

dplyr-count

*Count or tally GInteractions per group*


---

## Description

Count or tally GInteractions per group

## Usage

```
## S3 method for class 'GroupedGInteractions'
tally(x, wt = NULL, sort = FALSE, name = NULL)

## S3 method for class 'GroupedGInteractions'
count(x, ..., wt = NULL, sort = FALSE, name = NULL)

## S3 method for class 'GInteractions'
count(x, ..., wt = NULL, sort = FALSE, name = NULL)
```

## Arguments

x	A grouped GInteractions object
wt	<a href="#">&lt;data-masking&gt;</a> Frequency weights. Can be NULL or a variable: <ul style="list-style-type: none"> <li>• If NULL (the default), counts the number of rows in each group.</li> <li>• If a variable, computes <code>sum(wt)</code> for each group.</li> </ul>
sort	If TRUE, will show the largest groups at the top.
name	The name of the new column in the output.
...	<a href="#">&lt;data-masking&gt;</a> Variables to group by.

**Value**

a `S4Vectors::DataFrame()` object, with an added column with the count/tally per group.

**Examples**

```
gi <- read.table(text = "
chr1 11 20 chr1 21 30 + +
chr1 11 20 chr1 51 55 + +
chr1 11 30 chr1 51 55 - -
chr1 11 30 chr2 51 60 - -",
col.names = c(
  "seqnames1", "start1", "end1",
  "seqnames2", "start2", "end2", "strand1", "strand2")
) |>
  as_ginteractions() |>
  mutate(score = runif(4), type = c('cis', 'cis', 'cis', 'trans'))

#####
# 1. Tally groups
#####

gi

gi |> group_by(strand1) |> tally()

gi |> group_by(type) |> tally()

gi |> group_by(type) |> tally(wt = score)

#####
# 2. Count per groups
#####

gi |> count(type)

gi |> group_by(type) |> count(strand1)

gi |> group_by(type, strand1) |> count(wt = score)
```

---

dplyr-filter

*Subset a GInteractions with tidyverse-like filter*


---

**Description**

Subset a GInteractions with tidyverse-like filter

**Usage**

```
## S3 method for class 'GInteractions'
filter(.data, ...)
```



**Arguments**

`.data` a `GInteractions` object

`...` Expressions that return a logical value, and are defined in terms of the variables in `.data`. If multiple expressions are included, they are combined with the `&` operator. Only rows for which all conditions evaluate to `TRUE` are kept.

**Value**

a `GInteractions` object.

**Examples**

```
gi <- read.table(text = "
chr1 1 10 chr1 1 10
chr1 2 10 chr2 1 10
chr3 3 10 chr3 1 10
chr4 4 10 chr4 1 10
chr5 5 10 chr5 1 10",
col.names = c(
  "seqnames1", "start1", "end1",
  "seqnames2", "start2", "end2")
) |>
  as_ginteractions() |>
  mutate(cis = seqnames1 == seqnames2, score = runif(5)*100, gc = runif(5))
gi

#####
# 1. Filter metadata columns from GInteractions by condition
#####

gi |> filter(gc > 0.1)
gi |> filter(gc > 0.1, score > 50)
gi |> filter(cis)

#####
# 2. On-the-fly calculations
#####

gi
gi |> filter(start1 >= start2 + 3)
gi |> filter(score * gc > score * 0.5)
```

**Description**

Group `GInteractions` by columns

**Usage**

```
## S3 method for class 'GInteractions'
group_by(.data, ..., .add = FALSE)

## S3 method for class 'DelegatingGInteractions'
group_by(.data, ..., .add = FALSE)

## S3 method for class 'GroupedGInteractions'
ungroup(x, ...)
```

**Arguments**

<code>.data, x</code>	a (Grouped)GInteractions object
<code>...</code>	Column(s) to group by.
<code>.add</code>	When FALSE, the default, <code>group_by()</code> will override existing groups. To add to the existing groups, use <code>.add = TRUE</code> .

**Value**

a GroupedGInteractions object. When a (Anchored)PinnedGInteractions object is grouped, both anchoring and pinning are dropped.

**Examples**

```
gi <- read.table(text = "
chr1 11 20 chr1 21 30
chr1 11 20 chr1 51 55
chr1 11 30 chr1 51 55
chr1 11 30 chr2 51 60",
col.names = c(
  "seqnames1", "start1", "end1",
  "seqnames2", "start2", "end2")
) |>
  as_ginteractions() |>
  mutate(type = c('cis', 'cis', 'cis', 'trans'), score = runif(4))

#####
# 1. Group by core column
#####

gi |> group_by(end1)

gi |> group_by(end1, end2) |> group_data()

#####
# 2. Group by metadata column
#####

gi |> group_by(type) |> group_data()
```

```
#####
# 3. Combine core and metadata column grouping
#####

gi |> group_by(end1, type)
gi |> group_by(end1, type) |> group_data()

#####
# 4. Create a new column and group by this new variable
#####

gi |> group_by(class = c(1, 2, 1, 2))

#####
# 5. Replace or add groups to a GroupedGInteractions
#####

ggi <- gi |> group_by(class = c(1, 2, 1, 2))
ggi |> group_data()
ggi |> group_by(type) |> group_data()
ggi |> group_by(type, .add = TRUE) |> group_data()

#####
# 6. Ungroup GInteractions
#####

ggi <- gi |> group_by(type, class = c(1, 2, 1, 2))
ggi
ungroup(ggi, type)
ungroup(ggi, class)
```

---

dplyr-mutate

*Mutate columns from a GInteractions object*


---

## Description

Mutate columns from a GInteractions object

## Usage

```
## S3 method for class 'GInteractions'
mutate(.data, ...)
```

## Arguments

.data	a GInteractions object
...	Optional named arguments specifying which the columns in .data to create/modify.

**Value**

a GInteractions object.

**Examples**

```

gi <- read.table(text = "
chr1 10 20 chr1 50 51
chr1 10 50 chr2 30 40",
col.names = c("chr1", "start1", "end1", "chr2", "start2", "end2")) |>
  as_ginteractions(seqnames1 = chr1, seqnames2 = chr2)

#####
# 1. Add metadata columns to a GInteractions object
#####

gi |>
  mutate(type = c('cis', 'trans'), score = runif(2)) |>
  mutate(type2 = type)

#####
# 2. More complex, nested or inplace changes
#####

gi |>
  mutate(type = c('cis', 'trans'), score = runif(2)) |>
  mutate(type2 = type) |>
  mutate(count = c(1, 2), score = count * 2, new_col = paste0(type2, score))

#####
# 3. Core GInteractions columns can also be modified
#####

gi |>
  mutate(start1 = 1, end1 = 10, width2 = 30, strand2 = c('-', '+'))

# Note how the core columns are modified sequentially

gi |>
  mutate(start1 = 1, end1 = 10)

gi |>
  mutate(start1 = 1, end1 = 10, width1 = 50)

#####
# 4. Evaluating core GInteractions columns
#####

gi |>
  mutate(
    score = runif(2),
    cis = seqnames1 == seqnames2,
    distance = ifelse(cis, start2 - end1, NA)
  )

```

)

---

`dplyr-rename`*Rename columns from a GInteractions with tidyverse-like rename*

---

**Description**

Rename columns from a GInteractions with tidyverse-like rename

**Usage**

```
## S3 method for class 'GInteractions'
rename(.data, ...)
```

**Arguments**

```
.data      a GInteractions object
...        Use new_name = old_name to rename selected variables.
```

**Value**

a GInteractions object.

**Examples**

```
gi <- read.table(text = "
chr1 10 20 chr1 50 51
chr1 10 50 chr2 30 40",
col.names = c("chr1", "start1", "end1", "chr2", "start2", "end2")) |>
  as_ginteractions(seqnames1 = chr1, seqnames2 = chr2) |>
  mutate(type = c('cis', 'trans'), score = runif(2))

#####
# 1. Rename metadata columns to a GInteractions object
#####

gi |> rename(interaction_type = type, GC = score)
```

---

dplyr-select

*Select columns within GInteractions metadata columns*


---

**Description**

Select columns within GInteractions metadata columns

**Usage**

```
## S3 method for class 'GInteractions'
select(.data, ..., .drop_ranges = FALSE)
```

**Arguments**

<code>.data</code>	a GInteractions object
<code>...</code>	Integer indicating rows to keep.
<code>.drop_ranges</code>	if TRUE, returns a DataFrame object. In this case, it enables selection of any column including core GInteractions columns.

**Value**

a GInteractions object.

**Examples**

```
gi <- read.table(text = "
chr1 1 10 chr1 1 10
chr2 1 10 chr2 1 10
chr3 1 10 chr3 1 10
chr4 1 10 chr4 1 10
chr5 1 10 chr5 1 10",
col.names = c(
  "seqnames1", "start1", "end1",
  "seqnames2", "start2", "end2")
) |>
  as_ginteractions() |>
  mutate(score = runif(5)*100, cis = TRUE, gc = runif(5))

#####
# 1. Select metadata columns from GInteractions by index
#####

gi |> select(2, 1)
gi |> select(-3)

#####
# 2. Select metadata columns from GInteractions by name
#####
```

```

gi |> select(gc, score)

#####
# 3. Select metadata columns from GInteractions with <tidy-select>
#####

gi |> select(contains('s'))
gi |> select(matches('^s'))

#####
# 4. Select core and metadata columns with .drop_ranges = TRUE
#####

gi |> select(matches('^s'), .drop_ranges = TRUE)

```

---

dplyr-slice

*Slice a GInteractions rows by their index*


---

## Description

Slice a GInteractions rows by their index

## Usage

```
## S3 method for class 'GInteractions'
slice(.data, ...)
```

## Arguments

.data            a GInteractions object  
...               Integer indicating rows to keep.

## Value

a GInteractions object.

## Examples

```

gi <- read.table(text = "
chr1 1 10 chr1 1 10
chr2 1 10 chr2 1 10
chr3 1 10 chr3 1 10
chr4 1 10 chr4 1 10
chr5 1 10 chr5 1 10",
col.names = c(
  "seqnames1", "start1", "end1",
  "seqnames2", "start2", "end2")
) |>

```

```

as_ginteractions()

#####
# 1. Slice a GInteractions
#####

gi |> slice(1, 2, 3)
gi |> slice(-3)
gi |> slice(1:2, 5:4)

```

---

dplyr-summarize

*Summarize GInteractions per group*


---

## Description

Summarize GInteractions per group

## Usage

```
## S3 method for class 'GroupedGInteractions'
summarise(.data, ...)
```

```
## S3 method for class 'GroupedGInteractions'
summarize(.data, ...)
```

## Arguments

<code>.data</code>	a (grouped) GInteractions object
<code>...</code>	Name-value pairs of summary functions. The name will be the name of the variable in the result.

## Value

a `S4Vectors::DataFrame()` object:

- The rows come from the underlying `group_keys()`.
- The columns are a combination of the grouping keys and the summary expressions that you provide.
- GInteractions class is **not** preserved, as a call to `summarize` fundamentally creates a new data frame

## Examples

```

gi <- read.table(text = "
chr1 11 20 chr1 21 30 + +
chr1 11 20 chr1 51 55 + +
chr1 11 30 chr1 51 55 - -
chr1 11 30 chr2 51 60 - -",

```



```

col.names = c(
  "seqnames1", "start1", "end1",
  "seqnames2", "start2", "end2", "strand1", "strand2")
) |>
  as_ginteractions() |>
  mutate(score = runif(4), type = c('cis', 'cis', 'cis', 'trans'))

#####
# 1. Summarize a single column
#####

gi

gi |> group_by(type) |> summarize(m = mean(score))

gi |> group_by(strand1) |> summarize(m = mean(score))

df <- gi |>
  group_by(strand1) |>
  summarize(m = mean(score), n = table(seqnames2))
df

df$n

#####
# 2. Summarize by multiple columns
#####

gi |>
  group_by(strand1, seqnames2) |>
  summarise(m = mean(score), n = table(type))

```

---

ginteractions-anchor *Manage GInteractions anchors with plyranges*

---

## Description

Manage GInteractions anchors with plyranges

## Usage

```
## S3 method for class 'AnchoredPinnedGInteractions'
anchor(x)
```

```
## S3 method for class 'AnchoredPinnedGInteractions'
unanchor(x)
```

```
## S3 method for class 'PinnedGInteractions'
anchor_start(x)
```

```
## S3 method for class 'PinnedGInteractions'
anchor_end(x)

## S3 method for class 'PinnedGInteractions'
anchor_center(x)

## S3 method for class 'PinnedGInteractions'
anchor_3p(x)

## S3 method for class 'PinnedGInteractions'
anchor_5p(x)

## S3 method for class 'AnchoredPinnedGInteractions'
anchor_start(x)

## S3 method for class 'AnchoredPinnedGInteractions'
anchor_end(x)

## S3 method for class 'AnchoredPinnedGInteractions'
anchor_center(x)

## S3 method for class 'AnchoredPinnedGInteractions'
anchor_3p(x)

## S3 method for class 'AnchoredPinnedGInteractions'
anchor_5p(x)
```

### Arguments

x                    A PinnedGInteractions object

### Value

- anchor\_\* functions return an AnchoredPinnedGInteractions object.
- anchor returns a character string indicating where the pinned anchors are anchored at.
- unanchor removes the anchoring for a AnchoredPinnedGInteractions object.

### Examples

```
gi <- read.table(text = "
chr1 11 20 chr1 21 30 + +
chr1 11 20 chr1 51 55 + +
chr1 11 30 chr1 51 55 - -
chr1 11 30 chr2 51 60 - -",
col.names = c(
  "seqnames1", "start1", "end1",
  "seqnames2", "start2", "end2", "strand1", "strand2")
) |>
as_ginteractions() |>
```

```

mutate(score = runif(4), type = c('cis', 'cis', 'cis', 'trans'))

#####
# 1. Anchoring pinned genomic interactions with plyranges
#####

gi |> pin_by("second") |> anchor_end()

```

---

ginteractions-count-overlaps

*Count overlaps between a query GInteractions and a GRanges*

---

## Description

Count overlaps between a query GInteractions and a GRanges

## Usage

```

## S3 method for class 'PinnedGInteractions'
count_overlaps(x, y, maxgap = -1L, minoverlap = 0L)

## S3 method for class 'GInteractions'
count_overlaps(x, y, maxgap = -1L, minoverlap = 0L)

## S3 method for class 'PinnedGInteractions'
count_overlaps_directed(x, y, maxgap = -1L, minoverlap = 0L)

## S3 method for class 'GInteractions'
count_overlaps_directed(x, y, maxgap = -1L, minoverlap = 0L)

```

## Arguments

x	A (Pinned)GInteractions object
y	A GRanges object
maxgap, minoverlap	See <a href="#">?countOverlaps</a> in the <b>GenomicRanges</b> package for a description of these arguments

## Value

An integer vector of same length as x.

## Pinned GInteractions

When using `count_overlaps()` with a PinnedGInteractions object, only the pinned anchors are used to check for overlap with y. This is equivalent to specifying `use.region="both"` in `InteractionSet::findOverlaps()`.

## Examples

```

gi <- read.table(text = "
chr1 11 20 - chr1 21 30 +
chr1 11 20 - chr1 51 55 +
chr1 21 30 - chr1 51 55 +
chr1 21 30 - chr2 51 60 +",
col.names = c(
  "seqnames1", "start1", "end1", "strand1",
  "seqnames2", "start2", "end2", "strand2"
)
) |> as_ginteractions() |> mutate(id = 1:4, type = 'gi')

gr <- GenomicRanges::GRanges(
  c("chr1:20-30:+", "chr2:55-65:-")
) |> plyranges::mutate(id = 1:2, type = 'gr')

gi

gr

#####
# 1. Count overlaps between GInteractions and a subject GRanges
#####

count_overlaps(gi, gr)

count_overlaps_directed(gi, gr)

#####
# 2. Count overlaps between PinnedGInteractions and a subject GRanges
#####

gi |> pin_by("first") |> count_overlaps(gr)

gi |> pin_by("second") |> count_overlaps(gr)

gi |> pin_by("first") |> count_overlaps_directed(gr)

gi |> pin_by("second") |> count_overlaps_directed(gr)

```

---

ginteractions-export    *Export GInteractions as bedpe or pairs files*

---

## Description

`write_*` functions are provided to export a `GInteractions` object into these two file formats. See 4DN documentation ([https://github.com/4dn-dcic/pairix/blob/master/pairs\\_format\\_specification.md](https://github.com/4dn-dcic/pairix/blob/master/pairs_format_specification.md)) and UCSC documentation (<https://bedtools.readthedocs.io/en/latest/content/general-usage.html#bedpe-format>) for more details.

**Usage**

```
write_bedpe(x, file, scores = NULL)

write_pairs(x, file, seqlengths = GenomeInfoDb::seqlengths(x))
```

**Arguments**

x	a GInteractions object.
file	path to a .bedpe or .pairs file to save the genomic interactions.
scores	Name of column to extract scores from.
seqlengths	Named vector indicating the chromosome sizes.

**Value**

TRUE

**Examples**

```
gi <- read.table(text = "
chr1 100 200 chr1 5000 5100 bedpe_example1 30 + -
chr1 1000 5000 chr1 3000 3800 bedpe_example2 100 + -",
col.names = c(
  "seqnames1", "start1", "end1",
  "seqnames2", "start2", "end2", "name", "score", "strand1", "strand2")
) |> as_ginteractions()

write_bedpe(gi, 'gi.bedpe')
write_pairs(gi, 'gi.pairs')
```

---

ginteractions-filter-overlaps

*Filter GInteractions overlapping with a GRanges*

---

**Description**

Filter GInteractions overlapping with a GRanges

**Usage**

```
## S3 method for class 'PinnedGInteractions'
filter_by_overlaps(x, y, maxgap = -1L, minoverlap = 0L)

## S3 method for class 'GInteractions'
filter_by_overlaps(x, y, maxgap = -1L, minoverlap = 0L)

## S3 method for class 'PinnedGInteractions'
filter_by_non_overlaps(x, y, maxgap = -1L, minoverlap = 0L)
```

```
## S3 method for class 'GInteractions'
filter_by_non_overlaps(x, y, maxgap = -1L, minoverlap = 0L)
```

### Arguments

```
x          A (Pinned)GInteractions object
y          A GRanges object
maxgap, minoverlap
           See ?countOverlaps in the GenomicRanges package for a description of these
           arguments
```

### Value

An integer vector of same length as x.

### Pinned GInteractions

When using `filter_by_overlaps()` with a `PinnedGInteractions` object, only the pinned anchors are used to check for overlap with `y`. This is equivalent to specifying `use.region="both"` in `InteractionSet::findOverlaps()`.

### Examples

```
gi <- read.table(text = "
  chr1 11 20 - chr1 21 30 +
  chr1 11 20 - chr1 51 55 +
  chr1 21 30 - chr1 51 55 +
  chr1 21 30 - chr2 51 60 +",
  col.names = c(
    "seqnames1", "start1", "end1", "strand1",
    "seqnames2", "start2", "end2", "strand2")
) |> as_ginteractions() |> mutate(id = 1:4, type = 'gi')

gr <- GenomicRanges::GRanges(
  c("chr1:20-30:+", "chr2:55-65:-")
) |> plyranges::mutate(id = 1:2, type = 'gr')

gi

gr

#####
# 1. Filter GInteractions overlapping with a subject GRanges
#####

filter_by_overlaps(gi, gr)

filter_by_non_overlaps(gi, gr)

#####
```

```
# 2. Filter PinnedGInteractions overlapping with a subject GRanges
#####

gi |> pin_by("first") |> filter_by_overlaps(gr)

gi |> pin_by("first") |> filter_by_non_overlaps(gr)

gi |> pin_by("second") |> filter_by_overlaps(gr)

gi |> pin_by("second") |> filter_by_non_overlaps(gr)
```

---

ginteractions-find-overlaps

*Find overlaps between a query GInteractions and a GRanges*

---

## Description

Find overlaps between a query GInteractions and a GRanges

## Usage

```
## S3 method for class 'PinnedGInteractions'
find_overlaps(x, y, maxgap = -1L, minoverlap = 0L, suffix = c(".x", ".y"))

## S3 method for class 'GInteractions'
find_overlaps(x, y, maxgap = -1L, minoverlap = 0L, suffix = c(".x", ".y"))

## S3 method for class 'PinnedGInteractions'
find_overlaps_directed(
  x,
  y,
  maxgap = -1L,
  minoverlap = 0L,
  suffix = c(".x", ".y")
)

## S3 method for class 'GInteractions'
find_overlaps_directed(
  x,
  y,
  maxgap = -1L,
  minoverlap = 0L,
  suffix = c(".x", ".y")
)
```

**Arguments**

x	A (Pinned)GInteractions object
y	A GRanges object
maxgap, minoverlap	See <a href="#">?findOverlaps</a> in the <b>GenomicRanges</b> package for a description of these arguments
suffix	Suffix to add to metadata columns (character vector of length 2, default to c(".x", ".y")).

**Value**

a GInteractions object with rows corresponding to the GInteractions in x that overlap y.

**Rationale**

find\_overlaps() will search for any overlap between GInteractions in x and GRanges in y. It will return a GInteractions object of length equal to the number of times x overlaps y. This GInteractions will have additional metadata columns corresponding to the metadata from y. find\_overlaps\_directed() takes the strandness of each object into account.

**Pinned GInteractions**

When using find\_overlaps() with a PinnedGInteractions object, only the pinned anchors are used to check for overlap with y. This is equivalent to specifying use.region="both" in InteractionSet::findOverlaps().

**Examples**

```
gi <- read.table(text = "
chr1 11 20 - chr1 21 30 +
chr1 11 20 - chr1 51 55 +
chr1 21 30 - chr1 51 55 +
chr1 21 30 - chr2 51 60 +",
col.names = c(
  "seqnames1", "start1", "end1", "strand1",
  "seqnames2", "start2", "end2", "strand2"
))
) |> as_ginteractions() |> mutate(id = 1:4, type = 'gi')

gr <- GenomicRanges::GRanges(
  c("chr1:20-30:+", "chr2:55-65:-")
) |> plyranges::mutate(id = 1:2, type = 'gr')

gi

gr

#####
# 1. Find overlaps between GInteractions and a subject GRanges
#####
```



```

find_overlaps(gi, gr)

find_overlaps_directed(gi, gr)

#####
# 2. Find overlaps between PinnedGInteractions and a subject GRanges
#####

gi |> pin_by("first") |> find_overlaps(gr)

gi |> pin_by("second") |> find_overlaps(gr)

gi |> pin_by("first") |> find_overlaps_directed(gr)

gi |> pin_by("second") |> find_overlaps_directed(gr)

```

---

```
ginteractions-join-overlap-left
```

*Join overlaps between a query GInteractions and a GRanges*

---

## Description

Join overlaps between a query GInteractions and a GRanges

## Usage

```

## S3 method for class 'PinnedGInteractions'
join_overlap_left(x, y, maxgap = -1L, minoverlap = 0L, suffix = c(".x", ".y"))

## S3 method for class 'GInteractions'
join_overlap_left(x, y, maxgap = -1L, minoverlap = 0L, suffix = c(".x", ".y"))

## S3 method for class 'PinnedGInteractions'
join_overlap_left_directed(
  x,
  y,
  maxgap = -1L,
  minoverlap = 0L,
  suffix = c(".x", ".y")
)

## S3 method for class 'GInteractions'
join_overlap_left_directed(
  x,
  y,
  maxgap = -1L,
  minoverlap = 0L,

```

```

  suffix = c(".x", ".y")
)

```

### Arguments

x	A (Pinned)GInteractions object
y	A GRanges object
maxgap, minoverlap	See <a href="#">?countOverlaps</a> in the <b>GenomicRanges</b> package for a description of these arguments
suffix	Suffix to add to metadata columns (character vector of length 2, default to c(".x", ".y")).

### Value

An integer vector of same length as x.

### Examples

```

gi <- read.table(text = "
  chr1 11 20 - chr1 21 30 +
  chr1 11 20 - chr1 51 55 +
  chr1 21 30 - chr1 51 55 +
  chr1 21 30 - chr2 51 60 +",
  col.names = c(
    "seqnames1", "start1", "end1", "strand1",
    "seqnames2", "start2", "end2", "strand2")
) |> as_ginteractions() |> mutate(id = 1:4, type = 'gi')

gr <- GenomicRanges::GRanges(
  c("chr1:20-30:+", "chr2:55-65:-")
) |> plyranges::mutate(id = 1:2, type = 'gr')

gi

gr

#####
# 1. Join overlaps between GInteractions and a subject GRanges
#####

join_overlap_left(gi, gr)

join_overlap_left_directed(gi, gr)

#####
# 2. Join overlaps between PinnedGInteractions and a subject GRanges
#####

gi |> pin_by("first") |> join_overlap_left(gr)

```

```
gi |> pin_by("first") |> join_overlap_left_directed(gr)
gi |> pin_by("second") |> join_overlap_left(gr)
gi |> pin_by("second") |> join_overlap_left_directed(gr)
```

---

GM12878\_HiCCUPS

*Loops identified in GM12878 with HiCCUPS*

---

### Description

File obtained from GEO entry GSE63525 (GSE63525\_GM12878\_primary+replicate\_HiCCUPS\_loolist.txt.gz).  
Rao SS, Huntley MH, Durand NC, Stamenova EK et al. A 3D map of the human genome at kilobase resolution reveals principles of chromatin looping. Cell 2014 Dec 18;159(7):1665-80. PMID: 25497547

### Usage

```
GM12878_HiCCUPS
```

### Format

An object of class GInteractions of length 9448.

### Value

A GInteractions object

### Source

<https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE63525>

---

group-group\_data

*GInteractions grouping metadata*

---

### Description

GInteractions grouping metadata

**Usage**

```
## S3 method for class 'GroupedGInteractions'
group_data(.data)

## S3 method for class 'GroupedGInteractions'
group_keys(.tbl, ...)

## S3 method for class 'GroupedGInteractions'
group_indices(.data, ...)

## S3 method for class 'GInteractions'
group_vars(x)

## S3 method for class 'GroupedGInteractions'
group_vars(x)

## S3 method for class 'GroupedGInteractions'
groups(x)

## S3 method for class 'GroupedGInteractions'
group_size(x)

## S3 method for class 'GroupedGInteractions'
n_groups(x)
```

**Arguments**

```
.data, .tbl, x    a GInteractions object
...              Ignored.
```

**Value**

a GInteractions object.

**Examples**

```
gi <- read.table(text = "
chr1 11 20 chr1 21 30
chr1 11 20 chr1 51 55
chr1 11 30 chr1 51 55
chr1 11 30 chr2 51 60",
  col.names = c(
    "seqnames1", "start1", "end1",
    "seqnames2", "start2", "end2")
) |>
  as_ginteractions() |>
  mutate(type = c('cis', 'cis', 'cis', 'trans'), score = runif(4))

ggi <- gi |> group_by(end1)
```

```
ggi
group_data(ggi)
group_keys(ggi)
group_rows(ggi)
group_indices(ggi)
group_vars(ggi)
groups(ggi)
group_size(ggi)
n_groups(ggi)
```

---

mutate

*reexports*

---

## Description

reexports

## Usage

```
mutate(.data, ...)

slice(.data, ..., .by = NULL, .preserve = FALSE)

select(.data, ...)

filter(.data, ..., .by = NULL, .preserve = FALSE)

arrange(.data, ..., .by_group = FALSE)

rename(.data, ...)

group_by(.data, ..., .add = FALSE, .drop = group_by_drop_default(.data))

group_data(.data)

group_keys(.tbl, ...)

group_indices(.data, ...)

group_vars(x)

groups(x)

group_size(x)

n_groups(x)

group_rows(.data)
```

```
ungroup(x, ...)
summarize(.data, ..., .by = NULL, .groups = NULL)
summarise(.data, ..., .by = NULL, .groups = NULL)
tally(x, wt = NULL, sort = FALSE, name = NULL)
count(x, ..., wt = NULL, sort = FALSE, name = NULL)
anchor(x)
unanchor(x)
anchor_start(x)
anchor_end(x)
anchor_center(x)
anchor_3p(x)
anchor_5p(x)
stretch(x, extend)
find_overlaps(x, y, maxgap, minoverlap, suffix = c(".x", ".y"))
find_overlaps_directed(x, y, maxgap, minoverlap, suffix = c(".x", ".y"))
count_overlaps(x, y, maxgap, minoverlap)
count_overlaps_directed(x, y, maxgap, minoverlap)
filter_by_overlaps(x, y, maxgap = -1L, minoverlap = 0L)
filter_by_non_overlaps(x, y, maxgap, minoverlap)
join_overlap_left(x, y, maxgap, minoverlap, suffix = c(".x", ".y"))
join_overlap_left_directed(x, y, maxgap, minoverlap, suffix = c(".x", ".y"))
```

**Value**

Depending on the re-exported function

**Examples**

```
1 + 1
```

---

pair_granges	<i>Pairwise combination of a GRanges object</i>
--------------	---

---

**Description**

Create a GInteractions object from a GRanges object, containing all possible entry pairs

**Usage**

```
pair_granges(x)
```

**Arguments**

x                    A GRanges object

**Value**

A GInteractions object

**Examples**

```
gr <- read.table(text = "  
chr1 100 200  
chr1 5000 5100  
chr1 1000 5000  
chr2 3000 3800",  
col.names = c(  
  "seqnames", "start", "end"  
) |> plyranges::as_granges()  
  
pair_granges(gr)
```

---

pin	<i>Pin GInteractions by anchors set (anchors1 or anchors2).</i>
-----	---

---

**Description**

Pin GInteractions by anchors set (anchors1 or anchors2).

**Usage**

```
pin(x, anchors)

pin_by(x, anchors)

pinned_anchors(x)

unpin(x)

## S4 method for signature 'GroupedGInteractions,character'
pin(x, anchors)

## S4 method for signature 'GroupedGInteractions,numeric'
pin(x, anchors)

## S4 method for signature 'GInteractions,character'
pin(x, anchors)

## S4 method for signature 'GInteractions,numeric'
pin(x, anchors)

## S4 method for signature 'PinnedGInteractions,missing'
pin(x, anchors)

## S4 method for signature 'PinnedGInteractions,character'
pin(x, anchors)

## S4 method for signature 'PinnedGInteractions,numeric'
pin(x, anchors)

## S4 method for signature 'AnchoredPinnedGInteractions,character'
pin(x, anchors)

## S4 method for signature 'AnchoredPinnedGInteractions,numeric'
pin(x, anchors)

pin_first(x)

pin_second(x)

pin_anchors1(x)

pin_anchors2(x)

## S4 method for signature 'AnchoredPinnedGInteractions'
unpin(x)

## S4 method for signature 'PinnedGInteractions'
```



```

unpin(x)

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

## S4 method for signature 'PinnedGInteractions'
pinned_anchors(x)

## S4 method for signature 'AnchoredPinnedGInteractions'
pinned_anchors(x)

```

### Arguments

```

x                a GInteractions object
anchors          Anchors to pin on ("first" or "second")

```

### Value

- `pin_*` functions return a `PinnedGInteractions` object.
- `pin` returns a numerical value indicating which set of anchors is pinned.
- `unpin` removes the pinning of a `PinnedGInteractions` object.
- `pinned_anchors` returns an `(Anchored)GenomicRanges` object corresponding to the pinned anchors of a `PinnedGInteractions` object.

### Examples

```

gi <- read.table(text = "
chr1 11 20 chr1 21 30
chr1 11 20 chr1 51 55
chr1 11 30 chr1 51 55
chr1 11 30 chr2 51 60",
col.names = c(
  "seqnames1", "start1", "end1",
  "seqnames2", "start2", "end2")
) |>
  as_ginteractions() |>
  mutate(type = c('cis', 'cis', 'cis', 'trans'), score = runif(4))

#####
# 1. Pin by first anchors
#####

gi |> pin_by("first")

gi |> pin_first()

gi |> pin_anchors1()

#####
# 2. Pin by second anchors

```

```
#####
gi |> pin_by("second")

gi |> pin_second()

gi |> pin_anchors2()

#####
# 3. Unpin
#####

gi |> pin("second") |> unpin()
```

---

plyranges-flank	<i>Generate flanking regions from pinned anchors of a GInteractions object with plyranges</i>
-----------------	---

---

## Description

Generate flanking regions from pinned anchors of a GInteractions object with plyranges

## Usage

```
flank_downstream(x, width)

## S3 method for class 'Ranges'
flank_downstream(x, width)

## S3 method for class 'PinnedGInteractions'
flank_downstream(x, width)

flank_upstream(x, width)

## S3 method for class 'Ranges'
flank_upstream(x, width)

## S3 method for class 'PinnedGInteractions'
flank_upstream(x, width)

flank_right(x, width)

## S3 method for class 'Ranges'
flank_right(x, width)

## S3 method for class 'PinnedGInteractions'
flank_right(x, width)
```

```

flank_left(x, width)

## S3 method for class 'Ranges'
flank_left(x, width)

## S3 method for class 'PinnedGInteractions'
flank_left(x, width)

```

### Arguments

x	a PinnedGInteractions object
width	The width of the flanking region relative to the ranges in x. Either an integer vector of length 1 or an integer vector the same length as x. The width can be negative in which case the flanking region is reversed.

### Value

A PinnedGInteractions object

### Examples

```

gi <- read.table(text = "
chr1 11 20 chr1 21 30 + +
chr1 11 20 chr1 51 55 + +
chr1 11 30 chr1 51 55 - -
chr1 11 30 chr2 51 60 - -",
col.names = c(
  "seqnames1", "start1", "end1",
  "seqnames2", "start2", "end2", "strand1", "strand2")
) |>
  as_ginteractions() |>
  mutate(score = runif(4), type = c('cis', 'cis', 'cis', 'trans'))

#####
# 1. Simple flanking
#####

gi

gi |> pin_by("first") |> flank_left(-2)

gi |> pin_by("second") |> flank_upstream(4)

#####
# 2. Chained flanking of each set of anchors
#####

gi |>
  pin_by("first") |> flank_left(2) |>
  pin_by("second") |> flank_right(2)

```

---

plyranges-shift

*Shift pinned anchors of a GInteractions object with plyranges*

---

## Description

Shift pinned anchors of a GInteractions object with plyranges

## Usage

```
shift_downstream(x, shift)

## S3 method for class 'Ranges'
shift_downstream(x, shift)

## S3 method for class 'PinnedGInteractions'
shift_downstream(x, shift)

shift_upstream(x, shift)

## S3 method for class 'Ranges'
shift_upstream(x, shift)

## S3 method for class 'PinnedGInteractions'
shift_upstream(x, shift)

shift_right(x, shift)

## S3 method for class 'Ranges'
shift_right(x, shift)

## S3 method for class 'PinnedGInteractions'
shift_right(x, shift)

shift_left(x, shift)

## S3 method for class 'Ranges'
shift_left(x, shift)

## S3 method for class 'PinnedGInteractions'
shift_left(x, shift)
```

## Arguments

x	a PinnedGInteractions object
shift	The amount to move the genomic interval in the Ranges object by. Either a non-negative integer vector of length 1 or an integer vector the same length as x.

**Value**

A PinnedGInteractions object

**Examples**

```

gi <- read.table(text = "
chr1 11 20 chr1 21 30 + +
chr1 11 20 chr1 51 55 + +
chr1 11 30 chr1 51 55 - -
chr1 11 30 chr2 51 60 - -",
col.names = c(
  "seqnames1", "start1", "end1",
  "seqnames2", "start2", "end2", "strand1", "strand2")
) |>
as_ginteractions() |>
mutate(score = runif(4), type = c('cis', 'cis', 'cis', 'trans'))

#####
# 1. Simple shifting
#####

gi

gi |> pin_by("first") |> shift_left(15)

gi |> pin_by("second") |> shift_downstream(10)

#####
# 2. Chained shifting of each set of anchors
#####

gi |>
  pin_by("first") |> shift_downstream(20) |>
  pin_by("second") |> shift_upstream(20)

```

---

plyranges-stretch

*Stretch pinned anchors of a GInteractions object with plyranges*

---

**Description**

Stretch pinned anchors of a GInteractions object with plyranges

**Usage**

```
## S3 method for class 'AnchoredPinnedGInteractions'
stretch(x, extend)
```

```
## S3 method for class 'PinnedGInteractions'
stretch(x, extend)
```

**Arguments**

`x` a PinnedGInteractions object  
`extend` The amount to alter the width of a Ranges object by. Either an integer vector of length 1 or an integer vector the same length as `x`.

**Value**

A PinnedGInteractions object

**Examples**

```
gi <- read.table(text = "
chr1 11 20 chr1 21 30 + +
chr1 11 20 chr1 51 55 + +
chr1 11 30 chr1 51 55 - -
chr1 11 30 chr2 51 60 - -",
col.names = c(
  "seqnames1", "start1", "end1",
  "seqnames2", "start2", "end2", "strand1", "strand2")
) |>
  as_ginteractions() |>
  mutate(score = runif(4), type = c('cis', 'cis', 'cis', 'trans'))

#####
# 1. Simple stretching
#####

gi

gi |> pin_by("first") |> anchor_start() |> stretch(15)

gi |> pin_by("second") |> anchor_center() |> stretch(10)

gi |> pin_by("second") |> anchor_3p() |> stretch(20)

#####
# 2. Chained stretching of each set of anchors
#####

gi |>
  pin_by("first") |> anchor_start() |> stretch(20) |>
  pin_by("second") |> stretch(20)
```

---

 replace\_anchors

*Replace anchors of a GInteractions*


---

**Description**

Replace anchors of a GInteractions

**Usage**

```

replace_anchors(x, id, value)

## S4 method for signature 'GInteractions,character,GenomicRanges'
replace_anchors(x, id, value)

## S4 method for signature 'GInteractions,numeric,GenomicRanges'
replace_anchors(x, id, value)

## S4 method for signature 'PinnedGInteractions,missing,GenomicRanges'
replace_anchors(x, id, value)

## S4 method for signature 'AnchoredPinnedGInteractions,missing,GRanges'
replace_anchors(x, id, value)

## S4 method for signature 'AnchoredPinnedGInteractions,numeric,GRanges'
replace_anchors(x, id, value)

```

**Arguments**

x	a (Pinned)GInteractions object
id	Which anchors to replace ("first" or "second"). Ignored if the GInteractions is already pinned to a specific set of anchors.
value	A GRanges object vector the same length as x.

**Value**

a (Pinned)GInteractions object.

**Examples**

```

gi <- read.table(text = "
chr1 11 20 chr1 21 30
chr1 11 20 chr1 51 55
chr1 11 30 chr1 51 55
chr1 11 30 chr2 51 60",
col.names = c(
  "seqnames1", "start1", "end1",
  "seqnames2", "start2", "end2")
) |>
as_ginteractions() |>
mutate(type = c('cis', 'cis', 'cis', 'trans'), score = runif(4))

#####
# 1. Replace anchors of a GInteractions object
#####

gi |> replace_anchors(2, value = anchors1(gi))

gi |> replace_anchors(1, value = anchors2(gi))

```

```

gi |> replace_anchors(1, value = GenomicRanges::GRanges(c(
  "chr1:1-2", "chr1:2-3", "chr1:3-4", "chr1:4-5"
)))

#####
# 2. Replace anchors of a pinned GInteractions object
#####

gi |> pin_by(1) |> replace_anchors(value = anchors1(gi))

gi |> replace_anchors(1, value = anchors2(gi))

gi |>
  pin_by(1) |>
  replace_anchors(value = GenomicRanges::GRanges(c(
    "chr1:1-2", "chr1:2-3", "chr1:3-4", "chr1:4-5"
  ))) |>
  pin_by(2) |>
  replace_anchors(value = GenomicRanges::GRanges(c(
    "chr2:1-2", "chr2:2-3", "chr2:3-4", "chr2:4-5"
  )))

```

---

set\_seqnames1

*Internal GInteractions setters*


---

## Description

Internal GInteractions setters

## Usage

```
set_seqnames1(x, value)
```

```
set_seqnames2(x, value)
```

```
set_start1(x, value)
```

```
set_start2(x, value)
```

```
set_end1(x, value)
```

```
set_end2(x, value)
```

```
set_width1(x, value)
```

```
set_width2(x, value)
```



```
set_strand1(x, value)

set_strand2(x, value)

## S4 replacement method for signature 'GInteractions'
first(x) <- value

## S4 replacement method for signature 'GInteractions'
second(x) <- value

## S4 method for signature 'GInteractions,factor'
set_seqnames1(x, value)

## S4 method for signature 'GInteractions,factor'
set_seqnames2(x, value)

## S4 method for signature 'GInteractions,numeric'
set_start1(x, value)

## S4 method for signature 'GInteractions,numeric'
set_start2(x, value)

## S4 method for signature 'GInteractions,numeric'
set_end1(x, value)

## S4 method for signature 'GInteractions,numeric'
set_end2(x, value)

## S4 method for signature 'GInteractions,numeric'
set_width1(x, value)

## S4 method for signature 'GInteractions,numeric'
set_width2(x, value)

## S4 method for signature 'AnchoredPinnedGInteractions,numeric'
set_width1(x, value)

## S4 method for signature 'AnchoredPinnedGInteractions,numeric'
set_width2(x, value)

## S4 method for signature 'GInteractions,character'
set_strand1(x, value)

## S4 method for signature 'GInteractions,character'
set_strand2(x, value)
```

### Arguments

x                    a GInteractions object

value            a value passed to the corresponding field

**Value**

A modified GInteractions

# Index

## \* internal

- ce10\_ARCC, 11
- ce10\_REs, 12
- delegating-ginteractions-methods, 12
- GM12878\_HiCCUPS, 35
- mutate, 37
- set\_seqnames1, 48
- \$, GInteractions-method (anchors1), 5
  
- add\_pairdist, 4
- anchor (mutate), 37
- anchor.AnchoredPinnedGInteractions (ginteractions-anchor), 25
- anchor\_3p (mutate), 37
- anchor\_3p.AnchoredPinnedGInteractions (ginteractions-anchor), 25
- anchor\_3p.PinnedGInteractions (ginteractions-anchor), 25
- anchor\_5p (mutate), 37
- anchor\_5p.AnchoredPinnedGInteractions (ginteractions-anchor), 25
- anchor\_5p.PinnedGInteractions (ginteractions-anchor), 25
- anchor\_center (mutate), 37
- anchor\_center.AnchoredPinnedGInteractions (ginteractions-anchor), 25
- anchor\_center.PinnedGInteractions (ginteractions-anchor), 25
- anchor\_end (mutate), 37
- anchor\_end.AnchoredPinnedGInteractions (ginteractions-anchor), 25
- anchor\_end.PinnedGInteractions (ginteractions-anchor), 25
- anchor\_start (mutate), 37
- anchor\_start.AnchoredPinnedGInteractions (ginteractions-anchor), 25
- anchor\_start.PinnedGInteractions (ginteractions-anchor), 25
  
- anchors, DelegatingGInteractions-method (delegating-ginteractions-methods), 12
- anchors1, 5
- anchors1, DelegatingGInteractions-method (delegating-ginteractions-methods), 12
- anchors1, GInteractions-method (anchors1), 5
- anchors2 (anchors1), 5
- anchors2, DelegatingGInteractions-method (delegating-ginteractions-methods), 12
- anchors2, GInteractions-method (anchors1), 5
- annotate, 7
- annotate, GInteractions, GRanges, character-method (annotate), 7
- annotate\_directed (annotate), 7
- annotate\_directed, GInteractions, GRanges, character-method (annotate), 7
- arrange, 3
- arrange (mutate), 37
- arrange.GInteractions (dplyr-arrange), 14
- as\_ginteractions, 9
  
- ce10\_ARCC, 11
- ce10\_REs, 12
- count, 3
- count (mutate), 37
- count.GInteractions (dplyr-count), 15
- count.GroupedGInteractions (dplyr-count), 15
- count\_overlaps (mutate), 37
- count\_overlaps.GInteractions (ginteractions-count-overlaps), 27
- count\_overlaps.PinnedGInteractions (ginteractions-count-overlaps),

- 27
- count\_overlaps\_directed (mutate), 37
- count\_overlaps\_directed.GInteractions  
(ginteractions-count-overlaps),  
27
- count\_overlaps\_directed.PinnedGInteractions  
(ginteractions-count-overlaps),  
27
- countOverlaps, 27, 30, 34
- data.frame(), 9
- DataFrame(), 9, 16, 24
- delegating-ginteractions-methods, 12
- dplyr-arrange, 14
- dplyr-count, 15
- dplyr-filter, 16
- dplyr-group\_by, 17
- dplyr-mutate, 19
- dplyr-rename, 21
- dplyr-select, 22
- dplyr-slice, 23
- dplyr-summarise (dplyr-summarize), 24
- dplyr-summarize, 24
- end1 (anchors1), 5
- end1,DelegatingGInteractions-method  
(delegating-ginteractions-methods),  
12
- end1,GInteractions-method (anchors1), 5
- end2 (anchors1), 5
- end2,DelegatingGInteractions-method  
(delegating-ginteractions-methods),  
12
- end2,GInteractions-method (anchors1), 5
- filter, 3
- filter (mutate), 37
- filter.GInteractions (dplyr-filter), 16
- filter\_by\_non\_overlaps (mutate), 37
- filter\_by\_non\_overlaps.GInteractions  
(ginteractions-filter-overlaps),  
29
- filter\_by\_non\_overlaps.PinnedGInteractions  
(ginteractions-filter-overlaps),  
29
- filter\_by\_overlaps (mutate), 37
- filter\_by\_overlaps.GInteractions  
(ginteractions-filter-overlaps),  
29
- filter\_by\_overlaps.PinnedGInteractions  
(ginteractions-filter-overlaps),  
29
- find\_overlaps (mutate), 37
- find\_overlaps.GInteractions  
(ginteractions-find-overlaps),  
31
- find\_overlaps.PinnedGInteractions  
(ginteractions-find-overlaps),  
31
- find\_overlaps\_directed (mutate), 37
- find\_overlaps\_directed.GInteractions  
(ginteractions-find-overlaps),  
31
- find\_overlaps\_directed.PinnedGInteractions  
(ginteractions-find-overlaps),  
31
- findOverlaps, 32
- first<- ,GInteractions-method  
(set\_seqnames1), 48
- flank\_downstream (plyranges-flank), 42
- flank\_left (plyranges-flank), 42
- flank\_right (plyranges-flank), 42
- flank\_upstream (plyranges-flank), 42
- ginteractions-anchor, 25
- ginteractions-annotate (annotate), 7
- ginteractions-count-overlaps, 27
- ginteractions-export, 28
- ginteractions-filter-overlaps, 29
- ginteractions-find-overlaps, 31
- ginteractions-getters (anchors1), 5
- ginteractions-join-overlap-left, 33
- ginteractions-pin (pin), 39
- ginteractions-setters (set\_seqnames1),  
48
- GM12878\_HiCCUPS, 35
- group-group\_data, 35
- group\_by, 3
- group\_by (mutate), 37
- group\_by.DelegatingGInteractions  
(dplyr-group\_by), 17
- group\_by.GInteractions  
(dplyr-group\_by), 17
- group\_data (mutate), 37
- group\_data.GroupedGInteractions  
(group-group\_data), 35
- group\_indices (mutate), 37

- group\_indices.GroupedGInteractions (group-group\_data), 35
- group\_keys(mutate), 37
- group\_keys.GroupedGInteractions (group-group\_data), 35
- group\_rows(mutate), 37
- group\_size(mutate), 37
- group\_size.GroupedGInteractions (group-group\_data), 35
- group\_vars(mutate), 37
- group\_vars.GInteractions (group-group\_data), 35
- group\_vars.GroupedGInteractions (group-group\_data), 35
- groups(mutate), 37
- groups.GroupedGInteractions (group-group\_data), 35
- InteractionSet::GInteractions(), 9
- join\_overlap\_left(mutate), 37
- join\_overlap\_left.GInteractions (ginteractions-join-overlap-left), 33
- join\_overlap\_left.PinnedGInteractions (ginteractions-join-overlap-left), 33
- join\_overlap\_left\_directed(mutate), 37
- join\_overlap\_left\_directed.GInteractions (ginteractions-join-overlap-left), 33
- join\_overlap\_left\_directed.PinnedGInteractions (ginteractions-join-overlap-left), 33
- mcols, DelegatingGInteractions-method (delegating-ginteractions-methods), 12
- mutate, 3, 37
- mutate.GInteractions (dplyr-mutate), 19
- n\_groups(mutate), 37
- n\_groups.GroupedGInteractions (group-group\_data), 35
- pair\_granges, 39
- pin, 39
- pin, AnchoredPinnedGInteractions, character-method (pin), 39
- pin, GInteractions, character-method (pin), 39
- pin, GInteractions, numeric-method (pin), 39
- pin, GroupedGInteractions, character-method (pin), 39
- pin, GroupedGInteractions, numeric-method (pin), 39
- pin, PinnedGInteractions, character-method (pin), 39
- pin, PinnedGInteractions, missing-method (pin), 39
- pin, PinnedGInteractions, numeric-method (pin), 39
- pin\_anchors1 (pin), 39
- pin\_anchors2 (pin), 39
- pin\_by (pin), 39
- pin\_first (pin), 39
- pin\_second (pin), 39
- pinned\_anchors (pin), 39
- pinned\_anchors, AnchoredPinnedGInteractions-method (pin), 39
- pinned\_anchors, PinnedGInteractions-method (pin), 39
- plyinteractions (plyinteractions-package), 3
- plyinteractions-package, 3
- plyranges-flank, 42
- plyranges-shift, 44
- plyranges-stretch, 45
- ranges1 (anchors1), 5
- ranges1, DelegatingGInteractions-method (delegating-ginteractions-methods), 12
- ranges1, GInteractions-method (anchors1), 5
- ranges2 (anchors1), 5
- ranges2, DelegatingGInteractions-method (delegating-ginteractions-methods), 12
- ranges2, GInteractions-method (anchors1), 5
- reexports (mutate), 37
- regions, DelegatingGInteractions-method (delegating-ginteractions-methods), 12

- rename (mutate), 37
- rename.GInteractions (dplyr-rename), 21
- replace\_anchors, 46
- replace\_anchors, AnchoredPinnedGInteractions, missing, GenomicRanges-method (replace\_anchors), 46
- replace\_anchors, AnchoredPinnedGInteractions, numeric, GenomicRanges-method (replace\_anchors), 46
- replace\_anchors, GInteractions, character, GenomicRanges-method (replace\_anchors), 46
- replace\_anchors, GInteractions, numeric, GenomicRanges-method (replace\_anchors), 46
- replace\_anchors, PinnedGInteractions, missing, GenomicRanges-method (replace\_anchors), 46
  
- second<-, GInteractions-method (set\_seqnames1), 48
- select, 3
- select (mutate), 37
- select.GInteractions (dplyr-select), 22
- seqinfo, DelegatingGInteractions-method (delegating-ginteractions-methods), 12
- seqnames1 (anchors1), 5
- seqnames1, DelegatingGInteractions-method (delegating-ginteractions-methods), 12
- seqnames1, GInteractions-method (anchors1), 5
- seqnames2 (anchors1), 5
- seqnames2, DelegatingGInteractions-method (delegating-ginteractions-methods), 12
- seqnames2, GInteractions-method (anchors1), 5
- set\_end1 (set\_seqnames1), 48
- set\_end1, GInteractions, numeric-method (set\_seqnames1), 48
- set\_end2 (set\_seqnames1), 48
- set\_end2, GInteractions, numeric-method (set\_seqnames1), 48
- set\_seqnames1, 48
- set\_seqnames1, GInteractions, factor-method (set\_seqnames1), 48
- set\_seqnames2 (set\_seqnames1), 48
- set\_seqnames2, GInteractions, factor-method (set\_seqnames1), 48
- set\_start1 (set\_seqnames1), 48
- set\_start1, GInteractions, numeric-method (set\_seqnames1), 48
- set\_start2 (set\_seqnames1), 48
- set\_start2, GInteractions, numeric-method (set\_seqnames1), 48
- set\_strand1, GInteractions, character-method (set\_seqnames1), 48
- set\_strand2 (set\_seqnames1), 48
- set\_strand2, GInteractions, character-method (set\_seqnames1), 48
- set\_width1, AnchoredPinnedGInteractions, numeric-method (set\_seqnames1), 48
- set\_width1, GInteractions, numeric-method (set\_seqnames1), 48
- set\_width2 (set\_seqnames1), 48
- set\_width2, AnchoredPinnedGInteractions, numeric-method (set\_seqnames1), 48
- set\_width2, GInteractions, numeric-method (set\_seqnames1), 48
- shift\_downstream (plyranges-shift), 44
- shift\_left (plyranges-shift), 44
- shift\_right (plyranges-shift), 44
- shift\_upstream (plyranges-shift), 44
- show, DelegatingGInteractions-method (delegating-ginteractions-methods), 12
- slice, 3
- slice (mutate), 37
- slice.GInteractions (dplyr-slice), 23
- start1 (anchors1), 5
- start1, DelegatingGInteractions-method (delegating-ginteractions-methods), 12
- start1, GInteractions-method (anchors1), 5
- start2 (anchors1), 5
- start2, DelegatingGInteractions-method (delegating-ginteractions-methods), 12
- start2, GInteractions-method (anchors1), 5
- strand1 (anchors1), 5
- strand1, DelegatingGInteractions-method (delegating-ginteractions-methods), 12
- strand1, GInteractions-method (anchors1), 5
- strand2 (anchors1), 5

strand2, DelegatingGInteractions-method  
(delegating-ginteractions-methods),  
12

strand2, GInteractions-method  
(anchors1), 5

stretch (mutate), 37

stretch.AnchoredPinnedGInteractions  
(plyranges-stretch), 45

stretch.PinnedGInteractions  
(plyranges-stretch), 45

summarise (mutate), 37

summarise.GroupedGInteractions  
(dplyr-summarize), 24

summarize, 3

summarize (mutate), 37

summarize.GroupedGInteractions  
(dplyr-summarize), 24

tally, 3

tally (mutate), 37

tally.GroupedGInteractions  
(dplyr-count), 15

unanchor (mutate), 37

unanchor.AnchoredPinnedGInteractions  
(ginteractions-anchor), 25

ungroup (mutate), 37

ungroup.GroupedGInteractions  
(dplyr-group\_by), 17

unpin (pin), 39

unpin, AnchoredPinnedGInteractions-method  
(pin), 39

unpin, GInteractions-method (pin), 39

unpin, PinnedGInteractions-method (pin),  
39

width1 (anchors1), 5

width1, DelegatingGInteractions-method  
(delegating-ginteractions-methods),  
12

width1, GInteractions-method (anchors1),  
5

width2 (anchors1), 5

width2, DelegatingGInteractions-method  
(delegating-ginteractions-methods),  
12

width2, GInteractions-method (anchors1),  
5

write\_bedpe (ginteractions-export), 28

write\_pairs (ginteractions-export), 28