

# genomeIntervals

November 11, 2009

## R topics documented:

<code>c</code> . . . . .	1
<code>core_annotated</code> . . . . .	2
<code>distance_to_nearest</code> . . . . .	3
<code>gen_ints</code> . . . . .	4
<code>Genome_intervals-class</code> . . . . .	4
<code>genomeIntervals-package</code> . . . . .	6
<code>Genome_intervals_stranded-class</code> . . . . .	7
<code>getGffAttribute</code> . . . . .	9
<code>interval_overlap</code> . . . . .	10
<code>interval_union</code> . . . . .	11
<code>parseGffAttributes</code> . . . . .	13
<code>readGff3</code> . . . . .	14

<b>Index</b>	<b>16</b>
--------------	-----------

---

`c` *Combine genome intervals objects*

---

### Description

S3 methods for combining several genome intervals into a single one.

### Usage

```
## S3 method for class 'Genome_intervals':  
c(...)  
## S3 method for class 'Genome_intervals_stranded':  
c(...)
```

### Arguments

... [Genome\\_intervals](#) or [Genome\\_intervals\\_stranded](#) objects.

### Details

If the arguments have mixed classes ( both [Genome\\_intervals](#) or [Genome\\_intervals\\_stranded](#)), then they are coerced to [Genome\\_intervals](#) before combination. Otherwise, the common class is used.

**Value**

A single `Genome_intervals` or `Genome_intervals_stranded` object. Input objects are combined in their order of appearance in the the argument list.

If any input argument is not a `Genome_intervals`, `list(...)` is returned instead.

**Note**

These methods will be converted to S4 once the necessary dispatch on `...` is supported.

**Examples**

```
# load toy examples
data("gen_ints")

# combine i and j returns a Genome_intervals_stranded object
c(i, j)

# combine a not-stranded and a stranded returns a not-stranded object
c(as(i, "Genome_intervals"), j)
```

---

core\_annotated

*Genome intervals with minimal annotation*

---

**Description**

returns a copy of the input (stranded) genome intervals object with annotations restricted to the minimally required ones.

**Usage**

```
core_annotated(x)
```

**Arguments**

`x` A `Genome_intervals` or `Genome_intervals_stranded` object.

**Value**

A copy of `x` with the annotation slot restricted to `seq_name`, `inter_base` and `strand` (the latter only if `x` is a `Genome_intervals_stranded` object).

**Examples**

```
# load toy examples
data("gen_ints")

# add some non-core annotations to i
annotation(i)$comment = "some non-core annotation"

# i with all annotations
i

# core annotations only
```

```

core_annotated(i)

## Not run:
# with different annotation columns, i and j cannot be combined
c( i, j )
## End(Not run)

# core annotated versions can
c( core_annotated(i), core_annotated(j) )

```

---

```

distance_to_nearest

```

*Distance in bases to the closest interval(s)*

---

## Description

Given two objects, `from` and `to`, compute the distance in bases of each `from` interval to the nearest `to` interval(s). The distance between a base and the next inter-bases on either side values 0.5. Thus, base - base and inter-base - inter-base intervals distances are integer, whereas base - inter-base intervals distances are half-integers.

## Usage

```

## S4 method for signature 'Genome_intervals,
##   Genome_intervals':
distance_to_nearest(from, to)
## S4 method for signature 'Genome_intervals_stranded,
##   Genome_intervals_stranded':
distance_to_nearest(from, to)

```

## Arguments

`from`            A [Genome\\_intervals](#) or [Genome\\_intervals\\_stranded](#) object.  
`to`                A [Genome\\_intervals](#) or [Genome\\_intervals](#) object.

## Details

A wrapper calling [intervals::distance\\_to\\_nearest](#) by `seq_name` and by `strand` (if both `from` and `to` are [Genome\\_intervals\\_stranded](#) objects). Thus, if both are stranded, distances are computed over each strand separately. One object must be coerced to [Genome\\_intervals](#) if this is not wished.

## Value

A numeric vector of distances with one element for each row of `from`.

## See Also

[intervals::distance\\_to\\_nearest](#)

**Examples**

```
## load toy examples
data(gen_ints)

## i in close_intervals notation
close_intervals(i)

## j in close_intervals notation
close_intervals(j)

## distances from i to j
dn = distance_to_nearest(i, j)
dn

## distance == 0 if and only if the interval overlaps another one:
io = interval_overlap(i, j)
if( any( ( sapply(io, length) >0 ) != (!is.na(dn) & dn ==0) ) )
      stop("The property 'distance == 0 if and only if the interval overlaps another one' is violated")
)

## distances without strand-specificity
distance_to_nearest(
  as(i, "Genome_intervals"),
  as(j, "Genome_intervals")
)
```

---

gen\_ints

*Genome Intervals examples*


---

**Description**

Toy examples for testing functions and running examples of the package genomeIntervals.

**Usage**

```
data(gen_ints)
```

**Format**

Two Genome\_intervals\_stranded objects, i and j, without inter-base intervals and a third one, k, with.

---

Genome\_intervals-class

*Class "Genome\_intervals"*


---

**Description**

A set of genomic intervals without specified strand. Genomic intervals are intervals over the integers with two further annotations: seq\_name (a chromosome or more generally a sequence of origin) and inter\_base (logical) that states whether the interval is to be understood as an interval over bases (such as coding-sequence) or inter-bases (such as restriction sites or insertion positions).

**Slots**

**.Data:** See [Intervals\\_full](#)

**annotation:** A "data.frame" with the same number of rows as .Data. It has a column named `seq_name` that is a factor and does not contain missing values. `seq_name` is used to represent the chromosome or more generally the sequence of origin of the intervals. `annotation` has a column named `inter_base` that is logical and does not contain missing values. `inter_base` is FALSE if the interval is to be understood as an interval over bases (such as coding-sequence) and TRUE if it is over inter-bases (such as restriction site or an insertion position). Like base intervals, inter-base interval are encoded over the integers. An inter-base at position `n` indicates the space between base `n` and `n+1`.

**closed:** See [Intervals\\_full](#)

**type:** See [Intervals\\_full](#)

**Extends**

Class "[Intervals\\_full](#)", directly. Class "[Intervals\\_virtual](#)", by class "[Intervals\\_full](#)", distance 2. Class "[matrix](#)", by class "[Intervals\\_full](#)", distance 3. Class "[array](#)", by class "[Intervals\\_full](#)", distance 4. Class "[structure](#)", by class "[Intervals\\_full](#)", distance 5. Class "[vector](#)", by class "[Intervals\\_full](#)", distance 6, with explicit coerce.

**Methods**

```
[ signature(x = "Genome_intervals"):...
[[ signature(x = "Genome_intervals"):...
[[<- signature(x = "Genome_intervals"):...
$ signature(x = "Genome_intervals"):...
$<- signature(x = "Genome_intervals"):...
annotation signature(object = "Genome_intervals"):...
annotation<- signature(object = "Genome_intervals"):...
coerce signature(from = "Genome_intervals", to = "Intervals_full"):...
coerce signature(from = "Genome_intervals", to = "character"):...
distance_to_nearest signature(from = "Genome_intervals", to = "Genome_intervals"):
  ...
inter_base signature(x = "Genome_intervals"):...
inter_base<- signature(x = "Genome_intervals"):...
interval_complement signature(x = "Genome_intervals"):...
interval_intersection signature(x = "Genome_intervals"):...
interval_overlap signature(from = "Genome_intervals", to = "Genome_intervals"):
  ...
interval_union signature(x = "Genome_intervals"):...
seq_name signature(x = "Genome_intervals"):...
seq_name<- signature(x = "Genome_intervals"):...
size signature(x = "Genome_intervals"):...
type<- signature(x = "Genome_intervals"):...
```

**Note**

A `Genome_intervals` is a "`Intervals_full`" of type `Z` (i.e. a set of intervals over the integers). The annotation slot can carry further columns that can serve as annotations.

**See Also**

`Genome_intervals_stranded` for a derived class that allows stranded genomic intervals.

**Examples**

```
# The "Genome_intervals" class

i <- new(
  "Genome_intervals",
  matrix(
    c(1,2,
      3,5,
      4,6,
      8,9
    ),
    byrow = TRUE,
    ncol = 2
  ),
  closed = matrix(
    c(
      TRUE, FALSE,
      TRUE, FALSE,
      TRUE, TRUE,
      TRUE, FALSE
    ),
    byrow = TRUE,
    ncol = 2
  ),
  annotation = data.frame(
    seq_name = factor(c("chr01","chr01", "chr02","chr02"),
      inter_base = c(FALSE, FALSE, TRUE, TRUE)
    )
  )

colnames(i) <- c("start", "end")

# print
print(i)

# size (number of bases per interval)
size(i)
```

---

genomeIntervals-package

*Operations on genomic intervals*

---

**Description**

Tools for operation on genomic intervals.

## Details

Package: genomeIntervals  
Version: 0.9.6  
Date: 2009-01-15  
Type: Package  
Depends: R (>= 2.8.0), intervals (>= 0.10.3), Biobase, methods  
Suggests:  
License: Artistic 2.0  
BiocViews: DataImport, Infrastructure, Genetics  
LazyLoad: yes

### Index:

[Genome\\_intervals](#) Class "Genome\_intervals"

[Genome\\_intervals\\_stranded](#) Class "Genome\_intervals\_stranded"

[distance\\_to\\_nearest](#) Distance in bases to the closest interval(s)

[gen\\_ints](#) Genome Intervals examples

[getGffAttribute](#) Pull one or more key/value pairs from gffAttributes strings

[interval\\_overlap](#) Assess overlap from one set of genomic intervals to another

[interval\\_complement](#) Compute the complement of a set of genomic intervals

[interval\\_intersection](#) Compute the intersection of one or more sets of genomic intervals

[interval\\_union](#) Compute the union of genomic intervals in one or more genomic interval matrices

[parseGffAttributes](#) Parse out the gffAttributes column of a Genome\_intervals object

[readGff3](#) Make a Genome\_intervals\_stranded object from a GFF file

### Author(s)

Julien Gagneur <gagneur@embl.de>, Richard Bourgon.

Maintainer: Julien Gagneur <gagneur@embl.de>

### See Also

[intervals](#)

---

Genome\_intervals\_stranded-class  
*Class "Genome\_intervals\_stranded"*

---

### Description

A set of genomic intervals with a specified strand.

**Slots**

**.Data:** See [Genome\\_intervals](#)

**annotation:** A `data.frame` (see [Genome\\_intervals](#) for basic requirements). The annotation moreover has a `strand` column that is a factor with exactly two levels (typically "+" and "-").

**closed:** See [Genome\\_intervals](#)

**type:** See [Genome\\_intervals](#)

**Extends**

Class "[Genome\\_intervals](#)", directly. Class "[Intervals\\_full](#)", by class "[Genome\\_intervals](#)", distance 2. Class "[Intervals\\_virtual](#)", by class "[Genome\\_intervals](#)", distance 3. Class "[matrix](#)", by class "[Genome\\_intervals](#)", distance 4. Class "[array](#)", by class "[Genome\\_intervals](#)", distance 5. Class "[structure](#)", by class "[Genome\\_intervals](#)", distance 6. Class "[vector](#)", by class "[Genome\\_intervals](#)", distance 7, with explicit coerce.

**Methods**

**coerce** signature(from = "Genome\_intervals\_stranded", to = "character"):

...

**distance\_to\_nearest** signature(from = "Genome\_intervals\_stranded", to = "Genome\_interv

...

**interval\_complement** signature(x = "Genome\_intervals\_stranded"):...

**interval\_intersection** signature(x = "Genome\_intervals\_stranded"):...

**interval\_overlap** signature(to = "Genome\_intervals\_stranded", from = "Genome\_interv

...

**interval\_union** signature(x = "Genome\_intervals\_stranded"):...

**strand** signature(x = "Genome\_intervals\_stranded"):...

**strand<-** signature(x = "Genome\_intervals\_stranded"):...

**See Also**

[Genome\\_intervals](#) the parent class without strand.

**Examples**

```
# The "Genome_intervals_stranded" class
j <- new(
  "Genome_intervals_stranded",
  matrix(
    c(1, 2,
      3, 5,
      4, 6,
      8, 9
    ),
    byrow = TRUE,
    ncol = 2
  ),
  closed = matrix(
    c(
```

```

                                FALSE, FALSE,
                                TRUE, FALSE,
                                TRUE, TRUE,
                                TRUE, FALSE
                                ),
                                byrow = TRUE,
                                ncol = 2
                                ),
annotation = data.frame(
                                seq_name = factor( c("chr01","chr01", "chr02","chr02") ),
                                strand = factor( c("+", "+", "+", "-") ),
                                inter_base = c(FALSE,FALSE,FALSE,TRUE)
                                )
)

## print
print(j)

## size of each interval as count of included bases
size(j)

## close intervals left and right (canonical representation)
close_intervals(j)

```

---

getGffAttribute      *Pull one or more key/value pairs from gffAttributes strings*

---

## Description

GFF files contain a string, with key/value pairs separated by “;”, and the key and value separated by “=”. This function quickly extracts one or more key/value pairs.

## Usage

```
getGffAttribute(gi, attribute)
```

## Arguments

gi                    A [Genome\\_intervals](#) object.  
attribute            A vector of key names.

## Value

A matrix with the same number of rows as gi, and one column per element of attribute.

## See Also

See [parseGffAttributes](#) for more complete parsing. See the function [readGff3](#) for loading a GFF file.

**Examples**

```

# Get file path
libPath <- installed.packages()["genomeIntervals", "LibPath"]
filePath <- file.path(
  libPath,
  "genomeIntervals",
  "example_files"
)

# Load gff
gff <- readGff3( file.path( filePath, "sgd_simple.gff"), isRightOpen=FALSE)

## head of full gff annotations
head(annotation(gff))

# extract ID and Parent attributes
idpa = getGffAttribute( gff, c( "ID", "Parent" ) )

head(idpa)

```

---

interval\_overlap    *Assess overlap from one set of genomic intervals to another*

---

**Description**

Given two objects, a 'from' and a 'to', assess which intervals in 'to' overlap which of 'from'.

**Usage**

```

## S4 method for signature 'Genome_intervals,
##   Genome_intervals':
interval_overlap(
  from, to,
  check_valid = TRUE
)

## S4 method for signature 'Genome_intervals_stranded,
##   Genome_intervals_stranded':
interval_overlap(
  from, to,
  check_valid = TRUE
)

```

**Arguments**

from            A `Genome_intervals` or `Genome_intervals_stranded` object.  
to                A `Genome_intervals` or `Genome_intervals_stranded` object.  
check\_valid    Should `validObject` be called before passing to compiled code?

**Details**

A wrapper calling `intervals:interval_overlap` by `seq_name` and by `strand` (if both `to` and `from` are "Genome\_intervals\_stranded" objects).

**Value**

A list, with one element for each row of `from`. The elements are vectors of indices, indicating which `to` rows overlap each `from`. A list element of length 0 indicates a `from` with no overlapping `to` intervals.

**Examples**

```
data(gen_ints)
# i as entered
i

# i in close_intervals notation
close_intervals(i)

# j in close_intervals notation
close_intervals(j)

# list of intervals of j overlapping intervals of i
interval_overlap(i, j)
```

---

interval\_union      *Genome interval set operations*

---

**Description**

Compute interval set operations on "Genome\_intervals" or "Genome\_intervals\_stranded" objects.

**Usage**

```
## S4 method for signature 'Genome_intervals':
interval_union(x, ...)
## S4 method for signature 'Genome_intervals_stranded':
interval_union(x, ...)

## S4 method for signature 'Genome_intervals':
interval_complement(x)
## S4 method for signature 'Genome_intervals_stranded':
interval_complement(x)

## S4 method for signature 'Genome_intervals':
interval_intersection(x, ...)
## S4 method for signature 'Genome_intervals_stranded':
interval_intersection(x, ...)
```

**Arguments**

`x` A "Genome\_intervals" or "Genome\_intervals\_stranded" object.  
 ... Optionally, additional objects of the same class as `x`.

**Details**

Wrappers calling the corresponding functions of the package `intervals` by same `seq_name`, `inter_base` and if needed `strand`. Note that the union of single input object `x` returns the reduced form of `x`, i.e. the interval representation of the covered set.

**Value**

A single object of appropriate class, representing the union, complement or intersection of intervals computed over entries with same `seq_name`, `inter_base` and also `strand` if all passed objects are of the class "Genome\_intervals\_stranded".

**See Also**

[interval\\_union](#), [interval\\_complement](#), [interval\\_intersection](#) and [reduce](#) from the package `intervals`.

**Examples**

```
## load toy examples
data(gen_ints)
## content of i object
i

## complement
interval_complement(i)

## reduced form (non-overlapping interval representation of the covered set)
interval_union(i)

## union
interval_union(i[1:2,], i[1:4,])

# map to genome intervals and union again
i.nostrand = as(i, "Genome_intervals")
interval_union(i.nostrand)

## intersection with a second object
# print i and j in closed interval notation
close_intervals(i)
close_intervals(j)

# interval_intersection
interval_intersection(i, j)

# interval intersection non-stranded
interval_intersection(i.nostrand, as(j, "Genome_intervals"))
```

---

parseGffAttributes *Parse out the gffAttributes column of a Genome\_intervals object*

---

### Description

GFF files contain a string, with key/value pairs separated by “;”, and the key and value separated by “=”. This function parses such strings into a list of vectors with named elements.

### Usage

```
parseGffAttributes(gi)
```

### Arguments

gi                    A `Genome_intervals` object.

### Value

A list, with one element per row of `gi`. Each element is a character vector with named components. Names correspond to keys, and components correspond to values.

### Note

Key/value pairs which are missing the “=” symbol, or which have nothing between it and the “;” delimiter or end of line, will generate a NA value, with a warning. Any key/value “pairs” with more than one “=” cause an error.

### See Also

In many cases, `getGffAttribute`, in this package, is easier and faster. See the function `readGff3` for loading a GFF file.

### Examples

```
# Get file path
libPath <- installed.packages()["genomeIntervals", "LibPath"]
filePath <- file.path(
  libPath,
  "genomeIntervals",
  "example_files"
)

# Load gff and parse attributes
gff <- readGff3( file.path( filePath, "sgd_simple.gff"), isRightOpen = FALSE )
gfatt <- parseGffAttributes(gff)

head( gfatt )
```

---

`readGff3`*Make a `Genome_intervals_stranded` object from a GFF file*

---

## Description

Make a `Genome_intervals_stranded` object from a gff file in gff3 format.

## Usage

```
readGff3(file, isRightOpen=TRUE)
```

## Arguments

<code>file</code>	The name of the gff file to read.
<code>isRightOpen</code>	Although a proper GFF3 file follows the convention of right-open intervals, improper GFF files following the right-closed convention are frequently found. Set <code>isRightOpen = FALSE</code> in this case.

## Details

The file must follow gff3 format specifications as in <http://www.sequenceontology.org/gff3.shtml>. The file is read as a table. Meta-information (lines starting with ###) are not parsed. A “.” in, for example, the gff file’s *score* or *frame* field will be converted to NA. When the GFF file follows the right-open interval convention (`isRightOpen` is TRUE), then GFF entries for which end base equals first base are recognized as zero-length features and loaded as `inter_base` intervals.

## Value

A `Genome_intervals_stranded` object image of the gff file. The GFF3 fields `seqid`, `source`, `type`, `score`, `strand`, `phase` and `attributes` are stored in the annotation slot and renamed as `seq_name`, `source`, `type`, `score`, `strand`, `phase` and `gffAttributes` respectively.

## Note

Potential FASTA entries at the end of the file are ignored.

## See Also

The functions `getGffAttribute` and `parseGffAttributes` for parsing GFF attributes.

## Examples

```
# Get file path
libPath <- installed.packages()["genomeIntervals", "LibPath"]
filePath <- file.path(
  libPath,
  "genomeIntervals",
  "example_files"
)
```

```
# Load SGD gff
# SGD does not comply to the GFF3 right-open interval convention
gff <- readGff3( file.path( filePath, "sgd_simple.gff"), isRightOpen = FALSE)

head(gff,10)

head(annotation(gff),10)
```

# Index

## \*Topic classes

Genome\_intervals-class, 4  
 Genome\_intervals\_stranded-class,

7

## \*Topic datasets

gen\_ints, 4

## \*Topic package

genomeIntervals-package, 6

[, Genome\_intervals-method  
 (Genome\_intervals-class), 4

[<-, Genome\_intervals, ANY, missing, Genome\_intervals-method  
 (Genome\_intervals-class), 4

[[, Genome\_intervals-method  
 (Genome\_intervals-class), 4

[[<-, Genome\_intervals-method  
 (Genome\_intervals-class), 4

\$, Genome\_intervals-method  
 (Genome\_intervals-class), 4

\$<-, Genome\_intervals-method  
 (Genome\_intervals-class), 4

annotation, Genome\_intervals-method  
 (Genome\_intervals-class), 4

annotation<-, Genome\_intervals, ANY-method  
 (Genome\_intervals-class), 4

array, 5, 8

c, 1

coerce, Genome\_intervals, character-method  
 (Genome\_intervals-class), 4

coerce, Genome\_intervals, Intervals\_full-method  
 (Genome\_intervals-class), 4

coerce, Genome\_intervals\_stranded, character-method  
 (Genome\_intervals\_stranded-class), 7

core\_annotated, 2

core\_annotated, Genome\_intervals-method  
 (core\_annotated), 2

core\_annotated, Genome\_intervals\_stranded-method  
 (core\_annotated), 2

distance\_to\_nearest, 3, 7

distance\_to\_nearest, Genome\_intervals, Genome\_intervals\_stranded-method  
 (distance\_to\_nearest), 3

distance\_to\_nearest, Genome\_intervals\_stranded,  
 (distance\_to\_nearest), 3

gen\_ints, 4, 7

Genome\_intervals, 1-3, 7-9, 13

Genome\_intervals-class, 4

Genome\_intervals\_stranded, 1-3, 5,  
 7, 13, 14

Genome\_intervals\_stranded-class,  
 7

genomeIntervals-method  
 (genomeIntervals-package),  
 6

genomeIntervals-package, 6

getGffAttribute, 7, 9, 13, 14

i(gen\_ints), 4

inter\_base  
 (Genome\_intervals-class), 4

inter\_base, Genome\_intervals-method  
 (Genome\_intervals-class), 4

inter\_base<-  
 (Genome\_intervals-class), 4

inter\_base<-, Genome\_intervals-method  
 (Genome\_intervals-class), 4

interval\_complement, 7, 12

interval\_complement  
 (interval\_union), 11

interval\_complement, Genome\_intervals-method  
 (interval\_union), 11

interval\_complement, Genome\_intervals\_stranded-  
 (interval\_union), 11

interval\_intersection, 7, 12

interval\_intersection  
 (interval\_union), 11

interval\_intersection, Genome\_intervals-method  
 (interval\_union), 11

interval\_intersection, Genome\_intervals\_stranded-  
 (interval\_union), 11

interval\_overlap, 7, 10

interval\_overlap, ANY, missing-method  
 (interval\_overlap), 10

interval\_overlap, Genome\_intervals, Genome\_inter-  
 (interval\_overlap), 10

interval\_overlap, Genome\_intervals\_stranded-class, 10  
     (interval\_overlap), 10  
     vector, 5, 8  
 interval\_overlap, missing, ANY-method  
     (interval\_overlap), 10  
 interval\_union, 7, 11, 12  
 interval\_union, Genome\_intervals-method  
     (interval\_union), 11  
 interval\_union, Genome\_intervals\_stranded-method  
     (interval\_union), 11  
 intervals, 7  
 intervals::distance\_to\_nearest,  
     3  
 intervals:interval\_overlap, 10  
 Intervals\_full, 4, 5, 8  
 Intervals\_virtual, 5, 8  
  
 j(gen\_ints), 4  
  
 k(gen\_ints), 4  
  
 matrix, 5, 8  
  
 parseGffAttributes, 7, 9, 12, 14  
  
 readGff3, 7, 9, 13, 13  
 reduce, 12  
  
 seq\_name  
     (Genome\_intervals-class), 4  
 seq\_name, Genome\_intervals-method  
     (Genome\_intervals-class), 4  
 seq\_name<-  
     (Genome\_intervals-class), 4  
 seq\_name<-, Genome\_intervals-method  
     (Genome\_intervals-class), 4  
 show, Genome\_intervals-method  
     (Genome\_intervals-class), 4  
 size, Genome\_intervals-method  
     (Genome\_intervals-class), 4  
 strand  
     (Genome\_intervals\_stranded-class),  
     7  
 strand, Genome\_intervals\_stranded-method  
     (Genome\_intervals\_stranded-class),  
     7  
 strand<-  
     (Genome\_intervals\_stranded-class),  
     7  
 strand<-, Genome\_intervals\_stranded-method  
     (Genome\_intervals\_stranded-class),  
     7  
 structure, 5, 8  
  
 type<-, Genome\_intervals-method  
     (Genome\_intervals-class), 4