Package 'HelloRanges'

November 12, 2026

Type Package

Title Introduce *Ranges to bedtools users

Version 1.38.0

Author Michael Lawrence

Maintainer Michael Lawrence <lawremi@gmail.com>

Description Translates bedtools command-line invocations to R code calling functions from the Bioconductor *Ranges infrastructure. This is intended to educate novice Bioconductor users and to compare the syntax and semantics of the two frameworks.

License GPL (>= 2)

Imports docopt, stats, tools, utils

Depends methods, BiocGenerics, S4Vectors (>= 0.17.39), IRanges (>= 2.13.12), GenomicRanges (>= 1.31.10), Biostrings (>= 2.41.3), BSgenome, GenomicFeatures (>= 1.31.5), VariantAnnotation (>= 1.19.3), Rsamtools, GenomicAlignments (>= 1.15.7), rtracklayer (>= 1.33.8), Seqinfo, SummarizedExperiment, BiocIO

Suggests GenomeInfoDb, HelloRangesData, BiocStyle, RUnit, TxDb.Hsapiens.UCSC.hg19.knownGene

biocViews Sequencing, Annotation, Coverage, GenomeAnnotation, DataImport, SequenceMatching, VariantAnnotation

git_url https://git.bioconductor.org/packages/HelloRanges

git_branch RELEASE_3_23

git_last_commit 66a66ac

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Contents

argparsing 2

bedtools_closest . . . 3
bedtools_complement . . . 6
bedtools_coverage . . . 7
bedtools_flank . . . 10
bedtools_genomecov . . . 11
bedtools_getfasta . . . 14
bedtools_groupby . . . 15
bedtools_intersect . . . 18
bedtools_jaccard . . . 20
bedtools_makewindows . . . 23
bedtools_map . . . 24
bedtools_merge . . . 27
bedtools_multiinter . . . 29
bedtools_nuc . . . 30
bedtools_shift . . . 32
bedtools_slop . . . 34
bedtools_subtract . . . 35
bedtools_unionbedg . . . 37
distmode . . . 39
pair . . . 40

Index **42**

`argparsing` *Argument parsing details*

Description

HelloRanges uses **docopt** for parsing the argument string passed as the `cmd` argument to functions like `bedtools_intersect`. `bedtools` has its own style of argument formatting. Here we document the subtle differences.

Details

Here are the specific differences:

- **docopt** requires that multi-character arguments are prefixed by two hyphens, e.g., '`--bed`'. However, `bedtools` expects only a single hyphen. It turns out **docopt** is robust to the single-hyphen case, except for the first argument. Since the typical convention is to first indicate the file, e.g., '`-a`' or '`-i`', this incompatibility does not often arise in practice.
- **docopt** does not allow values of argument to be space-separated, while `bedtools` often expects space separation for multi-valued arguments. As a compromise, HelloRanges expects the values to be comma-separated. Thus, '`-b b.bed c.bed`' needs to be '`-b b.bed,c.bed`'.
- Most shells support nested commands within parentheses, e.g., '`-b < (grep foo file.bed)`', but **docopt** does not support that. Instead, nested commands should be enclosed in double quotes, e.g., '`-b < "grep foo file.bed"`'. Such constructs are supported via `pipe`.

Author(s)

Michael Lawrence

`bedtools_closest` *bedtools_closest*

Description

Finds the features in one dataset that are closest to those in another. Supports restriction by strand, upstream, downstream, and overlap. There are several methods for resolving ties. Optionally returns the distance.

Usage

```
bedtools_closest(cmd = "--help")
R_bedtools_closest(a, b, s = FALSE, S = FALSE, d = FALSE,
                   D = c("none", "ref", "a", "b"), io = FALSE, iu = FALSE,
                   id = FALSE, fu = FALSE, fd = FALSE,
                   t = c("all", "first", "last"), mdb = c("each", "all"),
                   k = 1L, names = NULL, filenames = FALSE, N = FALSE)
do_bedtools_closest(a, b, s = FALSE, S = FALSE, d = FALSE,
                    D = c("none", "ref", "a", "b"), io = FALSE, iu = FALSE,
                    id = FALSE, fu = FALSE, fd = FALSE,
                    t = c("all", "first", "last"), mdb = c("each", "all"),
                    k = 1L, names = NULL, filenames = FALSE, N = FALSE)
```

Arguments

`cmd`	String of bedtools command line arguments, as they would be entered at the shell. There are a few incompatibilities between the **docopt** parser and the bedtools style. See argument parsing.
`a`	Path to a BAM/BED/GFF/VCF/etc file, a BED stream, a file object, or a ranged data structure, such as a GRanges. Each feature in `a` is compared to `b` in search of nearest neighbors. Use `"stdin"` for input from another process (presumably while running via `Rscript`). For streaming from a subprocess, prefix the command string with "<", e.g., `"<grep foo file.bed"`. Any streamed data is assumed to be in BED format.
`b`	Like `a`, except supports multiple datasets, either as a vector/list or a comma-separated string. Also supports file glob patterns, i.e., strings containing the wildcard, "*".
`s`	Require same strandedness. That is, find the closest feature in `b` that overlaps `a` on the *same* strand. By default, overlaps are reported without respect to strand. Note that this is the exact opposite of Bioconductor behavior.
`S`	Require opposite strandedness. That is, find the closest feature in `b` that overlaps `a` on the *opposite* strand. By default, overlaps are reported without respect to strand.

d	In addition to the closest feature in b, report its distance to a as an extra column. The reported distance for overlapping features will be 0.
D	Like d, report the closest feature in b, and its distance to a as an extra column. However unlike d, D conveys a notion of upstream that is useful with other arguments. See details.
io	Ignore features in b that overlap a. That is, we want close, yet not touching features only.
iu	Ignore features in b that are upstream of features in a. This option requires D and follows its orientation rules for determining what is "upstream".
id	Ignore features in b that are downstream of features in a. This option requires D and follows its orientation rules for determining what is "downstream".
fu	Choose first from features in b that are upstream of features in a. This option requires D and follows its orientation rules for determining what is "upstream".
fd	Choose first from features in b that are downstream of features in a. This option requires D and follows its orientation rules for determining what is "downstream".
t	Specify how ties for closest feature should be handled. This occurs when two features in b have exactly the same "closeness" with a. By default, all such features in b are reported. The modes options are "all", "first" and "last".
mdb	How multiple databases should be resolved, either "each" (find closest in each b dataset independently) or "all" (combine all b datasets prior to the search).
k	**Not supported yet.** Report the k closest hits. Default is 1. If t is "all", all ties will still be reported.
names	When using multiple databases, provide an alias for each to use instead of their integer index. If a single string, can be comma-separated.
filenames	When using multiple databases, use their complete filename instead of their integer index.
N	**Not yet supported**, but related use cases are often solved by passing a single argument to nearest. Require that the query and the closest hit have different names. For BED, the 4th column is compared.

Details

As with all commands, there are three interfaces to the closest command:

bedtools_closest Parses the bedtools command line and compiles it to the equivalent R code.

R_bedtools_closest Accepts R arguments corresponding to the command line arguments and compiles the equivalent R code.

do_bedtools_closest Evaluates the result of R_bedtools_closest. Recommended **only** for demonstration and testing. It is best to integrate the compiled code into an R script, after studying it.

The generated code includes calls to utilities like nearest, precede and follow. nearest lacks the ability to restrict its search by direction/overlap, so some complex code is generated to support all of the argument combinations.

Arguments io, iu, id, fu, and fd require a notion of upstream/downstream to be defined via D, which accepts one of these values:

ref Report distance with respect to the reference genome. B features with a lower (start, stop) are "upstream".

a Report distance with respect to A. When A is on the - strand, "upstream" means B has a higher (start,stop).

b Report distance with respect to B. When B is on the - strand, "upstream" means A has a higher (start,stop).

Value

A language object containing the compiled R code, evaluating to a Pairs object with the closest hits from `a` and `b`. If `d` or `D` is `TRUE`, has a metadata column called "distance".

Author(s)

Michael Lawrence

References

http://bedtools.readthedocs.io/en/latest/content/tools/closest.html

See Also

nearest-methods for the various ways to find the nearest ranges.

Examples

```
## Not run:
setwd(system.file("unitTests", "data", "closest", package="HelloRanges"))

## End(Not run)

## basic
bedtools_closest("-a a.bed -b b.bed -d")
## strand-specific
bedtools_closest("-a strand-test-a.bed -b strand-test-b.bed -s")
## break ties
bedtools_closest("-a close-a.bed -b close-b.bed -t first")
## multiple databases
bedtools_closest("-a mq1.bed -b mdb1.bed,mdb2.bed,mdb3.bed -names a,b,c")
## ignoring upstream
bedtools_closest("-a d.bed -b d_iu.bed -D ref -iu")
```

bedtools_complement *bedtools_complement*

Description

Finds regions of the genome that are not covered by a genomic dataset.

Usage

```
bedtools_complement(cmd = "--help")
R_bedtools_complement(i, g)
do_bedtools_complement(i, g)
```

Arguments

cmd	String of bedtools command line arguments, as they would be entered at the shell. There are a few incompatibilities between the **docopt** parser and the bedtools style. See argument parsing.
i	Path to a BAM/BED/GFF/VCF/etc file, a BED stream, a file object, or a ranged data structure, such as a GRanges. Use "stdin" for input from another process (presumably while running via Rscript). For streaming from a subprocess, prefix the command string with "<", e.g., "<grep foo file.bed". Any streamed data is assumed to be in BED format.
g	A genome file, identifier or Seqinfo object that defines the order and size of the sequences.

Details

As with all commands, there are three interfaces to the complement command:

bedtools_complement Parses the bedtools command line and compiles it to the equivalent R code.

R_bedtools_complement Accepts R arguments corresponding to the command line arguments and compiles the equivalent R code.

do_bedtools_complement Evaluates the result of R_bedtools_complement. Recommended **only** for demonstration and testing. It is best to integrate the compiled code into an R script, after studying it.

The generated code is subtracts, via setdiff, the ranges from the set of ranges representing the entire genome.

While it may be tempting to call gaps instead, it is very unlikely to behave as expected. The GenomicRanges set operations treat all three strand values (+, -, *) as separate spaces. gaps takes as its universe the genome on all three strands, rather than just the "*" strand, resulting in extraneous stranded ranges.

Value

A language object containing the compiled R code, evaluating to a GRanges object with the complementary ranges.

Author(s)

Michael Lawrence

References

http://bedtools.readthedocs.io/en/latest/content/tools/complement.html

See Also

setops-methods for the various set operations.

Examples

```
## Not run:
setwd(system.file("unitTests", "data", "coverage", package="HelloRanges"))

## End(Not run)
bedtools_complement("-i a.bed -g test.genome")
```

---

bedtools_coverage *bedtools_coverage*

Description

Compute the coverage of one or more datasets over a set of query ranges.

Usage

```
bedtools_coverage(cmd = "--help")
R_bedtools_coverage(a, b, hist = FALSE, d = FALSE, counts = FALSE,
                    f = 1e-09, F = 1e-09, r = FALSE, e = FALSE,
                    s = FALSE, S = FALSE, split = FALSE, g = NA_character_,
                    header = FALSE, sortout = FALSE)
do_bedtools_coverage(a, b, hist = FALSE, d = FALSE, counts = FALSE,
                     f = 1e-09, F = 1e-09, r = FALSE, e = FALSE,
                     s = FALSE, S = FALSE, split = FALSE, g = NA_character_,
                     header = FALSE, sortout = FALSE)
```

Arguments

cmd	String of bedtools command line arguments, as they would be entered at the shell. There are a few incompatibilities between the **docopt** parser and the bedtools style. See argument parsing.
a	Path to a BAM/BED/GFF/VCF/etc file, a BED stream, a file object, or a ranged data structure, such as a GRanges. The coverage is computed over these ranges. Use "stdin" for input from another process (presumably while running via Rscript). For streaming from a subprocess, prefix the command string with "<", e.g., "<grep foo file.bed". Any streamed data is assumed to be in BED format.

b	Like a, except supports multiple datasets, either as a vector/list or a comma-separated string. Also supports file glob patterns, i.e., strings containing the wildcard, "*". The coverage is computed by counting how many of these ranges overlap positions in a.
hist	Report a histogram of coverage for each feature in a as well as a summary histogram for *all* features in a. See below for the structure of the returned table.
d	Report the depth at each position in each a feature. Positions reported are one based. Each position and depth follow the complete a feature.
counts	Only report the count of overlaps, not fraction, etc. Restricted by f and r.
f	Minimum overlap required as a fraction of A [default: any overlap].
F	Minimum overlap required as a fraction of B [default: any overlap].
r	Require that the fraction of overlap be reciprocal for a and b. In other words, if f is 0.90 and r is TRUE, this requires that b overlap at least 90% of a and that a also overlaps at least 90% of b.
e	Require that the minimum fraction be satisfied for a *OR* b. In other words, if e is TRUE with f=0.90 and F=0.10 this requires that either 90% of a is covered OR 10% of b is covered. If FALSE, both fractions would have to be satisfied.
s	Force strandedness. That is, only count ranges in b that overlap a on the same strand. By default, coverage is computed without respect to strand. Note that this is the exact opposite of Bioconductor behavior.
S	Require opposite strandedness. That is, count the features in b that overlap a on the *opposite* strand. By default, coverage is computed without respect to strand.
split	Treat split BAM (i.e., having an 'N' CIGAR operation) or BED12 entries as compound ranges with gaps, i.e., as GRangesList objects.
g	A genome file, identifier or Seqinfo object that defines the order and size of the sequences.
header	Ignored.
sortout	Sort the result by position.

Details

As with all commands, there are three interfaces to the coverage command:

bedtools_coverage Parses the bedtools command line and compiles it to the equivalent R code.

R_bedtools_coverage Accepts R arguments corresponding to the command line arguments and compiles the equivalent R code.

do_bedtools_coverage Evaluates the result of R_bedtools_coverage. Recommended **only** for demonstration and testing. It is best to integrate the compiled code into an R script, after studying it.

Typically, we compute coverage with coverage, but features like fractional overlap restriction and histograms add (educational) complexity. One key trick is the [,List,GenomicRanges method, which lets us extract coverage vectors for specific regions (see the generated code).

Value

A language object containing the compiled R code, evaluating to a GRanges object with coverage information. The exact type of information depends on the mode:

`default` Three metadata columns: "count" (the number of overlapping ranges), "covered" (the number of bases covered in the query), "fraction" (the fraction of bases covered).

`d` Metadata column "coverage" is an RleList with position-level coverage (depth). This is what we typically refer to as coverage in Bioconductor.

`hist` Metadata column "coverage" is a list of DataFrames. Each DataFrame contains a histogram of the coverage depth over that range, with columns "coverage" (the coverage value), "count" (the number of positions with that coverage), "len" (the length of the region, all the same) and "fraction" (the fraction of positions at that coverage). There is also a "coverage" component on `metadata(ans)` with the same histogram aggregated over all query ranges.

Author(s)

Michael Lawrence

References

http://bedtools.readthedocs.io/en/latest/content/tools/coverage.html

See Also

coverage-methods for ways to compute coverage.

Examples

```
## Not run:
setwd(system.file("unitTests", "data", "coverage", package="HelloRanges"))

## End(Not run)

## default behavior
bedtools_coverage("-a a.bed -b b.bed")
## histogram
bedtools_coverage("-a a.bed -b b.bed -hist -g test.genome")
## per-position depth
bedtools_coverage("-a a.bed -b b.bed -d -g test.genome")
```

bedtools_flank *bedtools_flank*

Description

Compute flanking regions.

Usage

```
bedtools_flank(cmd = "--help")
R_bedtools_flank(i, b = 0, l = 0, r = 0, s = FALSE, pct = FALSE,
                 g = NULL, header = FALSE)
do_bedtools_flank(i, b = 0, l = 0, r = 0, s = FALSE, pct = FALSE,
                  g = NULL, header = FALSE)
```

Arguments

cmd	String of bedtools command line arguments, as they would be entered at the shell. There are a few incompatibilities between the **docopt** parser and the bedtools style. See argument parsing.
i	Path to a BAM/BED/GFF/VCF/etc file, a BED stream, a file object, or a ranged data structure, such as a GRanges. Use "stdin" for input from another process (presumably while running via Rscript). For streaming from a subprocess, prefix the command string with "<", e.g., "<grep foo file.bed". Any streamed data is assumed to be in BED format.
b	Increase the BED/GFF/VCF range by the same number base pairs in each direction. Integer.
l	The number of base pairs to subtract from the start coordinate. Integer.
r	The number of base pairs to add to the end coordinate. Integer.
s	Define l and r based on strand. For example. if used, l is 500 for a negative-stranded feature, it will add 500 bp to the end coordinate.
pct	Define l and r as a fraction of the feature length. E.g. if used on a 1000bp feature, and l is 0.50, will add 500 bp upstream..
g	Genome file, identifier or Seqinfo object that defines the order and size of the sequences.
header	Ignored.

Details

As with all commands, there are three interfaces to the flank command:

bedtools_flank Parses the bedtools command line and compiles it to the equivalent R code.

R_bedtools_flank Accepts R arguments corresponding to the command line arguments and compiles the equivalent R code.

do_bedtools_flank Evaluates the result of R_bedtools_flank. Recommended **only** for demonstration and testing. It is best to integrate the compiled code into an R script, after studying it.

We compute the flanks with flank, although flank only computes one side at a time, so we may call it multiple times.

Value

A language object containing the compiled R code, evaluating to a GRanges object.

Author(s)

Michael Lawrence

References

http://bedtools.readthedocs.io/en/latest/content/tools/flank.html

See Also

intra-range-methods for flank.

Examples

```
## Not run:
setwd(system.file("unitTests", "data", "flank", package="HelloRanges"))

## End(Not run)
## 5 on both sides
r <- bedtools_flank("-i a.bed -b 5 -g tiny.genome")
## 5 on left side
bedtools_flank("-i a.bed -l 5 -r 0 -g tiny.genome")
## define left/right in terms of transcription direction
bedtools_flank("-i a.bed -l 5 -r 0 -s -g tiny.genome")
```

---

bedtools_genomecov *bedtools_genomecov*

Description

Compute coverage over the genome. By default, this computes a per-chromosome histogram of the coverage, but options allow for per-position coverage to be returned in different ways.

Usage

```
bedtools_genomecov(cmd = "--help")
R_bedtools_genomecov(i, g = NA_character_, d = FALSE, dz = FALSE,
                     bg = FALSE, bga = FALSE, split = FALSE,
                     strand = c("any", "+", "-"), `5` = FALSE, `3` = FALSE,
                     max = NULL, scale = 1, pc = FALSE, fs = NULL)
do_bedtools_genomecov(i, g = NA_character_, d = FALSE, dz = FALSE,
                      bg = FALSE, bga = FALSE, split = FALSE,
                      strand = c("any", "+", "-"), `5` = FALSE, `3` = FALSE,
                      max = NULL, scale = 1, pc = FALSE, fs = NULL)
```

Arguments

cmd	String of bedtools command line arguments, as they would be entered at the shell. There are a few incompatibilities between the **docopt** parser and the bedtools style. See argument parsing.
i	Path to a BAM/BED/GFF/VCF/etc file, a BED stream, a file object, or a ranged data structure, such as a GRanges. Use "stdin" for input from another process (presumably while running via Rscript). For streaming from a subprocess, prefix the command string with "<", e.g., "<grep foo file.bed". Any streamed data is assumed to be in BED format.
g	Genome file, identifier or Seqinfo object that defines the order and size of the sequences.
d	Report the depth at each genome position. This causes a GPos object to be returned.
dz	Same as d, except the zero coverage positions are dropped.
bg	Like d, except returns a GRanges object, which is useful for operating on runs of coverage. Zero coverage runs are dropped.
bga	Like bg, except the zero coverage runs are retained.
split	Treat split BAM (i.e., having an 'N' CIGAR operation) or BED12 entries as compound ranges with gaps, i.e., as GRangesList objects.
strand	Calculate coverage of intervals from a specific strand.
5	Calculate coverage of 5' positions (instead of entire interval).
3	Calculate coverage of 3' positions (instead of entire interval).
max	Combine all positions with a depth >= max into a single bin in the histogram.
scale	Scale the coverage by a constant factor. Each coverage value is multiplied by this factor before being reported. Useful for normalizing coverage by, e.g., reads per million (RPM).
pc	Calculates coverage of intervals from left point of a pair reads to the right point. Works for BAM files only.
fs	Forces to use the given fragment size instead of read length.

Details

As with all commands, there are three interfaces to the genomecov command:

`bedtools_genomecov` Parses the bedtools command line and compiles it to the equivalent R code.

`R_bedtools_genomecov` Accepts R arguments corresponding to the command line arguments and compiles the equivalent R code.

`do_bedtools_genomecov` Evaluates the result of `R_bedtools_genomecov`. Recommended **only** for demonstration and testing. It is best to integrate the compiled code into an R script, after studying it.

We typically compute the coverage with `coverage`. Computing the histogram requires more work.

Value

A language object containing the compiled R code, evaluating to an object that depends on the mode:

`default`	A DataFrame that is a per-chromosome histogram of the coverage that includes the whole genome margin. Includes columns "seqnames" (the chromosome name, or "genome"), "coverage" (the coverage value), "count" (the count of positions covered at that value), "len" (the length of the chromosome/genome), "fraction" (the fraction of bases covered at the value).
`d`, `dz`	A GPos object with per-position coverage values.
`bg`, `bga`	A GRanges object with coverage runs.

Author(s)

Michael Lawrence

References

http://bedtools.readthedocs.io/en/latest/content/tools/genomecov.html

See Also

intra-range-methods for genomecov.

Examples

```
## Not run:
setwd(system.file("unitTests", "data", "genomecov", package="HelloRanges"))

## End(Not run)
## get coverage runs as a GRanges
bedtools_genomecov("-i y.bed -bg -g test.genome")
## get coverage depth as a GPos, dropping zero values, ignore junctions
bedtools_genomecov("-i three_blocks.bam -dz -split")
## custom fragment size
bedtools_genomecov("-i chip.bam -bg -fs 100")
```

bedtools_getfasta *bedtools_getfasta*

Description

Query sequence from a FASTA file given a set of ranges, including compound regions like transcripts and junction reads. This assumes the sequence is DNA.

Usage

```
bedtools_getfasta(cmd = "--help")
R_bedtools_getfasta(fi, bed, s = FALSE, split = FALSE)
do_bedtools_getfasta(fi, bed, s = FALSE, split = FALSE)
```

Arguments

cmd	String of bedtools command line arguments, as they would be entered at the shell. There are a few incompatibilities between the **docopt** parser and the bedtools style. See argument parsing.
fi	Path to a FASTA file, or an XStringSet object.
bed	Path to a BAM/BED/GFF/VCF/etc file, a BED stream, a file object, or a ranged data structure, such as a GRanges, as the query. Use "stdin" for input from another process (presumably while running via Rscript). For streaming from a subprocess, prefix the command string with "<", e.g., "<grep foo file.bed". Any streamed data is assumed to be in BED format.
s	Force strandedness. If the feature occupies the antisense strand, the sequence will be reverse complemented.
split	Given BED12 or BAM input, extract and concatenate the sequences from the blocks (e.g., exons).

Details

As with all commands, there are three interfaces to the getfasta command:

bedtools_getfasta Parses the bedtools command line and compiles it to the equivalent R code.

R_bedtools_getfasta Accepts R arguments corresponding to the command line arguments and compiles the equivalent R code.

do_bedtools_getfasta Evaluates the result of R_bedtools_getfasta. Recommended **only** for demonstration and testing. It is best to integrate the compiled code into an R script, after studying it.

It is recommended to retrieve reference sequence using a **BSgenome** package, either custom or provided by Bioconductor. Call getSeq to query for specific regions of the BSgenome object. If one must access a file, consider converting it to 2bit or FA (razip) format for indexed access using import and its which argument.

But if one must access a FASTA file, we need to read all of it with readDNAStringSet and extract regions using x[gr], where gr is a GRanges or GRangesList.

Value

A language object containing the compiled R code, evaluating to a DNAStringSet object.

Author(s)

Michael Lawrence

References

http://bedtools.readthedocs.io/en/latest/content/tools/getfasta.html

See Also

getSeq, the primary sequence query interface.

Examples

```
## Not run:
setwd(system.file("unitTests", "data", "getfasta", package="HelloRanges"))

## End(Not run)
    ## simple query
    bedtools_getfasta("--fi t.fa -bed blocks.bed")
    ## get spliced transcript/read sequence
    bedtools_getfasta("--fi t.fa -bed blocks.bed -split")
```

---

bedtools_groupby *bedtools_groupby*

Description

Query sequence from a FASTA file given a set of ranges, including compound regions like transcripts and junction reads. This assumes the sequence is DNA.

Usage

```
bedtools_groupby(cmd = "--help")
R_bedtools_groupby(i, g = 1:3, c, o = "sum", delim=",")
do_bedtools_groupby(i, g = 1:3, c, o = "sum", delim=",")
```

Arguments

cmd	String of bedtools command line arguments, as they would be entered at the shell. There are a few incompatibilities between the **docopt** parser and the bedtools style. See argument parsing.

i	Path to a BAM/BED/GFF/VCF/etc file, a BED stream, a file object, or a ranged data structure, such as a GRanges. Use "stdin" for input from another process (presumably while running via Rscript). For streaming from a subprocess, prefix the command string with "<", e.g., "<grep foo file.bed". Any streamed data is assumed to be in BED format.
g	Column index(es) for grouping the input. Columns may be comma-separated. By default, the grouping is by range.
c	Specify columns (by integer index) from the input file to operate upon (see o option, below). Multiple columns can be specified in a comma-delimited list.
o	Specify the operations (by name) that should be applied to the columns indicated in c. Multiple operations can be specified in a comma-delimited list. Recycling is used to align c and o. See the details for the available operations.
delim	Delimiter character used to collapse strings.

Details

As with all commands, there are three interfaces to the groupby command:

bedtools_groupby Parses the bedtools command line and compiles it to the equivalent R code.

R_bedtools_groupby Accepts R arguments corresponding to the command line arguments and compiles the equivalent R code.

do_bedtools_groupby Evaluates the result of R_bedtools_groupby. Recommended **only** for demonstration and testing. It is best to integrate the compiled code into an R script, after studying it.

The workhorse for aggregation in R is aggregate and we have extended its interface to make it more convenient. See aggregate for details.

The following operations are supported (with R translation):

sum sum(X)

min min(X)

max max(X)

absmin min(abs(X))

absmax max(abs(X))

mean mean(X)

median median(X)

mode distmode(X)

antimode distmode(X, anti=TRUE)

collapse unstrsplit(X, delim)

distinct unstrsplit(unique(X), delim)

count lengths(X)

count_distinct lengths(unique(X))

sstdev sd(X) freqtable(X) firstdrop(heads(X, 1L)) lastdrop(tails(X, 1L))

For the sake of simplicity, and because the use cases are not clear, we do not support aggregation of every column. Here are some of the restrictions:

- No support for the last column of GFF (the ragged list of attributes).
- No support for the INFO, FORMAT and GENO fields of VCF.
- No support for the FLAG field of BAM (`bedtools` does not support this either).

Value

A language object containing the compiled R code, generally evaluating to a DataFrame, with a column for each grouping variable and each summarized variable. As a special case, if there are no grouping variables specified, then the grouping is by range, and an aggregated GRanges is returned.

Note

We admit that using column subscripts for `c` makes code hard to read. All the more reason to just write R code.

Author(s)

Michael Lawrence

References

http://bedtools.readthedocs.io/en/latest/content/tools/groupby.html

See Also

aggregate-methods for general aggregation.

Examples

```
## Not run:
setwd(system.file("unitTests", "data", "groupby", package="HelloRanges"))

## End(Not run)
    ## aggregation by range
    bedtools_groupby("-i values3.header.bed -c 5")
    ## average variant qualities by chromosome and reference base
## Not run:
    indexTabix(bgzip("a_vcfSVtest.vcf", overwrite=TRUE), "vcf")

## End(Not run)
    bedtools_groupby("-i a_vcfSVtest.vcf.bgz -g 1,4 -c 6 -o mean")
```

bedtools_intersect *bedtools_intersect*

Description

Finds and/or counts the intersections between two ranged datasets.

Usage

```
bedtools_intersect(cmd = "--help")
R_bedtools_intersect(a, b, ubam = FALSE, bed = FALSE, wa = FALSE, wb = FALSE,
                     loj = FALSE, wo = FALSE, wao = FALSE, u = FALSE,
                     c = FALSE, v = FALSE, f = 1e-09, F = 1e-09,
                     r = FALSE, e = FALSE, s = FALSE, S = FALSE,
                     split = FALSE, g = NA_character_, header = FALSE,
                     names = NULL, filenames = FALSE, sortout = FALSE)
do_bedtools_intersect(a, b, ubam = FALSE, bed = FALSE, wa = FALSE, wb = FALSE,
                      loj = FALSE, wo = FALSE, wao = FALSE, u = FALSE,
                      c = FALSE, v = FALSE, f = 1e-09, F = 1e-09,
                      r = FALSE, e = FALSE, s = FALSE, S = FALSE,
                      split = FALSE, g = NA_character_, header = FALSE,
                      names = NULL, filenames = FALSE, sortout = FALSE)
```

Arguments

cmd	String of bedtools command line arguments, as they would be entered at the shell. There are a few incompatibilities between the **docopt** parser and the bedtools style. See argument parsing.
a	Path to a BAM/BED/GFF/VCF/etc file, a BED stream, a file object, or a ranged data structure, such as a GRanges. Each feature in a is compared to b in search of overlaps. Use "stdin" for input from another process (presumably while running via Rscript). For streaming from a subprocess, prefix the command string with "<", e.g., "<grep foo file.bed". Any streamed data is assumed to be in BED format.
b	Like a, except supports multiple datasets, either as a vector/list or a comma-separated string. Also supports file glob patterns, i.e., strings containing the wildcard, "*".
ubam	**Not supported yet.** Write uncompressed BAM output. The default is write compressed BAM output.
bed	When using BAM input, return a GRanges (with a "blocks" column) instead of a GAlignments. For VCF input, return a VRanges instead of a VCF object.
wa	Return the original entry in a for each overlap.
wb	Return the original entry in b for each overlap.
loj	Perform a 'left outer join'. That is, for each feature in a report each overlap with b. If no overlaps are found, report an empty range for b on the "." sequence. Implies wa=TRUE and wb=TRUE.

wo	Return the number of base pairs of overlap between the two features as the "overlap_width" metadata column. Implies wa=TRUE and wb=TRUE.
wao	Like wo, except it additionally implies loj=TRUE.
u	Like wa, except only the unique entries in a are returned.
c	Like wa, except also count the number of hits in b for each range in a and return the count as the "overlap_count" metadata column.
v	Like wa, except only report those entries in a that have *no* overlap in b.
f	Minimum overlap required as a fraction of a [default: any overlap].
F	Minimum overlap required as a fraction of b [default: any overlap].
r	Require that the fraction of overlap be reciprocal for a and b. In other words, if f is 0.90 and r is TRUE, this requires that b overlap at least 90% of a and that a also overlaps at least 90% of b.
e	Require that the minimum fraction be satisfied for a *OR* b. In other words, if e is TRUE with f=0.90 and F=0.10 this requires that either 90% of a is covered OR 10% of b is covered. If FALSE, both fractions would have to be satisfied.
s	Require same strandedness. That is, find the intersect feature in b that overlaps a on the *same* strand. By default, overlaps are reported without respect to strand. Note that this is the exact opposite of Bioconductor behavior.
S	Require opposite strandedness. That is, find the intersect feature in b that overlaps a on the *opposite* strand. By default, overlaps are reported without respect to strand.
split	Treat split BAM (i.e., having an 'N' CIGAR operation) or BED12 entries as compound ranges with gaps, i.e., as GRangesList objects.
g	A genome file, identifier or Seqinfo object that defines the order and size of the sequences.
header	Ignored.
names	When using multiple databases, provide an alias for each to use instead of their integer index. If a single string, can be comma-separated.
filenames	When using multiple databases, use their complete filename instead of their integer index.
sortout	Sort the result by genomic coordinate.

Details

As with all commands, there are three interfaces to the intersect command:

bedtools_intersect Parses the bedtools command line and compiles it to the equivalent R code.

R_bedtools_intersect Accepts R arguments corresponding to the command line arguments and compiles the equivalent R code.

do_bedtools_intersect Evaluates the result of R_bedtools_intersect. Recommended **only** for demonstration and testing. It is best to integrate the compiled code into an R script, after studying it.

This is by far the most complex bedtools command, and it offers a dizzying list of parameters, many of which are redundant or mutually exclusive. The complexity of the generated code is highest when using the fractional restriction feature, since no such support exists in the GenomicRanges overlap routines.

Value

A language object containing the compiled R code, evaluating to a ranges object, the exact type of which depends on the arguments. If both `wa` and `wb` are `TRUE`, return a Pairs object with the original, matched up ranges, possibly with metadata columns. By default, the return value is a GAlignments for BAM input, a VCF object for VCF input, or a GRanges for any other type of input. If `bed` is `TRUE`, BAM input is converted to a GRanges, containing a "blocks" column (encoding the junctions) if the input is BAM. If the input is VCF, `bed=TRUE` converts the input to a VRanges.

Author(s)

Michael Lawrence

References

http://bedtools.readthedocs.io/en/latest/content/tools/intersect.html

See Also

setops-methods for set operations including intersect, findOverlaps-methods for different ways to detect overlaps.

Examples

```
## Not run:
setwd(system.file("unitTests", "data", "intersect", package="HelloRanges"))

## End(Not run)

## return intersecting ranges
bedtools_intersect("-a a.bed -b a.bed")
## add count
bedtools_intersect("-a a.bed -b b.bed -c")
## restrict by strand and fraction of overlap
bedtools_intersect("-a a.bed -b b.bed -c -s -f 0.1")
## return original 'a' ranges
bedtools_intersect("-a a.bed -b b.bed -wa")
## return both 'a' and 'b' ranges, along with overlap widths
bedtools_intersect("-a a.bed -b b.bed -wo")
## same as above, except left outer join
bedtools_intersect("-a a.bed -b b.bed -wao")
## consider read junction structure
bedtools_intersect("-a three_blocks.bam -b three_blocks_nomatch.bed -split")
```

---

bedtools_jaccard *bedtools_jaccard*

Description

Compare two sets of genomic regions using the Jaccard statistic, defined as the total width of the intersection, divided by the total width of the union.

Usage

```
bedtools_jaccard(cmd = "--help")
R_bedtools_jaccard(a, b, f = 1e-09, F = 1e-09, r = FALSE, e = FALSE,
                   s = FALSE, S = FALSE, split = FALSE)
do_bedtools_jaccard(a, b, f = 1e-09, F = 1e-09, r = FALSE, e = FALSE,
                    s = FALSE, S = FALSE, split = FALSE)
```

Arguments

cmd	String of bedtools command line arguments, as they would be entered at the shell. There are a few incompatibilities between the **docopt** parser and the bedtools style. See argument parsing.
a	Path to a BAM/BED/GFF/VCF/etc file, a BED stream, a file object, or a ranged data structure, such as a GRanges. Use "stdin" for input from another process (presumably while running via Rscript). For streaming from a subprocess, prefix the command string with "<", e.g., "<grep foo file.bed". Any streamed data is assumed to be in BED format.
b	Like a, except supports multiple datasets, either as a vector/list or a comma-separated string. Also supports file glob patterns, i.e., strings containing the wildcard, "*".
f	Minimum overlap required as a fraction of a [default: any overlap].
F	Minimum overlap required as a fraction of b [default: any overlap].
r	Require that the fraction of overlap be reciprocal for a and b. In other words, if f is 0.90 and r is TRUE, this requires that b overlap at least 90% of a and that a also overlaps at least 90% of b.
e	Require that the minimum fraction be satisfied for a *OR* b. In other words, if e is TRUE with f=0.90 and F=0.10 this requires that either 90% of a is covered OR 10% of b is covered. If FALSE, both fractions would have to be satisfied.
s	Require same strandedness. That is, find the jaccard feature in b that overlaps a on the *same* strand. By default, overlaps are reported without respect to strand. Note that this is the exact opposite of Bioconductor behavior.
S	Require opposite strandedness. That is, find the jaccard feature in b that overlaps a on the *opposite* strand. By default, overlaps are reported without respect to strand.
split	Treat split BAM (i.e., having an 'N' CIGAR operation) or BED12 entries as compound ranges with gaps, i.e., as GRangesList objects.

Details

As with all commands, there are three interfaces to the jaccard command:

bedtools_jaccard Parses the bedtools command line and compiles it to the equivalent R code.

R_bedtools_jaccard Accepts R arguments corresponding to the command line arguments and compiles the equivalent R code.

`do_bedtools_jaccard` Evaluates the result of `R_bedtools_jaccard`. Recommended **only** for demonstration and testing. It is best to integrate the compiled code into an R script, after studying it.

This is mostly just `intersect` and `union`, except when fractional overlap restrictions are involved.

Value

A language object containing the compiled R code, evaluating to a a DataFrame with four columns:

`intersection`	total width of intersection
`union`	total width of union
`jaccard`	the jaccard statistic
`n_intersections`	the number of ranges representing the intersection

Author(s)

Michael Lawrence

References

http://bedtools.readthedocs.io/en/latest/content/tools/jaccard.html

See Also

setops-methods for set operations including intersect and union.

Examples

```
## Not run:
setwd(system.file("unitTests", "data", "jaccard", package="HelloRanges"))

## End(Not run)

## basic
bedtools_jaccard("-a a.bed -b a.bed")
## excluding the gaps in compound ranges
bedtools_jaccard("-a three_blocks_match.bed -b e.bed -split")
## strand and fractional overlap restriction
bedtools_jaccard("-a aMixedStrands.bed -b bMixedStrands.bed -s -f 0.8")
```

bedtools_makewindows *bedtools_makewindows*

Description

Generate a partitioning/tiling or set of sliding windows over the genome or a set of ranges.

Usage

```
bedtools_makewindows(cmd = "--help")
R_bedtools_makewindows(b, g = NA_character_, w, s, n)
do_bedtools_makewindows(b, g = NA_character_, w, s, n)
```

Arguments

cmd	String of bedtools command line arguments, as they would be entered at the shell. There are a few incompatibilities between the **docopt** parser and the bedtools style. See argument parsing.
b	Path to a BAM/BED/GFF/VCF/etc file, a BED stream, a file object, or a ranged data structure, such as a GRanges. Use "stdin" for input from another process (presumably while running via Rscript). For streaming from a subprocess, prefix the command string with "<", e.g., "<grep foo file.bed". Any streamed data is assumed to be in BED format. Windows are generated with each range. Exclusive with g.
g	A genome file, identifier or Seqinfo object that defines the order and size of the sequences. Specifying this generates windows over the genome. Exclusive with b.
w	Window size, exclusive with n.
s	Step size (generates sliding windows).
n	Number of windows, exclusive with w.

Details

As with all commands, there are three interfaces to the makewindows command:

bedtools_makewindows Parses the bedtools command line and compiles it to the equivalent R code.

R_bedtools_makewindows Accepts R arguments corresponding to the command line arguments and compiles the equivalent R code.

do_bedtools_makewindows Evaluates the result of R_bedtools_makewindows. Recommended **only** for demonstration and testing. It is best to integrate the compiled code into an R script, after studying it.

We view the generation of a partitioning (or tiling) as a distinct use case from the generation of sliding windows. The two use cases correspond to the tile and slidingWindows functions, respectively.

Value

A language object containing the compiled R code, evaluating to a a GRangesList containing the windows for each range (or chromosome).

Author(s)

Michael Lawrence

References

http://bedtools.readthedocs.io/en/latest/content/tools/makewindows.html

See Also

tile-methods for generating windows.

Examples

```
## Not run:
setwd(system.file("unitTests", "data", "makewindows", package="HelloRanges"))

## End(Not run)

## tiles of width 5000
bedtools_makewindows("-b input.bed -w 5000")
## sliding windows, 5kb wide, every 2kb
bedtools_makewindows("-b input.bed -w 5000 -s 2000")
## 3 tiles in each range
bedtools_makewindows("-b input.bed -n 3")
## 3 tiles for each chromosome of the genome
bedtools_makewindows("-g test.genome -n 3")
```

---

bedtools_map *bedtools_map*

Description

Group ranges by overlap with query ranges and aggregate. By default, the scores are summed.

Usage

```
bedtools_map(cmd = "--help")
R_bedtools_map(a, b, c = "5", o = "sum", f = 1e-09, F = 1e-09,
               r = FALSE, e = FALSE, s = FALSE, S = FALSE, header = FALSE,
               split = FALSE, g = NA_character_, delim=",")
do_bedtools_map(a, b, c = "5", o = "sum", f = 1e-09, F = 1e-09,
                r = FALSE, e = FALSE, s = FALSE, S = FALSE, header = FALSE,
                split = FALSE, g = NA_character_, delim=",")
```

Arguments

cmd	String of bedtools command line arguments, as they would be entered at the shell. There are a few incompatibilities between the **docopt** parser and the bedtools style. See argument parsing.
a	Path to a BAM/BED/GFF/VCF/etc file, a BED stream, a file object, or a ranged data structure, such as a GRanges. Use "stdin" for input from another process (presumably while running via Rscript). For streaming from a subprocess, prefix the command string with "<", e.g., "<grep foo file.bed". Any streamed data is assumed to be in BED format. Windows are generated with each range. Exclusive with g. A summary of b is computed for each range.
b	Like a, except supports multiple datasets, either as a vector/list or a comma-separated string. Also supports file glob patterns, i.e., strings containing the wildcard, "*". Ranges that map to the same range in a are aggregated.
c	Specify columns (by integer index) from the input file to operate upon (see o option, below). Multiple columns can be specified in a comma-delimited list. Defaults to the score column.
o	Specify the operations (by name) that should be applied to the columns indicated in c. Multiple operations can be specified in a comma-delimited list. Recycling is used to align c and o. See bedtools_groupby for the available operations. Defaults to the "sum" operation.
f	Minimum overlap required as a fraction of a [default: any overlap].
F	Minimum overlap required as a fraction of b [default: any overlap].
r	Require that the fraction of overlap be reciprocal for a and b. In other words, if f is 0.90 and r is TRUE, this requires that b overlap at least 90% of a and that a also overlaps at least 90% of b.
e	Require that the minimum fraction be satisfied for a *OR* b. In other words, if e is TRUE with f=0.90 and F=0.10 this requires that either 90% of a is covered OR 10% of b is covered. If FALSE, both fractions would have to be satisfied.
s	Require same strandedness. That is, find the jaccard feature in b that overlaps a on the *same* strand. By default, overlaps are reported without respect to strand. Note that this is the exact opposite of Bioconductor behavior.
S	Require opposite strandedness. That is, find the jaccard feature in b that overlaps a on the *opposite* strand. By default, overlaps are reported without respect to strand.
header	Ignored.
split	Treat split BAM (i.e., having an 'N' CIGAR operation) or BED12 entries as compound ranges with gaps, i.e., as GRangesList objects.
g	A genome file, identifier or Seqinfo object that defines the order and size of the sequences.
delim	Delimiter character used to collapse strings.

Details

As with all commands, there are three interfaces to the map command:

`bedtools_map` Parses the bedtools command line and compiles it to the equivalent R code.

`R_bedtools_map` Accepts R arguments corresponding to the command line arguments and compiles the equivalent R code.

`do_bedtools_map` Evaluates the result of `R_bedtools_map`. Recommended **only** for demonstration and testing. It is best to integrate the compiled code into an R script, after studying it.

Computing overlaps with `findOverlaps` generates a Hits object, which we can pass directly to `aggregate` to aggregate the subject features that overlap the same range in the query.

There are several commands in the `bedtools` suite that might be approximately implemented by passing multiple files to `b` and specifying the aggregate expression `table(b)`. That counts how many ranges from each database/sample overlap a given query. The covered commands are: `bedtools annotate -counts`, `bedtools multicov` and `bedtools tag`.

Value

A language object containing the compiled R code, evaluating to a DataFrame with a "grouping" column corresponding to `as(hits, "List")`, and a column for each summary.

Note

We do not support the `bedtools null` argument, because it seems more sensible to just let R decide on the value of statistics when a group is empty.

Author(s)

Michael Lawrence

References

http://bedtools.readthedocs.io/en/latest/content/tools/map.html

See Also

findOverlaps-methods for finding hits, Hits-class for manipulating them, aggregate-methods for aggregating them.

Examples

```
## Not run:
setwd(system.file("unitTests", "data", "map", package="HelloRanges"))

## End(Not run)

## default behavior
bedtools_map("-a ivls.bed -b values.bed")
## take the mode of the scores
bedtools_map("-a ivls.bed -b values.bed -o mode")
## collapse the chromosome names
bedtools_map("-a ivls.bed -b test.gff2 -c 1 -o collapse")
## collapse the names, restricted by fractional overlap
bedtools_map("-a ivls2.bed -b values5.bed -c 4 -o collapse -f 0.7")
```

bedtools_merge *bedtools_merge*

Description

Collapse overlapping and adjacent ranges into a single range, i.e., reduce the ranges. Then, group the original ranges by reduced range and aggregate. By default, the scores are summed.

Usage

```
bedtools_merge(cmd = "--help")
R_bedtools_merge(i, s = FALSE, S = c("any", "+", "-"), d = 0L, c = NULL,
                 o = "sum", delim = ",")
do_bedtools_merge(i, s = FALSE, S = c("any", "+", "-"), d = 0L, c = NULL,
                  o = "sum", delim = ",")
```

Arguments

cmd	String of bedtools command line arguments, as they would be entered at the shell. There are a few incompatibilities between the **docopt** parser and the bedtools style. See argument parsing.
i	Path to a BAM/BED/GFF/VCF/etc file, a BED stream, a file object, or a ranged data structure, such as a GRanges. Use "stdin" for input from another process (presumably while running via Rscript). For streaming from a subprocess, prefix the command string with "<", e.g., "<grep foo file.bed". Any streamed data is assumed to be in BED format. These are the ranges that are merged.
s	Require same strandedness. That is, find the jaccard feature in b that overlaps a on the *same* strand. By default, overlaps are reported without respect to strand. Note that this is the exact opposite of Bioconductor behavior.
S	Force merge for one specific strand only. Follow with + or - to force merge from only the forward or reverse strand, respectively. By default, merging is done without respect to strand.
d	Maximum distance between features allowed for features to be merged. Default is 0. That is, overlapping and/or book-ended features are merged.
c	Specify columns (by integer index) from the input file to operate upon (see o option, below). Multiple columns can be specified in a comma-delimited list.
o	Specify the operations (by name) that should be applied to the columns indicated in c. Multiple operations can be specified in a comma-delimited list. Recycling is used to align c and o. See bedtools_groupby for the available operations. Defaults to the "sum" operation.
delim	Delimiter character used to collapse strings.

Details

As with all commands, there are three interfaces to the `merge` command:

`bedtools_merge` Parses the bedtools command line and compiles it to the equivalent R code.

`R_bedtools_merge` Accepts R arguments corresponding to the command line arguments and compiles the equivalent R code.

`do_bedtools_merge` Evaluates the result of `R_bedtools_merge`. Recommended **only** for demonstration and testing. It is best to integrate the compiled code into an R script, after studying it.

The workhorse for reduction is `reduce`. Passing `with.revmap=TRUE` to `reduce` causes it to return a list of integers, which can be passed directly to `aggregate` to aggregate the original ranges.

Since the grouping information is preserved in the result, this function serves as a proxy for `bedtools cluster`.

Value

A language object containing the compiled R code, evaluating to a DataFrame with a "grouping" column corresponding to `as(hits, "List")`, and a column for each summary.

Author(s)

Michael Lawrence

References

http://bedtools.readthedocs.io/en/latest/content/tools/merge.html

See Also

bedtools_groupby for more details on bedtools-style aggregation, reduce for merging, aggregate-methods for aggregating.

Examples

```
## Not run:
setwd(system.file("unitTests", "data", "merge", package="HelloRanges"))

## End(Not run)
## default behavior, sum the score
bedtools_merge("-i a.bed")
## count the seqnames
bedtools_merge("-i a.bed -c 1 -o count")
## collapse the names using "|" as the delimiter
bedtools_merge("-i a.names.bed -delim \"|\" -c 4  -o collapse")
## collapse the names and sum the scores
bedtools_merge("-i a.full.bed -c 4,5  -o collapse,sum")
## count and sum the scores
bedtools_merge("-i a.full.bed -c 5  -o count,sum")
## only merge the positive strand features
bedtools_merge("-i a.full.bed -S +")
```

bedtools_multiinter *bedtools_multiinter*

Description

Summarize the ranges according to disjoin and annotate each disjoint range with the samples that overlap the range.

Usage

```
bedtools_multiinter(cmd = "--help")
R_bedtools_multiinter(i, header=FALSE, names=NULL, g=NA_character_,
                      empty=FALSE)
do_bedtools_multiinter(i, header=FALSE, names=NULL, g=NA_character_,
                       empty=FALSE)
```

Arguments

cmd	String of bedtools command line arguments, as they would be entered at the shell. There are a few incompatibilities between the **docopt** parser and the bedtools style. See argument parsing.
i	Paths to BAM/BED/GFF/VCF/etc files (vector or comma-separated), or a list of objects.
header	Ignored.
names	Provide an alias for each to use for each i instead of their integer index. If a single string, can be comma-separated.
g	A genome file, identifier or Seqinfo object that defines the order and size of the sequences.
empty	Report empty regions (i.e., regions not covered in any of the files). This essentially yields a partitioning of the genome (and thus requires g to be specified).

Details

As with all commands, there are three interfaces to the multiinter command:

bedtools_multiinter Parses the bedtools command line and compiles it to the equivalent R code.

R_bedtools_multiinter Accepts R arguments corresponding to the command line arguments and compiles the equivalent R code.

do_bedtools_multiinter Evaluates the result of R_bedtools_multiinter. Recommended **only** for demonstration and testing. It is best to integrate the compiled code into an R script, after studying it.

The workhorse is disjoin. Passing with.revmap=TRUE to disjoin causes it to return a list of integers, which we use to extract the sample identifiers. The empty case requires a bit more code, because we have to combine the disjoint ranges with the gaps.

Value

A language object containing the compiled R code, evaluating to a GRanges with a column "i" indicating the sample memberships.

Author(s)

Michael Lawrence

References

http://bedtools.readthedocs.io/en/latest/content/tools/multiinter.html

See Also

disjoin for forming disjoint ranges.

Examples

```
## Not run:
setwd(system.file("unitTests", "data", "multiinter", package="HelloRanges"))

## End(Not run)
## default behavior
bedtools_multiinter("-i a.bed,b.bed,c.bed")
## custom names
bedtools_multiinter("-i a.bed,b.bed,c.bed -names A,B,C")
## include empty regions, i.e., partition the genome
bedtools_multiinter("-i a.bed,b.bed,c.bed -names A,B,C -empty -g test.genome")
```

---

bedtools_nuc *bedtools_nuc*

Description

Summarize DNA sequences over the specified ranges.

Usage

```
bedtools_nuc(cmd = "--help")
R_bedtools_nuc(fi, bed, s = FALSE, pattern = NULL, fullHeader = FALSE)
do_bedtools_nuc(fi, bed, s = FALSE, pattern = NULL, fullHeader = FALSE)
```

Arguments

cmd	String of bedtools command line arguments, as they would be entered at the shell. There are a few incompatibilities between the **docopt** parser and the bedtools style. See argument parsing.
fi	Path to a FASTA file, or an XStringSet.
bed	Path to a BAM/BED/GFF/VCF/etc file, a BED stream, a file object, or a ranged data structure, such as a GRanges, as the query. Use "stdin" for input from another process (presumably while running via Rscript). For streaming from a subprocess, prefix the command string with "<", e.g., "<grep foo file.bed". Any streamed data is assumed to be in BED format.
s	Force strandedness. If the feature occupies the antisense strand, the sequence will be reverse complemented.
pattern	Optional sequence pattern to count in each subsequence.
fullHeader	Use the full FASTA header as the names. By default, use just the first word.

Details

As with all commands, there are three interfaces to the nuc command:

bedtools_nuc Parses the bedtools command line and compiles it to the equivalent R code.

R_bedtools_nuc Accepts R arguments corresponding to the command line arguments and compiles the equivalent R code.

do_bedtools_nuc Evaluates the result of R_bedtools_nuc. Recommended **only** for demonstration and testing. It is best to integrate the compiled code into an R script, after studying it.

Computes AT/GC percentage and counts each type of base. Relies on Biostrings utilities like letterFrequency and alphabetFrequency. The counting of pattern occurrences uses vcountPattern.

Value

A language object containing the compiled R code, evaluating to a DataFrame with summary statistics including the AC and GT percentage, and the counts of each type of base. Also includes the count of pattern, if specified.

Author(s)

Michael Lawrence

References

http://bedtools.readthedocs.io/en/latest/content/tools/nuc.html

See Also

letterFrequency for summarizing sequences, matchPattern for pattern matching.

Examples

```
## Not run:
setwd(system.file("unitTests", "data", "nuc", package="HelloRanges"))

## End(Not run)
    ## default behavior, note the two dashes in '--fi'
    bedtools_nuc("--fi test.fasta -bed a.bed")
    ## with pattern counting
    bedtools_nuc("--fi test.fasta -bed a.bed -pattern ATA")
```

`bedtools_shift` *bedtools_shift*

Description

Compute shifting regions.

Usage

```
bedtools_shift(cmd = "--help")
R_bedtools_shift(i, s = 0, m = 0, p = 0, pct = FALSE, g = NULL, header = FALSE)
do_bedtools_shift(i, s = 0, m = 0, p = 0, pct = FALSE, g = NULL, header = FALSE)
```

Arguments

`cmd`	String of bedtools command line arguments, as they would be entered at the shell. There are a few incompatibilities between the **docopt** parser and the bedtools style. See argument parsing.
`i`	Path to a BAM/BED/GFF/VCF/etc file, a BED stream, a file object, or a ranged data structure, such as a GRanges. Use `"stdin"` for input from another process (presumably while running via `Rscript`). For streaming from a subprocess, prefix the command string with "<", e.g., `"<grep foo file.bed"`. Any streamed data is assumed to be in BED format.
`s`	Amount to shift all features.
`m`	Amount to shift negative strand features.
`p`	Amount to shift positive strand features.
`pct`	Define `l` and `r` as a fraction of the feature length. E.g. if used on a 1000bp feature, and `l` is 0.50, will shift 500 bp upstream..
`g`	Genome file, identifier or Seqinfo object that defines the order and size of the sequences.
`header`	Ignored.

Details

As with all commands, there are three interfaces to the `shift` command:

`bedtools_shift` Parses the bedtools command line and compiles it to the equivalent R code.

`R_bedtools_shift` Accepts R arguments corresponding to the command line arguments and compiles the equivalent R code.

`do_bedtools_shift` Evaluates the result of `R_bedtools_shift`. Recommended **only** for demonstration and testing. It is best to integrate the compiled code into an R script, after studying it.

This is a fairly straight-forward application of `shift`.

Value

A language object containing the compiled R code, evaluating to a GRanges, or similar, object. In principle, this is an endomorphism.

Author(s)

Michael Lawrence

References

http://bedtools.readthedocs.io/en/latest/content/tools/shift.html

See Also

intra-range-methods for `shift`.

Examples

```
## Not run:
setwd(system.file("unitTests", "data", "shift", package="HelloRanges"))

## End(Not run)
## shift all ranges by 5
bedtools_shift("-i a.bed -s 5 -g tiny.genome")
## shift only the negative strand features by 5
bedtools_shift("-i a.bed -p 0 -m 5 -g tiny.genome")
```

bedtools_slop *bedtools_slop*

Description

Widen ranges on the left and/or right side.

Usage

```
bedtools_slop(cmd = "--help")
R_bedtools_slop(i, b = 0, l = 0, r = 0, s = FALSE, pct = FALSE,
                g = NULL, header = FALSE)
do_bedtools_slop(i, b = 0, l = 0, r = 0, s = FALSE, pct = FALSE,
                 g = NULL, header = FALSE)
```

Arguments

cmd	String of bedtools command line arguments, as they would be entered at the shell. There are a few incompatibilities between the **docopt** parser and the bedtools style. See argument parsing.
i	Path to a BAM/BED/GFF/VCF/etc file, a BED stream, a file object, or a ranged data structure, such as a GRanges. Use "stdin" for input from another process (presumably while running via Rscript). For streaming from a subprocess, prefix the command string with "<", e.g., "<grep foo file.bed". Any streamed data is assumed to be in BED format.
b	Widen the same number base pairs in each direction.
l	The number of base pairs to subtract from the start coordinate.
r	The number of base pairs to add to the end coordinate.
s	Define l and r based on strand. For example. if TRUE, l=500 for a negative-stranded feature will add 500 bp to the end coordinate.
pct	Define l and r as a fraction of the feature length. E.g. if used on a 1000bp feature, and l is 0.50, will add 500 bp upstream.
g	Genome file, identifier or Seqinfo object that defines the order and size of the sequences.
header	Ignored.

Details

As with all commands, there are three interfaces to the slop command:

bedtools_slop Parses the bedtools command line and compiles it to the equivalent R code.

R_bedtools_slop Accepts R arguments corresponding to the command line arguments and compiles the equivalent R code.

do_bedtools_slop Evaluates the result of R_bedtools_slop. Recommended **only** for demonstration and testing. It is best to integrate the compiled code into an R script, after studying it.

This is a fairly straight-forward application of resize and the + operator on GRanges.

Value

A language object containing the compiled R code, evaluating to a GRanges, or similar, object. In principle, this is an endomorphism.

Author(s)

Michael Lawrence

References

http://bedtools.readthedocs.io/en/latest/content/tools/slop.html

See Also

intra-range-methods for resize.

Examples

```
## Not run:
setwd(system.file("unitTests", "data", "slop", package="HelloRanges"))

## End(Not run)
## widen on both ends
bedtools_slop("-i a.bed -b 5 -g tiny.genome")
## widen only on the left end
bedtools_slop("-i a.bed -l 5 -r 0 -g tiny.genome")
```

---

bedtools_subtract *bedtools_subtract*

Description

Subtracts one set of ranges from another, either by position or range.

Usage

```
bedtools_subtract(cmd = "--help")
R_bedtools_subtract(a, b, f = 1e-09, F = 1e-09, r = FALSE, e = FALSE,
                    s = FALSE, S = FALSE, A = FALSE, N = FALSE,
                    g = NA_character_)
do_bedtools_subtract(a, b, f = 1e-09, F = 1e-09, r = FALSE, e = FALSE,
                     s = FALSE, S = FALSE, A = FALSE, N = FALSE,
                     g = NA_character_)
```

Arguments

cmd	String of bedtools command line arguments, as they would be entered at the shell. There are a few incompatibilities between the **docopt** parser and the bedtools style. See argument parsing.
a	Path to a BAM/BED/GFF/VCF/etc file, a BED stream, a file object, or a ranged data structure, such as a GRanges. Each feature in a is compared to b in search of overlaps. Use "stdin" for input from another process (presumably while running via Rscript). For streaming from a subprocess, prefix the command string with "<", e.g., "<grep foo file.bed". Any streamed data is assumed to be in BED format.
b	Like a, except supports multiple datasets, either as a vector/list or a comma-separated string. Also supports file glob patterns, i.e., strings containing the wildcard, "*".
f	Minimum overlap required as a fraction of a [default: any overlap].
F	Minimum overlap required as a fraction of b [default: any overlap].
r	Require that the fraction of overlap be reciprocal for a and b. In other words, if f is 0.90 and r is TRUE, this requires that b overlap at least 90% of a and that a also overlaps at least 90% of b.
e	Require that the minimum fraction be satisfied for a *OR* b. In other words, if e is TRUE with f=0.90 and F=0.10 this requires that either 90% of a is covered OR 10% of b is covered. If FALSE, both fractions would have to be satisfied.
s	Require same strandedness. That is, find the subtract feature in b that overlaps a on the *same* strand. By default, overlaps are reported without respect to strand. Note that this is the exact opposite of Bioconductor behavior.
S	Require opposite strandedness. That is, find the subtract feature in b that overlaps a on the *opposite* strand. By default, overlaps are reported without respect to strand.
A	Remove entire feature if any overlap. If a feature in a overlaps one in b, the entire feature is removed.
N	Same as A=TRUE except when considering f the numerator in the fraction is the sum of the overlap for all overlapping features in b.
g	A genome file, identifier or Seqinfo object that defines the order and size of the sequences.

Details

As with all commands, there are three interfaces to the subtract command:

bedtools_subtract Parses the bedtools command line and compiles it to the equivalent R code.

R_bedtools_subtract Accepts R arguments corresponding to the command line arguments and compiles the equivalent R code.

do_bedtools_subtract Evaluates the result of R_bedtools_subtract. Recommended **only** for demonstration and testing. It is best to integrate the compiled code into an R script, after studying it.

We typically subtract sets of ranges using `setdiff`; however, that will not work here, because we cannot merge the ranges in `a`.

The algorithm has two modes: by position (where ranges are clipped) and by range (where ranges are discarded entirely). The position mode is the default. We find overlaps, optionally restrict them, and for each range in `a`, we subtract all of the qualifying intersections in `b`.

When `A` or `N` are `TRUE`, we use the second mode. In the simplest case, that is just `subsetByOverlaps` with `invert=TRUE`, but fractional overlap restrictions and `N` make that more complicated.

Value

A language object containing the compiled R code, evaluating to a GRanges object, except when `A` or `N` are `TRUE`, where the value might be a GRanges, GAlignments or VCF object, depending on the input.

Author(s)

Michael Lawrence

References

http://bedtools.readthedocs.io/en/latest/content/tools/subtract.html

See Also

setops-methods for set operations including setdiff, findOverlaps-methods for different ways to detect overlaps.

Examples

```
## Not run:
setwd(system.file("unitTests", "data", "subtract", package="HelloRanges"))

## End(Not run)

## simple case, position-wise subtraction
bedtools_subtract("-a a.bed -b b.bed")
## fractional overlap restriction
bedtools_subtract("-a a.bed -b b.bed -f 0.5")
## range-wise subtraction
bedtools_subtract("-a a.bed -b b.bed -A -f 0.5")
```

`bedtools_unionbedg` *bedtools_unionbedg*

Description

Summarize the ranges according to `disjoin` and construct a matrix of scores (disjoint range by sample/file). Empty cells are filled with `NA`.

Usage

```
bedtools_unionbedg(cmd = "--help")
R_bedtools_unionbedg(i, header=FALSE, names=NULL, g=NA_character_,
                     empty=FALSE)
do_bedtools_unionbedg(i, header=FALSE, names=NULL, g=NA_character_,
                      empty=FALSE)
```

Arguments

cmd	String of bedtools command line arguments, as they would be entered at the shell. There are a few incompatibilities between the **docopt** parser and the bedtools style. See argument parsing.
i	Paths to BAM/BED/GFF/VCF/etc files (vector or comma-separated), or a list of objects.
header	Ignored.
names	Provide an alias for each to use for each i instead of their integer index. If a single string, can be comma-separated.
g	A genome file, identifier or Seqinfo object that defines the order and size of the sequences.
empty	Report empty regions (i.e., regions not covered in any of the files). This essentially yields a partitioning of the genome (and thus requires g to be specified).

Details

As with all commands, there are three interfaces to the unionbedg command:

bedtools_unionbedg Parses the bedtools command line and compiles it to the equivalent R code.

R_bedtools_unionbedg Accepts R arguments corresponding to the command line arguments and compiles the equivalent R code.

do_bedtools_unionbedg Evaluates the result of R_bedtools_unionbedg. Recommended **only** for demonstration and testing. It is best to integrate the compiled code into an R script, after studying it.

This is essentially the same operation as bedtools_multiinter, except we build a score matrix and embed it into a SummarizedExperiment. This is a bit tricky and relies on the as.matrix,AtomicList-method coercion.

Value

A language object containing the compiled R code, evaluating to a RangedSummarizedExperiment with an assay called "score".

Author(s)

Michael Lawrence

References

http://bedtools.readthedocs.io/en/latest/content/tools/unionbedg.html

See Also

disjoin for forming disjoint ranges, RangedSummarizedExperiment-class for SummarizedExperiment objects.

Examples

```
## Not run:
setwd(system.file("unitTests", "data", "unionbedg", package="HelloRanges"))

## End(Not run)

## combine three samples
bedtools_unionbedg("-i a.bedGraph,b.bedGraph,c.bedGraph -names A,B,C")
## include empty ranges (filled with NAs)
bedtools_unionbedg("-i a.bedGraph,b.bedGraph,c.bedGraph -names A,B,C -empty -g test.genome")
```

`distmode` *Compute the mode of a distribution*

Description

Computes the mode (and "antimode") of a distribution. It is not clear whether this is a generally useful statistic, but `bedtools` defined it, so we did for completeness.

Usage

```
distmode(x, anti = FALSE)
```

Arguments

`x`	The vector for which the mode is computed.
`anti`	Whether to return the value with the least representation, instead of the highest.

Details

There are methods for List subclasses and ordinary vectors/factors. The List methods are useful for aggregation.

Value

The value that is the most (or least) prevalent in the `x`.

Author(s)

Michael Lawrence

See Also

Not to be confused with the data mode of a vector.

bedtools_map for an example in the context of aggregation.

Examples

```
distmode(c(1L, 2L, 1L))
```

---

pair *Pair up two vectors*

Description

Creates a Pairs from two vectors, optionally via a left outer join.

Usage

```
pair(x, y, ...)
## S4 method for signature 'Vector,Vector'
pair(x, y, by = findMatches(x, y), all.x = FALSE,
                   NA.VALUE = y[NA])
```

Arguments

x	The "first" vector.
y	The "second" vector.
by	The Hits-class object that matches up the elements into pairs.
all.x	If TRUE, keep every member of x, even if it has no hits. The "second" component is filled with the NA.VALUE.
NA.VALUE	The value to fill holes in y when all.x is TRUE.
...	Arguments for methods.

Details

This might move to **S4Vectors** at some point. It is distinct from simple Pairs construction, because it performs transformations like a left outer join. More options might be added in the future.

For GRanges and other ranged objects, pair adds "." to the seqlevels, because that is the seqname of the missing GRanges.

Value

A Pairs object

Author(s)

Michael Lawrence

See Also

Pairs-class, created by this function. bedtools_intersect, whose loj argument motivated the creation of this function.

Index

aggregate, *16*, *26*, *28*
aggregate-methods, *26*, *28*
alphabetFrequency, *31*
argparsing, 2
argument parsing, *3*, *6, 7*, *10*, *12*, *14, 15*, *18*, *21*, *23*, *25*, *27*, *29*, *31, 32*, *34*, *36*, *38*

bedtools_closest, 3
bedtools_complement, 6
bedtools_coverage, 7
bedtools_flank, 10
bedtools_genomecov, 11
bedtools_getfasta, 14
bedtools_groupby, 15, *25*, *27, 28*
bedtools_intersect, 18, *41*
bedtools_jaccard, 20
bedtools_makewindows, 23
bedtools_map, 24, *40*
bedtools_merge, 27
bedtools_multiinter, 29, *38*
bedtools_nuc, 30
bedtools_shift, 32
bedtools_slop, 34
bedtools_subtract, 35
bedtools_unionbedg, 37

coverage, *8*, *13*
coverage-methods, *9*

disjoin, *29, 30*, *37*, *39*
distmode, *16*, 39
distmode,CompressedAtomicList-method *(*distmode*)*, 39
distmode,factor-method *(*distmode*)*, 39
distmode,SimpleList-method *(*distmode*)*, 39
distmode,vector-method *(*distmode*)*, 39
do_bedtools_closest *(*bedtools_closest*)*, 3
do_bedtools_complement *(*bedtools_complement*)*, 6
do_bedtools_coverage *(*bedtools_coverage*)*, 7
do_bedtools_flank *(*bedtools_flank*)*, 10
do_bedtools_genomecov *(*bedtools_genomecov*)*, 11
do_bedtools_getfasta *(*bedtools_getfasta*)*, 14
do_bedtools_groupby *(*bedtools_groupby*)*, 15
do_bedtools_intersect *(*bedtools_intersect*)*, 18
do_bedtools_jaccard *(*bedtools_jaccard*)*, 20
do_bedtools_makewindows *(*bedtools_makewindows*)*, 23
do_bedtools_map *(*bedtools_map*)*, 24
do_bedtools_merge *(*bedtools_merge*)*, 27
do_bedtools_multiinter *(*bedtools_multiinter*)*, 29
do_bedtools_nuc *(*bedtools_nuc*)*, 30
do_bedtools_shift *(*bedtools_shift*)*, 32
do_bedtools_slop *(*bedtools_slop*)*, 34
do_bedtools_subtract *(*bedtools_subtract*)*, 35
do_bedtools_unionbedg *(*bedtools_unionbedg*)*, 37

findOverlaps, *26*
findOverlaps-methods, *20*, *26*, *37*
flank, *11*
follow, *4*

gaps, *6*
getSeq, *14, 15*

Hits, *26*
Hits-class, *26*

import, *14*

intersect, *22*
intra-range-methods, *11*, *13*, *33*, *35*

letterFrequency, *31*

matchPattern, *31*
mode, *40*

nearest, *4*
nearest-methods, *5*

pair, 40
pair,GAlignments,GenomicRanges-method *(pair)*, 40
pair,GenomicRanges,GenomicRanges-method *(pair)*, 40
pair,SummarizedExperiment,GenomicRanges-method *(pair)*, 40
pair,Vector,Vector-method *(pair)*, 40
Pairs, *40*
pipe, *2*
precede, *4*

R_bedtools_closest *(bedtools_closest)*, 3
R_bedtools_complement *(bedtools_complement)*, 6
R_bedtools_coverage *(bedtools_coverage)*, 7
R_bedtools_flank *(bedtools_flank)*, 10
R_bedtools_genomecov *(bedtools_genomecov)*, 11
R_bedtools_getfasta *(bedtools_getfasta)*, 14
R_bedtools_groupby *(bedtools_groupby)*, 15
R_bedtools_intersect *(bedtools_intersect)*, 18
R_bedtools_jaccard *(bedtools_jaccard)*, 20
R_bedtools_makewindows *(bedtools_makewindows)*, 23
R_bedtools_map *(bedtools_map)*, 24
R_bedtools_merge *(bedtools_merge)*, 27
R_bedtools_multiinter *(bedtools_multiinter)*, 29
R_bedtools_nuc *(bedtools_nuc)*, 30
R_bedtools_shift *(bedtools_shift)*, 32
R_bedtools_slop *(bedtools_slop)*, 34
R_bedtools_subtract *(bedtools_subtract)*, 35
R_bedtools_unionbedg *(bedtools_unionbedg)*, 37
RangedSummarizedExperiment-class, *39*
readDNAStringSet, *14*
reduce, *27*, *28*
resize, *35*

setdiff, *6*, *37*
setops-methods, *7*, *20*, *22*, *37*
shift, *33*
slidingWindows, *23*
subsetByOverlaps, *37*

tile, *23*
tile-methods, *24*

union, *22*

vcountPattern, *31*