Package 'bedr'

October 12, 2022

Type Package

Title Genomic Region Processing using Tools Such as 'BEDTools', 'BEDOPS' and 'Tabix'

Version 1.0.7

Date 2019-04-01

Author Syed Haider [aut],
Daryl Waggott [aut],
Emilie Lalonde [ctb],
Clement Fung [ctb],
Paul C. Boutros [aut, cre, cph]

Maintainer Paul C. Boutros <pboutros@mednet.ucla.edu>

Description Genomic regions processing using open-source command line tools such as 'BEDTools', 'BEDOPS' and 'Tabix'.
These tools offer scalable and efficient utilities to perform genome arithmetic e.g indexing, formatting and merging.
bedr API enhances access to these tools as well as offers additional utilities for genomic regions processing.

Depends R (>= 3.0)

SystemRequirements Preferred genomic operations engine: 'BEDTools', 'BEDOPS' and 'Tabix (>= 1.3)'.

Suggests knitr (>= 1.4), rmarkdown (>= 0.9.5)

VignetteBuilder knitr

License GPL-2

Imports testthat (>= 0.7.1), VennDiagram (>= 1.6.4), data.table (>= 1.8.11), R.utils (>= 2.0.2), yaml (>= 2.1.10), parallel, grid

NeedsCompilation no

Repository CRAN

Date/Publication 2019-04-01 18:50:02 UTC

R topics documented:

bedr-package . . . 3
bed2index . . . 3
bed2vcf . . . 4
bedr . . . 5
bedr.join.multiple.region . . . 8
bedr.join.region . . . 9
bedr.merge.region . . . 11
bedr.plot.region . . . 12
bedr.setup . . . 14
bedr.snm.region . . . 14
bedr.sort.region . . . 16
bedr.subtract.region . . . 17
catv . . . 18
check.binary . . . 19
cluster.region . . . 20
convert2bed . . . 21
create.tmp.bed.file . . . 22
determine.input . . . 23
df2list . . . 23
download.datasets . . . 24
flank.region . . . 25
get.chr.length . . . 26
get.example.regions . . . 27
get.fasta . . . 27
get.random.regions . . . 29
grow.region . . . 31
in.region . . . 32
index2bed . . . 34
is.merged.region . . . 34
is.sorted.region . . . 35
is.valid.ref . . . 36
is.valid.region . . . 37
is.valid.seq . . . 38
jaccard . . . 40
modifyList2 . . . 41
order.region . . . 42
permute.region . . . 43
process.input . . . 45
query.ucsc . . . 46
read.vcf . . . 47
reldist . . . 48
size.region . . . 50
strsplit2matrix . . . 51
tabix . . . 51
table2venn . . . 53
test.region.similarity . . . 54

vcf2bed 56
write.vcf 57
%in.region% 58

Index **60**

`bedr-package` *A bedtools wrapper for working with genomic ranges in R*

Description

A bedtools wrapper that allows using a mix of internal R objects and external R files. A number of convenience functions are provided for simplifying analysis workflows in R.

Details

Package:	bedr
Type:	Package
Version:	1.0.4
Date:	2016-09-19
License:	GPL2

Author(s)

Daryl Waggott
Syed Haider

`bed2index` *bed dataframe to index string*

Description

convert a dataframe in bed format to an index string

Usage

```
bed2index(x, sort = TRUE)
```

Arguments

`x`	a object region object or index
`sort`	should the index be sorted

Value

Returns a vector of string based genomic regions

Author(s)

Daryl Waggott

Examples

```
test.regions <- get.random.regions(10);
bed2index(test.regions);
```

`bed2vcf` *convert bed to vcf*

Description

convert bed to vcf

Usage

```
bed2vcf(x, filename = NULL, zero.based = TRUE, header = NULL, fasta = NULL)
```

Arguments

`x`	the input bed object
`filename`	a filename to write to.
`zero.based`	is the file zero based i.e. bed format. defaults to true.
`header`	a list of things to put in the header. repeated elements such as INFO, FILTER, FORMAT can be put in data.frames.
`fasta`	build of the genome in fasta format

Author(s)

Daryl Waggott

Examples

```
## Not run:
  bed2vcf(x)

## End(Not run)
```

bedr *Main bedtools wrapper function.*

Description

Main bedtools wrapper function.

Usage

```
bedr(engine = "bedtools",
params = NULL,
input = list(),
method = NULL,
tmpDir = NULL,
deleteTmpDir = TRUE,
outputDir = NULL,
outputFile = NULL,
check.chr = TRUE,
check.zero.based = TRUE,
check.valid = TRUE,
check.sort = TRUE,
check.merge = TRUE,
verbose = TRUE
);
```

Arguments

engine	What analytical engine to use i.e. bedtools, bedops, tabix, unix
params	A string that includes all the extra parameters and arguments to the bedtools commmand. For example if you wanted to do a left outer join you would specificy method as intersect and use params = c("-loj -header"). If you leave input and method as defaults then this is this string represents the full command.
input	A list of input items to be used by bedtools. Each item should be named by its parameter name in bedtools for example input = list(a=xxx, b=yyy, i=zzz). Items can be R objects or external files. R objects need to be in bed format i.e. have chr, start, stop as the first three columns, or, have an position index as the first column or rowname i.e. chr1:100-1000.
method	What bedtools method. This can be intersect, sort, merge etc. See bedtools documentation for specifcis.
tmpDir	The directory to be used for writing files
deleteTmpDir	Should tmp files be deleted. helpful for diagnostics.
outputDir	The output directory. Only used if outputFile is specified. It defaults to the current working directory.

outputFile	The name of the output file. If this is specified the output will be sent to a file not an R object
check.chr	check for chr prefix
check.zero.based	check for zero based coordinates
check.valid	do all region integrity checks
check.sort	check if region is sorted
check.merge	check if region is merged
verbose	Should messages be printed to screen.

Value

The output of command with some parsing to keep it consistent with the input.

Author(s)

Daryl Waggott

See Also

iranges

Examples

```
if (check.binary("bedtools")) {
set.seed(666)

index <- get.example.regions();
a <- index[[1]];
b <- index[[2]];

### check
is.a.valid  <- is.valid.region(a);
is.b.valid  <- is.valid.region(b);
a <- a[is.a.valid];
b <- b[is.b.valid];

### sort
is.sorted <- is.sorted.region(a);
a.sort1 <- bedr(engine = "bedtools", input = list(i = a), method = "sort", params = "");
b.sort1 <- bedr(engine = "bedtools", input = list(i = b), method = "sort", params = "");
a.sort2 <- bedr(engine = "bedops",   input = list(i = a), method = "sort", params = "");

a.sort3 <- bedr.sort.region(a);
a.sort4 <- bedr.sort.region(a, engine = "unix", method = "natural");
a.sort5 <- bedr.sort.region(a, engine = "R", method = "natural");

### merge
is.merged <- is.merged.region(a.sort1);
a.merge1 <- bedr(engine = "bedtools", input = list(i = a.sort1), method = "merge", params = "");
```

```
b.merge1 <- bedr(engine = "bedtools", input = list(i = b.sort1), method = "merge", params = "");
a.merge2 <- bedr(engine = "bedops",  input = list(i = a.sort1), method = "merge", params = "");
# a.merge3 <- bedr.merge.region(a); this will throw an error b/c region is not sorted

### subtract
a.sub1 <- bedr(input = list(a = a.merge1, b = b.merge1), method = "subtract", params="");
a.sub2 <- bedr.subtract.region(a.merge1, b.merge1);

### in.region
is.region <- in.region(a.merge1, b.merge1);
#is.region <- a.merge1 %in.region% b.merge1
### intersect
# note for bedtools its recommended to bedr.sort.regions before intersect for faster processing
# also if regions are not merged this can cause unexpected behaviour
a.int1 <- bedr(input = list(a = a.sort1, b = b.sort1), method = "intersect", params = "-loj");
a.int1 <- bedr(input = list(a = a.sort1, b = b.sort1), method="intersect",params ="-loj -sorted");
a.int2 <- bedr(input = list(a = a.merge1, b = b.merge1), method="intersect",params="-loj -sorted");
a.int3 <- bedr.join.region(a.merge1, b.merge1);

### multiple join
d <- get.random.regions(100, chr="chr1", sort = TRUE);
a.mult <- bedr.join.multiple.region(x = list(a.merge1,b.merge1,bedr.sort.region(d)));

## Not run:
  ### groupby
 # note the "g" column number is based on bed format i.e. first three columns chr, start, stop
  # note the use of first, first, last on the region columns i.e. the union of the regions
  # note currently missing values are not dealt with in bedtools.  also the 5th column is
  # assumed to be "score" and gets a default "-1" not a "."
  cnv.gene <- bedr(
    input = list(i=cnv.gene), method = "groupby", params = paste(
      "-g 16 -c ",
      paste(1:15, collapse = ","),
      " -o ", "first,first,last, ",
      paste(rep("sum",12), collapse = ","),
      sep = ""
      )
    );

  ### example 1
  ###  workflow adding gene names to exome sequencing target file
  # download refseq genes from ucsc or query biomart for ensemble gene names.
  # format them in basic bed format.
  # sort, merge target
  # sort, merge -nms target.  Overlapping genes/features get merged.
  # this may not be ideal if there are some really big features.
  # intersect -loj target, genes.
  # alternatively, do not merge the target and apply the merge after the intersect.
  # this will provide precision at the level of the exon.

## End(Not run)
}
```

`bedr.join.multiple.region` *join multiple region objects*

Description

join multiple objects

Usage

```
bedr.join.multiple.region(
x = list(),
fraction.overlap = 1/1e9,
empty = FALSE,
missing.values = ".",
cluster = FALSE,
species = "human",
build = "hg19",
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
check.sort = TRUE,
check.merge = TRUE,
verbose = TRUE
)
```

Arguments

`x`	list of region objects
`fraction.overlap`	proportion of bases to be considered an overlap
`empty`	print rows if no match
`missing.values`	missing value character
`cluster`	TRUE/FALSE for clustering
`species`	species i.e. human or mouse
`build`	genome build to use for empty regions
`check.zero.based`	should 0 based coordinates be checked
`check.chr`	should chr prefix be checked
`check.valid`	check if region is valid
`check.sort`	check if region is sorted
`check.merge`	check if overlapping regions are merged
`verbose`	messages and checks

Author(s)

Daryl Waggott

References

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

Examples

```
if (check.binary("bedtools")) {

index <- get.example.regions();

a <- index[[1]];
b <- index[[2]];

a.sort <- bedr.sort.region(a);
b.sort <- bedr.sort.region(b);
d <- get.random.regions(100, chr="chr1", sort = TRUE);

a.mult <- bedr.join.multiple.region(x = list(a.sort,b.sort,bedr.sort.region(d)));
}
```

---

bedr.join.region *join two region objects using a left outer join*

Description

join two region objects using a left outer join

Usage

```
bedr.join.region(
x,
y,
fraction.overlap = 1/1e9,
reciporical = FALSE,
report.n.overlap = FALSE,
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
check.sort = TRUE,
check.merge = TRUE,
verbose = TRUE
)
```

Arguments

x	object a
y	object b
fraction.overlap	proportion of overlap to be considered a match
report.n.overlap	should the number of overlapping bases be reported
reciporical	should the fraction overlap be applied to object b as well
check.zero.based	should 0 based coordinates be checked
check.chr	should chr prefix be checked
check.valid	check if region is valid
check.sort	check if region is sorted
check.merge	check if overlapping regions are merged
verbose	messages and checks

Author(s)

Daryl Waggott

References

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

Examples

```
if (check.binary("bedtools")) {

index <- get.example.regions();

a <- index[[1]];
b <- index[[2]];

a.sort <- bedr.sort.region(a);
b.sort <- bedr.sort.region(b);

d <- bedr.join.region(a.sort, b.sort);
}
```

bedr.merge.region *merge i.e. collapse overlpaping regions*

Description

merge i.e. collapse overlpaping regions

Usage

```
bedr.merge.region(
x,
distance = 0,
list.names = TRUE,
number = FALSE,
stratify.by = NULL,
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
check.sort = TRUE,
verbose = TRUE
);
```

Arguments

x	input
distance	maximum distance between regions to be merged. defaults to 0 which means overlapping or bookended features. note that you can use negative distances to enforce a minimum overlap.
list.names	output list of names for merged items
number	output number of merged items
stratify.by	a column name indicating the groups to stratify merging within i.e. gene name. merging will not happen between groups.
check.zero.based	should 0 based coordinates be checked
check.chr	should chr prefix be checked
check.valid	should the region be checkded for integerity
check.sort	should the sort order be checked
verbose	should log messages and checking take place

Author(s)

Daryl Waggott

References

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

Examples

```
if (check.binary("bedtools")) {

index <- get.example.regions();

a <- index[[1]];

a.sort   <- bedr.sort.region(a);
a.merged <- bedr.merge.region(a.sort);

}
```

`bedr.plot.region` *Visualize regions or intervals*

Description

Visualize regions or intervals. e.g. VennDiagrams of intersections of distinct intervals, base pairs and genes.

Usage

```
bedr.plot.region(
input,
filename = NULL,
type = "venn",
feature = "interval",
fraction.overlap = 0.000000001,
group = NULL,
params = list(),
verbose = TRUE
)
```

Arguments

`input`	A list of input regions or indices
`filename`	The name of the output image file
`type`	The type of plot. only 'venn' is supported for intersections at the moment.
`feature`	How should the regions be intersected. By unique "interval", "gene", "size" or "other" to use the features in the first item in the input list.
`fraction.overlap`	Minimum overlap required as a fraction of A. Default is 1E-9 (i.e. 1bp).
`group`	A grouping parameter for barplots. Possible values include "input", "chr", or a categorical vector of length equal to the sum of the input.
`params`	Additional parameters for plotting or intersecting. See `venn.diagram` function for possible options.
`verbose`	Include text messages.

Details

By default a venn diagram is output. If a single input is used then the plot shows the number of unique and collapsed regions after applying a merge.

Value

Plots!

Author(s)

Daryl Waggott

Examples

```
if (check.binary("bedtools")) {
# example data
a <- get.random.regions(n = 1000, chr = "chr22", size.mean = 10)
b <- get.random.regions(n = 1000, chr = "chr22", size.mean = 10)
d <- get.random.regions(n = 1000, chr = "chr22", size.mean = 10)
e <- get.random.regions(n = 1000, chr = "chr22", size.mean = 10)
f <- get.random.regions(n = 1000, chr = "chr22", size.mean = 10)

pdf("bedr.plot.region.ex.pdf")

# basic venn diagrams
bedr.plot.region(input = list(a=a,b=b))
bedr.plot.region(input = list(a=a,b=b,d=d))
#bedr.plot.region(input = list(a=a,b=b,d=d,e=e))
#bedr.plot.region(input = list(a=a,b=b,d=d,e=e,f=f))

### change venn parameters
bedr.plot.region(
input = list(a=a,b=b,d=d),
params = list(lty = 2, label.col = "black", main = "Region Overlap")
)

### try with different
#bedr.plot.region(input = list(a=a,b=b), feature = "gene")
#bedr.plot.region(input = list(a=a,b=b), feature = "reference")
#bedr.plot.region(input = list(a=a,b=b), feature = "interval")
#bedr.plot.region(input = list(a=a,b=b), feature = "cluster")
#bedr.plot.region(input = list(a=a,b=b), feature = "bp")

dev.off()
}
```

`bedr.setup` *Initialize some config settings for bedr*

Description

Initialize some config settings for bedr. This includes downloading useful datasets if requested.

Usage

```
bedr.setup(datasets = "all", data.dir = paste0(Sys.getenv("HOME"), "/bedr/data"))
```

Arguments

`datasets`	A list of datasets to download. Defaults to "all" i.e. c("refgene","hg19","b37","hugo", "cosmic","clinvar")
`data.dir`	A directory to put the data. Defaults to ~/bedr/data

Details

The default config file is at ~/bedr/config.yml. It's in yaml format.

Author(s)

Daryl Waggott

Examples

```
## Not run:
  bedr.setup()

## End(Not run)
```

`bedr.snm.region` *sort a region file*

Description

Sort and merge regions object in one step

Usage

```
bedr.snm.region(
x,
method = "lexicographical",
distance = 0,
list.names = TRUE,
number = FALSE,
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
verbose = TRUE
)
```

Arguments

x	a object region object or index
method	natural or lexicographic
distance	distance between regions to be merged
list.names	output list of names for merged items
number	output number of merged items
check.zero.based	should 0 based coordinates be checked
check.chr	should chr prefix be checked
check.valid	should the region be checkded for integerity
verbose	should log messages and checking take place

Value

Sorted and merged regions object

Author(s)

Daryl Waggott

Examples

```
if (check.binary("bedtools")) {

index <- get.example.regions();

a <- index[[1]];

b <- bedr.snm.region(a);

}
```

bedr.sort.region *sort a region file*

Description

sort a region file

Usage

```
bedr.sort.region(
x,
method = "lexicographical",
engine = "R",
chr.to.num = c("X" = 23, "Y" = 24, "M" = 25),
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
check.merge = TRUE,
verbose = TRUE
)
```

Arguments

x	a region object or index
method	natural or lexicographic
engine	what analytical engine to use for sorting i.e. bedtools, bedops, gnu unix
chr.to.num	chromosome letter names to numbers map. Defaults to Homo sapiens i.e c("X" = 23, "Y" = 24, "M" = 25)
check.zero.based	should 0 based coordinates be checked
check.chr	should chr prefix be checked
check.valid	should the region be checked for integerity
check.merge	should overlapping regions be checked
verbose	should log messages and checking take place

Author(s)

Daryl Waggott

References

http://bedtools.readthedocs.org/en/latest/content/tools/sort.html

Examples

```
if (check.binary("bedtools")) {

index <- get.example.regions();
a <- index[[1]];
b <- bedr.sort.region(a);

}
```

---

bedr.subtract.region *subtracts features or ranges in object b from object a*

Description

subtracts features or ranges in object b from object a

Usage

```
bedr.subtract.region(
x,
y,
fraction.overlap = 1/1e9,
remove.whole.feature = TRUE,
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
check.sort = TRUE,
check.merge = TRUE,
verbose = TRUE
)
```

Arguments

x	item a
y	item b
fraction.overlap	what portion of A to be considered an overlap
remove.whole.feature	should whole feature be removed
check.zero.based	should 0 based coordinates be checked
check.chr	should chr prefix be checked
check.valid	should the region be checkded for integerity
check.sort	check if region is sorted
check.merge	check if overlapping regions are merged
verbose	messages and checks

Value

Regions exclusive to one object of regions.

Author(s)

Daryl Waggott

References

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

Examples

```
if (check.binary("bedtools")) {
index <- get.example.regions();

a <- index[[1]];
b <- index[[2]];
a <- bedr(engine = "bedtools", input = list(i = a), method = "sort", params = "");
b <- bedr(engine = "bedtools", input = list(i = b), method = "sort", params = "");
d <- bedr.subtract.region(a,b);
}
```

---

catv *outputs text if verbose flag is set*

Description

outputs text if verbose flag is set

Usage

```
catv(x)
```

Arguments

x	some text string

Value

Prints the text string

Author(s)

Daryl Waggott

Examples

```
verbose <- TRUE;
catv("Hello Universe!");
verbose <- FALSE;
catv("Goodbye Universe!")
```

---

check.binary *checks if binary is in the path*

Description

check if a binary is in the path. Specifically used for bedtools, bedops and tabix.

Usage

```
check.binary(x = "bedtools", verbose = TRUE)
```

Arguments

x	a string referring to a binary/executable i.e. bedtools, bedops, tabix
verbose	print log

Details

Used internally to determine functionality selection options.

Value

TRUE or FALSE

Author(s)

Daryl Waggott

Examples

```
check.binary("bedtools")
```

`cluster.region` *cluster intervals*

Description

cluster intervals

Usage

```
cluster.region(
x,
distance = 0,
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
check.sort = TRUE,
verbose = TRUE
)
```

Arguments

`x`	The region
`distance`	maximum distance between regions to be merged. defaults to 0 which means overlapping or bookended features. note that you can use negative distances to enforce a minimum overlap.
`check.zero.based`	should 0 based coordinates be checked
`check.chr`	should chr prefix be checked
`check.valid`	should the region be checkded for integerity
`check.sort`	should regions be checked for sort order
`verbose`	should log messages and checking take place

Details

clusters adjacent features of a specified distance.

Value

A data.frame in bed format

Author(s)

Daryl Waggott

References

http://bedtools.readthedocs.org/en/latest/content/tools/cluster.html

See Also

bedr.merge.region

Examples

```
if (check.binary("bedtools")) {

index <- get.example.regions();

a <- index[[1]];

b <- cluster.region(a, distance = 0);
d <- cluster.region(a, distance = 100);

}
```

---

convert2bed *convert object to bed format*

Description

checks if an object can be converted into a bed style data.frame then does the conversion.

Usage

```
convert2bed(
x,
set.type = TRUE,
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
check.sort = TRUE,
check.merge = TRUE,
verbose = TRUE
)
```

Arguments

x	A region index (i.e. chr1:10-100, ...) either as a vector or row.names/first column of a data.frame. Or a data.frame with the first three columns "chr", s"start", "end"
set.type	should the attribute input.type be set. Sometimes it is desirable to avoid setting it when applying intermediate conversion
check.zero.based	should 0 based coordinates be checked
check.chr	should chr prefix be checked

check.valid	should the region be checked for integerity
check.sort	should the region be checked to see if it is sorted
check.merge	should the region be checked for overlapping regions
verbose	messages and text

Details

Very useful to convert data before using other bedr functions

Value

Returns x converted to bedformat, as a data frame

Author(s)

Daryl Waggott

Examples

```
## Not run:
  a.bed <- convert.to.bed(a)

## End(Not run)
```

---

create.tmp.bed.file *output R objects as tmpfiles*

Description

output R objects as tmpfiles

Usage

```
create.tmp.bed.file(x, name = "bedr", tmpDir = NULL)
```

Arguments

x	region object
name	a prefix for the tmp file.
tmpDir	where should the temp files be put

Author(s)

Daryl Waggott

Examples

```
# create tmp file
```

`determine.input` *Determine input format*

Description

Determine input format whether its tabular or bed

Usage

```
determine.input(x, check.chr = FALSE, verbose = TRUE)
```

Arguments

`x`	input vector, matrix or dataframe
`check.chr`	check whether the coordinates are in chromosomal format with chr prefix
`verbose`	messages and checks

Value

integer value. index format (0), bed (1), index in first column (2), rownmames are index (3), unrecognized(4)

Author(s)

Daryl Waggott

Examples

```
if (check.binary("bedtools")) {

index <- get.example.regions();
a <- index[[1]];
bedr:::determine.input(a);
}
```

`df2list` *Data frame to list conversion*

Description

Take data frame and return a list of rownames where column value is not 0 i.e. missing

Usage

```
df2list(x, start.col = 1)
```

Arguments

x	data frame
start.col	offset from first column to ignore certain columns

Value

returns a list following cleanup and change of data structure

Author(s)

Daryl Waggott

Examples

```
## Not run:
  df2list(data.frame(a = 1:10, b = 11:20));

## End(Not run)
```

---

download.datasets *Download some useful datasets*

Description

Download some useful datasets. Some functions such as plotting and fasta querying require additional data.

Usage

```
download.datasets(datasets = "all", data.dir = paste0(Sys.getenv("HOME"), "/bedr/data"))
```

Arguments

datasets	A list of datasets to download. Defaults to "all" i.e. c("refgene","hg19","b37","hugo", "cosmic","clinvar")
data.dir	A directory to put the data. Defaults to ~/bedr/data

Details

External datasets are required for some bedr functionality. For example, plotting intersections based on genes, get.fasta, bed2vcf and validate.gene.names. If these datasets already exist you can simply place symlinks in a directory and use bedr.setup() to define the data.dir.

Value

some datasets.

Author(s)

Daryl Waggott

Examples

```
## Not run:
  download.datasets("cosmic");

## End(Not run)
```

`flank.region` *Get adjacent flanks from regions*

Description

Get adjacent flanks from regions

Usage

```
flank.region(
x,
n.add = NULL,
start.add = NULL,
end.add = NULL,
species = "human",
build = "hg19",
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
check.sort = TRUE,
check.merge = TRUE,
verbose = TRUE
)
```

Arguments

`x`	a object region object or index
`n.add`	the number of bases to be selected from each side of a region
`start.add`	the number of based to be selected from the start of a region
`end.add`	the number of based to be selected from the end of a region
`species`	the species i.e. human or mouse
`build`	the genome build i.e. hg19 or mm10
`check.zero.based`	should 0 based coordinates be checked

check.chr	should chr prefix be checked
check.valid	should the region be checkded for integerity
check.sort	should regions be checked for sort order
check.merge	should overlapping regions be checked
verbose	should log messages and checking take place

Author(s)

Daryl Waggott

References

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

Examples

```
if (check.binary("bedtools")) {

index <- get.example.regions();

a <- index[[1]];
a <- bedr(engine = "bedtools", input = list(i = a), method = "sort", params = "");
b <- flank.region(a, n.add = 20);

}
```

---

get.chr.length *gets the length of each chromosome for a species/build*

Description

Gets the length of each chromosome for a species/build. Choices are human (hg18, hg19, hg38), mouse(mm9, mm10)

Usage

```
get.chr.length(chr = NULL, species = "human", build = "hg19")
```

Arguments

chr	a vector of chromosomes to query. defaults to all.
species	species
build	build

Value

A dataframe with chromosome annotations

Author(s)

Daryl Waggott

Examples

```
size <- get.chr.length();
```

---

get.example.regions *return a set of regions for the examples and unit testing*

Description

return a set of regions for the examples and unit testing

Usage

```
get.example.regions()
```

Value

A list with three example regions

Author(s)

Daryl Waggott

Examples

```
index <- get.example.regions()
```

---

get.fasta *Query fasta sequence*

Description

Query fasta sequence using bedtools get.fasta

Usage

```
get.fasta(
x,
fasta = NULL,
bed12 = FALSE,
strand = FALSE,
output.fasta = FALSE,
use.name.field = FALSE,
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
check.sort = TRUE,
check.merge = TRUE,
verbose = TRUE
)
```

Arguments

x	region or index
fasta	a fasta file defaults to mini example hg19 human
bed12	should bed12 format be used
strand	strand specific i.e. reverse complement negative
output.fasta	output a fasta defaults to a data.frame for easier parsing
use.name.field	should the name field be used for
check.zero.based	check for zero based region
check.chr	check for "chr" prefix
check.valid	check for valid regions i.e. start < end
check.sort	check if region is sorted
check.merge	check if region is merged
verbose	print progress

Details

Uses bedtools getFasta to query a fasta file and load into R as a data.frame for easy parsing.

Note that the hg19 reference genome fasta is large and requires on the order of 4 GB RAM to avoid a segfault happens.

Value

A data.frame or fasta. The data.frame has is two columns corresponding to the region and the sequence.

Author(s)

Daryl Waggott, Syed Haider

References

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

Examples

```
if (check.binary("bedtools")) {

## Not run:

  # get the sequence for a set of regions as a data.frame
  index <- get.example.regions();
  a <- index[[1]];
  b <- get.fasta(bedr.sort.region(a));

  # this time output a fasta
  d <- get.fasta(b, output.fasta = TRUE);
  writeLines(d[[1]], con = "test.fasta");

  # get the region adjacent to a set of mutations in a vcf
  clinvar.vcf.example <- system.file(
    "extdata/clinvar_dbSNP138_example.vcf.gz",
    package = "bedr"
    );
  clinvar <- read.vcf(clinvar.vcf.example);

  # note that clinvar uses ncbi fasta which does not use "chr" and codes chrM as MT
  clinvar.bed <- data.frame(
    chr = paste0("chr", gsub("MT", "M", clinvar$vcf$CHROM)),
    start = clinvar$vcf$POS - 2,
    end = clinvar$vcf$POS + 1,
    stringsAsFactors = FALSE
    );

  # get trinucleotide sequences of variants on chr M only
  mutation.triplet <- get.fasta(
   clinvar.bed[which(clinvar.bed$chr == "chrM"), ],
   fasta = system.file("extdata/ucsc.hg19.chrM.fasta", package = "bedr"),
   check.chr = FALSE
   );

## End(Not run)
}
```

`get.random.regions` *generates a set of random regions*

Description

generates a set of random regions for a specific species/build. Choices are human (hg18, hg19), mouse(mm9, mm10). regions are sampled from a log-normal distribution.

Usage

```
get.random.regions(
n = 10,
chr = NULL,
species = "human",
build = "hg19",
size.mean = 10,
size.sd = 0.25,
mask.gaps = FALSE,
mask.repeats = FALSE,
sort.output = TRUE,
verbose = TRUE
)
```

Arguments

n	number of regions
chr	the chr or region
species	species
build	build
size.mean	region mean in log space
size.sd	region sd in log space
mask.gaps	should the gaps (Ns) in the human reference be ignored as potential start points. This drammatically increases memory and run time but is more appropriate in almost all settings. By default it's off.
mask.repeats	should the repeats from repeatMasker be ignored as potential start points. This drammatically increases memory and run time but is more appropriate in almost all settings. By default it's off.
sort.output	return a sorted index
verbose	words

Author(s)

Daryl Waggott

Examples

```
test.regions <- get.random.regions(100)
```

grow.region *Get adjacent flanks from regions*

Description

Get adjacent flanks from regions

Usage

```
grow.region(
x,
n.add = NULL,
start.add = NULL,
end.add = NULL,
species = "human",
build = "hg19",
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
check.sort = TRUE,
check.merge = TRUE,
verbose = TRUE
)
```

Arguments

x	a object region object or index
n.add	the number of bases to be selected from each side of a region
start.add	the number of based to be selected from the start of a region
end.add	the number of based to be selected from the end of a region
species	the species i.e. human or mouse
build	the genome build i.e. hg19 or mm10
check.zero.based	should 0 based coordinates be checked
check.chr	should chr prefix be checked
check.valid	should the region be checkded for integerity
check.sort	should regions be checked for sort order
check.merge	should overlapping regions be checked
verbose	should log messages and checking take place

Author(s)

Daryl Waggott

References

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

Examples

```
if (check.binary("bedtools")) {

index <- get.example.regions();

a <- index[[1]];
a <- bedr(engine = "bedtools", input = list(i = a), method = "sort", params = "");
b <- grow.region(a, n.add = 20);

}
```

`in.region` *checks if regions in object a are found in object b*

Description

checks if regions in object a are found in object b

Usage

```
in.region(
x,
y,
proportion.overlap = 1e-09,
reciprocal.overlap = FALSE,
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
check.sort = TRUE,
check.merge = TRUE,
verbose = FALSE
)
```

Arguments

`x`	first region index in the form chr:start-stop. regions in this index will be checked for intersection in the values of the second index.
`y`	second region index.
`proportion.overlap`	Defaults 1e-9 which is 1 bp. See details below for the different interpretation between 0 and 1 based overlap
`reciprocal.overlap`	Should the proportion.overlap be reciprocal

check.zero.based	should 0 based coordinates be checked
check.chr	should chr prefix be checked
check.valid	check if region is valid
check.sort	check if region is sorted
check.merge	check if overlapping regions are merged
verbose	prints some debugging information. currently it just checks if the input regions are overlapping

Details

The function can also be called using syntax similar to the %in% operator, for example "region1 %in.region% region2"

The default is to report TRUE if there is 1bp overlap in zero based bed format. That means that region chr1:10-20 and chr1:20-30 would not overlap. To switch to one based intuitive interpretation set proportion.overlap = 0.

Value

Returns a logical vector the length of x.

Author(s)

Daryl Waggott

References

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

Examples

```
if (check.binary("bedtools")) {

index <- get.example.regions();

a <- index[[1]];
b <- index[[2]];
a <- bedr(engine = "bedtools", input = list(i = a), method = "sort", params = "");
b <- bedr(engine = "bedtools", input = list(i = b), method = "sort", params = "");

d <- in.region(a,b);

# alternative calling
d <- a %in.region% b

}
```

index2bed *convert a region index into a bed file dataframe*

Description

convert a region index into a bed file dataframe

Usage

```
index2bed(x, set.type = TRUE)
```

Arguments

x	an index
set.type	should the attribute input.type be set. Sometimes it is desirable to avoid setting it when applying intermediate conversion

Author(s)

Daryl Waggott

Examples

```
if (check.binary("bedtools")) {

index <- get.example.regions();
a <- index[[1]];
a.bed <- index2bed(a);
}
```

is.merged.region *checks if region file is merged*

Description

checks if region file is merged

Usage

```
is.merged.region(
x,
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
check.sort = TRUE,
verbose = FALSE
)
```

Arguments

x	region or index
check.zero.based	should 0 based coordinates be checked
check.chr	should chr prefix be checked
check.valid	check if region is valid
check.sort	check if region is sorted
verbose	more words

Author(s)

Daryl Waggott

Examples

```
if (check.binary("bedtools")) {

index <- get.example.regions();

a <- index[[1]];

b <- is.merged.region(a);

}
```

---

is.sorted.region *checks if region file is sorted*

Description

checks if region file is sorted

Usage

```
is.sorted.region(
x,
method = "lex",
engine = "unix",
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
check.merge = TRUE,
verbose = FALSE
)
```

Arguments

x	The region index, bed file, or bed formatted object
method	lexicgraphical or natural, lex is required for many operations but natural is better for interpretation
engine	what analytical engine to use for sorting i.e. bedtools, bedops, gnu unix
check.zero.based	should 0 based coordinates be checked
check.chr	should chr prefix be checked
check.valid	check if region is valid
check.merge	check if region is merged
verbose	more words

Author(s)

Daryl Waggott

Examples

```
if (check.binary("bedtools")) {
index <- get.example.regions();

a <- index[[1]];

b <- is.sorted.region(a);
}
```

---

is.valid.ref *verifies the reference sequence in a vcf*

Description

verifies the reference sequence in a vcf

Usage

```
is.valid.ref(
x,
fasta = NULL,
strand = FALSE,
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
check.sort = TRUE,
check.merge = TRUE,
verbose = TRUE
)
```

Arguments

x	input bed object
fasta	a reference build in fasta format
strand	should strand be used. if reverse then the sequence will be reverse complemented
check.zero.based	should 0 based coordinates be checked
check.chr	should chr prefix be checked
check.valid	should the region be checkded for integerity
check.sort	should regions be checked for sort order
check.merge	should overlapping regions be checked
verbose	should log messages and checking take place

Value

a logical vector the length of the input

Author(s)

Daryl Waggott

Examples

```
vcf.path <- system.file("extdata/callerA.vcf.gz", package = "bedr");
vcf.data <- read.vcf(vcf.path, split.info = TRUE);
vcf.data$vcf <- vcf.data$vcf[,
c("CHROM", "POS", "END", setdiff(colnames(vcf.data$vcf), c("CHROM", "POS", "END")))
];
vcf.data$vcf$CHROM <- paste("chr", vcf.data$vcf$CHROM, sep = "");

## Not run:
# need reference sequence FASTA and index file to run this, as 'fasta' parameter
is.valid.ref(vcf.data);

## End(Not run)
```

---

is.valid.region *checks if region/index is valid*

Description

checks if region/index is valid

Usage

```
is.valid.region(
x,
check.zero.based = TRUE,
check.chr = TRUE,
throw.error = FALSE,
verbose = TRUE
)
```

Arguments

x	The region index, bed file, or bed formatted object
check.zero.based	should basic test for zero based coordinates be checked
check.chr	should the algorithm check for the "chr" prefix
throw.error	should an error be thrown. The default is to report a logical vector of inconsistencies.
verbose	should diagnostic messages be printed

Author(s)

Daryl Waggott

Examples

```
index <- get.example.regions();

a <- index[[1]];
is.valid <- is.valid.region(a);
a.valid <- a[is.valid];
```

---

is.valid.seq *verifies that sequences are correct given coordinates and a reference*

Description

verifies that sequences are correct given coordinates and a reference

Usage

```
is.valid.seq(
x,
querySeq,
fasta = NULL,
strand = FALSE,
```

```
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
check.sort = TRUE,
check.merge = TRUE,
verbose = TRUE
)
```

Arguments

x	input bed object
querySeq	a vector of sequences the same length as x
fasta	a reference build in fasta format
strand	should strand be used. if reverse then the sequence will be reverse complemented
check.zero.based	should 0 based coordinates be checked
check.chr	should chr prefix be checked
check.valid	should the region be checkded for integerity
check.sort	should regions be checked for sort order
check.merge	should overlapping regions be checked
verbose	should log messages and checking take place

Value

a logical vector the length of the input querySeq

Author(s)

Daryl Waggott, Syed Haider

Examples

```
if (check.binary("bedtools")) {
  index <- get.example.regions();
  a <- index[[1]];
  a <- get.fasta(bedr.sort.region(a));
  is.valid.seq(x = a, querySeq = a$sequence);
}
```

jaccard *calculate the jaccard distance between sets of intervals*

Description

calculate the jaccard distance between sets of intervals

Usage

```
jaccard(
x,
y,
proportion.overlap = 1e-09,
reciprocal.overlap = FALSE,
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
check.sort = TRUE,
check.merge = TRUE,
verbose = TRUE
)
```

Arguments

x	first region to be compared
y	second region to be compared
proportion.overlap	Defaults 1e-9 which is 1 bp. See details below for the different interpretation between 0 and 1 based overlap
reciprocal.overlap	Should the proportion.overlap be reciprocal
check.zero.based	should 0 based coordinates be checked
check.chr	should chr prefix be checked
check.valid	should the region be checkded for integerity
check.sort	should regions be checked for sort order
check.merge	should overlapping regions be checked
verbose	should log messages and checking take place

Details

The Jaccard metric is the ratio of intersections to unions. The process can be tweaked by changing the proportion of overlap and even growiwng the regions.

Value

A short vector.

Author(s)

Daryl Waggott

References

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

See Also

reldist

Examples

```
if (check.binary("bedtools")) {

index <- get.example.regions();

a <- index[[1]];
b <- index[[2]];
a <- bedr(engine = "bedtools", input = list(i = a), method = "sort", params = "");
b <- bedr(engine = "bedtools", input = list(i = b), method = "sort", params = "");
jaccard(a,b);

}
```

---

modifyList2 *Interface to R's modifyList*

Description

Interface to R's modifyList

Usage

```
modifyList2(x, val)
```

Arguments

x	a named list
val	a named list with components to be updated using x

Value

modified version of x

Author(s)

Daryl Waggott

See Also

modifyList

`order.region` *Gets the sort order of a region index similar to the order command*

Description

Helps if you don't want to use sort region on a huge dataset

Usage

```
order.region(
x,
method = "lex",
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
check.merge = TRUE
)
```

Arguments

`x`	index or bed style data.frame
`method`	natural or lexicographical (lex)
`check.zero.based`	should 0 based coordinates be checked
`check.chr`	should chr prefix be checked
`check.valid`	check if region is valid
`check.merge`	check if region is sorted and merged

Author(s)

Daryl Waggott

References

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

Examples

```
if (check.binary("bedtools")) {

index <- get.example.regions();

a <- index[[1]];
a <- bedr(engine = "bedtools", input = list(i = a), method = "sort", params = "");
a.order <- order.region(a)

b <- a[a.order];

}
```

`permute.region` *permute a set of regions*

Description

permute a set of regions

Usage

```
permute.region(
x,
stratify.by.chr = FALSE,
species = "human",
build = "hg19",
chr.names = paste0("chr",c(1:22,"X","Y","M")),
mask.gaps = FALSE,
gaps.file = NULL,
mask.repeats = FALSE,
repeats.file = NULL,
sort.output = TRUE,
is.checked = FALSE,
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
verbose = TRUE
)
```

Arguments

`x`	regions to permute
`stratify.by.chr`	Should the permutation be happen separetely for each chromosome. That is are chromosomes exchangeable.
`species`	species

build	the build of the reference
chr.names	names of the chromosomes to use
mask.gaps	should the gaps (Ns) in the human reference be ignored as potential start points. This drammatically increases memory and run time but is more appropriate in almost all settings. It defaults to off
gaps.file	database file of gaps. Defaults to Homo sapiens Hg19 gap.txt.gz file available through UCSC
mask.repeats	should the repeats from repeatMasker be ignored as potential start points. This drammatically increases memory and run time but is more appropriate in almost all settings. By default it's off
repeats.file	database file of repeats as supplied by UCSC containing RepMasker data e.g rmsk.txt.gz
sort.output	should the output be sorted
is.checked	Has the input data already be tested for validity. This is often done once before multiple permutations.
check.zero.based	should 0 based coordinates be checked
check.chr	should chr prefix be checked
check.valid	should the region be checkded for integerity
verbose	should log messages and checking take place

Details

1. Sampling with replacement on region length. 2. Sampling with replacement on start positions. Positions that contain Ns in the reference are set to 0 weight during sampling.

Regions that overlap the end of a chromosome or gap are trimmed.

Steps 1 and 2 are done within chromosomes if stratify.by.chr is set to true.

Value

A region object with randomized start positions.

Author(s)

Daryl Waggott

Examples

```
if (check.binary("bedtools")) {

index <- get.example.regions();
a <- index[[1]];
a <- bedr(engine = "bedtools", input = list(i = a), method = "sort", params = "");
a.perm <- permute.region(a);

}
```

process.input *process.input*

Description

process.input

Usage

```
process.input(
input,
tmpDir = NULL,
include.names = TRUE,
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
check.sort = TRUE,
check.merge = TRUE,
verbose = TRUE
)
```

Arguments

input	regions input or a file in one of the standard formats. these are bed, vcf, gff, bam, sam, csv, tsv, txt
tmpDir	The directory to be used for writing files
include.names	should the names of the input files be included in the output
check.zero.based	should 0 based coordinates be checked
check.chr	should chr prefix be checked
check.valid	should the region be checked for integerity
check.sort	should the region sorting be checked
check.merge	should overlapping regions be checked
verbose	messages and checks

Value

list of input files

Author(s)

Daryl Waggott

Examples

```
if (check.binary("bedtools")) {

index <- get.example.regions();
a <- index[[1]];
a <- bedr(engine = "bedtools", input = list(i = a), method = "sort", params = "");
a.processed <- process.input(a, verbose = FALSE)

}
```

`query.ucsc` *read a ucsc table into R*

Description

read a ucsc table into R

Usage

```
query.ucsc(
x,
mirror = "http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database",
download = TRUE,
overwrite.local = FALSE,
columns.keep = NULL,
verbose = TRUE
)
```

Arguments

`x`	a ucsc data table. Include the full path including "txt.gz" extenstion to load from a local file. Note that $HOME/bedr/data will be checked first before downloading.
`mirror`	the ucsc mirror
`download`	should the data be downloaded to $HOME/bedr/data/
`overwrite.local`	should the local version be overwritten if it exists
`columns.keep`	what columns to load. this can help with very large tables where you only want 'chr,start,end'. defaults to all. you may have to check the sql for the actual column names.
`verbose`	more words

Details

tables can be found at http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database

Value

A data.frame

Author(s)

Daryl Wagott

Examples

```
## Not run:
  query.ucsc("refGene");

## End(Not run)
```

---

read.vcf *Read a vcf into R*

Description

Read a vcf into R and parse it for downstream analysis

Usage

```
read.vcf(x, split.info = FALSE, split.samples = FALSE, nrows = -1, verbose = TRUE)
```

Arguments

x	A vcf
split.info	Split the info into columns
split.samples	Split the sample into columns. If multiple samples then a list matrices will be created, one matrix for each element in the FORMAT tag.
nrows	The the number of rows to be read. Set to 0 to parse the header.
verbose	print progress

Details

The function can be slow for splitting the INFO, FORMAT for large VCFs.

Value

VCF representation in R as a list. The first element in the list is the header, the second the body of the VCF. Every repeating tag in the header i.e. INFO, FORMAT is structured as matrix. If reading a multi-sample VCF and the split.sample = TRUE, then a matrix is added to the list for every tag in the FORMAT string.

Note that by default the vcf is returned as a data.table not a data.frame. Therefore there are some quirks i.e. subsetting via named columns a$vcf[,"CHROM", with = FALSE]. If in doubt just caset to data.frame.

Author(s)

Daryl Waggott

Examples

```
clinVar.vcf.example    <- system.file("extdata/clinvar_dbSNP138_example.vcf.gz", package = "bedr")
singleSample.vcf.example <- system.file("extdata/singleSampleOICR_example.vcf.gz", package = "bedr")
multiSample.vcf.example  <- system.file("extdata/multiSampleOICR_example.vcf.gz", package = "bedr")

# read a subset of NCBI clinVar vcf.  Note this has no samples.
vcf1.a <- read.vcf(clinVar.vcf.example)
vcf1.b <- read.vcf(clinVar.vcf.example, split.info = TRUE)

## Not run:

# same as above but split multiple samples
vcf1.c <- read.vcf(clinVar.vcf.example, split.info = TRUE, split.sample = TRUE)

# read a single-sample VCF
system.time(
  vcf2.a <- read.vcf(singleSample.vcf.example, split.info = TRUE, split.sample = TRUE)
  )

# read a multi-sample VCF
vcf3.a <- read.vcf(multiSample.vcf.example, split.info = FALSE, split.sample = TRUE);

# multi core example
options("cores"=9);
vcf2.a <- read.vcf(singleSample.vcf.example, split.info = TRUE, split.sample = TRUE)
options("cores"=1);


## End(Not run)
```

---

reldist *Calculate the relative distance between two sets of intervals*

Description

Calculate the relative distance between two sets of intervals

Usage

```
reldist(
x,
y,
detail = FALSE,
check.zero.based = TRUE,
```

```
check.chr = TRUE,
check.valid = TRUE,
check.sort =TRUE,
check.merge = TRUE,
verbose = TRUE
)
```

Arguments

x	firt region to be compared
y	second region to be compared
detail	should the relative distance be printed for every region
check.zero.based	should 0 based coordinates be checked
check.chr	should chr prefix be checked
check.valid	should the region be checkded for integerity
check.sort	should regions be checked for sort order
check.merge	should overlapping regions be checked
verbose	should log messages and checking take place

Details

The frequency of relative distances in bins spanning 0 to 0.5

Author(s)

Daryl Waggott

References

http://bedtools.readthedocs.org/en/latest/content/tools/reldist.html

See Also

jaccard

Examples

```
if (check.binary("bedtools")) {

index <- get.example.regions();

a <- index[[1]];
b <- index[[2]];
a <- bedr(engine = "bedtools", input = list(i = a), method = "sort", params = "");
b <- bedr(engine = "bedtools", input = list(i = b), method = "sort", params = "");
reldist(a,b);

}
```

`size.region` *Get region size*

Description

Get region size

Usage

```
size.region(
x,
zero.based = TRUE,
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
verbose = TRUE
)
```

Arguments

`x`	region in vector, matrix or dataframe format
`zero.based`	whether the coordinates are zero-based or 1
`check.zero.based`	should 0 based coordinates be checked
`check.chr`	should chr prefix be checked
`check.valid`	should the region be checked for integerity
`verbose`	messages and checks

Value

size/length of the region

Author(s)

Daryl Waggott

See Also

convert2bed

Examples

```
if (check.binary("bedtools")) {

index <- get.example.regions();

a <- index[[1]];
```

```
a.sizes <- bedr:::size.region(a);
}
```

`strsplit2matrix` *split a vector of strings into tabular data*

Description

split a vector of strings into tabular data

Usage

```
strsplit2matrix(x, split, fixed = FALSE, perl = FALSE)
```

Arguments

`x`	a character vector
`split`	the character or regex to split on
`fixed`	fixed i.e. no regex
`perl`	per style

Author(s)

Daryl Waggott

Examples

```
## Not run:
  a.bed <- strSplitToMatrix(x);

## End(Not run)
```

`tabix` *Main bedtools wrapper function.*

Description

Main bedtools wrapper function.

Usage

```
tabix(
region,
file.name,
params = NULL,
tmpDir = NULL,
deleteTmpDir = TRUE,
outputDir = NULL,
outputFile = NULL,
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
check.sort = TRUE,
check.merge = TRUE,
verbose = TRUE
)
```

Arguments

region	The regions to query the tabix'd file
file.name	The name of the bzipped/indexed tabix file to query
params	A string that includes all the extra parameters and arguments to the bedtools commmand. For example if you wanted to do a left outer join you would specificy method as intersect and use params = c("-loj -header"). If you leave input and method as defaults then this is this string represents the full command.
tmpDir	The directory to be used for writing files
deleteTmpDir	Should tmp files be deleted. helpful for diagnostics.
outputDir	The output directory. Only used if outputFile is specified. It defaults to the current working directory.
outputFile	The name of the output file. If this is specified the output will be sent to a file not an R object
check.chr	check for chr prefix
check.zero.based	check for zero based coordinates
check.valid	do all region integrity checks
check.sort	check if region is sorted
check.merge	check if region is merged
verbose	Should messages be printed to screen.

Value

The output of command with some parsing to keep it consistent with the input.

Author(s)

Daryl Waggott

See Also

genomicRanges

Examples

```
if (check.binary("tabix")) {
query.regions <- c("1:1000-100000", "1:1000000-1100000")
cosmic.vcf.example <- system.file(
"extdata/CosmicCodingMuts_v66_20130725_ex.vcf.gz",
package = "bedr"
)
cosmic.query <- tabix(query.regions, cosmic.vcf.example, check.chr = FALSE)
}
```

---

table2venn *Plot venn diagram*

Description

Plot venn diagram of regions intersect

Usage

```
table2venn(x, var.names)
```

Arguments

x	intersect table of regions
var.names	names of the overlapping regions

Value

venn diagram input list

Author(s)

Daryl Waggott

See Also

bedr.plot.region

`test.region.similarity` *Compare sets of regions via jaccard and relative distance using permutation*

Description

Compare sets of regions via jaccard and relative distance using permutation to get an empirical p-value.

Usage

```
test.region.similarity(
x,
y,
n = 1000,
stratify.by.chr = FALSE,
species = "human",
build = "hg19",
mask.gaps = FALSE,
mask.repeats = FALSE,
gaps.file = NULL,
repeats.file = NULL,
check.zero.based = TRUE,
check.chr = TRUE,
check.valid = TRUE,
verbose = TRUE
)
```

Arguments

`x`	first region to be compared. this is the region that is permuted
`y`	second region to be compared
`n`	the number of iterations to permute
`stratify.by.chr`	Should the permutation be happen separetely for each chromosome. That is are chromosomes exchangeable.
`species`	species
`build`	the build of the reference
`mask.gaps`	should the gaps (Ns) in the human reference be ignored as potential start points. This drammatically increases memory and run time but is more appropriate in almost all settings. By default it's off.
`mask.repeats`	should the repeats from repeatMasker be ignored as potential start points. This drammatically increases memory and run time but is more appropriate in almost all settings. By default it's off.

gaps.file	database file of gaps. Defaults to Homo sapiens Hg19 gap.txt.gz file available through UCSC
repeats.file	database file of repeats as supplied by UCSC containing RepMasker data e.g rmsk.txt.gz
check.zero.based	should 0 based coordinates be checked
check.chr	should chr prefix be checked
check.valid	should the region be checkded for integerity
verbose	should log messages and checking take place

Details

Iteratively permutes intervals and recalculates jaccard and reldist statistics.

Value

A list of results

Author(s)

Daryl Waggott

Examples

```
if (check.binary("bedtools")) {

index <- get.example.regions();

a <- index[[1]];
b <- index[[2]];
a <- bedr(engine = "bedtools", input = list(i = a), method = "sort", params = "");
b <- bedr(engine = "bedtools", input = list(i = b), method = "sort", params = "");

# a simple example
test.region.similarity(a, b, n = 8)

# note you can set the cores available to parallelize
options(cores = 4);
system.time(test.region.similarity(a, b, n = 8));

# a real example comparing the distribution of mRNA vs ncRNA genes in RefSeq
## Not run:

# more core
options(cores = 8);

# load refgene
refgene <- query.ucsc("refGene")
refgene <- refgene[,c("chrom","txStart","txEnd","name","name2","strand")]
```

```
# only include canonical chr
chr <- paste0("chr", c(1:22,"X","Y"));
refgene <- refgene[refgene$chrom

# remove genes with multiple positions
duplicated.gene <- duplicated(refgene$name2) | duplicated(rev(refgene$name2));
refgene <- refgene[!duplicated.gene,];

# only select pr coding genes
refgene.nm <- refgene[grepl("^NM",refgene$name),];
# only select non protein coding rna genes
refgene.nr <- refgene[grepl("^NR",refgene$name),];

# sort and merge
refgene.nm <- bedr.snm.region(refgene.nm,check.chr = FALSE);
refgene.nr <- bedr.snm.region(refgene.nr,check.chr = FALSE);

test.region.similarity(refgene.nm, refgene.nr, mask.unmapped = TRUE );

option(core = 1)

## End(Not run)

}
```

---

vcf2bed *convert a vcf to a bed file*

Description

Convert a vcf to a bed file. Currently, it needs to read into R via read.vcf

Usage

```
vcf2bed(x, filename = NULL, header = FALSE, other = NULL, verbose = TRUE)
```

Arguments

x	a vcf object
filename	the name of file if you want it exported
header	indicate if the bed file has header or not when exported
other	fields to include apart from chr, start, end.
verbose	more words

Value

A bed styled R object or an external file

Author(s)

Daryl Waggott

Examples

```
clinVar.vcf.example <- system.file("extdata/clinvar_dbSNP138_example.vcf.gz", package = "bedr")
x <- read.vcf(clinVar.vcf.example)
x.bed <- vcf2bed(x)
```

`write.vcf` *write a vcf object*

Description

write a vcf object

Usage

```
write.vcf(x, filename = NULL, verbose = TRUE)
```

Arguments

`x`	a vcf object
`filename`	a filename
`verbose`	more words

Details

The input needs to be a vcf object. This

Value

A vcf file

Author(s)

Daryl Waggott

References

vcf format specifications

Examples

```
vcf <- read.vcf(system.file("extdata/clinvar_dbSNP138_example.vcf.gz", package = "bedr"));
vcf$header <- c(vcf$header, NOTE="vcf processed by bedr")

## Not run:
  write.vcf(vcf, filename = paste(tempdir(), "/bedr.example.vcf", sep = ""));

## End(Not run)
```

`%in.region%` *checks if regions in object a are found in object b*

Description

checks if regions in object a are found in object b

Usage

```
x %in.region% y
```

Arguments

x	first region index in the form chr:start-stop. regions in this index will be checked for intersection in the values of the second index.
y	second region index.

Details

The function can also be called using syntax similar to the %in% operator, for example "region1 %in.region% region2"

Value

Returns a logical vector the length of x.

Author(s)

Daryl Waggott

Examples

```
if (check.binary("bedtools")) {

index <- get.example.regions();

a <- index[[1]];
b <- index[[2]];
```

```
a <- bedr.sort.region(a);
b <- bedr.sort.region(b);

d <- a %in.region% b

}
```

Index

∗ **bedops**
bedr, 5
tabix, 51
∗ **bedtools**
bedr, 5
index2bed, 34
strsplit2matrix, 51
tabix, 51
∗ **bed**
bed2index, 3
bedr, 5
convert2bed, 21
tabix, 51
∗ **binary**
check.binary, 19
∗ **cat**
catv, 18
∗ **cluster**
cluster.region, 20
∗ **config**
bedr.setup, 14
∗ **data**
download.datasets, 24
∗ **distance**
jaccard, 40
reldist, 48
∗ **example**
get.random.regions, 29
is.valid.region, 37
∗ **fasta**
get.fasta, 27
∗ **flank**
flank.region, 25
∗ **input**
determine.input, 23
process.input, 45
∗ **interval**
bedr, 5
tabix, 51
∗ **in**
%in.region%, 58
in.region, 32
∗ **join**
bedr.join.multiple.region, 8
bedr.join.region, 9
∗ **length**
get.chr.length, 26
∗ **merge**
bedr.merge.region, 11
cluster.region, 20
∗ **order**
order.region, 42
∗ **package**
bedr-package, 3
∗ **permute**
permute.region, 43
test.region.similarity, 54
∗ **plot**
bedr.plot.region, 12
∗ **region**
bedr, 5
size.region, 50
tabix, 51
∗ **sequence**
is.valid.ref, 36
is.valid.seq, 38
∗ **sort**
bedr.snm.region, 14
bedr.sort.region, 16
grow.region, 31
order.region, 42
∗ **subtract**
bedr.subtract.region, 17
∗ **tabix**
bedr, 5
tabix, 51
∗ **tmp**
create.tmp.bed.file, 22

∗ **ucsc**
query.ucsc, 46
∗ **vcf**
read.vcf, 47
vcf2bed, 56
write.vcf, 57
∗ **venn**
bedr.plot.region, 12
table2venn, 53
%in.region%, 58

bed2index, 3
bed2vcf, 4
bedr, 5
bedr-package, 3
bedr.join.multiple.region, 8
bedr.join.region, 9
bedr.merge.region, 11, *21*
bedr.plot.region, 12
bedr.setup, 14
bedr.snm.region, 14
bedr.sort.region, 16
bedr.subtract.region, 17

catv, 18
check.binary, 19
cluster.region, 20
convert2bed, 21
create.tmp.bed.file, 22

determine.input, 23
df2list, 23
download.datasets, 24

flank.region, 25

get.chr.length, 26
get.example.regions, 27
get.fasta, 27
get.random.regions, 29
grow.region, 31

in.region, 32
index2bed, 34
is.merged.region, 34
is.sorted.region, 35
is.valid.ref, 36
is.valid.region, 37
is.valid.seq, 38

jaccard, 40

modifyList2, 41

order.region, 42

permute.region, 43
process.input, 45

query.ucsc, 46

read.vcf, 47
reldist, 48

size.region, 50
strsplit2matrix, 51

tabix, 51
table2venn, 53
test.region.similarity, 54

vcf2bed, 56
venn.diagram, *12*

write.vcf, 57