Package 'multicrispr'

August 26, 2026

Title Multi-locus multi-purpose Crispr/Cas design

Version 1.23.0

Encoding UTF-8

Description This package is for designing Crispr/Cas9 and Prime Editing experiments.
It contains functions to (1) define and transform genomic targets, (2) find spacers
(4) count offtarget (mis)matches, and (5) compute Doench2016/2014 targeting efficiency.
Care has been taken for multicrispr to scale well towards large target sets,
enabling the design of large Crispr/Cas9 libraries.

License GPL-2

LazyData true

RoxygenNote 7.3.2

Depends R (>= 4.0)

Imports BiocGenerics, Biostrings, BSgenome, CRISPRseek, data.table,
Seqinfo, GenomicFeatures, GenomicRanges, ggplot2, grid,
karyoploteR, magrittr, methods, parallel, plyranges, Rbowtie,
reticulate, rtracklayer, stats, stringi, tidyr, tidyselect,
utils

Suggests AnnotationHub, BiocStyle, BSgenome.Hsapiens.UCSC.hg38,
BSgenome.Mmusculus.UCSC.mm10,
BSgenome.Scerevisiae.UCSC.sacCer1, ensembldb, IRanges,
GenomeInfoDb, knitr, magick, rmarkdown, testthat,
TxDb.Mmusculus.UCSC.mm10.knownGene

VignetteBuilder knitr

biocViews CRISPR, Software

BugReports https://github.com/bhagwataditya/multicrispr/issues

URL https://github.com/bhagwataditya/multicrispr

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

git_branch devel

git_last_commit aabd0f0

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-08-25

Author Aditya Bhagwat [aut, cre],
Richie ´Cotton [aut],
Rene Wiegandt [ctb],
Mette Bentsen [ctb],
Jens Preussner [ctb],
Michael Lawrence [ctb],
Hervé Pagès [ctb],
Johannes Graumann [sad],
Mario Looso [sad, rth]

Maintainer Aditya Bhagwat <aditya.bhagwat@uni-marburg.de>

Contents

add_genome_matches 2
add_inverse_strand 3
add_seq 4
add_target_matches 5
bed_to_granges 6
char_to_granges 7
double_flank 7
extend_for_pe 9
extend_pe_to_gg 10
extract_matchranges 10
extract_subranges 11
find_gg 12
find_primespacers 13
find_spacers 14
genes_to_granges 16
gr2dt 17
has_been_indexed 18
index_genome 18
index_targets 19
plot_intervals 20
plot_karyogram 21
score_ontargets 22
up_flank 24
write_ranges 26

Index **27**

`add_genome_matches` *Add genome matches*

Description

Add genome matches

Usage

```
add_genome_matches(
  spacers,
  bsgenome = getBSgenome(genome(spacers)[1]),
  mismatches = 2,
  pam = "NGG",
  offtargetmethod = c("bowtie", "pdict")[1],
  outdir = OUTDIR,
  indexedgenomesdir = INDEXEDGENOMESDIR,
  verbose = TRUE
)
```

Arguments

spacers	GRanges
bsgenome	BSgenome
mismatches	number
pam	string
offtargetmethod	'bowtie' or 'pdict'
outdir	bowtie output directory
indexedgenomesdir	directory with indexed genomes
verbose	TRUE (default) or FALSE

Value

GRanges

Examples

```
require(magrittr)
file <- system.file('extdata/SRF.bed', package='multicrispr')
bsgenome <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10
targets0 <- bed_to_granges(file, 'mm10')
targets <- extend(targets0)
spacers <- find_spacers(targets, bsgenome, complement = FALSE,
                        ontargetmethod = NULL, offtargetmethod = NULL)
spacers %<>% extract(1:100)
spacers %<>% add_genome_matches(bsgenome)
```

---

add_inverse_strand *Add inverse strand*

Description

Add inverse strand

Usage

```
add_inverse_strand(gr, verbose = FALSE, plot = FALSE, ...)
```

Arguments

gr	GRanges-class
verbose	TRUE or FALSE (default)
plot	TRUE or FALSE (default)
...	plot_intervals arguments

Value

GRanges-class

Examples

```
# PE example
#-----------
    require(magrittr)
    bsgenome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
    gr <- char_to_granges(c(PRNP = 'chr20:4699600:+',             # snp
                            HBB  = 'chr11:5227002:-',             # snp
                            HEXA = 'chr15:72346580-72346583:-',   # del
                            CFTR = 'chr7:117559593-117559595:+'), # ins
                          bsgenome)
    add_inverse_strand(gr, plot = TRUE)
# TFBS example
#-------------
    bedfile <- system.file('extdata/SRF.bed', package='multicrispr')
    gr <- bed_to_granges(bedfile, genome = 'mm10')
    add_inverse_strand(gr)
```

---

add_seq *Add sequence to GRanges*

Description

Add sequence to GRanges

Usage

```
add_seq(gr, bsgenome, verbose = FALSE, as.character = TRUE)
```

Arguments

gr	GRanges-class
bsgenome	BSgenome-class
verbose	TRUE or FALSE (default)
as.character	TRUE (default) or FALSE

Value

GRanges-class

Examples

```
# PE example
#-----------
    require(magrittr)
    bsgenome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
    gr <- char_to_granges(c(PRNP = 'chr20:4699600:+',             # snp
                            HBB  = 'chr11:5227002:-',             # snp
                            HEXA = 'chr15:72346580-72346583:-',   # del
                            CFTR = 'chr7:117559593-117559595:+'), # ins
                          bsgenome)
   (gr %<>% add_seq(bsgenome))

# TFBS example
#-------------
    bsgenome <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10
    bedfile  <- system.file('extdata/SRF.bed', package='multicrispr')
    gr <- bed_to_granges(bedfile, 'mm10')
    (gr %<>% add_seq(bsgenome))
```

`add_target_matches` *Add target matches*

Description

Add target matches

Usage

```
add_target_matches(
  spacers,
  targets,
  bsgenome,
  mismatches = 2,
  pam = "NGG",
  outdir = OUTDIR,
  verbose = TRUE
)
```

Arguments

`spacers`	GRanges
`targets`	GRanges
`bsgenome`	BSgenome
`mismatches`	number
`pam`	string
`outdir`	bowtie output directory
`verbose`	TRUE (default) or FALSE

Value

GRanges

Examples

```
require(magrittr)
file <- system.file('extdata/SRF.bed', package='multicrispr')
bsgenome <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10
targets0 <- bed_to_granges(file, 'mm10')
targets <- extend(targets0)
spacers <- find_spacers(targets, bsgenome, complement = FALSE,
                        ontargetmethod = NULL, offtargetmethod = NULL)
spacers %<>% add_target_matches(targets, bsgenome)
```

`bed_to_granges` *Read bedfile into GRanges*

Description

Read bedfile into GRanges

Usage

```
bed_to_granges(
  bedfile,
  genome,
  txdb = NULL,
  do_order = TRUE,
  plot = TRUE,
  verbose = TRUE
)
```

Arguments

`bedfile`	file path
`genome`	string: UCSC genome name (e.g. 'mm10')
`txdb`	NULL (default) or `TxDb-class` (used for gene annotation)
`do_order`	TRUE (default) or FALSE: order on seqnames and star?
`plot`	TRUE (default) or FALSE: plot karyogram?
`verbose`	TRUE (default) or FALSE

Value

`GRanges-class`

See Also

`char_to_granges`, `genes_to_granges`

Examples

```
bedfile  <- system.file('extdata/SRF.bed', package = 'multicrispr')
bsgenome <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10
(gr <- bed_to_granges(bedfile, genome='mm10'))
```

char_to_granges *Convert character vector into GRanges*

Description

Convert character vector into GRanges

Usage

```
char_to_granges(x, bsgenome)
```

Arguments

x	character vector
bsgenome	BSgenome-class

Value

GRanges-class

See Also

bed_to_granges, genes_to_granges

Examples

```
require(magrittr)
bsgenome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
x <- c(PRNP  = 'chr20:4699600:+',            # snp
       HBB   = 'chr11:5227002:-',           # snp
       HEXA = 'chr15:72346580-72346583:-',  # del
       CFTR = 'chr7:117559593-117559595:+') # ins
gr <- char_to_granges(x, bsgenome)
plot_intervals(gr, facet_var = c('targetname', 'seqnames'))
```

double_flank *Double flank*

Description

Double flank

Usage

```
double_flank(
  gr,
  upstart = -200,
  upend = -1,
  downstart = 1,
  downend = 200,
  strandaware = TRUE,
  plot = FALSE,
  linetype_var = "set",
  ...
)
```

Arguments

gr	GRanges-class
upstart	upstream flank start in relation to start(gr)
upend	upstream flank end in relation to start(gr)
downstart	downstream flank start in relation to end(gr)
downend	downstream flank end in relation to end(gr)
strandaware	TRUE (default) or FALSE
plot	TRUE or FALSE (default)
linetype_var	gr var mapped to linetype
...	passed to plot_intervals

Value

GRanges-class

Examples

```
# Prime Editing example
#----------------------
    require(magrittr)
    bsgenome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
    gr <- char_to_granges(c(PRNP = 'chr20:4699600:+',             # snp
                            HBB  = 'chr11:5227002:-',             # snp
                            HEXA = 'chr15:72346580-72346583:-',   # del
                            CFTR = 'chr7:117559593-117559595:+'), # ins
                          bsgenome)
    double_flank(gr, -10,  -1, +1, +20, plot = TRUE)

# TFBS example
#-------------
    bedfile  <- system.file('extdata/SRF.bed', package='multicrispr')
    gr  <-  bed_to_granges(bedfile, genome = 'mm10', plot = FALSE)
    double_flank(gr, plot = TRUE)
```

extend_for_pe *Extend ranges for prime editing*

Description

Extend target ranges to span in which to look for spacer-pam seqs

Usage

```
extend_for_pe(
  gr,
  bsgenome,
  nrt = 16,
  spacer = strrep("N", 20),
  pam = "NGG",
  plot = FALSE
)
```

Arguments

gr	GRanges-class
bsgenome	BSgenome-class
nrt	number: reverse transcription length
spacer	string: spacer pattern in extended IUPAC alphabet
pam	string: pam pattern in extended IUPAC alphabet
plot	TRUE (default) or FALSE

Details

Extend target ranges to find nearby spacers for prime editing

Value

GRanges-class

Examples

```
require(magrittr)
bsgenome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
gr <- char_to_granges(c( PRNP = 'chr20:4699600:+',             # snp
                         HBB  = 'chr11:5227002:-',             # snp
                         HEXA = 'chr15:72346580-72346583:-',   # del
                         CFTR = 'chr7:117559593-117559595:+'), # ins
                      bsgenome = bsgenome)
find_primespacers(gr, bsgenome)
(grext <- extend_for_pe(gr))
find_spacers(grext, bsgenome, complement = FALSE)
```

extend_pe_to_gg *Extend prime editing target to find GG sites*

Description

Extend prime editing target to find GG sites in accessible neighbourhood

Usage

```
extend_pe_to_gg(gr, nrt = 16, plot = FALSE)
```

Arguments

gr	target GRanges-class
nrt	n RT nucleotides (default 16, recommended 10-16)
plot	TRUE or FALSE (default)

Details

Extends each target range to the area in which to search for a prime editing GG duplet, as shown in the sketch below.

==========> —-GG———> —-GG———> ** <———GG— <———GG—- <==========

Value

GRanges-class

Examples

```
# PE example
#-----------
    require(magrittr)
    bsgenome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
    gr <- char_to_granges(c(PRNP = 'chr20:4699600:+',             # snp
                            HBB  = 'chr11:5227002:-',             # snp
                            HEXA = 'chr15:72346580-72346583:-',   # del
                            CFTR = 'chr7:117559593-117559595:+'), # ins
                          bsgenome)
    extend_pe_to_gg(gr, plot = TRUE)
```

extract_matchranges *Extract matching subranges*

Description

Extract subranges that match pattern

Usage

```
extract_matchranges(gr, bsgenome, pattern, plot = FALSE)
```

Arguments

gr	GRanges-class
bsgenome	BSgenome{BSgenome-class}
pattern	string: search pattern in extended IUPAC alphabet
plot	TRUE or FALSE (default)

Value

GRanges-class

Examples

```
# PE example
#------------
require(magrittr)
bsgenome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
gr <- char_to_granges(c(PRNP = 'chr20:4699600:+',             # snp
                        HBB  = 'chr11:5227002:-',             # snp
                        HEXA = 'chr15:72346580-72346583:-',   # del
                        CFTR = 'chr7:117559593-117559595:+'), # ins
                      bsgenome)
gr %<>% extend_for_pe()
pattern <- strrep('N',20) %>% paste0('NGG')
extract_matchranges(gr, bsgenome, pattern, plot = TRUE)

# TFBS examples
#--------------
bsgenome <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10
bedfile  <- system.file('extdata/SRF.bed', package='multicrispr')
gr <- bed_to_granges(bedfile, 'mm10') %>% extend()
extract_matchranges(gr, bsgenome, pattern = strrep('N',20) %>% paste0('NGG'))
```

---

extract_subranges *Extract subranges*

Description

Extract subranges from a GRanges-class object

Usage

```
extract_subranges(gr, ir, plot = FALSE)
```

Arguments

gr	GRanges-class
ir	IRanges-class: subranges to be extracted
plot	TRUE or FALSE (default)

Value

GRanges-class.

Examples

```
# Extract a subrange
gr <- GenomicRanges::GRanges(c(A = 'chr1:1-100:+', B = 'chr1:1-100:-'))
gr$targetname <- 'AB'
ir <- IRanges::IRanges(c(A = '1-10', A = '11-20', B = '1-10', B = '11-20'))
extract_subranges(gr, ir, plot = TRUE)

# Return empty GRanges for empty IRanges
extract_subranges(GenomicRanges::GRanges('chr1:345-456'), IRanges::IRanges())
```

---

find_gg *Find GG*

Description

Find GG

Usage

```
find_gg(gr)
```

Arguments

gr	GRanges-class

Value

GRanges-class

Examples

```
# PE example
#-----------
    require(magrittr)
    bsgenome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
    gr <- char_to_granges(c(PRNP = 'chr20:4699600:+',            # snp
                            HBB  = 'chr11:5227002:-',             # snp
                            HEXA = 'chr15:72346580-72346583:-',   # del
                            CFTR = 'chr7:117559593-117559595:+'), # ins
                          bsgenome)
    gr %<>% extend_pe_to_gg(plot = TRUE) %>% add_seq(bsgenome)
    find_gg(gr)
```

`find_primespacers` *Find prime editing spacers*

Description

Find prime editing spacers around target ranges

Usage

```
find_primespacers(
  gr,
  bsgenome,
  edits = get_plus_seq(bsgenome, gr),
  nprimer = 13,
  nrt = 16,
  ontargetmethod = c("Doench2014", "Doench2016")[1],
  offtargetmethod = c("bowtie", "pdict")[1],
  mismatches = 0,
  nickmatches = 2,
  indexedgenomesdir = INDEXEDGENOMESDIR,
  outdir = OUTDIR,
  verbose = TRUE,
  plot = TRUE,
  ...
)
```

Arguments

`gr`	GRanges-class
`bsgenome`	BSgenome-class
`edits`	character vector: desired edits on '+' strand. If named, names should be identical to those of `gr`
`nprimer`	n primer nucleotides (default 13, max 17)
`nrt`	n rev transcr nucleotides (default 16, recomm. 10-16)
`ontargetmethod`	'Doench2014' or 'Doench2016': on-target scoring method
`offtargetmethod`	'bowtie' or 'pdict'
`mismatches`	no of primespacer mismatches (default 0, to suppress offtarget analysis: -1)
`nickmatches`	no of nickspacer offtarget mismatches (default 2, to suppresses offtarget analysis: -1)
`indexedgenomesdir`	directory with indexed genomes (as created by `index_genome`)
`outdir`	directory whre offtarget analysis output is written
`verbose`	TRUE (default) or FALSE
`plot`	TRUE (default) or FALSE
`...`	passed to plot_intervals

Details

Below the architecture of a prime editing site. Edits can be performed anywhere in the revtranscript area.

spacer pam ————————=== primer revtranscript ————-================ 1.....17....GG..... .....CC..... ————-extension————-

Value

GRanges-class with prime editing spacer ranges and following mcols: * crisprspacer: N20 spacers * crisprpam: NGG PAMs * crisprprimer: primer (on PAM strand) * crisprtranscript: reverse transcript (on PAM strand) * crisprextension: 3' extension of gRNA contains: reverse transcription template + primer binding site sequence can be found on non-PAM strand * crisprextrange: genomic range of crispr extension * Doench2016|4: on-target efficiency scores * off0, off1, off2: number of offtargets with 0, 1, 2 mismatches * off: total number of offtargets: off = off0 + off1 + ... * nickrange: nickspacer range * nickspacer: nickspacer sequence * nickDoench2016|4: nickspacer Doench scores * nickoff: nickspacer offtarget counts

See Also

find_spacers to find standard crispr sites

Examples

```
# Find PE spacers for 4 clinically relevant loci (Anzalone et al, 2019)
    bsgenome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
    gr <- char_to_granges(c(
        PRNP = 'chr20:4699600:+',             # snp: prion disease
        HBB  = 'chr11:5227002:-',             # snp: sickle cell anemia
        HEXA = 'chr15:72346580-72346583:-',   # del: tay sachs disease
        CFTR = 'chr7:117559593-117559595:+'), # ins: cystic fibrosis
        bsgenome)
    spacers <- find_primespacers(gr, bsgenome)
    spacers <- find_spacers(extend_for_pe(gr), bsgenome, complement = FALSE)

# Edit PRNP locus for resistance against prion disease (Anzalone et al, 2019)
    bsgenome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
    gr <- char_to_granges(c(PRNP = 'chr20:4699600:+'), bsgenome)
    find_primespacers(gr, bsgenome)
    find_primespacers(gr, bsgenome, edits = 'T')
```

---

find_spacers *Find crispr spacers in targetranges*

Description

Find crispr spacers in targetranges

Usage

```
find_spacers(
  gr,
  bsgenome,
  spacer = strrep("N", 20),
  pam = "NGG",
  complement = TRUE,
  ontargetmethod = c("Doench2014", "Doench2016")[1],
  offtargetmethod = c("bowtie", "pdict")[1],
  offtargetfilterby = character(0),
  subtract_targets = FALSE,
  mismatches = 2,
  indexedgenomesdir = INDEXEDGENOMESDIR,
  outdir = OUTDIR,
  verbose = TRUE,
  plot = TRUE,
  ...
)
```

Arguments

gr	GRanges-class
bsgenome	BSgenome-class
spacer	string: spacer pattern in extended IUPAC alphabet
pam	string: pam pattern in extended IUPAC alphabet
complement	TRUE (default) or FALSE: also search in compl ranges?
ontargetmethod	'Doench2016','Doench2016' or NULL (no on-target score)
offtargetmethod	'bowtie', 'pdict', or NULL (no offtarget analysis)
offtargetfilterby	filter for best off-target counts by this variable
subtract_targets	TRUE or FALSE (default): whether to subtract target (mis)matches from offtarget counts
mismatches	0-3: allowed mismatches in offtargetanalysis (choose mismatch=-1 to suppress offtarget analysis)
indexedgenomesdir	directory with Bowtie-indexed genomes (as produced with index_genome)
outdir	directory where bowtie analysis results are written to
verbose	TRUE (default) or FALSE
plot	TRUE (default) or FALSE
...	passed to plot_intervals

Value

GRanges-class

See Also

find_primespacers to find prime editing spacers

Examples

```
# PE example
#-----------
    require(magrittr)
    bsgenome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
    gr <- char_to_granges(c(PRNP = 'chr20:4699600:+',             # snp
                            HBB  = 'chr11:5227002:-',             # snp
                            HEXA = 'chr15:72346580-72346583:-',   # del
                            CFTR = 'chr7:117559593-117559595:+'), # ins
                          bsgenome)
    plot_intervals(gr)
    find_primespacers(gr, bsgenome)
    find_spacers(extend_for_pe(gr), bsgenome, complement=FALSE, mismatches=0)
          # complement = FALSE because extend_for_pe  already
          # adds  reverse complements and does so in a strand-specific
          # manner

# TFBS example
#-------------
    bsgenome <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10
    bedfile  <- system.file('extdata/SRF.bed', package='multicrispr')
    gr <- bed_to_granges(bedfile, 'mm10') %>% extend()
    gr %<>% extract(1:100)
    find_spacers(gr, bsgenome, subtract_targets = TRUE)
```

---

genes_to_granges *Convert geneids into GRanges*

Description

Convert geneids into GRanges

Usage

```
genes_to_granges(geneids, txdb, complement = TRUE, plot = TRUE, verbose = TRUE)

genefile_to_granges(file, txdb, complement = TRUE, plot = TRUE)
```

Arguments

geneids	Gene identifier vector
txdb	TxDb-class or EnsDb-class
complement	TRUE (default) or FALSE: add complementary strand?
plot	TRUE (default) or FALSE
verbose	TRUE (default) or FALSE
file	Gene identifier file (one per row)

Value

GRanges-class

See Also

char_to_granges, bed_to_granges

Examples

```
# Entrez
#-------
    genefile <- system.file('extdata/SRF.entrez', package='multicrispr')
    geneids  <- as.character(read.table(genefile)[[1]])
    txdb     <- getFromNamespace('TxDb.Mmusculus.UCSC.mm10.knownGene',
                                 'TxDb.Mmusculus.UCSC.mm10.knownGene')
    (gr <- genes_to_granges(geneids, txdb))
    (gr <- genefile_to_granges(genefile, txdb))

# Ensembl
#--------
    # txdb <- AnnotationHub::AnnotationHub()[["AH75036"]]
    # genefile <- system.file('extdata/SRF.ensembl', package='multicrispr')
    # geneids <- as.character(read.table(genefile)[[1]])
    # (gr <- genes_to_granges(geneids, txdb))
    # (gr <- genefile_to_granges(genefile, txdb))
```

---

gr2dt *GRanges <-> data.table*

Description

GRanges <-> data.table

Usage

```
gr2dt(gr)

dt2gr(dt, seqinfo)
```

Arguments

gr	GRanges-class
dt	data.table
seqinfo	Seqinfo-class

Value

data.table (gr2dt) or GRanges (dt2gr)

Examples

```
bsgenome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
gr <- char_to_granges(c(PRNP = 'chr20:4699600:+',             # snp
                        HBB  = 'chr11:5227002:-',               # snp
                        HEXA = 'chr15:72346580-72346583:-',   # del
                        CFTR = 'chr7:117559593-117559595:+'), # ins
```

```
                            bsgenome)
(dt <- gr2dt(gr))
(gr <- dt2gr(dt, BSgenome::seqinfo(bsgenome)))
```

---

has_been_indexed *Has been indexed?*

Description

Has been indexed?

Usage

```
has_been_indexed(bsgenome, indexedgenomesdir = INDEXEDGENOMESDIR)
```

Arguments

bsgenome	BSgenome
indexedgenomesdir	directory with indexed genomes

Value

TRUE or FALSE

Examples

```
bsgenome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
has_been_indexed(bsgenome)
```

---

index_genome *Index genome*

Description

Bowtie index genome

Usage

```
index_genome(
  bsgenome,
  indexedgenomesdir = INDEXEDGENOMESDIR,
  download = TRUE,
  overwrite = FALSE
)
```

Arguments

bsgenome	BSgenome-class
indexedgenomesdir	string: directory with bowtie-indexed genome
download	TRUE (default) or FALSE: whether to download pre-indexed version if available
overwrite	TRUE or FALSE (default)

Details

Checks whether already available locally. If not, checks whether indexed version can be downloaded from our s3 storage. If not, builds the index with bowtie. This can take a few hours, but is a one-time operation.

Value

invisible(genomdir)

Examples

```
bsgenome <- BSgenome.Scerevisiae.UCSC.sacCer1::Scerevisiae
index_genome(bsgenome, indexedgenomesdir = tempdir())
```

---

index_targets *Index targets*

Description

Bowtie index targets

Usage

```
index_targets(
  targets,
  bsgenome = getBSgenome(genome(targets)[1]),
  outdir = OUTDIR,
  verbose = TRUE
)
```

Arguments

targets	GRanges-class
bsgenome	BSgenome-class
outdir	string: output directory
verbose	TRUE (default) or FALSE

Value

invisible(targetdir)

Examples

```
require(magrittr)
bsgenome <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10
bedfile  <- system.file('extdata/SRF.bed', package = 'multicrispr')
targets <- extend(bed_to_granges(bedfile, genome = 'mm10'))
index_targets(targets, bsgenome)
```

`plot_intervals` *Interval plot GRanges*

Description

Interval plot GRanges

Usage

```
plot_intervals(
  gr,
  xref = "targetname",
  y = default_y(gr),
  nperchrom = 2,
  nchrom = 4,
  color_var = "targetname",
  facet_var = "seqnames",
  linetype_var = default_linetype(gr),
  size_var = default_size_var(gr),
  alpha_var = default_alpha_var(gr),
  title = NULL,
  scales = "free"
)
```

Arguments

`gr`	GRanges-class
`xref`	gr var used for scaling x axis
`y`	'names' (default) or name of gr variable
`nperchrom`	number (default 1): n head (and n tail) targets shown per chromosome
`nchrom`	number (default 6) of chromosomes shown
`color_var`	'seqnames' (default) or other gr variable
`facet_var`	NULL(default) or gr variable mapped to facet
`linetype_var`	NULL (default) or gr variable mapped to linetype
`size_var`	NULL (default) or gr variable mapped to size
`alpha_var`	NULL or gr variable mapped to alpha
`title`	NULL or string: plot title
`scales`	'free', 'fixed', etc

Value

ggplot object

See Also

plot_karyogram

Examples

```
# SRF sites
    require(magrittr)
    bsgenome <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10
    bedfile <-  system.file('extdata/SRF.bed',  package = 'multicrispr')
    targets   <- bed_to_granges(bedfile, 'mm10', plot = FALSE)
    plot_intervals(targets)

# PE targets
    bsgenome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
    gr <- char_to_granges(c(PRNP = 'chr20:4699600:+',
                            HBB  = 'chr11:5227002:-',
                            HEXA = 'chr15:72346580-72346583:-',
                            CFTR = 'chr7:117559593-117559595:+'),
                          bsgenome)
    spacers <- find_primespacers(gr, bsgenome, plot = FALSE)
    plot_intervals(gr)
    plot_intervals(extend_for_pe(gr))
    plot_intervals(spacers)

# Empty gr
    plot_intervals(GenomicRanges::GRanges())
```

---

plot_karyogram *Karyo/Interval Plot GRanges(List)*

Description

Karyo/Interval Plot GRanges(List)

Usage

```
plot_karyogram(grlist, title = unique(genome(grlist)))
```

Arguments

grlist	GRanges-class
title	plot title

Value

list

See Also

plot_intervals

Examples

```
# Plot GRanges
    bedfile <-  system.file('extdata/SRF.bed',  package = 'multicrispr')
    gr <- bed_to_granges(bedfile, 'mm10', plot = FALSE)
    plot_karyogram(gr)

# Plot GRangesList
    flanks  <- up_flank(gr, stranded=FALSE)
    grlist <- GenomicRanges::GRangesList(sites = gr, flanks = flanks)
    plot_karyogram(grlist)
```

`score_ontargets` *Add on-target efficiency scores*

Description

Add Doench2014 or Doench2016 on-target efficiency scores

Usage

```
score_ontargets(
  spacers,
  bsgenome,
  ontargetmethod = c("Doench2014", "Doench2016")[1],
  chunksize = 10000,
  verbose = TRUE,
  plot = TRUE,
  ...
)
```

Arguments

`spacers`	`GRanges-class`: spacers
`bsgenome`	`BSgenome-class`
`ontargetmethod`	'Doench2014' (default) or 'Doench2016' (requires non-NULL argument python, virtualenv, or condaenv)
`chunksize`	Doench2016 is executed in chunks of chunksize
`verbose`	TRUE (default) or FALSE
`plot`	TRUE (default) or FALSE
`...`	passed to `plot_intervals`

Details

`add_ontargets` adds efficiency scores `filter_ontargets` adds efficiency scores and filters on them

Value

numeric vector

References

Doench 2014, Rational design of highly active sgRNAs for CRISPR-Cas9-mediated gene inactivation. Nature Biotechnology, doi: 10.1038/nbt.3026

Doench 2016, Optimized sgRNA design to maximize activity and minimize off-target effects of CRISPR-Cas9. Nature Biotechnology, doi: 10.1038/nbt.3437

Python module azimuth: github/MicrosoftResearch/azimuth

Examples

```
# Install azimuth
#----------------
    ## With reticulate
    # require(reticulate)
    # conda_create('azienv', c('python=2.7'))
    # use_condaenv('azienv')
    # py_install(c('azimuth', 'scikit-learn==0.17.1', 'biopython=='1.76'),
    #            'azienv', pip = TRUE)

    ## Directly
    # conda create --name azienv python=2.7
    # conda activate azienv
    # pip install scikit-learn==0.17.1
    # pip install biopython==1.76
    # pip install azimuth

# PE example
#-----------
    require(magrittr)
    bsgenome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
    targets <- char_to_granges(c(PRNP = 'chr20:4699600:+',             # snp
                                 HBB  = 'chr11:5227002:-',             # snp
                                 HEXA = 'chr15:72346580-72346583:-',   # del
                                 CFTR = 'chr7:117559593-117559595:+'), # ins
                               bsgenome)
    spacers <- find_primespacers(targets, bsgenome, ontargetmethod=NULL,
                                 offtargetmethod=NULL)
    spacers %<>% score_ontargets(bsgenome, 'Doench2014')
    # reticulate::use_condaenv('azienv')
    # reticulate::import('azimuth')
    # spacers %<>% score_ontargets(bsgenome, 'Doench2016')

# TFBS example
#-------------
    bedfile  <- system.file('extdata/SRF.bed', package = 'multicrispr')
    bsgenome <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10
    targets <- extend(bed_to_granges(bedfile, 'mm10'))
    spacers <- find_spacers(targets, bsgenome, ontargetmethod=NULL,
                            offtargetmethod=NULL)
    spacers %<>% score_ontargets(bsgenome, 'Doench2014')
    # reticulate::use_condaenv('azienv')
    # reticulate::import('azimuth')
    # spacers %>%  score_ontargets(bsgenome, 'Doench2016')
```

up_flank *Extend or Flank GRanges*

Description

Returns extensions, upstream flanks, or downstream flanks

Usage

```
up_flank(
  gr,
  start = -200,
  end = -1,
  strandaware = TRUE,
  bsgenome = NULL,
  verbose = FALSE,
  plot = FALSE,
  linetype_var = "set",
  ...
)

down_flank(
  gr,
  start = 1,
  end = 200,
  strandaware = TRUE,
  bsgenome = NULL,
  verbose = FALSE,
  plot = FALSE,
  linetype_var = "set",
  ...
)

extend(
  gr,
  start = -22,
  end = 22,
  strandaware = TRUE,
  bsgenome = NULL,
  verbose = FALSE,
  plot = FALSE,
  linetype_var = "set",
  ...
)
```

Arguments

gr	GRanges-class
start	number or vector (same length as gr): start definition, relative to gr start (up_flank, extend) or gr end (down_flank).

end	number or vector (same length as gr): end definition, relative to gr start (up_flank) or gr end (extend, down_flank).
strandaware	TRUE (default) or FALSE: consider strand information?
bsgenome	NULL (default) or BSgenome-class. Required to update gr$seq if present.
verbose	TRUE or FALSE (default)
plot	TRUE or FALSE (default)
linetype_var	string: gr var mapped to linetype
...	passed to plot_intervals

Details

up_flank returns upstream flanks, in relation to start(gr). down_flank returns downstream flanks, in relation to end(gr). extend returns extensions, in relation to start(gr) and end(gr)

Value

a GRanges-class

Examples

```
# PE example
#-----------
require(magrittr)
bsgenome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
gr <- char_to_granges(c(PRNP  = 'chr20:4699600:+',         # snp
                        HBB   = 'chr11:5227002:-',            # snp
                        HEXA = 'chr15:72346580-72346583:-',  # del
                        CFTR = 'chr7:117559593-117559595:+'),# ins
                      bsgenome = bsgenome)
gr %>% up_flank( -22,  -1, plot=TRUE)
gr %>% up_flank( c(-10,-20,-30,-40),  -1, plot=TRUE)
gr %>% up_flank( -22,  -1, plot=TRUE, strandaware=FALSE)

gr %>% down_flank(+1, +22, plot=TRUE)
gr %>% down_flank(+1, c(10, 20, 30, 40), plot=TRUE)
gr %>% down_flank(+1, +22, plot=TRUE, strandaware=FALSE)

gr %>% extend(   -10, +20, plot=TRUE)
gr %>% extend(   -10, +20, plot=TRUE, strandaware=FALSE)

# TFBS example
#-------------
    bedfile <- system.file('extdata/SRF.bed', package='multicrispr')
    gr <- bed_to_granges(bedfile, genome = 'mm10')
    gr %>% extend(plot = TRUE)
    gr %>% up_flank(plot = TRUE)
    gr %>% down_flank(plot = TRUE)
```

write_ranges *Write GRanges to file*

Description

Write GRanges to file

Usage

```
write_ranges(gr, file, verbose = TRUE)

read_ranges(file, bsgenome)
```

Arguments

gr	GRanges-class
file	file
verbose	TRUE (default) or FALSE
bsgenome	BSgenome-class

Value

GRanges-class for read_ranges

Examples

```
# Find PE spacers for 4 clinically relevant loci (Anzalone et al, 2019)
    bsgenome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
    gr <- char_to_granges(c(
        PRNP = 'chr20:4699600:+',             # snp: prion disease
        HBB  = 'chr11:5227002:-',             # snp: sickle cell anemia
        HEXA = 'chr15:72346580-72346583:-',   # del: tay sachs disease
        CFTR = 'chr7:117559593-117559595:+'), # ins: cystic fibrosis
        bsgenome)
    file <- file.path(tempdir(), 'gr.txt')
    write_ranges(gr, file)
    read_ranges(file, bsgenome)
```

Index

add_genome_matches, 2
add_inverse_strand, 3
add_seq, 4
add_target_matches, 5

bed_to_granges, 6, *7*, *17*
BSgenome, *11*

char_to_granges, *6*, 7, *17*

double_flank, 7
down_flank *(*up_flank*)*, 24
dt2gr *(*gr2dt*)*, 17

extend *(*up_flank*)*, 24
extend_for_pe, 9
extend_pe_to_gg, 10
extract_matchranges, 10
extract_subranges, 11

find_gg, 12
find_primespacers, 13, *15*
find_spacers, *14*, 14

genefile_to_granges *(*genes_to_granges*)*, 16
genes_to_granges, *6, 7*, 16
gr2dt, 17

has_been_indexed, 18

index_genome, *13*, *15*, 18
index_targets, 19

plot_intervals, *4*, 20, *21, 22*, *25*
plot_karyogram, *21*, 21

read_ranges *(*write_ranges*)*, 26

score_ontargets, 22

up_flank, 24

write_ranges, 26