Package 'CrispRVariants'

December 20, 2024

Type Package

Title Tools for counting and visualising mutations in a target location

Version 1.35.0

Author Helen Lindsay [aut, cre]

Description CrispRVariants provides tools for analysing the results of a CRISPR-Cas9 mutagenesis sequencing experiment, or other sequencing experiments where variants within a given region are of interest. These tools allow users to localize variant allele combinations with respect to any genomic location (e.g. the Cas9 cut site), plot allele combinations and calculate mutation rates with flexible filtering of unrelated variants.

biocViews ImmunoOncology, CRISPR, GenomicVariation, VariantDetection, GeneticVariability, DataRepresentation, Visualization, Sequencing

Depends R (>= 4.3.0), ggplot2 (>= 2.2.0)

Encoding UTF-8

License GPL-2

Imports AnnotationDbi, BiocParallel, Biostrings, methods, GenomeInfoDb, GenomicAlignments, GenomicRanges, grDevices, grid, gridExtra, IRanges, reshape2, Rsamtools, S4Vectors (>= 0.9.38), utils

Suggests BiocStyle, GenomicFeatures, knitr, rmarkdown, readxl, rtracklayer, sangerseqR, testthat, VariantAnnotation

VignetteBuilder knitr

NeedsCompilation no

RoxygenNote 7.2.3

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

git_branch devel

git_last_commit aa34499

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2024-12-20

Maintainer Helen Lindsay <helen.lindsay@chuv.ch>

Contents

`.explodeCigarOpCombs` *.explodeCigarOpCombs*

Description

Breaks cigar strings into individual operations

Usage

```
.explodeCigarOpCombs(cigar, ops = GenomicAlignments::CIGAR_OPS)
```

Arguments

`cigar`	character(m) A vector of cigar strings
`ops`	character(n) Which operations should be kept?

Value

The operations, as a CharacterList

Exploded cigar operations with operation widths

Author(s)

Helen Lindsay

`.formatVarLabels` *formatVarLabels*

Description

Internal CrispRVariants function for creating allele labels given variants and positions

Assume that the reference may be on the negative strand and regions are given with respect to the reference sequence.

Usage

```
.formatVarLabels(
  grl,
  labels,
  position = c("start", "end"),
  genome.to.pos = NULL,
  pos.to.lab = ":",
  as.string = TRUE
)

.findMismatches(
  alns,
  ref.seq,
  ref.start,
  regions = NULL,
  strand = "+",
  min.pct = 0
)
```

Arguments

`grl`	(GRangesList) A GRangesList of variants. The position of the variants is used in labels
`labels`	(character(n)) A vector of labels for each variant. In CrispRVariants, this is the size and type of the variant, e.g. "9D" for a 9 bp deletion.
`position`	One of "start" and "end". Determines whether the start or the end coordinate is used when labeling variants.
`genome.to.pos`	Optional named vector for transforming variant coordinates into another coordinate system (Default: NULL)
`pos.to.lab`	(character(1)) Character to join positions and labels (Default: ":", e.g. -3:9D)
`as.string`	Should individual variant labels be pasted into a single comma separated string when one alignment has multiple variants? (Default: TRUE)
`alns`	A GAlignments object, where the aligned sequences should span the reference sequence

ref.seq	A DNAString object, the sequence for comparison when checking for mismatches. The sequence does not necessarily have to match the mapping reference sequence. Must span all regions if regions are provided.
ref.start	(numeric(1)) The genomic start position of the reference sequence
regions	A GRanges object, regions to check for mismatches with coordinates relative to the reference sequence
strand	One of "+", "-"
min.pct	(numeric(1), between 0 and 100) Only return SNVs that occur at in least min.pct change, not any change at a position.

Value

A data frame of sequence indices, genomic position of mismatch and mismatch base

---

.getAxisCoords *.getAxisCoords*

Description

Manually specify x-tick locations and labels, as sometimes ggplot defaults are too dense. Used internally by CrispRVariants for creating alignment plot with plotAlignments.

Usage

```
.getAxisCoords(
  locations,
  labels = NULL,
  loc.boundaries = NULL,
  lab.boundaries = c(-1, 1),
  label.at = 5,
  min.tick.sep = 1
)
```

Arguments

locations	character(n) Actual x coordinates, or the desired range of the x coordinates. If labels are provided, all tick locations must be in locations and have a matching label.
labels	character(n) labels for the x axis ticks. Should be the same length as locations if provided. Note that if not all tick locations are included in locations, it must be possible to extrapolate labels from locations (Default: NULL)
loc.boundaries	numeric(i) Locations that must be included. (Default: NULL)
lab.boundaries	numeric(j) Labels that must be included. (Default: c(-1,1), for showing the cut sites). Boundaries must be in labels and have a matching tick location.
label.at	numeric(1) Add ticks when label modulo label.at is zero (Default = 5)
min.tick.sep	numeric(1) Minimum distance between ticks, excluding boundary ticks. (Default: 1)

Value

A list containing vectors named tick_locs and tick_labs

Author(s)

Helen Lindsay

`.intersperse` *.intersperse*

Description

create a vector of elements in outer interspersed with elements in inner. Similar to python zip. No element checking.

Usage

```
.intersperse(outer, inner)
```

Arguments

`outer`	vector that will be the first and last elements
`inner`	vector that will join elements of outer

Value

A vector interspersing elements of outer and inner. If outer is c(a,b,c) and inner is c(d,e), returns c(a,d,b,e,c)

Author(s)

Helen Lindsay

Examples

```
CrispRVariants:::.intersperse(c(1:10), c(1:9)*10)
```

`.invertKeepRanges` *Helper functions for selectAlnRegions*

Description

(.invertKeepRanges) Internal CrispRVariants function used by seqsToPartialAlns for checking arguments and getting region to delete. Returns FALSE if no region to delete found, or region to be deleted is entire target.

(.checkRelativeLocs) Shift keep to start at 1 if it is within the target

(.adjustRelativeInsLocs) Internal CrispRVariants function for shifting insertion locations relative to the target region when removing a segment of the alignments. Note that this function does not do input checking but assumes this has been done upstream. Insertions at the left border of a gap region are removed.

(.offsetIndices) Get indices of a vector grouped by "x", cumulatively adding offsets to each group according to "offset"

Usage

```
.invertKeepRanges(target, keep)

.checkRelativeLocs(target, keep)

.adjustRelativeInsLocs(target, keep, starts, gap_nchars)

.offsetIndices(x, offset)
```

Arguments

`target`	The complete region spanned by the alignments (GRanges)
`keep`	Region to display, relative to the target region, i.e. not genomic coords (IRanges or GRanges)
`starts`	numeric(n) Insertion locations. When plotting, the insertion symbol appears at the left border of the start location.
`gap_nchars`	character(n) Number of letters added to when joining segments before each region in keep. If first base of keep is 1, the first entry of gap_nchars should be 0.
`x`	A vector of group lengths
`offset`	A vector of offset lengths matching x

Value

Gaps between keep (IRanges), or FALSE if no gap ranges found

insertion_sites (data.frame) with modified start column

Author(s)

Helen Lindsay

Examples

```
CrispRVariants:::.offsetIndices(rep(2,5), c(0:4)*10)
```

`abifToFastq` *Read a file in ab1 (Sanger) format and convert to fastq*

Description

This is an R implementation of Wibowo Arindrarto's abifpy.py trimming module, which itself implement's Richard Mott's trimming algorithm See https://github.com/bow/abifpy for more details.

Usage

```
abifToFastq(
  seqname,
  fname,
  outfname,
  trim = TRUE,
  cutoff = 0.05,
  min_seq_len = 20,
  offset = 33,
  recall = FALSE
)
```

Arguments

`seqname`	name of sequence, to appear in fastq file
`fname`	filename of sequence in ab1 format
`outfname`	filename to append the fastq output to
`trim`	should low quality bases be trimmed from the ends? TRUE or FALSE
`cutoff`	probability cutoff
`min_seq_len`	minimum number of sequenced bases required in order to trim the read
`offset`	phred offset for quality scores
`recall`	Use sangerseqR to resolve the primary sequence if two sequences are present. May cause quality scores to be ignored. (Default: FALSE)

Details

Requires Bioconductor package SangerseqR

Value

None. Sequences are appended to the outfname.

Author(s)

Helen Lindsay

Examples

```
# Running this code will write the fastq file to "IM2033.fastq"
ab1_fname <- system.file("extdata", "IM2033.ab1", package = "CrispRVariants")
abifToFastq("IM2033", ab1_fname, "IM2033.fastq")
```

`addClipped` *Extrapolates mapping location from clipped, aligned reads*

Description

Extrapolates the mapping location of a read by assuming that the clipped regions should map adjacent to the mapped locations. This is not always a good assumption, particularly in the case of chimeric reads!

Usage

```
addClipped(bam, ...)

## S4 method for signature 'GAlignments'
addClipped(bam, ...)
```

Arguments

`bam`	A GAlignments object
`...`	additional arguments

Value

A `GRanges` representation of the extended mapping locations

Author(s)

Helen Lindsay

addCodonFrame *Internal CrispRVariants function for indicating codon frame on an alignment tile plot*

Description

Adds vertical dotted lines in intervals of three nucleotides. Codon frame is supplied, alignments are assumed not to span an intron-exon junction.

Usage

```
addCodonFrame(p, width, codon.frame)
```

Arguments

p	A ggplot object, typically from CrispRVariants:::makeAlignmentTilePlot
width	The number of nucleotides in the alignments
codon.frame	The leftmost starting location of the next codon - 1,2,or 3

Value

A ggplot object with added vertical lines indicating the frame

Author(s)

Helen Lindsay

alleles *Get allele names*

Description

Function to access allele names

Usage

```
alleles(obj, ...)

## S4 method for signature 'CrisprSet'
alleles(obj, ...)
```

Arguments

obj	An object containing variant alleles
...	additional arguments

Value

A data frame relating CIGAR strings to variant labels

Author(s)

Helen Lindsay

Examples

```
data("gol_clutch1")
alleles <- alleles(gol)
```

`alns` *Get alignments*

Description

Return alignments from an object that contains them. For a CrisprSet object, these are truncated, non-chimeric alignments

Usage

```
alns(obj, ...)

## S4 method for signature 'CrisprSet'
alns(obj, ...)
```

Arguments

`obj`	An object containing aligned sequences
`...`	additional arguments

Value

A GAlignmentsList of consensus sequences on the positive strand.

Author(s)

Helen Lindsay

Examples

```
data("gol_clutch1")
alns <- alns(gol)
```

`annotateGenePlot` *Plots and annotates transcripts*

Description

Plots the gene structure, annotates this with the target location

Usage

```
annotateGenePlot(
  txdb,
  target,
  target.colour = "red",
  target.size = 1,
  gene.text.size = 10,
  panel.spacing = grid::unit(c(0.1, 0.1, 0.1, 0.1), "lines"),
  plot.title = NULL,
  all.transcripts = TRUE
)
```

Arguments

`txdb`	A GenomicFeatures:TxDb object
`target`	Location of target (GRanges)
`target.colour`	Colour of box indicating targt region
`target.size`	Thickness of box indicating target region
`gene.text.size`	Size for figure label
`panel.spacing`	Unit object, margin size
`plot.title`	A title for the plot. If no plot.title is supplied, the title is the list of gene ids shown (default). If plot.title == FALSE, the plot will not have a title.
`all.transcripts`	If TRUE (default), all transcripts of genes overlapping the target are shown, including transcripts that do not themselves overlap the target. If FALSE, only the transcripts that overlap the target are shown.

Value

A ggplot2 plot of the transcript structures

`arrangePlots` *Arrange plots for plotVariants:CrisprSet*

Description

Arranges 3 plots in two rows. The vertical margins of the left.plot and right.plot constrained to be equal

Usage

```
arrangePlots(
  top.plot,
  left.plot,
  right.plot,
  fig.height = NULL,
  col.wdth.ratio = c(2, 1),
  row.ht.ratio = c(1, 6),
  left.plot.margin = grid::unit(c(0.1, 0.2, 3, 0.2), "lines")
)
```

Arguments

`top.plot`	ggplot grob, placed on top of the figure, spanning the figure width
`left.plot`	ggplot, placed in the second row on the left
`right.plot`	ggplot, placed in the second row on the right. y-axis labels are removed.
`fig.height`	Actual height for the figure. If not provided, figure height is the sum of the row.ht.ratio (Default: NULL)
`col.wdth.ratio`	Vector specifying column width ratio (Default: c(2, 1))
`row.ht.ratio`	Vector specifying row height ratio (Default: c(1,6))
`left.plot.margin`	Unit object specifying margins of left.plot. Margins of right.plot are constrained by the left.plot.

Value

The arranged plots

`barplotAlleleFreqs` *Plots barplots of the spectrum of variants for a sample set*

Description

For signature "matrix", this function optionally does a very naive classification of variants by size. Frameshift variant combinations are those whose sum is not divisible by three. Intron boundaries are *NOT* considered, use with caution! For signature "CrisprSet", the function uses the VariantAnnotation package to localize variant alleles with respect to annotated transcripts. Variants are annotated as "coding" when they are coding in any transcript.

(signature("CrisprSet")) Groups variants by size and type and produces a barplot showing the variant spectrum for each sample. Accepts all arguments accepted by barplotAlleleFreqs for signature("matrix"). Requires package "VariantAnnotation"

signature("matrix") Accepts a matrix of allele counts, with rownames being alleles and column names samples.

Usage

```
barplotAlleleFreqs(obj, ...)

## S4 method for signature 'CrisprSet'
barplotAlleleFreqs(
  obj,
  ...,
  txdb,
  min.freq = 0,
  include.chimeras = TRUE,
  group = NULL,
  palette = c("rainbow", "bluered")
)

## S4 method for signature 'matrix'
barplotAlleleFreqs(
  obj,
  category.labels = NULL,
  group = NULL,
  bar.colours = NULL,
  group.colours = NULL,
  legend.text.size = 10,
  axis.text.size = 10,
  legend.symbol.size = 1,
  snv.label = "SNV",
  novar.label = "no variant",
  chimera.label = "Other",
  include.table = TRUE,
  classify = TRUE
)
```

Arguments

obj	The object to be plotted
...	additional arguments
txdb	A transcript database object
min.freq	Include variants with at frequency least min.freq in at least one sample. (Default: 0, i.e. no cutoff)
include.chimeras	Should chimeric reads be included in results? (Default: TRUE)
group	A grouping factor for the columns in obj. Columns in the same group will be displayed in the same text colour (Default: NULL)
palette	Colour palette. Options are "rainbow", a quantitative palette (default) or "bluered", a gradient palette.
category.labels	Labels for each category, corresponding to the rows of obj. Only applicable when categories are provided, i.e. "classify" is FALSE. (Default: NULL)
bar.colours	Colours for the categories in the barplot. Colours must be provided if there are more than 6 different categories.
group.colours	Colours for the text labels for the experimental groups A set of 15 different colours is provided.
legend.text.size	The size of the legend text, in points.
axis.text.size	The size of the axis text, in points
legend.symbol.size	The size of the symbols in the legend
snv.label	The row label for single nucleotide variants
novar.label	The row label for non-variant sequences
chimera.label	The row label for chimeric (non-linearly aligned) variant alleles
include.table	Should a table of allele (variant combination) counts and total sequences be plotted? (Default: TRUE)
classify	If TRUE, performs a naive classification by size (Default:TRUE)

Value

A ggplot2 barplot of the allele distribution and optionally a table of allele counts

Author(s)

Helen Lindsay

Examples

```
data("gol_clutch1")
barplotAlleleFreqs(variantCounts(gol))

# Just show the barplot without the counts table:
barplotAlleleFreqs(variantCounts(gol), include.table = FALSE)
```

`collapsePairs` *Internal CrispRVariants function for collapsing pairs with concordant indels*

Description

Given a set of alignments to a target region, finds read pairs. Compares insertion/deletion locations within pairs using the cigar string. Pairs with non-identical indels are excluded. Pairs with identical indels are collapsed to a single read, taking the consensus sequence of the pairs.

Usage

```
collapsePairs(
  alns,
  use.consensus = TRUE,
  keep.unpaired = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

`alns`	A GAlignments object. We do not use GAlignmentPairs because amplicon-seq can result in pairs in non-standard pairing orientation. Must include BAM flag, must not include unmapped reads.
`use.consensus`	Should the consensus sequence be used if pairs have a mismatch? Setting this to be TRUE makes this function much slower (Default: TRUE)
`keep.unpaired`	Should unpaired and chimeric reads be included? (Default: TRUE)
`verbose`	Report statistics on reads kept and excluded
`...`	Additional items with the same length as alns, that should be filtered to match alns.

Value

The alignments, with non-concordant pairs removed and concordant pairs represented by a single read.

Author(s)

Helen Lindsay

consensusSeqs *Get consensus sequences for variant alleles*

Description

Return consensus sequences of variant alleles. At present, chimeric alignments are not included.

Usage

```
consensusSeqs(obj, ...)

## S4 method for signature 'CrisprSet'
consensusSeqs(obj, ..., top.n = NULL, min.freq = 0, min.count = 1)
```

Arguments

obj	An object containing aligned sequences
...	additional arguments
top.n	(Integer n) If specified, return variants ranked at least n according to frequency across all samples (Default: 0, i.e. no cutoff)
min.freq	(Float n least one sample (Default: 0)
min.count	(Integer n) Return variants with count greater than n in at least one sample (Default: 0)

Value

A DNAStringSet of consensus sequences on the positive strand.

Author(s)

Helen Lindsay

Examples

```
data("gol_clutch1")
seqs <- consensusSeqs(gol, sample = 2)
```

countDeletions *Count the number of reads containing an insertion or deletion*

Description

Counts the number of reads containing a deletion or insertion (indel) of any size in a set of aligned reads. For countDeletions and countInsertions Reads may be filtered according to whether they contain more than one indel of the same or different types.

Usage

```
countDeletions(alns, ...)

## S4 method for signature 'GAlignments'
countDeletions(
  alns,
  ...,
  multi.del = FALSE,
  del.and.ins = FALSE,
  del.ops = c("D")
)

countInsertions(alns, ...)

## S4 method for signature 'GAlignments'
countInsertions(
  alns,
  ...,
  ins.and.del = FALSE,
  multi.ins = FALSE,
  del.ops = c("D")
)

countIndels(alns)

## S4 method for signature 'GAlignments'
countIndels(alns)

indelPercent(alns)

## S4 method for signature 'GAlignments'
indelPercent(alns)
```

Arguments

alns	The aligned reads
...	extra arguments

multi.del	If TRUE, returns the exact number of deletions, i.e., if one read contains 2 deletions, it contributes 2 to the total count (default: FALSE)
del.and.ins	If TRUE, counts deletions regardless of whether reads also contain insertions. If FALSE, counts reads that contain deletions but not insertions (default: FALSE)
del.ops	Cigar operations counted as deletions. Default: c("D")
ins.and.del	If TRUE, counts insertions regardless of whether reads also contain deletions If FALSE, counts reads that contain insertions but not deletions (default: FALSE)
multi.ins	If TRUE, returns the exact number of insertions, i.e., if one read contains 2 insertions, it contributes 2 to the total count (default: FALSE)

Value

countDeletions: The number of reads containing a deletion (integer)

countInsertions: The number of reads containing an insertion (integer)

countIndels: The number of reads containing at least one insertion

indelPercent: The percentage of reads containing an insertion or deletion (numeric)

Author(s)

Helen Lindsay

Examples

```
bam_fname <- system.file("extdata", "gol_F1_clutch_2_embryo_4_s.bam",
                         package = "CrispRVariants")
bam <- GenomicAlignments::readGAlignments(bam_fname, use.names = TRUE)
countDeletions(bam)
countInsertions(bam)
countIndels(bam)
indelPercent(bam)
```

CrisprRun-class *CrisprRun class*

Description

A ReferenceClass container for a single sample of alignments narrowed to a target region. Typically CrisprRun objects will not be accessed directly, but if necessary via a CrisprSet class which contains a list of CrisprRun objects. Note that the CrispRVariants plotting functions don't work on CrisprRun objects.

Arguments

bam	a GAlignments object containing (narrowed) alignments to the target region. Filtering of the bam should generally be done before initialising a CrisprRun object
target	The target location, a GRanges object
genome.ranges	A GRangesList of genomic coordinates for the cigar operations. If bam is a standard GAlignments object, this is equivalent to cigarRangesAlongReferenceSpace + start(bam)
rc	(reverse complement) Should the alignments be reverse complemented, i.e. displayed with respect to the negative strand? (Default: FALSE)
name	A name for this set of reads, used in plots if present (Default: NULL)
chimeras	Off-target chimeric alignments not in bam. (Default: empty)
verbose	Print information about initialisation progress (Default: FALSE)

Fields

`alns` A GAlignments object containing the narrowed reads. Note that if the alignments are represented with respect to the reverse strand, the "start" remains with repect to the forward strand, whilst the cigar and the sequence are reverse complemented.

`name` The name of the sample

`cigar_labels` A vector of labels for the reads, based on the cigar strings, optionally renumbered with respect to a new zero point (e.g. the cut site) and shortened to only insertion and deletion locations. Set at initialisation of a CrisprSet object, but not at initialisation of a CrisprRun object.

`chimeras` Chimeric, off-target alignments corresponding to alignments in alns

Methods

`getCigarLabels( target, target.loc, genome_to_target, ref, separate.snv, rc, match.label, mismatch.labe`
Description: Sets the "cig_labels" field, returns the cigar labels.

Input parameters: target: (GRanges) the counting region. target.loc: The location of the cut site with respect to the target genome_to_target: A vector with names being genomic locations and values being locations with respect to the cut site separate.snv: Should single nucleotide variants be called? (Default: TRUE) match.label: Label for non-variant reads (Default: no variant) mismatch.label: Label for single nucleotide variants (Default: SNV) rc: Should the variants be displayed with respect to the negative strand? (Default: FALSE) keep.ops: CIGAR operations to remain in the variant label (usually indels) upstream: distance upstream of the cut site to call SNVs downstream: distance downstream of the cut site to call SNVs regions: IRanges(k) Regions for counting insertions and deletions. Insertions on the right border are not counted. snv.regions Regions for calling SNVS

`getInsertionSeqs(target)` Description: Return a table relating insertion sequences to alignment indices Input parameters:

Author(s)

Helen Lindsay

See Also

CrisprSet

Examples

```
# readsToTarget with signature("GAlignments", "GRanges") returns a
# CrisprRun object

bam_fname <- system.file("extdata", "gol_F1_clutch_1_embryo_1_s.bam",
                         package = "CrispRVariants")
param <- Rsamtools::ScanBamParam(what = c("seq", "flag"))
alns <- GenomicAlignments::readGAlignments(bam_fname, param = param,
         use.names = TRUE)

reference <- Biostrings::DNAString("GGTCTCTCGCAGGATGTTGCTGG")
gd <- GenomicRanges::GRanges("18", IRanges::IRanges(4647377, 4647399), strand = "+")

crispr_run <- readsToTarget(alns, target = gd, reference = reference,
                            name = "Sample name", target.loc = 17)

# Alternatively, CrisprRun objects can be accessed from a CrisprSet object
# e.g. crispr_set$crispr_runs[[1]]
```

CrisprSet-class *CrisprSet class*

Description

A ReferenceClass container for holding a set of narrowed alignments, each corresponding to the same target region. Individual samples are represented as CrisprRun objects. CrisprRun objects with no on-target reads are excluded. CrisprSet objects are constructed with readsToTarget or readsToTargets. For most use cases, a CrisprSet object should not be initialized directly.

Arguments

crispr.runs	A list of CrisprRun objects, typically representing individual samples within an experiment
reference	The reference sequence, must be the same length as the target region
target	The target location (GRanges). Variants will be counted over this region. Need not correspond to the guide sequence.
rc	Should the alignments be reverse complemented, i.e. displayed w.r.t the reverse strand? (default: FALSE)
names	A list of names for each of the samples, e.g. for displaying in plots. If not supplied, the names of the crispr.runs are used, which default to the filenames of the bam files if available (Default: NULL)

renumbered	Should the variants be renumbered using target.loc as the zero point? If TRUE, variants are described by the location of their 5'-most base with respect to the target.loc. A 3bp deletion starting 5bp 5' of the cut site would be labelled as -5:3D (Default: TRUE)
target.loc	The location of the Cas9 cut site with respect to the supplied target. (Or some other central location). Can be displayed on plots and used as the zero point for renumbering variants. For a target region with the PAM location from bases 21-23, the target.loc is base 17 (default: 17)
match.label	Label for sequences with no variants (default: "no variant")
mismatch.label	Label for sequences with only single nucleotide variants (default: "SNV")
bpparam	A BiocParallel parameter object specifying how many cores to use. The parallelisable step is calling SNVs. Parallelisation is by sample. (default: SerialParam, i.e. no parallelization)
verbose	If true, prints information about initialisation progress (default: TRUE)

Fields

`crispr_runs` A list of CrisprRun objects, typically corresponding to samples of an experiment.

`ref` The reference sequence for the target region, as a Biostrings::DNAString object

`cigar_freqs` A matrix of counts for each variant

`target` The target location, as a GRanges object

Methods

`classifyCodingBySize(var_type, cutoff = 10)` Description: This is a naive classification of variants as frameshift or in-frame Coding indels are summed, and indels with sum divisible by 3 are considered frameshift. Note that this may not be correct for variants that span an intron-exon boundary Input paramters: var_type: A vector of var_type. Only variants with var_type == "coding" are considered. Intended to work with classifyVariantsByLoc cutoff: Variants are divided into those less than and greater than "cutoff" (Default: 10) Result: A character vector with a classification for each variant allele

`classifyVariantsByLoc(txdb, add_chr = TRUE, verbose = TRUE, ...)` Description: Uses the VariantAnnotation package to look up the location of the variants. VariantAnnotation allows multiple classification tags per variant, this function returns a single tag. The following preference order is used: spliceSite > coding > intron > fiveUTR > threeUTR > promoter > intergenic

Input parameters: txdb: A BSgenome transcription database add_chr: Add "chr" to chromosome names to make compatible with UCSC (default: TRUE) verbose: Print progress (default: TRUE) ...: Filtering arguments for variantCounts

Return value: A vector of classification tags, matching the rownames of .self$cigar_freqs (the variant count table)

`classifyVariantsByType(...)` Description: Classifies variants as insertions, deletions, or complex (combinations). In development Input parameters: ... Optional arguments to "variantCounts" for filtering variants before classification Return value: A named vector classifying variant alleles as insertions, deletions, etc

`consensusAlleles( cig_freqs = .self$cigar_freqs, return_nms = TRUE, match.ops = c("M", "X", "=") )`
Description: Get variants by their cigar string, make the pairwise alignments for the consensus sequence for each variant allele

Input parameters: cig_freqs: A table of variant allele frequencies (by default: .self$cigar_freqs, but could also be filtered) return_nms: If true, return a list of sequences and labels (Default:FALSE) match.ops: CIGAR operations for 1-1 alignments

Return: A DNAStringSet of the consensus sequences for the specified alleles, or a list containing the consensus sequences and names for the labels if return_nms = TRUE

`filterUniqueLowQual(min_count = 2, max_n = 0, verbose = TRUE)` Description: Deletes reads containing rare variant combinations and more than a minimum number of ambiguity characters within the target region. These are assumed to be alignment errors.

Input parameters: min_count: the number of times a variant combination must occur across all samples to keep (default: 2, i.e. a variant must occur at least twice in one or more samples to keep) max_n: maximum number of ambiguity ("N") bases a read with a rare variant combination may contain. (default: 0) verbose: If TRUE, print the number of sequences removed (default: TRUE)

`filterVariants( cig_freqs = NULL, names = NULL, columns = NULL, include.chimeras = TRUE )`
Description: Relabels specified variants in a table of variant allele counts as non-variant, e.g. variants known to exist in control samples. Accepts either a size, e.g. "1D", or a specific mutation, e.g. "-4:3D". For alleles that include one variant to be filtered and one other variant, the other variant will be retained. If SNVs are included, these will be removed entirely, but note that SNVs are only called in reads that do not contain an insertion/deletion variant

Input parameters: cig_freqs: A table of variant allele counts (Default: NULL, i.e. .self$cigar_freqs) names: Labels of variants alleles to remove (Default: NULL) columns: Indices or names of control samples. Remove all variants that occur in these columns. (Default: NULL) include.chimeras: Should chimeric reads be included? (Default: TRUE)

`heatmapCigarFreqs( as.percent = TRUE, x.size = 8, y.size = 8, x.axis.title = NULL, x.angle = 90, min.freq =`
Description: Internal method for CrispRVariants:plotFreqHeatmap, optionally filters the table of variants, then a table of variant counts, coloured by counts or proportions.

Input parameters: as.percent: Should colours represent the percentage of reads per sample (TRUE) or the actual counts (FALSE)? (Default: TRUE) x.size: Font size for x axis labels (Default: 8) y.size: Font size for y axis labels (Default: 8) x.axis.title: Title for x axis min.freq: Include only variants with frequency at least min.freq in at least one sample min.count: Include only variants with count at least min.count in at least one sample top.n: Include only the n most common variants type: Should labels show counts or proportions? (Default: counts) header: What should be displayed in the header of the heatmap. Default: total count for type = "counts" or proportion of reads shown in the matrix for type = "proportions". If "counts" is selected, total counts will be shown for both types. "efficiency" shows the mutation efficiency (calculated with default settings) order: Reorder the columns according to this order (Default: NULL) alleles: Names of alleles to include. Selection of alleles takes place after filtering (Default: NULL). exclude: Names of alleles to exclude (Default: NULL) create.plot: Should the plot be created (TRUE, default), or the data used in the plot returned. ...: Extra filtering or plotting options

Return value: A ggplot2 plot object. Call "print(obj)" to display

See also: CrispRVariants::plotFreqHeatmap

`makePairwiseAlns(cig_freqs = .self$cigar_freqs, ...)` Description: Get variants by their cigar string, make the pairwise alignments for the consensus sequence for each variant allele

Input parameters: cig_freqs: A table of variant allele frequencies (by default: .self$cigar_freqs, but could also be filtered) ...: Extra arguments for CrispRVariants::seqsToAln, e.g. which symbol should be used for representing deleted bases

`mutationEfficiency( snv = c("non_variant", "include", "exclude"), include.chimeras = TRUE, exclude.cols`
Description: Calculates summary statistics for the mutation efficiency, i.e. the percentage of reads that contain a variant. Reads that do not contain and insertion or deletion, but do contain a single nucleotide variant (snv) can be considered as mutated, non-mutated, or not included in efficiency calculations as they are ambiguous. Note: mutationEfficiency does not treat partial alignments differently

Input parameters: snv: One of "include" (consider reads with mismatches to be mutated), "exclude" (do not include reads with snvs in efficiency calculations), and "non_variant" (consider reads with mismatches to be non-mutated). include.chimeras: Should chimeras be counted as variants? (Default: TRUE) exclude.cols: A list of column names to exclude from calculation, e.g. if one sample is a control (default: NULL, i.e. include all columns) group: A grouping variable. Efficiency will be calculated per group, instead of for individual. Cannot be used with exclude.cols. filter.vars: Variants that should not be counted as mutations. filter.cols: Column names to be considered controls. Variants occuring in a control sample will not be counted as mutations. count.alleles: If TRUE, also report statistics about the number of alleles per sample/per group. (Default: FALSE) per.sample: Return efficiencies for each sample (Default: TRUE) min.freq: Minimum frequency for counting alleles. Does not apply to calculating efficiency. To filter when calculating efficiency, first use "variantCounts". (Default: 0, i.e. no filtering) Return value: A vector of efficiency statistics per sample and overall, or a matrix if a group is supplied.

`plotVariants( min.freq = 0, min.count = 0, top.n = nrow(.self$cigar_freqs), alleles = NULL, renumbered = .s`
Description: Internal method for CrispRVariants:plotAlignments, optionally filters the table of variants, then plots variants with respect to the reference sequence, collapsing insertions and displaying insertion sequences below the plot.

Input parameters: min.freq: i(in at least one sample min.count i (integer) include variants that occur at leas i times in at least one sample top.n: n (integer) Plot only the n most frequent variants (default: plot all) Note that if there are ties in variant ranks, top.n only includes ties with all members ranking <= top.n alleles: Alleles to include after filtering. Default NULL means use all alleles that pass filtering. renumbered: If TRUE, the x-axis is numbered with respect to the target (cut) site. If FALSE, x-axis shows genomic locations. (default: TRUE) add.other Add a blank row named "Other" for chimeric alignments, if there are any (Default: TRUE) create.plot Data is plotted if TRUE and returned without if FALSE. (Default: TRUE) plot.regions Subregion of the target to plot (Default: NULL) allow.partial Should partial alignments be allowed? (Default: TRUE) ... additional arguments for plotAlignments

Return value: A ggplot2 plot object. Call "print(obj)" to display

Author(s)

Helen Lindsay

See Also

`readsToTarget` and `readsToTargets` for initialising a CrisprSet, `CrisprRun` container for sample data.

Examples

```
# Load the metadata table
md_fname <- system.file("extdata", "gol_F1_metadata_small.txt", package = "CrispRVariants")
md <- read.table(md_fname, sep = "\t", stringsAsFactors = FALSE)

# Get bam filenames and their full paths
bam_fnames <- sapply(md$bam.filename, function(fn){
 system.file("extdata", fn, package = "CrispRVariants")})

reference <- Biostrings::DNAString("GGTCTCTCGCAGGATGTTGCTGG")
gd <- GenomicRanges::GRanges("18", IRanges::IRanges(4647377, 4647399), strand = "+")

crispr_set <- readsToTarget(bam_fnames, target = gd, reference = reference,
                            names = md$experiment.name, target.loc = 17)
```

`dispatchDots` *dispatchDots*

Description

Update default values for func with values from dot args

Usage

```
dispatchDots(func, ..., call = FALSE)
```

Arguments

`func`	Function to call
`...`	dot args to pass to function
`call`	If TRUE, call the function with the argument list and return this result (Default: FALSE)

Value

A list of arguments to pass to func, or if call is TRUE, the result of calling func with these arguments.

Author(s)

Helen Lindsay

Examples

```
# Set up a function to dispatch dot arguments to:
f <- function(a=1, b=2, c=3){
 print(c(a,b,c))
}
# Set up a function for passing dots:
```

```
g <- function(...){
 CrispRVariants:::dispatchDots(f, ...)
}

g(a = 5)
g(a = 5, call = TRUE)
# Unrelated arguments will not be passed on
g(a = 5, d = 6)
```

`excludeFromBam` *Removes reads from a bam file*

Description

Returns a GAlignments excluding reads based on either name and/or location

Usage

```
excludeFromBam(bam, exclude.ranges = GRanges(), exclude.names = NA)
```

Arguments

`bam`	a GAlignments object
`exclude.ranges`	Regions to exclude, as `GRanges`.
`exclude.names`	A character vector of alignments names to exclude

Value

The bam minus the excluded regions

Author(s)

Helen Lindsay

`findChimeras` *Find chimeric reads*

Description

Find chimeric reads, assuming that the GAlignments object does not contain multimapping reads. That is, read names that appear more than ones in the file are considered chimeras. Chimeric reads are reads that cannot be mapped as a single, linear alignment. Reads from structual rearrangements such as inversions can be mapped as chimeras. Note that the indices of all chimeric reads are returned, these are not separated into individual chimeric sets.

Usage

```
findChimeras(bam, by.flag = FALSE)
```

Arguments

bam	A GAlignments object, must include names
by.flag	Can the chimeras be detected just using the supplementary alignment flag? (Default: FALSE). If TRUE, detects supplementary alignments and returns reads with the same name as a supplementary alignment (quicker). If FALSE, all alignments with duplicated names are returned.

Value

A vector of indices of chimeric sequences within the original bam

Author(s)

Helen Lindsay

See Also

plotChimeras for plotting chimeric alignment sets.

Examples

```
bam_fname <- system.file("extdata", "gol_F1_clutch_2_embryo_4_s.bam",
                         package = "CrispRVariants")
bam <- GenomicAlignments::readGAlignments(bam_fname, use.names = TRUE)
chimera_indices <- findChimeras(bam)
chimeras <- bam[chimera_indices]
```

---

findSNVs *Find frequent SNVs*

Description

Find single nucleotide variants (SNVs) above a specified frequency in a table of variants.

Usage

```
findSNVs(obj, ...)

## S4 method for signature 'CrisprSet'
findSNVs(obj, ..., freq = 0.25, include.chimeras = TRUE)
```

Arguments

obj	An object containing variant counts
...	additional arguments
freq	minimum frequency snv to return (Default: 0.25)
include.chimeras	include chimeric reads when calculating SNV frequencies (Default: TRUE)

Value

A vector of SNVs and their frequencies

Author(s)

Helen Lindsay

---

getChimeras *Get chimeric alignments*

Description

Return chimeric alignments from a collection of aligned sequences

Usage

```
getChimeras(obj, ...)

## S4 method for signature 'CrisprSet'
getChimeras(obj, ..., sample)
```

Arguments

obj	An object containing aligned sequences
...	additional arguments
sample	The sample name or sample index to return

Value

A GAlignment object containing the chimeric read groups

Author(s)

Helen Lindsay

Examples

```
data("gol_clutch1")
chimeras <- getChimeras(gol, sample = 2)
```

getInsertionsTable *getInsertionsTable*

Description

Returns a table of insertion sequences present in a GAlignments object. This table is aggregated and used by plotAlignments.

Usage

```
getInsertionsTable(alns, pos = 1L)
```

Arguments

alns	A GAlignments object
pos	(Integer(1)) The amount by which to shift genomic coordinates upstream to get coordinates relative to a display region

Value

A data frame of insertion sequences, genomic and relative locations

Author(s)

Helen Lindsay

gol_clutch1 *Variant sequences from golden clutch 1 (Burger et al)*

Description

This dataset is a subset of the crispant data for the golden gene used by Burger et al (submitted).

Usage

```
data(gol_clutch1)
```

Format

A CrisprSet object countaining 8 samples

Details

- gol The variants as a CrisprSet object

Value

A CrisprSet object named "gol"

indelLabels *indelLabels*

Description

Makes allele labels for insertion / deletion variants

Usage

```
indelLabels(
  alns,
  rc = FALSE,
  genome.to.pos = NULL,
  keep.ops = c("I", "D", "N"),
  regions = NULL,
  as.string = TRUE,
  ...
)
```

Arguments

alns	(GAligments) aligned reads for finding variants
rc	Should the variants be displayed with respect to the negative strand? (Default: FALSE)
genome.to.pos	A vector with names being genomic locations and values being positions to use in labels (Default: NULL)
keep.ops	CIGAR operations to remain in the variant label (usually indels)
regions	IRanges(k) Regions for counting insertions and deletions. Insertions on the right border are not counted.
as.string	Return labels as strings (Default: TRUE)
...	extra formatting arguments

Value

A vector of labels for alns

Author(s)

Helen Lindsay

makeAlignmentTilePlot *Internal CrispRVariants function for creating the plotAlignments background*

Description

Takes a matrix of characters, x and y locations and colours, creates a ggplot geom_tile plot with tiles labelled by the characters.

Usage

```
makeAlignmentTilePlot(
  m,
  ref,
  xlab,
  plot.text.size,
  axis.text.size,
  xtick.labs,
  xtick.breaks,
  tile.height
)
```

Arguments

m	A matrix with column headings Var1: y location, Var2: x location, cols: tile fill colour, isref: transparency value text_cols: text colour
ref	The reference sequence, only used for checking the number of x-tick labels when x-tick breaks are not supplied
xlab	Label for the x axis
plot.text.size	Size for text within plot
axis.text.size	Size for text on axes
xtick.labs	x axis labels
xtick.breaks	Locations of x labels
tile.height	Controls whitespace between tiles

Value

A ggplot object

Author(s)

Helen Lindsay

mergeChimeras *mergeChimeras*

Description

Merges chimeric alignments where the individual segments border an unmapped region (a long deletion). If bases of the read are mapped to both ends of the gap, the multimapped reads are only included in the leftmost genomic segment. If there are more than max_unmapped unmapped bases between the mapped bases, the read is not considered mergeable. Currently experimental and only tested with reads mapped by bwa mem.

Usage

```
mergeChimeras(
  bam,
  chimera_idxs = NULL,
  verbose = TRUE,
  max_read_overlap = 10,
  max_unmapped = 4,
  name = NULL
)
```

Arguments

bam	A GenomicAlignments::GAlignments object
chimera_idxs	Indices of chimeric reads within bam
verbose	Should information about the number of mergeable alignments be printed? (Default: TRUE)
max_read_overlap	Maximum number of bases in a mergeable read that are aligned to two genomic locations (Default: 10)
max_unmapped	Maximum number of bases in a mergeable read that are unmapped and located between two mapped segments (Default: 4)
name	Name of the sample, used when reporting verbose output.

Value

A list of the merged and unmerged chimeric alignments

Author(s)

Helen Lindsay

`mergeCrisprSets` *Merge two CrisprSets*

Description

Merge two CrisprSet objects sharing a reference and target location

Usage

```
mergeCrisprSets(x, y, ...)

## S4 method for signature 'CrisprSet,CrisprSet'
mergeCrisprSets(
  x,
  y,
  ...,
  x.samples = NULL,
  y.samples = NULL,
  names = NULL,
  order = NULL
)
```

Arguments

`x`	A CrisprSet object
`y`	A second CrisprSet object
`...`	extra arguments
`x.samples`	A subset of column names or indices to keep from CrispRSet x (Default: NULL, i.e. keep all)
`y.samples`	A subset of column names or indices to keep from CrispRSet y (Default: NULL, i.e. keep all)
`names`	New names for the merged CrisprSet object (Default: NULL)
`order`	A list of sample names, matching the names in x and y, specifying the order of the samples in the new CrisprSet. (Not implemented yet)

Value

A merged CrisprSet object

Author(s)

Helen Lindsay

Examples

```
# Load the metadata table
md_fname <- system.file("extdata", "gol_F1_metadata_small.txt", package = "CrispRVariants")
md <- read.table(md_fname, sep = "\t", stringsAsFactors = FALSE)

# Get bam filenames and their full paths
bam_fnames <- sapply(md$bam.filename, function(fn){
 system.file("extdata", fn, package = "CrispRVariants")})

reference <- Biostrings::DNAString("GGTCTCTCGCAGGATGTTGCTGG")
gd <- GenomicRanges::GRanges("18", IRanges::IRanges(4647377, 4647399),
      strand = "+")

crispr_set1 <- readsToTarget(bam_fnames[c(1:4)], target = gd,
      reference = reference, names = md$experiment.name[1:4], target.loc = 17)
crispr_set2 <- readsToTarget(bam_fnames[c(5:8)], target = gd,
      reference = reference, names = md$experiment.name[5:8], target.loc = 17)
mergeCrisprSets(crispr_set1,crispr_set2)
```

`mismatchLabels` *nonindelLabels*

Description

Make variant labels for variants without an insertion or deletion

Usage

```
mismatchLabels(
  alns,
  target,
  ref.seq,
  regions = NULL,
  min.pct = 0,
  mismatch.label = "SNV",
  genome.to.pos = NULL,
  as.string = TRUE
)
```

Arguments

`alns`	A GAlignments object, where the aligned sequences should span the reference sequence
`target`	(GRanges(1)) The region for counting mismatches
`ref.seq`	A DNAString object, the sequence for comparison when checking for mismatches. The sequence does not necessarily have to match the mapping reference sequence. Must span all regions if regions are provided.

regions	A GRanges object, regions to check for mismatches with coordinates relative to the reference sequence
min.pct	(numeric(1), between 0 and 100) Only return SNVs that occur at in least min.pct change, not any change at a position.
mismatch.label	(character(1)) Label to append to the start of mismatch strings, if returning as a single string (Default: "SNV:")
genome.to.pos	Optional named vector for transforming variant coordinates into another coordinate system (Default: NULL)
as.string	Should individual variant labels be pasted into a single comma separated string when one alignment has multiple variants? (Default: TRUE)

Value

A data frame of sequence indices, genomic position of mismatch and mismatch base

---

`mutationEfficiency` *Get mutation efficiency*

Description

Returns the percentage of sequences that contain at least one mutation.

Usage

```
mutationEfficiency(obj, ...)

## S4 method for signature 'CrisprSet'
mutationEfficiency(
  obj,
  ...,
  snv = c("non_variant", "include", "exclude"),
  include.chimeras = TRUE,
  exclude.cols = NULL,
  filter.vars = NULL,
  filter.cols = NULL,
  group = NULL
)
```

Arguments

obj	An object containing variant counts
...	additional arguments
snv	Single nucleotide variants (SNVs) may be considered as mutations ("include"), treated as ambiguous sequences and not counted at all ("exclude"), or treated as non-mutations, e.g. sequencing errors or pre-existing SNVs ("non_variant", default)

include.chimeras
: Should chimeric alignments be counted as variants when calculating mutation efficiency (Default: TRUE

exclude.cols
: A vector of names of columns in the variant counts table that will not be considered when counting mutation efficiency

filter.vars
: Variants to remove before calculating efficiency. May be either a variant size, e.g. "1D", or a particular variant/variant combination, e.g. -5:3D

filter.cols
: A vector of control sample names. Any variants present in the control samples will be counted as non-variant, unless they also contain another indel. Note that this is not compatible with counting snvs as variants.

group
: A grouping vector. If provided, efficiency will be calculated per group (Default: NULL)

Value

A vector of efficiency statistics per sample and overall, or a matrix of efficiency statistics per group if a group is provided

Author(s)

Helen Lindsay

Examples

```
data("gol_clutch1")
mutationEfficiency(gol)
```

---

narrowAlignments *Narrow a set of aligned reads to a target region*

Description

Aligned reads are narrowed to the target region. In the case of reads with deletions spanning the boundaries of the target, reads are narrowed to the start of the deletion,

Usage

```
narrowAlignments(alns, target, ...)

## S4 method for signature 'GAlignments,GRanges'
narrowAlignments(
  alns,
  target,
  ...,
  reverse.complement,
  minoverlap = NULL,
  verbose = FALSE,
```

```
  clipping.ops = c("S", "H"),
  match.ops = c("M", "X", "=")
)
```

Arguments

alns	A GAlignments object including a metadata column "seq" containing the sequence
target	A GRanges object
...	additional arguments
reverse.complement	Should the aligned reads be reverse complemented?
minoverlap	Minimum overlapping region between alignments and target. If not specified, alignments must span the entire target region. (Default: NULL)
verbose	(Default: FALSE)
clipping.ops	CIGAR operations corresponding to clipping (Default: c("S","H"))
match.ops	CIGAR operations corresponding to a match, i.e. a non-indel position (Default: c("M","X","="))

Value

The narrowed alignments (GAlignments)

Author(s)

Helen Lindsay

Examples

```
bam_fname <- system.file("extdata", "gol_F1_clutch_2_embryo_4_s.bam",
                         package = "CrispRVariants")
bam <- GenomicAlignments::readGAlignments(bam_fname, use.names = TRUE)
target <- GenomicRanges::GRanges("18", IRanges::IRanges(4647377, 4647399),
          strand = "+")
narrowAlignments(bam, target, reverse.complement = FALSE)
```

---

plotAlignments *Plot alignments with respect to a reference sequence*

Description

(signature("CrisprSet")) Wrapper for CrisprSet$plotVariants. Optionally filters a CrisprSet frequency table, then plots variants. More information in CrisprSet

(signature("DNAString")) Plots a set of pairwise alignments to a reference sequence. Alignments should all be the same length as the reference sequences. This is achieved by removing insertions with respect to the reference, see seqsToAln. Insertions are indicated by symbols in the plot and a table showing the inserted sequences below the plot. The default options are intended for a figure 6-8 inches wide, with figure height best chosen according to the number of different variants and insertions to be displayed.

Usage

```
plotAlignments(obj, ...)

## S4 method for signature 'CrisprSet'
plotAlignments(
  obj,
  ...,
  min.freq = 0,
  min.count = 1,
  top.n = 50,
  renumbered = obj$pars[["renumbered"]],
  add.other = TRUE,
  create.plot = TRUE
)

## S4 method for signature 'character'
plotAlignments(
  obj,
  ...,
  alns,
  ins.sites,
  highlight.pam = TRUE,
  show.plot = FALSE,
  target.loc = 17,
  pam.start = NA,
  pam.end = NA,
  ins.size = 2,
  legend.cols = 3,
  xlab = NULL,
  xtick.labs = NULL,
  xtick.breaks = NULL,
  plot.text.size = 2,
  axis.text.size = 8,
  legend.text.size = 6,
  highlight.guide = TRUE,
  guide.loc = NULL,
  tile.height = 0.55,
```

```
  max.insertion.size = 20,
  min.insertion.freq = 5,
  line.weight = 1,
  legend.symbol.size = ins.size,
  add.other = FALSE,
  codon.frame = NULL,
  style = c("all", "mismatches")
)

## S4 method for signature 'DNAString'
plotAlignments(
  obj,
  ...,
  alns,
  ins.sites,
  highlight.pam = TRUE,
  show.plot = FALSE,
  target.loc = 17,
  pam.start = NA,
  pam.end = NA,
  ins.size = 2,
  legend.cols = 3,
  xlab = NULL,
  xtick.labs = NULL,
  xtick.breaks = NULL,
  plot.text.size = 2,
  axis.text.size = 8,
  legend.text.size = 6,
  highlight.guide = TRUE,
  guide.loc = NULL,
  tile.height = 0.55,
  max.insertion.size = 20,
  min.insertion.freq = 5,
  line.weight = 1,
  legend.symbol.size = ins.size,
  add.other = FALSE,
  codon.frame = NULL
)
```

Arguments

obj	The object to be plotted
...	Additional arguments
min.freq	i (one sample (default: 0, i.e no frequency cutoff)
min.count	i (integer) only plot variants with count >= i in at least one sample (default: 0, i.e no count cutoff)
top.n	(integer) Plot only the n most frequent variants (default: 50)

renumbered — If TRUE, the x-axis is numbered with respect to the target (default: TRUE)

add.other — Add a blank row labelled "Other" to the plot, for combining with plotFreqHeatmap (default: TRUE (signature "CrisprSet") FALSE (signature "matrix"))

create.plot — Should the data be plotted? If false, returns the data used for plotting (Default: TRUE)

alns — A named character vector of aligned sequences, with insertions removed

ins.sites — A table of insertion_sites, which must include cols named "start", "cigar", "seq" and "count" for the start of the insertion in the corresponding sequence

highlight.pam — should location of PAM with respect to the target site be indicated by a box? (Default: TRUE) If TRUE, and pam.start and pam.end are not supplied, PAM is inferred from target.loc

show.plot — Should the plot be displayed (TRUE) or just returned as a ggplot object (FALSE). (Default: FALSE)

target.loc — The location of the zero point / cleavage location. Base n, where the zero point is between bases n and n+1

pam.start — The first location of the PAM with respect to the reference.

pam.end — The last location of the PAM with respect to the reference. Default is two bases after the pam.start

ins.size — The size of the symbols representing insertions within the plot.

legend.cols — The number of columns in the legend. (Default:3)

xlab — A title for the x-axis (Default: NULL)

xtick.labs — Labels for the x-axis ticks (Default: NULL)

xtick.breaks — Locations for x-axis tick breaks (Default: NULL)

plot.text.size — The size of the text inside the plot

axis.text.size — The size of the axis labels

legend.text.size — The size of the legend labels

highlight.guide — Should the guide be indicated by a box in the reference sequence? (Default: TRUE)

guide.loc — The location of the guide region to be highlighted, as an IRanges object. Will be inferred from target.loc if highlight.guide = TRUE and no guide.loc is supplied, assuming the guide plus PAM is 23bp (Default: NULL)

tile.height — The height of the tiles within the plot. (Default: 0.55)

max.insertion.size — The maximum length of an insertion to be shown in the legend. If max.insertion.size = n, an insertion of length m > n will be annotated as "mI" in the figure. (Default: 20)

min.insertion.freq — Display inserted sequences with frequency at least x amongst the sequences with an insertion of this size and length (Default: 5)

line.weight — The line thickness for the vertical line indicating the zero point (cleavage site) and the boxes for the guide and PAM. (Default: 1)

legend.symbol.size
: The size of the symbols indicating insertions in the legend. (Default: ins.size)

codon.frame
: Codon position of the leftmost nucleotide. If provided, codon positions in the specified frame are indicated. (Default: NULL)

style
: One of "all" (colour all tiles) and "mismatches" (colour only mismatch positions)

Value

A ggplot2 figure

Author(s)

Helen Lindsay

See Also

seqsToAln, ggplot

Examples

```
#Load a CrisprSet object and plot
data("gol_clutch1")
plotAlignments(gol)
```

---

plotChimeras *Display a dot plot of chimeric alignments*

Description

Produces a dot plot of a set of chimeric alignments. For chimeric alignments, a single read is split into several, possibly overlapping alignmed blocks. Aligned sections of chimeric reads can be separated by large genomic distances, or on separate chromosomes. plotChimeras produces a dot plot, each aligned block highlighted, and chromosomes shown in different colours. Large gaps between aligned segments are collapsed and indicated on the plot with horizontal lines. The X-axis shows each base of the entire read. Note that the mapping to the fwd strand is shown if all strands agree. The chimeric alignments must be sorted!

Usage

```
plotChimeras(
  chimeric.alns,
  max.gap = 10,
  tick.sep = 20,
  text.size = 10,
  title.size = 16,
  gap.pad = 20,
  legend.title = "Chromosome",
  xangle = 90,
```

```
  wrt.forward = FALSE,
  annotate.within = 20,
  annotations = GenomicRanges::GRanges()
)
```

Arguments

chimeric.alns	A GAlignments object containing only the chimeric reads to be plotted
max.gap	If aligned segments are separated by more than max.gap,
tick.sep	How many bases should separate tick labels on plot. Default 20.
text.size	Size of X and Y tick labels on plot. Default 12
title.size	Size of X and Y axis labels on plot. Default 16
gap.pad	How much should aligned blocks be separated by? (Default: 20)
legend.title	Title for the legend. Default "Chromosome"
xangle	Angle for x axis text (Default 90, i.e vertical)
wrt.forward	Should chimeric alignments where all members map to the negative strand be displayed with respect to the forward strand, i.e. as the cigar strand is written (TRUE), or the negative strand (FALSE) (Default: FALSE)
annotate.within	annot_aln ranges in "annotations" within n bases of a chimeric alignment (Default 50)
annotations	A list of GRanges. Any that overlap with the chimeric alignments are highlighed in the plot.

Value

A ggplot2 dotplot of the chimeric alignments versus the reference sequence

Author(s)

Helen Lindsay

See Also

findChimeras for finding chimeric alignment sets.

Examples

```
bam_fname <- system.file("extdata", "gol_F1_clutch_2_embryo_4_s.bam",
                         package = "CrispRVariants")
bam <- GenomicAlignments::readGAlignments(bam_fname, use.names = TRUE)
# Choose a single chimeric read set to plot:
chimeras <- bam[names(bam) == "AB3092"]

# This read aligns in 3 pieces, all on chromosome 18.
# The plot shows the alignment annot_alns a small duplication and
# a long gap.
plotChimeras(chimeras)
```

plotFreqHeatmap *Plot a table of counts with colours indicating frequency*

Description

Creates a heatmap from a matrix of counts or proportions, where tiles are coloured by the proportion and labeled with the value.

Usage

```
plotFreqHeatmap(obj, ...)

## S4 method for signature 'matrix'
plotFreqHeatmap(
  obj,
  ...,
  col.sums = NULL,
  header = NA,
  header.name = "Total",
  group = NULL,
  group.colours = NULL,
  as.percent = TRUE,
  x.axis.title = NULL,
  x.size = 6,
  y.size = 8,
  x.angle = 90,
  legend.text.size = 6,
  plot.text.size = 3,
  line.width = 1,
  x.hjust = 1,
  legend.position = "right",
  x.labels = NULL,
  legend.key.height = grid::unit(1, "lines")
)

## S4 method for signature 'CrisprSet'
plotFreqHeatmap(
  obj,
  ...,
  top.n = 50,
  min.freq = 0,
  min.count = 1,
  type = c("counts", "proportions"),
  order = NULL,
  alleles = NULL
)
```

Arguments

obj	A matrix of counts with rows = feature, columns = sample
...	additional arguments
col.sums	Alternative column sums to be used for calculating the tile colours if as.percent = TRUE, e.g. if "obj" is a subset of a larger data set. If "NULL" (default), the column sums of "obj" are used.
header	Alternative column titles, e.g. column sums for the unfiltered data set when obj is a subset. If set to "NA", column sums of obj are displayed. If "NULL", no header is displayed (Default: NA).
header.name	Label for the header row (Default: "Total")
group	Grouping factor for columns. If supplied, columns are ordered to match the levels (Default: NULL)
group.colours	Colours for column groups, should match levels of "group". If "NULL", groups are coloured differently (Default: NULL)
as.percent	Should colours represent the percentage of reads per sample (TRUE) or the actual counts (FALSE)? (Default: TRUE)
x.axis.title	A title for the x-axis. (Default: NULL)
x.size	Font size for x-labels (Default: 16)
y.size	Font size for y-labels (Default: 16)
x.angle	Angle for x-labels (Default: 90, i.e. vertical)
legend.text.size	Font size for legend (Default: 16)
plot.text.size	Font size counts within plot (Default: 3)
line.width	Line thickness of title box'
x.hjust	Horizontal justification of x axis labels (Default: 1)
legend.position	The position of the legend (Default: right)
x.labels	X-axis labels (Default: NULL, column.names of the matrix, doesn't do anything at the moment)
legend.key.height	The height of the legend key, as a "unit" object. (See unit).
top.n	Show the n top ranked variants. Note that if the nth and n+1th variants have equal rank, they will not be shown. (Default: 50)
min.freq	i (one sample (Default: 0, i.e no frequency cutoff)
min.count	i (integer) only plot variants with count >= i in at least one sample (default: 0, i.e no count cutoff)
type	Plot either "counts" or "proportions"
order	A list of column names or indices specifying the order of the columns in the plot
alleles	A list of alleles to include. Can be used to display only alleles of interest or to order the alleles. The default value NULL means all alleles passing the frequency cut offs will be included.

Value

The ggplot2 plot of the variant frequencies

Examples

```
#Load a CrisprSet object for plotting
data("gol_clutch1")

# Plot the frequency heatmap
plotFreqHeatmap(gol)
```

`plotVariants` *Plot alignments, frequencies and location of target sequence*

Description

Combines a plot of transcript structure, alleles aligned with respect to a reference genome and a heatmap of counts or proportions of each allele in a set of data.

Usage

```
plotVariants(obj, ...)

## S4 method for signature 'CrisprSet'
plotVariants(
  obj,
  ...,
  txdb = NULL,
  add.chr = TRUE,
  plotAlignments.args = list(),
  plotFreqHeatmap.args = list()
)
```

Arguments

`obj`	The object to be plotted
`...`	extra arguments for plot layout
`txdb`	GenomicFeatures:TxDb object (default: NULL)
`add.chr`	If target chromosome does not start with "chr", e.g. "chr5", add the "chr" prefix. (Default:TRUE)
`plotAlignments.args`	Extra arguments for plotAlignments
`plotFreqHeatmap.args`	Extra arguments for plotFreqHeatmap

Value

A ggplot2 plot of the variants

See Also

arrangePlots for general layout options and annotateGenePlot for options relating to the transcript plot.

Examples

```
#Load a CrisprSet object for plotting
data("gol_clutch1")

#Load the transcript db.  This is a subset of the Ensembl Danio Rerio v73 gtf
# for the region 18:4640000-4650000 which includes the targeted gol gene

library(GenomicFeatures)
fn <- system.file("extdata", "Danio_rerio.Zv9.73.gol.sqlite",
                  package = "CrispRVariants")
txdb <- loadDb(fn)

# Plot the variants
p <- plotVariants(gol, txdb = txdb)

#In the above plot, the bottom margin is too large, the legend is
#cut off, and the text within the plots should be larger.
#These issues can be fixed with some adjustments:
p <- plotVariants(gol, txdb = txdb,
                  plotAlignments.args = list(plot.text.size = 4, legend.cols = 2),
                  plotFreqHeatmap.args = list(plot.text.size = 4),
                  left.plot.margin = grid::unit(c(0.1,0.2,0.5,1), "lines"))
```

---

rcAlns *Internal CrispRVariants function for determining read orientation*

Description

Function for determining whether reads should be oriented to the target strand, always displayed on the positive strand, or oriented to

Usage

```
rcAlns(target.strand, orientation)
```

Arguments

target.strand	The target strand (one of "+","-","*")
orientation	One of "target", "opposite" and "positive" (Default: "target")

Value

A logical value indicating whether reads should be reverse complemented

Author(s)

Helen Lindsay

`readsByPCRPrimer` *Finds overlaps between aligned reads and PCR primers*

Description

Short reads amplified with PCR primers should start and end at defined positions. However, the ends of an aligned read may be clipped as sequencing technologies are prone to making errors at the start and end. readsByPCRPrimer extrapolates the genomic location of entire reads from their aligned sections by adding clipped sections, then finds near exact matches to a set of PCR primers. Note that this is not always a good assumption, and is misleading in the case of chimeric reads where sections clipped in one part of a chimera are aligned in another.

Usage

```
readsByPCRPrimer(bam, primers, ...)

## S4 method for signature 'GAlignments,GRanges'
readsByPCRPrimer(
  bam,
  primers,
  ...,
  tolerance = 0,
  verbose = TRUE,
  ignore.strand = TRUE,
  allow.partial = TRUE,
  chimera.idxs = NULL
)

## S4 method for signature 'GRanges,GRanges'
readsByPCRPrimer(
  bam,
  primers,
  ...,
  tolerance = 0,
  verbose = TRUE,
  ignore.strand = TRUE,
  allow.partial = TRUE,
  chimera.idxs = NULL
)
```

Arguments

bam	A set of aligned reads
primers	A set of ranges that the unclipped reads may map to
...	Additional arguments
tolerance	Number of bases by which reads and primers may differ at each end (Default: 0)
verbose	Print number of full and partial matches (Default: TRUE)
ignore.strand	Passed to findOverlaps and disjoin. Should strand be ignored when finding overlaps. (Default: TRUE)
allow.partial	Should reads that do not match the PCR boundaries, but map to a region covered by only one primer be considered matches? (Default: TRUE)
chimera.idxs	Indices of chimeric reads within the bam. If specified, chimeras overlapping multiple pcr primers will be removed.

Value

A Hits object where "query" is the index with respect to bam and "subject" is the index with respect to the primers.

Author(s)

Helen Lindsay

See Also

GRanges, GAlignments

---

readsToTarget *Trims reads to a target region.*

Description

Trims aligned reads to one or several target regions, optionally reverse complementing the alignments.

Usage

```
readsToTarget(reads, target, ...)

## S4 method for signature 'GAlignments,GRanges'
readsToTarget(
  reads,
  target,
  ...,
  reverse.complement = TRUE,
```

```
  chimeras = NULL,
  collapse.pairs = FALSE,
  use.consensus = FALSE,
  store.chimeras = FALSE,
  verbose = TRUE,
  name = NULL,
  minoverlap = NULL,
  orientation = c("target", "opposite", "positive")
)

## S4 method for signature 'GAlignmentsList,GRanges'
readsToTarget(
  reads,
  target,
  ...,
  reference = reference,
  names = NULL,
  reverse.complement = TRUE,
  target.loc = 17,
  chimeras = NULL,
  collapse.pairs = FALSE,
  use.consensus = FALSE,
  orientation = c("target", "opposite", "positive"),
  minoverlap = NULL,
  verbose = TRUE
)

## S4 method for signature 'character,GRanges'
readsToTarget(
  reads,
  target,
  ...,
  reference,
  reverse.complement = TRUE,
  target.loc = 17,
  exclude.ranges = GRanges(),
  exclude.names = NA,
  chimeras = c("count", "exclude", "ignore", "merge"),
  collapse.pairs = FALSE,
  use.consensus = FALSE,
  orientation = c("target", "opposite", "positive"),
  names = NULL,
  minoverlap = NULL,
  verbose = TRUE
)

readsToTargets(reads, targets, ...)
```

```
## S4 method for signature 'character,GRanges'
readsToTargets(
  reads,
  targets,
  ...,
  references,
  primer.ranges = NULL,
  target.loc = 17,
  reverse.complement = TRUE,
  collapse.pairs = FALSE,
  use.consensus = FALSE,
  ignore.strand = TRUE,
  names = NULL,
  bpparam = BiocParallel::SerialParam(),
  orientation = c("target", "opposite", "positive"),
  chimera.to.target = 5,
  verbose = TRUE
)

## S4 method for signature 'GAlignmentsList,GRanges'
readsToTargets(
  reads,
  targets,
  ...,
  references,
  primer.ranges = NULL,
  target.loc = 17,
  reverse.complement = TRUE,
  collapse.pairs = FALSE,
  use.consensus = FALSE,
  ignore.strand = TRUE,
  names = NULL,
  bpparam = BiocParallel::SerialParam(),
  chimera.to.target = 5,
  orientation = c("target", "opposite", "positive"),
  verbose = TRUE
)
```

Arguments

reads	A GAlignments object, or a character vector of the filenames
target	A GRanges object specifying the range to narrow alignments to
...	Extra arguments for initialising CrisprSet
reverse.complement	(Default: TRUE) Should the alignments be oriented to match the strand of the target? If TRUE, targets located strand and targets on the negative strand with respect to the negative strand. If FALSE, the parameter 'orientation' must be set

	to determine the orientation. 'reverse.complement' will be replaced by 'orientation' in a later release.
chimeras	Flag to determine how chimeric reads are treated. One of "ignore", "exclude", and "merge". Default "count", "merge" not implemented yet
collapse.pairs	If reads are paired, should pairs be collapsed? (Default: FALSE) Note: only collapses primary alignments, and assumes that there is only one primary alignment per read.
use.consensus	Take the consensus sequence for non-matching pairs? If FALSE, the sequence of the first read is used. Can be very slow. (Default: FALSE)
store.chimeras	Should chimeric reads be stored? (Default: FALSE)
verbose	Print progress and statistics (Default: TRUE)
name	An experiment name for the reads. (Default: NULL)
minoverlap	Minimum number of bases the aligned read must share with the target site. If not specified, the aligned read must completely span the target region. (Default: NULL)
orientation	One of "target" (reads are displayed on the same strand as the target) "opposite" (reads are displayed on the opposite) strand from the target or "positive" (reads are displayed on the forward strand regardless of the strand of the target) (Default:"target")
reference	The reference sequence
names	Experiment names for each bam file. If not supplied, filenames are used.
target.loc	The zero point for renumbering (Default: 17)
exclude.ranges	Ranges to exclude from consideration, e.g. homologous to a pcr primer.
exclude.names	Alignment names to exclude
targets	A set of targets to narrow reads to
references	A set of reference sequences matching the targets. References for negative strand targets should be on the negative strand.
primer.ranges	A set of GRanges, corresponding to the targets. Read lengths are typically greater than target regions, and it can be that reads span multiple targets. If primer.ranges are available, they can be used to assign such reads to the correct target.
ignore.strand	Should strand be considered when finding overlaps? (See findOverlaps)
bpparam	A BiocParallel parameter for parallelising across reads. Default: no parallelisation. (See bpparam)
chimera.to.target	Number of bases that may separate a chimeric read set from the target.loc for it to be assigned to the target. (Default: 5)

Value

(signature("GAlignments", "GRanges")) A CrisprRun object

(signature("character", "GRanges")) A CrisprSet object

Author(s)

Helen Lindsay

Examples

```
# Load the metadata table
md_fname <- system.file("extdata", "gol_F1_metadata_small.txt", package = "CrispRVariants")
md <- read.table(md_fname, sep = "\t", stringsAsFactors = FALSE)

# Get bam filenames and their full paths
bam_fnames <- sapply(md$bam.filename, function(fn){
 system.file("extdata", fn, package = "CrispRVariants")})

reference <- Biostrings::DNAString("GGTCTCTCGCAGGATGTTGCTGG")
gd <- GenomicRanges::GRanges("18", IRanges::IRanges(4647377, 4647399),
       strand = "+")

crispr_set <- readsToTarget(bam_fnames, target = gd, reference = reference,
                            names = md$experiment.name, target.loc = 17)
```

`readTargetBam` *Internal CrispRVariants function for reading and filtering a bam file*

Description

Includes options for excluding reads either by name or range. The latter is useful if chimeras are excluded. Reads are excluded before chimeras are detected, thus a chimeric read consisting of two sections, one of which overlaps an excluded region, will not be considered chimeric. Chimeric reads can be ignored, excluded, which means that all sections of a chimeric read will be removed, or merged, which means that chimeras will be collapsed into a single read where possible. (Not implemented yet) If chimeras = "merge", chimeric reads are merged if all segments

Usage

```
readTargetBam(
  file,
  target,
  exclude.ranges = GRanges(),
  exclude.names = NA,
  chimera.to.target = 5,
  chimeras = c("count", "ignore", "exclude", "merge"),
  max.read.overlap = 10,
  max.unmapped = 4,
  by.flag = TRUE,
  verbose = TRUE
)
```

Arguments

file	The name of a bam file to read in
target	A GRanges object containing a single target range
exclude.ranges	A GRanges object of regions that should not be counted, e.g. primer or cloning vector sequences that have a match in the genome
exclude.names	A vector of read names to exclude.
chimera.to.target	Maximum distance between endpoints of chimeras and target.loc for assigning chimeras to targets (default: 5)
chimeras	Flag to determine how chimeric reads are treated. One of "ignore", "exclude", "count" and "merge". Default "ignore".
max.read.overlap	Maximum number of bases mapped to two positions for chimeras to be merged (Default: 10)
max.unmapped	Maximum number of bases that are unmapped for chimeras to be merged (Default: 4)
by.flag	Is the supplementary alignment flag set? Used for identifying chimeric alignments, function is much faster if TRUE. Not all aligners set this flag. If FALSE, chimeric alignments are identified using read names (Default: TRUE)
verbose	Print stats about number of alignments read and filtered. (Default: TRUE)

Value

A GenomicAlignments::GAlignment obj

---

refFromAlns *refFromAlns*

Description

Reconstruct the reference sequence from alignments reads using the CIGAR

Usage

```
refFromAlns(alns, location, ...)

## S4 method for signature 'GAlignments,ANY'
refFromAlns(alns, location, ..., keep.names = FALSE)

## S4 method for signature 'GAlignments,GRanges'
refFromAlns(alns, location, ...)
```

Arguments

alns	Alignments to use for inferring the reference sequence
location	The location to infer the reference for.
...	additional arguments
keep.names	Should read names be added to the result if present? (Default: FALSE)

Value

The reference sequences corresponding to the provided alignments

A DNAStringSet (signature = c("GAlignments", "ANY"))

A DNAString (signature = c("GAlignments", "GRanges"))

Author(s)

Helen Lindsay

Examples

```
bam <- system.file("extdata", "bam/ab1_ptena_wildtype_looking_embryo_1_s.bam",
                   package = "CrispRVariants")
alns <- GenomicAlignments::readGAlignments(bam,
          param = Rsamtools::ScanBamParam(tag = "MD", what = "seq"))
# To get the reference sequence from the a given location:
location <- GenomicRanges::GRanges("chr17", IRanges::IRanges(23648420,23648430))
refFromAlns(alns, location = location)
```

---

reverseCigar *Reverses the order of operations in a cigar string*

Description

For example, the string "20M5D15M" would become "15M5D20M"

Usage

```
reverseCigar(cigars)
```

Arguments

cigars	the cigar strings.

Value

The reversed cigar string

`rmMultiPCRChimera` *Remove chimeric reads overlapping multiple primers*

Description

Finds and removes sets of chimeric read alignments that overlap more than one guide, i.e. that cannot be unambiguously assigned to a single guide.

Usage

```
rmMultiPCRChimera(readnames, pcrhits, chimera_idxs, ...)

## S4 method for signature 'character,Hits,integer'
rmMultiPCRChimera(readnames, pcrhits, chimera_idxs, ..., verbose = TRUE)
```

Arguments

`readnames`	A set of read names, used for identifying chimeric read sets
`pcrhits`	A mapping between indices of reads and a set of pcr primers
`chimera_idxs`	location of chimeric reads within the bam
`...`	Additional arguments
`verbose`	Display information about the chimeras (Default: TRUE)

Value

pcrhits, with chimeric reads mapping to different primers omitted.

Author(s)

Helen Lindsay

`selectOps` *selectOps*

Description

select CIGAR operations in a region of interest.

Usage

```
selectOps(cigar, ...)

## S4 method for signature 'character'
selectOps(
  cigar,
  ...,
  ops = GenomicAlignments::CIGAR_OPS,
  op.regions = NULL,
  pos = 1L
)
```

Arguments

cigar	CIGAR strings
...	Extra arguments (Not currently used)
ops	CIGAR operations to consider (Default: all)
op.regions	(GRanges) Return operations only in these regions
pos	An offset for the cigar ranges

Value

A GRanges list of opertion locations in reference space with a metadata column for the operation width in query space.

Author(s)

Helen Lindsay

---

seqsToAln *Creates a text alignment from a set of cigar strings*

Description

Creates a one-to-one text alignment of a set of cigar strings with respect to the reference sequence by collapsing insertions and introducing gaps across deletions.

When genomic coordinates for the alignment start and the target region are provided, aligned sequences are cropped to the target region

Given a character vector of pairwise alignments and a region to display, trims alignments to the display regions, joined by a separator "join". Alignments should be equal length, e.g. created by seqsToAln

Usage

```
seqsToAln(
  cigar,
  dnaseq,
  target,
  del_char = "-",
  aln_start = NULL,
  reverse_complement = FALSE,
  allow_partial = FALSE
)

selectAlnRegions(
  alns,
  reference,
  target,
  keep,
  join = "/ %s /",
  border.gaps = FALSE
)
```

Arguments

cigar	A list of cigar strings to align
dnaseq	The set of sequences corresponding to the cigars, as Biostrings::DNAStrings
target	The target region to return, as GRanges. Sequences overlapping the target region are trimmed to exactly match it.
del_char	The character to represent deleted bases. Default "-"
aln_start	Genomic start locations of aligned sequences. Should be used in conjunction with target_start and target_end.
reverse_complement	(Default: FALSE)
allow_partial	Are alignments that do not span the target region allowed? (Default: FALSE)
alns	Character vector of pairwise alignments, with insertions removed
reference	Reference sequence
keep	Region to display, relative to the target region, i.e. not genomic coords (IRanges or GRanges)
join	character(1) String used for joining alignment segments. Can accept a placeholder to fill in the number of bases deleted with " deleted
border.gaps	(logical(1)) Should bases deleted from the borders be shown? (Default: FALSE)

Value

The sequences with insertions collapsed and deletions padded

A list of the truncated alignments (alns) and reference (ref)

Author(s)

Helen Lindsay

`setDNATileColours` *Sets colours for plotting aligned DNA sequences.*

Description

Sets tile colours for `plotAlignments` with a DNA alphabet

Usage

```
setDNATileColours(m)
```

Arguments

`m`	A matrix with a column named "value" of the characters at each tile position.

Value

A matrix with additional columns specifying tile and text colours

Author(s)

Helen Lindsay

`setMismatchTileColours` *Sets colours for plotting mismatches in aligned DNA sequences.*

Description

Sets tile colours for `plotAlignments` with a DNA alphabet.

Usage

```
setMismatchTileColours(m)
```

Arguments

`m`	A data frame of nucleotides and plotting locations, e.g. created by `transformAlnsToLong`

Value

A matrix with additional columns specifying tile and text colours

Author(s)

Helen Lindsay

transformAlnsToLong *Transform data for plotting*

Description

Orders and transforms a reference sequence and a set of aligned sequences into long format, i.e. one observation (tile position) per row. Used internally by `plotAlignments`.

Usage

```
transformAlnsToLong(ref, alns, add.other = FALSE)
```

Arguments

`ref`	The reference sequence
`alns`	Character vector of aligned sequences
`add.other`	Add a blank row labelled "Other" (Default: FALSE)

Value

A matrix of characters and plotting locations

Author(s)

Helen Lindsay

variantCounts *Get variant counts*

Description

Returns a matrix of counts where rows are sequence variants and columns are samples

Usage

```
variantCounts(obj, ...)

## S4 method for signature 'CrisprSet'
variantCounts(
  obj,
  ...,
  top.n = NULL,
  min.freq = 0,
  min.count = 1,
  include.chimeras = TRUE,
```

```
  include.nonindel = TRUE,
  result = "counts",
  filter.vars = NULL
)
```

Arguments

obj	An object containing variant counts
...	Additional arguments
top.n	(Integer n) If specified, return variants ranked at least n according to frequency across all samples (Default: 0, i.e. no cutoff)
min.freq	(Float n least one sample (Default: 0)
min.count	(Integer n) Return variants with count greater than n in at least one sample (Default: 0)
include.chimeras	Should chimeric reads be included in the counts table? (Default: TRUE)
include.nonindel	Should sequences without indels be returned? (Default:TRUE)
result	Return variants as either counts ("counts", default) or proportions ("proportions")
filter.vars	Labels of variants alleles to remove (Default: NULL)

Value

A matrix of counts where rows are variants and columns are samples

Author(s)

Helen Lindsay

Examples

```
data("gol_clutch1")

#Return a matrix of the 5 most frequent variants
variantCounts(gol, top.n = 5)
```

---

writeFastq *Append a sequence to a fastq file*

Description

Used by abifToFastq to write sanger sequences to fastq format As abifToFastq appends output to files, writeFastq checks that sequence names are unique. This function is faster with checking switched off.

Usage

```
writeFastq(outf, vals, allow_spaces = FALSE, check = TRUE)
```

Arguments

outf	Name of fastq file to append sequence
vals	A list containing entries named "seq" (sequence) and "quals" (quality scores, in ASCII format)
allow_spaces	Should spaces in the sequence name be substituted with underscores? TRUE or FALSE
check	Check whether reads with the same name already exist in the output fastq. (Default: TRUE)

Value

None. The sequences in "vals" are written to outf

Author(s)

Helen Lindsay

Index