

Package ‘PICB’

February 10, 2025

Title piRNA Cluster Builder

Version 0.99.15

Description piRNAs (short for PIWI-interacting RNAs) and their PIWI protein partners play a key role in fertility and maintaining genome integrity by restricting mobile genetic elements (transposons) in germ cells. piRNAs originate from genomic regions known as piRNA clusters. The piRNA Cluster Builder (PICB) is a versatile toolkit designed to identify genomic regions with a high density of piRNAs. It constructs piRNA clusters through a stepwise integration of unique and multimapping piRNAs and offers wide-ranging parameter settings, supported by an optimization function that allows users to test different parameter combinations to tailor the analysis to their specific piRNA system. The output includes extensive metadata columns, enabling researchers to rank clusters and extract cluster characteristics.

License CC0

Encoding UTF-8

Imports utils, GenomicRanges, GenomicAlignments, GenomeInfoDb, Biostrings, Rsamtools, data.table, IRanges, seqinr, stats, openxlsx, dplyr, S4Vectors, methods

Suggests knitr, rtracklayer, testthat, BiocStyle, prettydoc, BSgenome, BSgenome.Dmelanogaster.UCSC.dm6, BiocManager, rmarkdown, ggplot2

VignetteBuilder knitr

BugReports <https://github.com/HaaseLab/PICB/issues>

URL <https://github.com/HaaseLab/PICB>

biocViews Genetics, GenomeAnnotation, Sequencing, FunctionalPrediction, Coverage, Transcriptomics

RoxygenNote 7.3.2

git_url <https://git.bioconductor.org/packages/PICB>

git_branch devel
git_last_commit 7163d82
git_last_commit_date 2025-02-06
Repository Bioconductor 3.21
Date/Publication 2025-02-09
Author Pavol Genzor [aut],
 Aleksandr Friman [aut],
 Daniel Stoyko [aut],
 Parthena Konstantinidou [aut],
 Franziska Ahrend [aut, cre] (ORCID:
<https://orcid.org/0009-0004-7464-3444>),
 Zuzana Loubalova [aut],
 Yuejun Wang [aut],
 Hernan Lorenzi [aut],
 Astrid D Haase [aut]
Maintainer Franziska Ahrend <haase-lab-bioinfo@nih.gov>

Contents

| | |
|-------------------------------|-----------|
| PICB-package | 2 |
| PICBannotate | 5 |
| PICBbuild | 6 |
| PICBexporttoexcel | 9 |
| PICBgetchromosomes | 10 |
| PICBimportfromexcel | 10 |
| PICBload | 11 |
| PICBloadfasta | 13 |
| PICBoptimize | 13 |
| PICBstrandanalysis | 15 |
| Index | 16 |

PICB-package

PICB: piRNA Cluster Builder

Description

piRNAs (short for PIWI-interacting RNAs) and their PIWI protein partners play a key role in fertility and maintaining genome integrity by restricting mobile genetic elements (transposons) in germ cells. piRNAs originate from genomic regions known as piRNA clusters. The piRNA Cluster Builder (PICB) is a versatile toolkit designed to identify genomic regions with a high density of piRNAs. It constructs piRNA clusters through a stepwise integration of unique and multimapping piRNAs and offers wide-ranging parameter settings, supported by an optimization function that allows users to test different parameter combinations to tailor the analysis to their specific piRNA system. The output includes extensive metadata columns, enabling researchers to rank clusters and extract cluster characteristics.

Main Functions

The package provides several core functions:

- PICBload: Load and preprocess BAM files containing piRNA alignments
- PICBbuild: Build piRNA seeds/cores/clusters from alignments
- PICBoptimize: Optimize parameters for cluster building
- PICBstrandanalysis: Add sense/antisense ratio of unique piRNAs per piRNAcluster
- PICBannotate: Annotate GRanges according to a piRNA library
- PICBloadfasta: Get SeqInfo object from a fasta file
- PICBexporttoexcel: Export cluster object into an Excel file
- PICBimporttoexcel: Import cluster object from an Excel file

Workflow

A typical PICB workflow consists of:

1. Loading alignments with PICBload
2. Building clusters with PICBbuild
3. Optional parameter optimization with PICBoptimize
4. Optional strand analysis with PICBstrandanalysis
5. Exporting results with PICBexporttoexcel

Author(s)

Maintainer: Franziska Ahrend <haase-lab-bioinfo@nih.gov> ([ORCID](#))

Authors:

- Pavol Genzor
- Aleksandr Friman <haase-lab-bioinfo@nih.gov>
- Daniel Stoyko
- Parthena Konstantinidou <haase-lab-bioinfo@nih.gov>
- Zuzana Loubalova <haase-lab-bioinfo@nih.gov>
- Yuejun Wang <haase-lab-bioinfo@nih.gov>
- Hernan Lorenzi <haase-lab-bioinfo@nih.gov>
- Astrid D Haase <haase-lab-bioinfo@nih.gov>

See Also

Useful links:

- <https://github.com/HaaseLab/PICB>
- Report bugs at <https://github.com/HaaseLab/PICB/issues>

Examples

```
# 0. Load PICB
library(PICB)

# 1. Load Required Genome from e.g. Seqinfo (check all options in Vignette or ReadMe)
myGenome <- GenomeInfoDb::Seqinfo(
  seqnames = c("chr2L", "chr2R", "chr3L", "chr3R", "chr4", "chrX", "chrY"),
  seqlengths = c(23513712, 25286936, 28110227, 32079331, 1348131, 23542271, 3667352)
)

# 2. Load Example Data and Process Alignments
bam_file <- system.file("extdata", "Fly_Ov1_chr2L_20To21mb_filtered.bam",
  package = "PICB")

myAlignments <- PICBload(
  BAMFILE = bam_file,
  REFERENCE.GENOME = myGenome,
  VERBOSE = FALSE
)

# 3. Build piRNA Clusters
myClusters <- PICBbuild(
  IN.ALIGNMENTS = myAlignments,
  REFERENCE.GENOME = myGenome,
  LIBRARY.SIZE = 12799826, # Usually calculated automatically
  VERBOSITY = 0
)$clusters

# 4. Optimize Parameters (Optional)
parameterExploration <- PICBoptimize(
  IN.ALIGNMENTS = myAlignments,
  REFERENCE.GENOME = myGenome,
  MIN.UNIQUE.ALIGNMENTS.PER.WINDOW = c(1, 2, 3, 4, 5),
  LIBRARY.SIZE = 12799826, # Usually calculated automatically
  VERBOSITY = 1
)

# 5. Perform Strand Analysis
myClustersWithStrand <- PICBstrandanalysis(
  IN.ALIGNMENTS = myAlignments,
  IN.RANGES = myClusters
)

# 6. Export Clusters
PICBexporttoexcel(
  IN.RANGES = myClustersWithStrand,
  EXCEL.FILE.NAME = "myClusters_demonstration.xlsx"
)

# 7. Import Ranges
importedClusters <- PICBimportfromexcel(
  EXCEL.FILE.NAME = system.file("extdata", "myClusters_demonstration.xlsx", package = "PICB")
)
```

PICBannotate *Annotate GRanges according to a piRNA library*

Description

Annotate GRanges according to a piRNA library

Usage

```
PICBannotate(
  INPUT.GRANGES,
  ALIGNMENTS,
  REFERENCE.GENOME = NULL,
  REPLICATE.NAME = NULL,
  LIBRARY.SIZE = length(ALIGNMENTS$unique) + length(ALIGNMENTS$multi.primary),
  PROVIDE.NON.NORMALIZED = FALSE,
  SEQ.LEVELS.STYLE = "UCSC",
  COMPUTE.1U.10A.FRACTIONS = FALSE
)
```

Arguments

INPUT.GRANGES GRanges (seeds/cores/clusters) to annotate

ALIGNMENTS list of alignments from PICBload

REFERENCE.GENOME name of genome. For example "BSgenome.Dmelanogaster.UCSC.dm6"

REPLICATE.NAME name of the replicate. NULL by default.

LIBRARY.SIZE number of reads in the library. By default computed as number of unique mapping alignments + number of primary multimapping alignments.

PROVIDE.NON.NORMALIZED provide annotations in non-normalized format. False by default.

SEQ.LEVELS.STYLE naming of chromosomes style. "UCSC" by default.

COMPUTE.1U.10A.FRACTIONS for each locus and each alignments type (unique mapping, primary multimapping, secondary multimapping) compute fraction 1U and 10A containing reads overlapping the locus. Default FALSE.

Value

GRanges object with extra annotation columns

Author(s)

Aleksandr Friman

Examples

```

library(BSgenome.Dmelanogaster.UCSC.dm6)
myGenome <- "BSgenome.Dmelanogaster.UCSC.dm6"
myAlignmentsFromPICBload <- PICBload(
  BAMFILE = system.file("extdata", "Fly_Ov1_chr2L_20To21mb_filtered.bam", package = "PICB"),
  REFERENCE.GENOME = myGenome,
  VERBOSE = FALSE
)
myRangesFromPICBbuild <- PICBbuild(
  IN.ALIGNMENTS = myAlignmentsFromPICBload,
  REFERENCE.GENOME = myGenome,
  VERBOSITY = 0
)

myClustersFromPICBbuildAnnotationsRemoved <- GenomicRanges::granges(myRangesFromPICBbuild$clusters)

PICBannotate(
  INPUT.GRANGES = myClustersFromPICBbuildAnnotationsRemoved,
  ALIGNMENTS = myAlignmentsFromPICBload,
  REFERENCE.GENOME = myGenome,
  PROVIDE.NON.NORMALIZED = TRUE
)

```

PICBbuild

Build piRNA seeds/cores/clusters from alignments

Description

Build piRNA seeds/cores/clusters from alignments

Usage

```

PICBbuild(
  IN.ALIGNMENTS,
  REFERENCE.GENOME,
  UNIQUEMAPPERS.SLIDING.WINDOW.WIDTH = 350,
  UNIQUEMAPPERS.SLIDING.WINDOW.STEP = round(UNIQUEMAPPERS.SLIDING.WINDOW.WIDTH/10, 0),
  PRIMARY.MULTIMAPPERS.SLIDING.WINDOW.WIDTH = 350,
  PRIMARY.MULTIMAPPERS.SLIDING.WINDOW.STEP =
    round(PRIMARY.MULTIMAPPERS.SLIDING.WINDOW.WIDTH/10, 0),
  SECONDARY.MULTIMAPPERS.SLIDING.WINDOW.WIDTH = 1000,
  SECONDARY.MULTIMAPPERS.SLIDING.WINDOW.STEP =
    round(SECONDARY.MULTIMAPPERS.SLIDING.WINDOW.WIDTH/10, 0),
  LIBRARY.SIZE = length(IN.ALIGNMENTS$unique) + length(IN.ALIGNMENTS$multi.primary),
  MIN.UNIQUE.ALIGNMENTS.PER.WINDOW = 2 * (UNIQUEMAPPERS.SLIDING.WINDOW.WIDTH/1000) *
    (LIBRARY.SIZE/1e+06),
  MIN.UNIQUE.SEQUENCES.PER.WINDOW = min(MIN.UNIQUE.ALIGNMENTS.PER.WINDOW,
    round(UNIQUEMAPPERS.SLIDING.WINDOW.WIDTH/50, 0)),

```

```

MIN.PRIMARY.MULTIMAPPING.ALIGNMENTS.PER.WINDOW = 4 *
    (PRIMARY.MULTIMAPPERS.SLIDING.WINDOW.WIDTH/1000) * (LIBRARY.SIZE/1e+06),
MIN.SECONDARY.MULTIMAPPING.ALIGNMENTS.PER.WINDOW = 0.2 *
    (SECONDARY.MULTIMAPPERS.SLIDING.WINDOW.WIDTH/1000) * (LIBRARY.SIZE/1e+06),
MIN.SEED.LENGTH = 2 * UNIQUEMAPPERS.SLIDING.WINDOW.WIDTH + 100,
MIN.COVERED.SEED.LENGTH = 0,
THRESHOLD.SEEDS.GAP = 0,
THRESHOLD.CORES.GAP = 0,
THRESHOLD.CLUSTERS.GAP = 0,
SEQ.LEVELS.STYLE = "UCSC",
MIN.OVERLAP = 5,
PROVIDE.NON.NORMALIZED = FALSE,
COMPUTE.1U.10A.FRACCTIONS = FALSE,
VERBOSITY = 2
)

```

Arguments

```

IN.ALIGNMENTS    list of alignments from PICBload
REFERENCE.GENOME
    name of genome. For example "BSgenome.Dmelanogaster.UCSC.dm6"
UNIQUEMAPPERS.SLIDING.WINDOW.WIDTH
    width of sliding window for unique mappers. 350 nt by default
UNIQUEMAPPERS.SLIDING.WINDOW.STEP
    step of sliding windows for unique mappers. width/10 by default
PRIMARY.MULTIMAPPERS.SLIDING.WINDOW.WIDTH
    width of sliding window for primary multimapping alignments. 350 nt by default
PRIMARY.MULTIMAPPERS.SLIDING.WINDOW.STEP
    step of sliding windows for primary multimapping alignments. width/10 by default
SECONDARY.MULTIMAPPERS.SLIDING.WINDOW.WIDTH
    width of sliding window for secondary multimapping alignments. 1000 nt by default
SECONDARY.MULTIMAPPERS.SLIDING.WINDOW.STEP
    step of sliding windows for secondary multimapping alignments. width/10 by default
LIBRARY.SIZE      number of reads in the library. By default computed as number of unique mapping alignments + number of primary multimapping alignments.
MIN.UNIQUE.ALIGNMENTS.PER.WINDOW
    absolute number of unique mapping alignments per window to call it. By default computed as 2 FPKM.
MIN.UNIQUE.SEQUENCES.PER.WINDOW
    absolute number of unique mapping sequences per window to call it. By default computed as width/50.
MIN.PRIMARY.MULTIMAPPING.ALIGNMENTS.PER.WINDOW
    absolute number of primary multimapping alignments per window to call it. By default computed as 4 FPKM.

```

| | |
|--|---|
| MIN.SECONDARY.MULTIMAPPING.ALIGNMENTS.PER.WINDOW | absolute number of secondary multimapping alignments per window to call it. By default computed as 0.2 FPKM. |
| MIN.SEED.LENGTH | minimum length of a seed. By default computed as 2x unique mapper window size + 100. |
| MIN.COVERED.SEED.LENGTH | minimum number of seed nucleotides covered by unique mappers. 0 by default. |
| THRESHOLD.SEEDS.GAP | minimum gap between seeds to not merge them. 0 by default. |
| THRESHOLD.CORES.GAP | minimum gap between cores to not merge them. 0 by default. |
| THRESHOLD.CLUSTERS.GAP | minimum gap between clusters to not merge them. 0 by default. |
| SEQ.LEVELS.STYLE | naming of chromosomes style. "UCSC" by default. |
| MIN.OVERLAP | minimum overlap between seeds and cores, as well as between cores and clusters 5 nt by default. |
| PROVIDE.NON.NORMALIZED | include non-normalized to the library size statistics in the output annotations |
| COMPUTE.1U.10A.FRACTIONS | for each locus and each alignments type (unique mapping, primary multimapping, secondary multimapping) compute fraction 1U and 10A containing reads overlapping the locus. Default FALSE. |
| VERBOSITY | verbosity level 0/1/2/3. 2 by default. |

Value

list of annotated GRanges objects named "seeds" for seeds, "cores" for cores, "clusters" for clusters

Author(s)

Pavol Genzor
 Daniel Stoyko
 Aleksandr Friman
 Franziska Ahrend

Examples

```
library(BSgenome.Dmelanogaster.UCSC.dm6)
myAlignmentsFromPICBload <- PICBload(
  BAMFILE = system.file("extdata", "Fly_Ov1_chr2L_20To21mb_filtered.bam", package = "PICB"),
  REFERENCE.GENOME = "BSgenome.Dmelanogaster.UCSC.dm6",
  VERBOSE = FALSE
)

outputOfPICBbuild <- PICBbuild(
```



```

    IN.ALIGNMENTS = myAlignmentsFromPICBload,
    REFERENCE.GENOME = "BSgenome.Dmelanogaster.UCSC.dm6",
    VERBOSITY = 0
  )

```

PICBexporttoexcel *Export cluster object into an Excel file*

Description

Export cluster object into an Excel file

Usage

```
PICBexporttoexcel(IN.RANGES = NULL, EXCEL.FILE.NAME = NULL)
```

Arguments

```

IN.RANGES            clustering object to export
EXCEL.FILE.NAME      file name to save

```

Value

no values returned

Author(s)

Aleksandr Friman
Franziska Ahrend

Examples

```

library(BSgenome.Dmelanogaster.UCSC.dm6)
myAlignmentsFromPICBload <- PICBload(
  BAMFILE = system.file("extdata", "Fly_Ov1_chr2L_20To21mb_filtered.bam", package = "PICB"),
  REFERENCE.GENOME = "BSgenome.Dmelanogaster.UCSC.dm6",
  VERBOSE = FALSE
)

outputOfPICBbuild <- PICBbuild(
  IN.ALIGNMENTS = myAlignmentsFromPICBload,
  REFERENCE.GENOME = "BSgenome.Dmelanogaster.UCSC.dm6",
  LIBRARY.SIZE = 12799826, #usually not necessary
  VERBOSITY = 0
)
PICBexporttoexcel(
  IN.RANGES = outputOfPICBbuild,
  EXCEL.FILE.NAME = "inst/extdata/myClusters_demonstration.xlsx"
)

```

PICBgetchromosomes *Get SeqInfo object from standard non-circular chromosome names from your genome*

Description

Get SeqInfo object from standard non-circular chromosome names from your genome

Usage

```
PICBgetchromosomes(REFERENCE.GENOME, SEQ.LEVELS.STYLE = "UCSC")
```

Arguments

REFERENCE.GENOME

name of genome. For example "BSgenome.Dmelanogaster.UCSC.dm6", or directly a SeqInfo object.

SEQ.LEVELS.STYLE

naming of chromosomes style. "UCSC" by default.

Value

SeqInfo object with standard non-circular chromosome names

Author(s)

Aleksandr Friman

Franziska Ahrend

Examples

```
library(BSgenome.Dmelanogaster.UCSC.dm6)
mySI <- PICBgetchromosomes("BSgenome.Dmelanogaster.UCSC.dm6", "UCSC")
```

PICBimportfromexcel *Import cluster object from an Excel file*

Description

Import cluster object from an Excel file

Usage

```
PICBimportfromexcel(EXCEL.FILE.NAME = NULL)
```

Arguments

EXCEL.FILE.NAME
file name to import from

Value

list of annotated GRanges objects named "seeds" for seeds, "cores" for cores, "clusters" for clusters

Author(s)

Aleksandr Friman

Examples

```
importedClusters <- PICBimportfromexcel(  
  EXCEL.FILE.NAME = system.file("extdata", "myClusters_demonstration.xlsx", package = "PICB")  
)
```

PICBload

Load and preprocess BAM files containing piRNA alignments

Description

Load and preprocess BAM files containing piRNA alignments

Usage

```
PICBload(  
  BAMFILE = NULL,  
  REFERENCE.GENOME = NULL,  
  SIMPLE.CIGAR = TRUE,  
  IS.SECONDARY.ALIGNMENT = NA,  
  STANDARD.CONTIGS.ONLY = TRUE,  
  PERFECT.MATCH.ONLY = FALSE,  
  FILTER.BY.FLAG = TRUE,  
  SELECT.FLAG = c(0, 16, 272, 256),  
  USE.SIZE.FILTER = TRUE,  
  READ.SIZE.RANGE = c(18, 50),  
  TAGS = c("NH", "NM"),  
  WHAT = c("flag"),  
  SEQ.LEVELS.STYLE = "UCSC",  
  GET.ORIGINAL.SEQUENCE = FALSE,  
  VERBOSE = TRUE  
)
```

Arguments

| | |
|------------------------|---|
| BAMFILE | name of the bam file to load. Should be sorted and indexed. |
| REFERENCE.GENOME | name of genome. For example "BSgenome.Dmelanogaster.UCSC.dm6" |
| SIMPLE.CIGAR | simpleCigar parameter of Rsamtools::ScanBamParam |
| IS.SECONDARY.ALIGNMENT | defines loading of primary/secondary alignments. Default value NA loads both primary and secondary. |
| STANDARD.CONTIGS.ONLY | use only standard chromosomes |
| PERFECT.MATCH.ONLY | load only alignments without mismatches |
| FILTER.BY.FLAG | enables filtering by flag. TRUE by default. |
| SELECT.FLAG | vector of flags to use. Default value c(0,16, 272, 256). |
| USE.SIZE.FILTER | enables filter by alignment size. True by default. |
| READ.SIZE.RANGE | allowed alignment sizes. c(18,50) by default. |
| TAGS | tags to import from bam file. c("NH","NM") by default. |
| WHAT | "what" parameter of Rsamtools::ScanBamParam. c("flag") by default. |
| SEQ.LEVELS.STYLE | naming of chromosomes style. "UCSC" by default. |
| GET.ORIGINAL.SEQUENCE | adds "seq" to WHAT. False by default. |
| VERBOSE | enables progress output. True by default. |

Value

list of GRanges objects named "unique" for unique mapping alignments, "multi.primary" for primary multimapping alignments, "multi.secondary" for secondary multimapping alignments

Author(s)

Pavol Genzor
 Daniel Stoyko
 Aleksandr Friman
 Franziska Ahrend

Examples

```
library(BSgenome.Dmelanogaster.UCSC.dm6)
PICBload(
  BAMFILE = system.file("extdata", "Fly_Ov1_chr2L_20To21mb_filtered.bam", package = "PICB"),
  REFERENCE.GENOME = "BSgenome.Dmelanogaster.UCSC.dm6",
  VERBOSE = FALSE
)
```

| | |
|---------------|---|
| PICBloadfasta | <i>Get SeqInfo object from a fasta file</i> |
|---------------|---|

Description

Get SeqInfo object from a fasta file

Usage

```
PICBloadfasta(FASTA.NAME = NULL)
```

Arguments

FASTA.NAME path to the fasta file

Value

SeqInfo object with all chromosome names and lengths from the fasta file

Author(s)

Aleksandr Friman

Examples

```
library(BSgenome.Dmelanogaster.UCSC.dm6)

# create temporary fasta file
chr2L_seq <- BSgenome.Dmelanogaster.UCSC.dm6[["chr2L"]]
chr2L_seq_set <- DNASTringSet(chr2L_seq)
names(chr2L_seq_set) <- "chr2L"
temp_fasta <- tempfile(fileext = ".fasta")
writeXStringSet(chr2L_seq_set, temp_fasta)

myGenome <- PICBloadfasta(FASTA.NAME = temp_fasta)
unlink(temp_fasta)
```

| | |
|--------------|--|
| PICBoptimize | <i>Runs PICBbuild multiple times with provided parameters and returns optimization data frame.</i> |
|--------------|--|

Description

Runs PICBbuild multiple times with provided parameters and returns optimization data frame.

Usage

```
PICBoptimize(
  IN.ALIGNMENTS,
  REFERENCE.GENOME,
  LIBRARY.SIZE = length(IN.ALIGNMENTS$unique) + length(IN.ALIGNMENTS$multi.primary),
  VERBOSITY = 2,
  PROVIDE.INFO.SEEDS.AND.CORES = FALSE,
  SEQ.LEVELS.STYLE = "UCSC",
  ...
)
```

Arguments

IN.ALIGNMENTS list of alignments from PICBload

REFERENCE.GENOME name of genome. For example "BSgenome.Dmelanogaster.UCSC.dm6"

LIBRARY.SIZE number of reads in the library. By default computed as number of unique mapping alignments + number of primary multimapping alignments.

VERBOSITY verbosity level 0/1/2/3. 2 by default.

PROVIDE.INFO.SEEDS.AND.CORES FALSE by default.

SEQ.LEVELS.STYLE naming of chromosomes style. "UCSC" by default.

... rest of the parameters used by PICBbuild and provided as iterable vectors

Value

Optimization values dataframe

Author(s)

Aleksandr Friman

Examples

```
library(BSgenome.Dmelanogaster.UCSC.dm6)
myAlignmentsFromPICBload <- PICBload(
  BAMFILE = system.file("extdata", "Fly_Ov1_chr2L_20To21mb_filtered.bam", package = "PICB"),
  REFERENCE.GENOME = "BSgenome.Dmelanogaster.UCSC.dm6",
  VERBOSE = FALSE
)

PICBoptimize(
  IN.ALIGNMENTS = myAlignmentsFromPICBload,
  REFERENCE.GENOME = "BSgenome.Dmelanogaster.UCSC.dm6",
  MIN.UNIQUE.ALIGNMENTS.PER.WINDOW = c(1, 2, 3, 4, 5)
)
```

PICBstrandanalysis *Add sense/antisense ratio of unique piRNAs per piRNA cluster*

Description

Add sense/antisense ratio of unique piRNAs per piRNA cluster

Usage

```
PICBstrandanalysis(IN.ALIGNMENTS, IN.RANGES, VERBOSE = TRUE)
```

Arguments

IN.ALIGNMENTS list of alignments from PICBload
IN.RANGES single GRanges object (seeds, cores or clusters from PICBbuild)
VERBOSE enables progress output. True by default.

Value

GRanges object with an additional annotation column

Author(s)

Parthena Konstantinidou
Zuzana Loubalova
Franziska Ahrend

Examples

```
library(BSgenome.Dmelanogaster.UCSC.dm6)
myAlignmentsFromPICBload <- PICBload(
  BAMFILE = system.file("extdata", "Fly_0v1_chr2L_20To21mb_filtered.bam", package = "PICB"),
  REFERENCE.GENOME = "BSgenome.Dmelanogaster.UCSC.dm6",
  VERBOSE = FALSE
)

outputOfPICBbuild <- PICBbuild(
  IN.ALIGNMENTS = myAlignmentsFromPICBload,
  REFERENCE.GENOME = "BSgenome.Dmelanogaster.UCSC.dm6",
  VERBOSITY = 0
)

outputOfPICBbuild$clusters <- PICBstrandanalysis(
  IN.ALIGNMENTS = myAlignmentsFromPICBload,
  IN.RANGES = outputOfPICBbuild$clusters
)
```

Index

* **internal**

PICB-package, [2](#)

PICB (PICB-package), [2](#)

PICB-package, [2](#)

PICBannotate, [5](#)

PICBbuild, [6](#)

PICBexporttoexcel, [9](#)

PICBgetchromosomes, [10](#)

PICBimportfromexcel, [10](#)

PICBload, [11](#)

PICBloadfasta, [13](#)

PICBoptimize, [13](#)

PICBstrandanalysis, [15](#)