Package 'amplican'

April 11, 2018

Type Package

Title Automated analysis of CRISPR experiments.

Description `amplican` performs alignment of the amplicon reads, normalizes gathered data, calculates multiple statistics (e.g. cut rates, frameshifts) and presents results in form of aggregated reports. Data and statistics can be broken down by experiments, barcodes, user defined groups, guides and amplicons allowing for quick identification of potential problems.

Version 1.0.0

URL https://github.com/valenlab/amplican

BugReports https://github.com/valenlab/amplican/issues

biocViews Technology, Alignment, qPCR, CRISPR

License GPL-3

LazyData true

Depends R (>= 3.4.0), methods, BiocGenerics (>= 0.22.0), Biostrings (>= 2.44.2), data.table (>= 1.10.4)

Imports utils (>= 3.4.1), S4Vectors (>= 0.14.3), ShortRead (>= 1.34.0), IRanges (>= 2.10.2), GenomicRanges (>= 1.28.4), GenomeInfoDb (>= 1.12.2), BiocParallel (>= 1.10.1), gtable (>= 0.2.0), gridExtra (>= 2.2.1), ggplot2 (>= 2.2.0), ggbio (>= 1.24.1), ggthemes (>= 3.4.0), waffle (>= 0.7.0), stringr (>= 1.2.0), stats (>= 3.4.1), matrixStats (>= 0.52.2), Matrix (>= 1.2-10), dplyr (>= 0.7.2), rmarkdown (>= 1.6), knitr (>= 1.16), clusterCrit (>= 1.2.7)

RoxygenNote 6.0.1

Suggests testthat, BiocStyle

Collate 'helpers_general.R' 'AlignmentsExperimentSet-class.R' 'helpers_rmd.R' 'amplicanReport.R' 'helpers_directory.R' 'helpers_warnings.R' 'helpers_filters.R' 'helpers_alignment.R' 'amplicanAlign.R' 'amplican.R' 'amplicanFilter.R' 'amplicanNormalize.R' 'amplicanSummarize.R' 'helpers_plots.R'

VignetteBuilder knitr

NeedsCompilation no

Author Kornel Labun [aut], Eivind Valen [cph, cre]

Maintainer Eivind Valen <eivind.valen@gmail.com>

R topics documented:

`amplican` *Automated analysis of CRISPR experiments.*

Description

Main goals:

1. Flexible pipeline for analysis of the CRISPR Mi-Seq or Hi-Seq data.
2. Compatible with GRanges and data.table style.
3. Precise quantification of mutation rates.
4. Prepare automatic reports as .Rmd files that are flexible and open for manipulation.
5. Provide specialized plots for deletions, insertions, mismatches, variants, heterogeneity of the reads.

Details

To learn more about amplican, start with the vignettes: `browseVignettes(package = "amplican")`

Author(s)

Maintainer: Eivind Valen <eivind.valen@gmail.com> [copyright holder]

Authors:

- Kornel Labun <kornel.labun@gmail.com>

See Also

Useful links:

- https://github.com/valenlab/amplican
- Report bugs at https://github.com/valenlab/amplican/issues

`amplicanAlign` *Align reads to amplicons.*

Description

amplicanAlign takes a configuration files, fastq reads and output directory to prepare alignments and summary. It uses global Needleman-Wunsch algorithm with parameters optimized for CRISPR experiment. After alignments, object of `AlignmentsExperimentSet` is returned that allows for coercion into GRanges (plus is for forward and minus for reverse reads). It is also possible to output alignments in other, additional formats.

Usage

```
amplicanAlign(config, fastq_folder, use_parallel = FALSE,
  average_quality = 30, min_quality = 20,
  scoring_matrix = Biostrings::nucleotideSubstitutionMatrix(match = 5,
  mismatch = -4, baseOnly = TRUE, type = "DNA"), gap_opening = 25,
  gap_extension = 0, fastqfiles = 0.5, primer_mismatch = 0)
```

Arguments

`config`	(string) The path to your configuration file. For example: `system.file("extdata", "config.txt"` Configuration file can contain additional columns, but first 11 columns have to follow the example config specification.
`fastq_folder`	(string) Path to FASTQ files. If not specified, FASTQ files should be in the same directory as config file.
`use_parallel`	(boolean) Set to TRUE, if you have registered multicore back-end with `register`.
`average_quality`	(numeric) The FASTQ file have a quality for each nucleotide, depending on sequencing technology there exist many formats. This package uses `readFastq` to parse the reads. If the average quality of the reads fall below value of `average_quality` then sequence is filtered. Default is 0.
`min_quality`	(numeric) Similar as in average_quality, but depicts the minimum quality for ALL nucleotides in given read. If one of nucleotides has quality BELLOW `min_quality`, then the sequence is filtered. Default is 20.
`scoring_matrix`	(matrix) Default is 'NUC44'. Pass desired matrix using `nucleotideSubstitutionMatrix`.

`gap_opening` (numeric) The opening gap score.

`gap_extension` (numeric) The gap extension score.

`fastqfiles` (numeric) Normally you want to use both FASTQ files. But in some special cases, you may want to use only the forward file, or only the reverse file. Possible options:

- 0 Use both FASTQ files.
- 0.5 Use both FASTQ files, but only for one of the reads (forward or reverse) is required to have primer perfectly matched to sequence - eg. use when reverse reads are trimmed of primers, but forward reads have forward primer in the sequence.
- 1 Use only the forward FASTQ file.
- 2 Use only the reverse FASTQ file.

`primer_mismatch` (numeric) Decide how many mismatches are allowed during primer matching of the reads, that groups reads by experiments. When `primer_mismatch = 0` no mismatches are allowed, which can increase number of unasssigned read.

Value

(AlignmentsExperimentSet) Check `AlignmentsExperimentSet` class for details. You can use `lookupAlignment` to examine alignments visually.

See Also

Other analysis steps: `amplicanConsensus`, `amplicanFilter`, `amplicanMap`, `amplicanNormalize`, `amplicanOverlap`, `amplicanPipeline`, `amplicanReport`, `amplicanSummarize`

Examples

```
# path to example config file
config <- system.file("extdata", "config.csv", package = "amplican")
# path to example fastq files
fastq_folder <- system.file("extdata", package = "amplican")
aln <- amplicanAlign(config, fastq_folder)
aln
```

---

amplicanConsensus *Extract consensus out of forward and reverse events.*

Description

When forward and reverse reads are in agreement on the events (eg. deletion) `amplicanConsensus` will mark forward event as TRUE indicating that he represents consensus. In cases where forward and reverse read agree only partially, for example, they share the same start of the deletion, but they have different end `amplicanConsensus` will pick the version of read with higher alignment score, in situation where both of the reads overlap expected cut site, otherwise both events will be rejected and marked FALSE. When there are events only on one of the strands they will be rejected.

Usage

```
amplicanConsensus(aln, cfgT, overlaps = "overlaps", promiscuous = TRUE)
```

Arguments

aln	(data.frame) Contains relevant events in GRanges style.
cfgT	(data.frame) Should be table containing at least positions of primers in the amplicons and their identifiers
overlaps	(character) Specifies which metadata column of aln indicates which events are overlapping expected cut site.
promiscuous	(boolean) Allows to relax consensus rules. When TRUE will allow Indels that are not confirmed by the other strand (when both are used).

Details

In situation where you have only forward or only reverse reads don't use this function and assign all TRUE to all of your events.

Consensus out of the forward + reverse reads is required for amplicanSummary, and amplicanConsensus requires amplicanOverlap.

Value

(bolean vector) Where TRUE means that given event represents consensus out of forward and reverse reads.

See Also

Other analysis steps: amplicanAlign, amplicanFilter, amplicanMap, amplicanNormalize, amplicanOverlap, amplicanPipeline, amplicanReport, amplicanSummarize

Examples

```
file_path <- system.file("test_data", "test_aln.csv", package = "amplican")
aln <- data.table::fread(file_path)
cfgT <- data.table::fread(
  system.file("test_data", "test_cfg.csv", package = "amplican"))
all(aln$consensus == amplicanConsensus(aln, cfgT))
```

---

amplicanFilter *Filter Events Overlapping Primers, PRIMER DIMERS and Low Alignment Score Events.*

Description

Very often alignments return deletions that are not real deletions, but rather artifact of incomplete reads eg.:

```
ACTGAAAAA------- <- this "deletion" should be filtered
ACTG----ACTGACTG
```

We call them Events Overlapping Primers and filter them together with reads that are potentially PRIMER DIMERS. This filter will also remove all events coming from reads with low alignment score - potential Off-targets.

Usage

```
amplicanFilter(aln, cfgT, PRIMER_DIMER)
```

Arguments

aln	(data.frame) Should contain events from alignments in GRanges style with columns eg. seqnames, width, start, end.
cfgT	(data.frame) Needs columns Forward_Primer, ReversePrimer and Amplicon.
PRIMER_DIMER	(numeric) Value specifying buffer for PRIMER DIMER detection. For a given read it will be recognized as PRIMER DIMER when alignment will introduce gap of size bigger than: `length of amplicon - (lengths of PRIMERS + PRIMER_DIMER value)`

Value

(aln) Reduced by events classified as PRIMER DIMER or overlapping primers.

See Also

findPD and findEOP

Other analysis steps: amplicanAlign, amplicanConsensus, amplicanMap, amplicanNormalize, amplicanOverlap, amplicanPipeline, amplicanReport, amplicanSummarize

Examples

```
file_path <- system.file("extdata", "results", "alignments",
                         "raw_events.csv", package = "amplican")
aln <- data.table::fread(file_path)
cfgT <- data.table::fread(
  system.file("extdata", "results", "config_summary.csv",
              package = "amplican"))
amplicanFilter(aln, cfgT, 30)
```

---

amplicanMap *Map events to their respective relative coordinates specified with UPPER case.*

Description

Translate coordinates of GRanges events so that they can be relative to the amplicon. As point zero we assume first left sided UPPER case letter in the amplicon. Be weary that events for amplicons without expected cut sites are filtered. Don't use this function, if you don't have expected cut sites specified and don't use any of the metaplots.

Usage

```
amplicanMap(aln, cfgT)
```

Arguments

aln	(data.frame) List of events to map to the relative coordinates.
cfgT	(data.frame) config table

Value

(GRanges) Same as events, but the coordinates are relative to the

See Also

Other analysis steps: amplicanAlign, amplicanConsensus, amplicanFilter, amplicanNormalize, amplicanOverlap, amplicanPipeline, amplicanReport, amplicanSummarize

Examples

```
# example config
config <- read.csv(system.file("extdata", "config.csv",
                   package = "amplican"))
# example events
events <- read.csv(system.file("extdata", "results", "alignments",
                   "raw_events.csv", package = "amplican"))
# make events relative to the UPPER case
amplicanMap(events, config)
```

---

amplicanNormalize *Remove events that can be found in Controls.*

Description

This function can adjust events for small differences between known annotations (amplicon sequences) and real DNA of the strain that was sequenced. Events from the control are grouped by add and their frequencies are calculated in respect to number of total reads in that groups. In next step events from the control are filtered according to min_freq, all events below are treated as sequencing errors and rejected. Finally, all events that can be found in treatment group that find their exact match (by non skipped columns) in control group are removed. All events from control group are returned back.

Usage

```
amplicanNormalize(aln, cfgT, add = c("guideRNA", "Group"),
  skip = c("counts", "score", "seqnames", "read_id", "strand", "overlaps",
  "consensus"), min_freq = 0.01)
```

Arguments

aln	(data.frame) Contains events from alignments.
cfgT	(data.frame) Config table with information about experiments.
add	(character vector) Columns from cfgT that should be included in event table for normalization matching. Defaults to c("guideRNA", "Group") , which means that only those events created by the same guideRNA in the same Group will be removed if found in Control.

skip	(character vector) Specifies which columns of aln to skip.
min_freq	(numeric) All events from control group below this frequency will be not included in filtering. Use this to filter out background noise and sequencing errors.

Value

(data.frame) Same as aln, but events are normalized. Events from Control are not changed. Additionally columns from add are added to the data.frame.

See Also

Other analysis steps: amplicanAlign, amplicanConsensus, amplicanFilter, amplicanMap, amplicanOverlap, amplicanPipeline, amplicanReport, amplicanSummarize

Examples

```
aln <- data.frame(seqnames = 1:5, start = 1, end = 2, width = 2,
                  counts = 101:105)
cfgT <- data.frame(ID = 1:5, guideRNA = rep("ACTG", 5),
                   Reads_Filtered = c(2, 2, 3, 3, 4),
                   Group = c("A", "A", "B", "B", "B"),
                   Control = c(TRUE, FALSE, TRUE, FALSE, FALSE))
# all events are same as in the control group, therefore are filtered out
# events from control groups stay
amplicanNormalize(aln, cfgT)
# events that are different from control group are preserved
aln[2, "start"] <- 3
amplicanNormalize(aln, cfgT)
```

---

amplicanOverlap *Check which events overlap expected cut sites.*

Description

To determine which deletions, insertions and mismatches (events) are probably created by CRISPR we check whether they overlap expected cut sites. Expected cut sites should be specified in UPPER CASE letters in the amplicon sequences.

Usage

```
amplicanOverlap(aln, cfgT, cut_buffer = 5, relative = FALSE)
```

Arguments

aln	(data.frame) Contains relevant events in GRanges style.
cfgT	(data.frame) Contains amplicon sequences.
cut_buffer	(numeric) Number of bases that should expand 5' and 3' of the specified expected cut sites.
relative	(boolean) Sets whether events are relative to the position of the target site.

Value

(bolean vector) Where TRUE means that given event overlaps cut site.

See Also

Other analysis steps: amplicanAlign, amplicanConsensus, amplicanFilter, amplicanMap, amplicanNormalize, amplicanPipeline, amplicanReport, amplicanSummarize

Examples

```
file_path <- system.file("test_data", "test_aln.csv", package = "amplican")
aln <- data.table::fread(file_path)
cfgT <- data.table::fread(
  system.file("test_data", "test_cfg.csv", package = "amplican"))
all(aln$overlaps == amplicanOverlap(aln, cfgT))
```

---

amplicanPipeline *Wraps main package functionality into one function.*

Description

amplicanPipeline is convenient wrapper around all functionality of the package with the most robust settings. It will generate all results in the `result_folder` and also knit prepared reports into 'reports' folder.

Usage

```
amplicanPipeline(config, fastq_folder, results_folder, knit_reports = TRUE,
  write_alignments_format = "txt", average_quality = 30, min_quality = 0,
  use_parallel = FALSE,
  scoring_matrix = Biostrings::nucleotideSubstitutionMatrix(match = 5,
  mismatch = -4, baseOnly = TRUE, type = "DNA"), gap_opening = 25,
  gap_extension = 0, fastqfiles = 0.5, primer_mismatch = 0,
  PRIMER_DIMER = 30, cut_buffer = 5, promiscuous_consensus = TRUE,
  normalize = c("guideRNA", "Group"))
```

Arguments

`config`	(string) The path to your configuration file. For example: `system.file("extdata", "config.txt"` Configuration file can contain additional columns, but first 11 columns have to follow the example config specification.
`fastq_folder`	(string) Path to FASTQ files. If not specified, FASTQ files should be in the same directory as config file.
`results_folder`	(string) Where do you want to store results? The package will create files in that folder so make sure you have writing permissions.
`knit_reports`	(boolean) whether function should "knit" all reports automatically for you (it is time consuming, be patient), when false reports will be prepared, but not knitted

write_alignments_format

(character vector) Whether amplicanPipeline should write alignments results to separate files. Alignments are also always saved as .rds object of AlignmentsExperimentSet class. Possible options are:

- "fasta" outputs alignments in fasta format where header indicates experiment ID, read id and number of reads
- "txt" simple format, read information followed by forward read and amplicon sequence followed by reverse read with its amplicon sequence eg.:

```
ID: ID_1 Count: 7
ACTGAAAAA--------
ACTG-----ACTGACTG

------G-ACTG
ACTGACTGACTG
```

- "None" Don't write any alignments to files.
- c("fasta", "txt") There are also possible combinations of above formats, pass a vector to get alignments in multiple formats.

average_quality

(numeric) The FASTQ file have a quality for each nucleotide, depending on sequencing technology there exist many formats. This package uses readFastq to parse the reads. If the average quality of the reads fall below value of average_quality then sequence is filtered. Default is 0.

min_quality (numeric) Similar as in average_quality, but depicts the minimum quality for ALL nucleotides in given read. If one of nucleotides has quality BELLOW min_quality, then the sequence is filtered. Default is 20.

use_parallel (boolean) Set to TRUE, if you have registered multicore back-end with register.

scoring_matrix (matrix) Default is 'NUC44'. Pass desired matrix using nucleotideSubstitutionMatrix.

gap_opening (numeric) The opening gap score.

gap_extension (numeric) The gap extension score.

fastqfiles (numeric) Normally you want to use both FASTQ files. But in some special cases, you may want to use only the forward file, or only the reverse file. Possible options:

- 0 Use both FASTQ files.
- 0.5 Use both FASTQ files, but only for one of the reads (forward or reverse) is required to have primer perfectly matched to sequence - eg. use when reverse reads are trimmed of primers, but forward reads have forward primer in the sequence.
- 1 Use only the forward FASTQ file.
- 2 Use only the reverse FASTQ file.

primer_mismatch

(numeric) Decide how many mismatches are allowed during primer matching of the reads, that groups reads by experiments. When primer_mismatch = 0 no mismatches are allowed, which can increase number of unasssigned read.

PRIMER_DIMER (numeric) Value specifying buffer for PRIMER DIMER detection. For a given read it will be recognized as PRIMER DIMER when alignment will introduce gap of size bigger than:
length of amplicon - (lengths of PRIMERS + PRIMER_DIMER value)

cut_buffer
: The number of bases by which extend expected cut sites (specified as UPPER case letters in the amplicon) in 5' and 3' directions.

promiscuous_consensus
: (boolean) Whether rules of amplicanConsensus should be promiscuous. When promiscuous, we allow indels that have no confirmation on the other strand.

normalize
: (character vector) If column 'Control' in config table has all FALSE/0 values then normalization is skipped. Otherwise, normalization is strict, which means events that are found in 'Control' TRUE group will be removed in 'Control' FALSE group. This parameter by default uses columns 'guideRNA' and 'Group' to impose additional restrictions on normalized events eg. only events created by the same 'guideRNA' in the same 'Group' will be normalized.

Value

(invisible) results_folder path

See Also

Other analysis steps: amplicanAlign, amplicanConsensus, amplicanFilter, amplicanMap, amplicanNormalize, amplicanOverlap, amplicanReport, amplicanSummarize

Examples

```
# path to example config file
config <- system.file("extdata", "config.csv", package = "amplican")
# path to example fastq files
fastq_folder <- system.file("extdata", package = "amplican")
# output folder
results_folder <- tempdir()

#full analysis, not knitting files automatically
amplicanPipeline(config, fastq_folder, results_folder, knit_reports = FALSE)
```

---

amplicanReport *Prepare reports as .Rmd files.*

Description

amplicanReport takes a configuration file, fastq reads and output directory to prepare summaries as an editable .Rmd file. You can specify whether you want to make summaries based on ID, Barcode, Group or even guideRNA and Amplicon. This function automatically knits all reports after creation. If you want to postpone knitting and edit reports, use .Rmd templates to create your own version of reports instead of this function.

Usage

```
amplicanReport(results_folder, levels = c("id", "barcode", "group", "guide",
  "amplicon", "summary"), report_files = c("id_report", "barcode_report",
  "group_report", "guide_report", "amplicon_report", "index"), cut_buffer = 5,
  xlab_spacing = 4, top = 5, knit_reports = TRUE)
```

Arguments

results_folder	(string) Folder containing results from the amplicanAlign function, do not change names of the files.
levels	(vector) Possible values are: "id", "barcode", "group", "guide", "amplicon", "summary". You can also input more than one value eg. c("id", "barcode") will create two separate reports for each level.
report_files	(vector) You can supply your own names of the files. For each of the levels there has to be one file name. Files are created in current working directory by default.
cut_buffer	(numeric) Default 5. A number of bases that is used around the specified cut site.
xlab_spacing	(numeric) Default is 4. Spacing of the ticks on the x axis of plots.
top	(numeric) Default is 5. How many of the top most frequent unassigned reads to report? It is only relevant when you used forward and reverse reads. We align them to each other as we could not specify correct amplicon.
knit_reports	(boolean) Whether to knit reports automatically.

Value

(string) Path to the folder with results.

See Also

Other analysis steps: amplicanAlign, amplicanConsensus, amplicanFilter, amplicanMap, amplicanNormalize, amplicanOverlap, amplicanPipeline, amplicanSummarize

Examples

```
results_folder <- tempdir()
amplicanReport(results_folder, report_files = file.path(results_folder,
                                                        c("id_report",
                                                          "barcode_report",
                                                          "group_report",
                                                          "guide_report",
                                                          "amplicon_report",
                                                          "index")),
               knit_reports = FALSE)
```

---

amplicanSummarize *Summarize how many reads have frameshift and how many reads have deletions.*

Description

Before using this function make sure events are filtered to represent consensus with amplicanConsensus, if you use both forward and reverse reads. If you want to calculate metrics over expected cut site, filter events using amplicanOverlap.

Usage

```
amplicanSummarize(aln, cfgT)
```

Arguments

aln	(data.frame) Contains events from the alignments.
cfgT	(data.frame) Config file with the experiments details.

Details

Adds columns to cfgT:

- ReadsCut Count of reads with deletions overlapping expected cut site.
- Reads_Frameshifted Count of reads with frameshift overlapping expected cut site.

Value

(data.frame) As cfgT, but with extra columns.

See Also

Other analysis steps: amplicanAlign, amplicanConsensus, amplicanFilter, amplicanMap, amplicanNormalize, amplicanOverlap, amplicanPipeline, amplicanReport

Examples

```
file_path <- system.file("extdata", "results", "alignments",
                         "events_filtered_shifted_normalized.csv",
                         package = "amplican")
aln <- data.table::fread(file_path)
cfgT <- data.table::fread(
  system.file("extdata", "results", "config_summary.csv",
              package = "amplican"))
amplicanSummarize(aln, cfgT)
```

---

amplican_print_reads *Pretty print forward and reverse reads aligned to each other.*

Description

Usefull and needed for barcode reports.

Usage

```
amplican_print_reads(forward, reverse)
```

Arguments

forward	(character or vector of characters) Forward reads.
reverse	(character or vector of characters) Will be reverse complemented before alignment.

Value

Vector with alignments ready to be printed.

Examples

```
# load example data
unassigned_file <- system.file('extdata', 'results',  'alignments',
                               'unassigned_reads.csv', package = 'amplican')
unassigned <- data.table::setDF(data.table::fread(unassigned_file))
# sort by frequency
unassigned <- unassigned[order(unassigned$BarcodeFrequency,
                               decreasing = TRUE), ]
# print alignment of most frequent unassigned reads
cat(amplican_print_reads(unassigned[1, 'Forward'],
                         unassigned[1, 'Reverse']),
          sep = "\n")
```

`comb_along` *Generate all combinations along string exchanging m characters at a time with dictionary letters.*

Description

Generate all combinations along string `seq` swapping `m` characters at a time with letters defined in dictionary `letters`. Allows, for instance, to create a list of possible primers with two mismatches.

Usage

```
comb_along(seq, m = 2, letters = c("A", "C", "T", "G"))
```

Arguments

`seq`	(character) input character to permutate
`m`	(integer) number of elements to permutate at each step
`letters`	(character vector) dictionary source for combinations of elements

Value

(character vector) all unique combinations of permutated string

Examples

```
comb_along("AC")
comb_along("AAA", 1)
comb_along("AAA")
comb_along("AAA", 3)
comb_along("AAAAAAAAAA")
```

findEOP *Find Events Overlapping Primers.*

Description

Very often alignments return deletions that are not real deletions, but rather artifact of incomplete reads eg.:

```
ACTGAAAAA------- <- this "deletion" should be filtered
ACTG----ACTGACTG
```

Usage

```
findEOP(aln, cfgT)
```

Arguments

aln	(data.frame) Should contain events from alignments in GRanges style with columns eg. seqnames, width, start, end.
cfgT	(data.frame) Needs columns Forward_Primer, ReversePrimer and Amplicon.

Value

(logical vector) where TRUE indicates events that are overlapping primers

See Also

findPD findLQR

Other filters: findLQR, findPD

Examples

```
file_path <- system.file("extdata", "results", "alignments",
                         "raw_events.csv", package = "amplican")
aln <- data.table::fread(file_path)
cfgT <- data.table::fread(
  system.file("extdata", "results", "config_summary.csv",
              package = "amplican"))
findEOP(aln, cfgT)
```

findLQR *Find Off-targets and Fragmented alignments from reads.*

Description

Will try to detect off-targets and low quality alignments (outliers). It tries k-means clustering on normalized number of events per read and read alignment score. If there are 3 clusters (decided based on silhouette criterion) cluster with high event count and low alignment score will be marked for filtering. When there is less than 1000 scores in `aln` it will filter nothing.

Usage

```
findLQR(aln)
```

Arguments

`aln`	(data.frame) Should contain events from alignments in GRanges style with columns eg. seqnames, width, start, end, score.

Value

(logical vector) where TRUE indicates events that are potential off-targets or low quality alignments.

See Also

`findPD` `findEOP`

Other filters: `findEOP`, `findPD`

Examples

```
file_path <- system.file("extdata", "results", "alignments",
                         "raw_events.csv", package = "amplican")
aln <- data.table::fread(file_path)
aln <- aln[seqnames == "ID_1"] # for first experiment
findLQR(aln)
```

findPD *Find PRIMER DIMER reads.*

Description

Use to filter reads that are most likely PRIMER DIMERS.

Usage

```
findPD(aln, cfgT, PRIMER_DIMER = 30)
```

Arguments

aln	(data.frame) Should contain events from alignments in GRanges style with columns eg. seqnames, width, start, end.
cfgT	(data.frame) Needs columns Forward_Primer, ReversePrimer and Amplicon.
PRIMER_DIMER	(numeric) Value specifying buffer for PRIMER DIMER detection. For a given read it will be recognized as PRIMER DIMER when alignment will introduce gap of size bigger than: `length of amplicon - (lengths of PRIMERS + PRIMER_DIMER value)`

Value

(logical) Where TRUE indicates event classified as PRIMER DIMER

See Also

findEOP findLQR

Other filters: findEOP, findLQR

Examples

```
file_path <- system.file("extdata", "results", "alignments",
                         "raw_events.csv", package = "amplican")
aln <- data.table::fread(file_path)
cfgT <- data.table::fread(
  system.file("extdata", "results", "config_summary.csv",
              package = "amplican"))
findPD(aln, cfgT)
```

---

metaplot_deletions *MetaPlots deletions using ggplot2 and ggbio.*

Description

This function plots deletions in relation to the amplicons for given `selection` vector that groups values by given config `group`. All reads should already be converted to their relative position to their respective amplicon using amplicanMap. Top plot is for the forward reads and bottom plot is for reverse reads.

Usage

```
metaplot_deletions(alnmt, config, group, selection, over = "overlaps")
```

Arguments

alnmt	(data.frame) Loaded alignment information from events_filtered_shifted_normalized.csv file.
config	(data.frame) Loaded table from config_summary.csv file.
group	(string) Name of the column from the config file to use for grouping. Events are subselected based on this column and values from selection.

selection	(string or vector of strings) Values from config column specified in group argument.
over	(string) Specify which column contains overlaps with expected cut sites generated by amplicanOverlap

Value

(deletions metaplot) ggplot2 object of deletions metaplot

See Also

Other specialized plots: metaplot_insertions, metaplot_mismatches, plot_cuts, plot_deletions, plot_heterogeneity, plot_insertions, plot_mismatches, plot_variants

Examples

```
#example config
config <- read.csv(system.file("extdata", "results", "config_summary.csv",
                               package = "amplican"))
#example alignments results
alignments_file <- system.file("extdata", "results", "alignments",
                               "events_filtered_shifted_normalized.csv",
                               package = "amplican")
alignments <- read.csv(alignments_file)
metaplot_deletions(alignments[alignments$consensus, ],
                   config, "Group", "Betty")
```

---

metaplot_insertions *MetaPlots insertions using ggplot2 and ggbio.*

Description

This function plots insertions in relation to the amplicons for given selection vector that groups values by given config group. All reads should already be converted to their relative position to their respective amplicon using amplicanMap. Top plot is for the forward reads and bottom plot is for reverse reads.

Usage

```
metaplot_insertions(alnmt, config, group, selection)
```

Arguments

alnmt	(data.frame) Loaded alignment information from alignments_events.csv file.
config	(data.frame) Loaded table from config_summary.csv file.
group	(string) Name of the column from the config file to use for grouping. Events are subselected based on this column and values from selection.
selection	(string or vector of strings) Values from config column specified in group argument.

Value

(insertions metaplot) ggplot2 object of insertions metaplot

See Also

Other specialized plots: metaplot_deletions, metaplot_mismatches, plot_cuts, plot_deletions, plot_heterogeneity, plot_insertions, plot_mismatches, plot_variants

Examples

```
#example config
config <- read.csv(system.file("extdata", "results", "config_summary.csv",
                               package = "amplican"))
#example alignments results
alignments_file <- system.file("extdata", "results", "alignments",
                               "events_filtered_shifted_normalized.csv",
                               package = "amplican")
alignments <- read.csv(alignments_file)
metaplot_insertions(alignments[alignments$consensus, ], config,
                    "Group", "Betty")
```

---

metaplot_mismatches *MetaPlots mismatches using ggplot2 and ggbio.*

Description

Plots mismatches in relation to the amplicons for given `selection` vector that groups values by given config `group`. All reads should already be converted to their relative position to their respective amplicon using amplicanMap. Zero position on new coordinates is the most left UPPER case letter of the respective amplicon. This function filters out all alignment events that have amplicons without UPPER case defined. Top plot is for the forward reads and bottom plot is for reverse reads.

Usage

```
metaplot_mismatches(alnmt, config, group, selection)
```

Arguments

`alnmt`	(data.frame) Loaded alignment information from alignments_events.csv file.
`config`	(data.frame) Loaded table from config_summary.csv file.
`group`	(string) Name of the column from the config file to use for grouping. Events are subselected based on this column and values from selection.
`selection`	(string or vector of strings) Values from config column specified in group argument.

Value

(mismatches metaplot) ggplot2 object of mismatches metaplot

See Also

Other specialized plots: metaplot_deletions, metaplot_insertions, plot_cuts, plot_deletions, plot_heterogeneity, plot_insertions, plot_mismatches, plot_variants

Examples

```
#example config
config <- read.csv(system.file("extdata", "results", "config_summary.csv",
                               package = "amplican"))
#example alignments results
alignments_file <- system.file("extdata", "results", "alignments",
                               "events_filtered_shifted_normalized.csv",
                               package = "amplican")
alignments <- read.csv(alignments_file)
metaplot_mismatches(alignments[alignments$consensus & alignments$overlaps, ],
                    config, "Group", "Betty")
```

---

plot_cuts *Plots cuts using ggplot2 and ggbio.*

Description

This function plots cuts in relation to the amplicon with distinction for each ID.

Usage

```
plot_cuts(alignments, config, id, cut_buffer = 5, xlab_spacing = 4)
```

Arguments

alignments	(data.frame) Loaded alignment information from alignments_events.csv file.
config	(data.frame) Loaded table from config_summary.csv file.
id	(string or vector of strings) Name of the ID column from config file or name of multiple IDs if it is possible to group them. First amplicon will be used as the basis for plot.
cut_buffer	(numeric) Default is 5, you should specify the same as used in the analysis.
xlab_spacing	(numeric) Spacing of the x axis labels. Default is 4.

Value

(cuts plot) ggplot2 object of cuts plot

See Also

Other specialized plots: metaplot_deletions, metaplot_insertions, metaplot_mismatches, plot_deletions, plot_heterogeneity, plot_insertions, plot_mismatches, plot_variants

Examples

```
#example config
config <- read.csv(system.file("extdata", "results", "config_summary.csv",
                               package = "amplican"))
#example alignments results
alignments_file <- system.file("extdata", "results", "alignments",
                               "events_filtered_shifted_normalized.csv",
                               package = "amplican")
alignments <- read.csv(alignments_file)
plot_cuts(alignments[alignments$consensus & alignments$overlaps, ],
          config, c('ID_1','ID_3'))
```

`plot_deletions` *Plots deletions using ggplot2 and ggbio.*

Description

This function plots deletions in relation to the amplicon, assumes events are relative to the expected cut site. Top plot is for the forward reads, middle one shows amplicon sequence, and bottom plot is for reverse reads.

Usage

```
plot_deletions(alignments, config, id, cut_buffer = 5, xlab_spacing = 4,
  over = "overlaps")
```

Arguments

`alignments`	(data.frame) Loaded alignment information from alignments.csv file.
`config`	(data.frame) Loaded table from config_summary.csv file.
`id`	(string or vector of strings) Name of the ID column from config file or name of multiple IDs if it is possible to group them. First amplicon will be used as the basis for plot.
`cut_buffer`	(numeric) Default is 5, you should specify the same as used in the analysis.
`xlab_spacing`	(numeric) Spacing of the x axis labels. Default is 4.
`over`	(string) Specify which columns contains overlaps with expected cut sites generated by `amplicanOverlap`

Value

(deletions plot) ggplot2 object of deletions plot

See Also

Other specialized plots: `metaplot_deletions`, `metaplot_insertions`, `metaplot_mismatches`, `plot_cuts`, `plot_heterogeneity`, `plot_insertions`, `plot_mismatches`, `plot_variants`

Examples

```
#example config
config <- read.csv(system.file("extdata", "results", "config_summary.csv",
                               package = "amplican"))
#example alignments results
alignments_file <- system.file("extdata", "results", "alignments",
                               "events_filtered_shifted_normalized.csv",
                               package = "amplican")
alignments <- read.csv(alignments_file)
plot_deletions(alignments[alignments$consensus, ], config, c('ID_1','ID_3'))
```

`plot_height` *Get figure height in inches for number of elements on y axis.*

Description

Helper function to calculate figure height based on number of elements to plot for automating sizes of figures in knited reports.

Usage

```
plot_height(x)
```

Arguments

x	(numeric) number of elements to fit onto height axis

Value

(numeric) In inches

Examples

```
plot_height(20)
```

`plot_heterogeneity` *Plots heterogeneity of the reads using ggplot2 and ggbio.*

Description

This function creates stacked barplot explaining reads heterogeneity. It groups reads by user defined levels and measures how unique are reads in this level. Uniqueness of reads is simplified to the bins and colored according to the color gradient. Default color black indicates very high heterogeneity of the reads. The more yellow (default) the more similar are reads and less heterogeneous.

Usage

```
plot_heterogeneity(alignments, config, level = "ID", colors = c("#000000",
  "#F0E442"), bins = c(0, 5, seq(10, 100, 10)))
```

Arguments

alignments	(data.frame) Loaded alignment information from alignments_events.csv file.
config	(data.frame) Loaded table from config_summary.csv file.
level	(string) Name of the column from config file specifying levels to group by.
colors	(html colors vector) Two colours for gradient, eg. c('#000000', '#F0E442').
bins	(numeric vector) Numeric vector from 0 to 100 specifying bins eg. c(0, 5, seq(10, 100, 10)).

Value

(heterogeneity plot) ggplot2 object of heterogeneity plot

See Also

Other specialized plots: metaplot_deletions, metaplot_insertions, metaplot_mismatches, plot_cuts, plot_deletions, plot_insertions, plot_mismatches, plot_variants

Examples

```
#example config
config <- read.csv(system.file("extdata", "results", "config_summary.csv",
                               package = "amplican"))
#example alignments results
alignments_file <- system.file("extdata", "results", "alignments",
                               "events_filtered_shifted_normalized.csv",
                               package = "amplican")
alignments <- read.csv(alignments_file)
plot_heterogeneity(alignments[alignments$consensus, ], config)
```

---

plot_insertions *Plots insertions using ggplot2 and ggbio.*

Description

This function plots insertions in relation to the amplicon. Top plot is for the forward reads, middle one shows amplicon sequence, and bottom plot is for reverse reads.

Usage

```
plot_insertions(alignments, config, id, cut_buffer = 5, xlab_spacing = 4)
```

Arguments

alignments	(data.frame) Loaded alignment information from alignments_events.csv file.
config	(data.frame) Loaded table from config_summary.csv file.
id	(string or vector of strings) Name of the ID column from config file or name of multiple IDs if it is possible to group them. First amplicon will be used as the basis for plot.
cut_buffer	(numeric) Default is 5, you should specify the same as used in the analysis.
xlab_spacing	(numeric) Spacing of the x axis labels. Default is 4.

Value

(insertions plot) ggplot2 object of insertions plot

See Also

Other specialized plots: metaplot_deletions, metaplot_insertions, metaplot_mismatches, plot_cuts, plot_deletions, plot_heterogeneity, plot_mismatches, plot_variants

Examples

```
#example config
config <- read.csv(system.file("extdata", "results", "config_summary.csv",
                               package = "amplican"))
#example alignments results
alignments_file <- system.file("extdata", "results", "alignments",
                               "events_filtered_shifted_normalized.csv",
                               package = "amplican")
alignments <- read.csv(alignments_file)
plot_insertions(alignments[alignments$consensus, ], config, c('ID_1','ID_3'))
```

---

plot_mismatches *Plots mismatches using ggplot2 and ggbio.*

Description

Plots mismatches in relation to the amplicon, assumes your reads are relative to the respective amplicon sequences predicted cut sites. Top plot is for the forward reads, middle one shows amplicon sequence, and bottom plot is for reverse reads.

Usage

```
plot_mismatches(alignments, config, id, cut_buffer = 5, xlab_spacing = 4)
```

Arguments

alignments	(data.frame) Loaded alignment information from alignments_events.csv file.
config	(data.frame) Loaded table from config_summary.csv file.
id	(string or vector of strings) Name of the ID column from config file or name of multiple IDs, if it is possible to group them. They have to have the same amplicon, amplicons on the reverse strand will be reverse complemented to match forward strand amplicons.
cut_buffer	(numeric) Default is 5, you should specify the same as used in the analysis.
xlab_spacing	(numeric) Spacing of the x axis labels. Default is 4.

Value

(mismatches plot) ggplot2 object of mismatches plot

See Also

Other specialized plots: metaplot_deletions, metaplot_insertions, metaplot_mismatches, plot_cuts, plot_deletions, plot_heterogeneity, plot_insertions, plot_variants

Examples

```
#example config
config <- read.csv(system.file("extdata", "results", "config_summary.csv",
                               package = "amplican"))
#example alignments results
alignments_file <- system.file("extdata", "results", "alignments",
                               "events_filtered_shifted_normalized.csv",
                               package = "amplican")
alignments <- read.csv(alignments_file)
plot_mismatches(alignments[alignments$consensus, ], config,
                c('ID_1', 'ID_3'))
```

---

plot_variants *Plots most frequent variants using ggplot2 and ggbio.*

Description

This function plots variants in relation to the amplicon. Shows sequences of top mutants without aggregating on deletions, insertions and mismatches.

Usage

```
plot_variants(alignments, config, id, cut_buffer = 5, top = 10,
  annot = "codon")
```

Arguments

alignments	(data.frame) Loaded alignment information from alignments_events.csv file.
config	(data.frame) Loaded table from config_summary.csv file.
id	(string or vector of strings) Name of the ID column from config file or name of multiple IDs if it is possible to group them. First amplicon will be used as the basis for plot.
cut_buffer	(numeric) Default is 5, you should specify the same as used in the analysis.
top	(numeric) Specify number of most frequent reads to plot. By default it is 10. Check plot_heterogeneity to see how many reads will be enough to give good overview of your variants.
annot	("codon" or NA) What to display for annotation top plot. When NA will not display anything.

Details

Top plot shows all six possible frames for given amplicon. Amino acids are colored as follows:

Small nonpolar	G, A, S, T	Orange
Hydrophobic	C, V, I, L, P, F, Y, M, W	Green
Polar	N, Q, H	Magenta
Negatively charged	D, E	Red
Positively charged	K, R	Blue
Other	eg. *, U, +	Grey

Variant plot shows amplicon reference, UPPER letters which were the basis for window selection are highlighted with dashed white box (guideRNA). Black triangles are reflecting insertion points. Dashed letters indicate deletions. Table associated with variant plot represents:

- Freq - Frequency of given read in experiment. Variants are ordered by frequency value.
- Count - Represents raw count of this variant reads in experiment.
- F - Sum of deletion and insertion widths of events overlapping presented window. Green background indicates frameshift.

Value

(variant plot) ggplot2 object of variants plot

Note

This function is inspired by `plotAlignments`.

See Also

Other specialized plots: `metaplot_deletions`, `metaplot_insertions`, `metaplot_mismatches`, `plot_cuts`, `plot_deletions`, `plot_heterogeneity`, `plot_insertions`, `plot_mismatches`

Examples

```
#example config
config <- read.csv(system.file("extdata", "results", "config_summary.csv",
                               package = "amplican"))
#example alignments results
alignments_file <- system.file("extdata", "results", "alignments",
                               "events_filtered_shifted_normalized.csv",
                               package = "amplican")
alignments <- read.csv(alignments_file)
plot_variants(alignments[alignments$consensus & alignments$overlaps, ],
              config, c('ID_1','ID_3'))
```

Index