

# Package: amplican (via r-universe)

October 2, 2024

**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** 1.27.0

**URL** <https://github.com/valenlab/amplican>

**BugReports** <https://github.com/valenlab/amplican/issues>

**biocViews** ImmunoOncology, Technology, Alignment, qPCR, CRISPR

**License** GPL-3

**LinkingTo** Rcpp

**Depends** R (>= 3.5.0), methods, BiocGenerics (>= 0.22.0), Biostrings (>= 2.44.2), pwalgn, data.table (>= 1.10.4-3)

**Imports** Rcpp, 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 (>= 3.3.4), 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), cluster (>= 2.1.4)

**RoxygenNote** 7.3.1

**Suggests** testthat, BiocStyle, GenomicAlignments

**Collate** 'helpers\_general.R' 'AlignmentsExperimentSet-class.R'  
'RcppExports.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'  
'ggforce\_bezier.R' 'helpers\_plots.R'

**VignetteBuilder** knitr

**Encoding** UTF-8

**Repository** <https://bioc.r-universe.dev>

**RemoteUrl** <https://github.com/bioc/amplican>

**RemoteRef** HEAD

**RemoteSha** 345126698110d02358ae347b0832105bc84b374f

## Contents

|  |    |
|--|----|
| amplican . . . . .                     | 3  |
| amplicanAlign . . . . .                | 3  |
| amplicanConsensus . . . . .            | 6  |
| amplicanFilter . . . . .               | 7  |
| amplicanMap . . . . .                  | 8  |
| amplicanNormalize . . . . .            | 9  |
| amplicanOverlap . . . . .              | 10 |
| amplicanPipeline . . . . .             | 11 |
| amplicanPipelineConservative . . . . . | 14 |
| amplicanReport . . . . .               | 18 |
| amplicanSummarize . . . . .            | 19 |
| amplican_print_reads . . . . .         | 20 |
| cigarsToEvents . . . . .               | 21 |
| comb_along . . . . .                   | 22 |
| findEOP . . . . .                      | 22 |
| findLQR . . . . .                      | 23 |
| findPD . . . . .                       | 24 |
| geom_bezier . . . . .                  | 25 |
| getEvents . . . . .                    | 28 |
| is_hdr_strict . . . . .                | 29 |
| metaplot_deletions . . . . .           | 30 |
| metaplot_insertions . . . . .          | 31 |
| metaplot_mismatches . . . . .          | 32 |
| pairToEvents . . . . .                 | 33 |
| plot_cuts . . . . .                    | 33 |
| plot_deletions . . . . .               | 34 |
| plot_height . . . . .                  | 35 |
| plot_heterogeneity . . . . .           | 36 |
| plot_insertions . . . . .              | 37 |
| plot_mismatches . . . . .              | 38 |
| plot_variants . . . . .                | 39 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>42</b> |
|--------------|-----------|

---

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,
  filter_n = FALSE,
  batch_size = 1e+06,
  scoring_matrix = Biostrings::nucleotideSubstitutionMatrix(match = 5, mismatch = -4,
    baseOnly = FALSE, type = "DNA"),
  gap_opening = 25,
  gap_extension = 0,
  fastqfiles = 0.5,
  primer_mismatch = 0,
  donor_mismatch = 3,
  donor_strict = FALSE
)

```

**Arguments**

|                 |  |
|-----------------|--|
| config          | (string) The path to your configuration file. For example: <code>system.file("extdata", "config.txt", package = "amplican")</code> . 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.  |
| average_quality | (numeric) The FASTQ file have a quality for each nucleotide, depending on sequencing technology there exist many formats. This package uses <a href="#">readFastq</a> 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 BELOW min_quality, then the sequence is filtered. Default is 20.   |
| filter_n        | (boolean) Whether to filter out reads containing N base.   |
| batch_size      | (numeric) How many reads to analyze at a time? Needed for filtering of large fastq files.  |
| scoring_matrix  | (matrix) Default is 'NUC44'. Pass desired matrix using <a href="#">nucleotideSubstitutionMatrix</a> .  |
| 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 unassigned read.

donor\_mismatch

(numeric) How many events of length 1 (mismatches, deletions and insertions of length 1) are allowed when aligning toward the donor template. This parameter is only used when donor template is specified. The higher the parameter the less strict will be algorithm accepting read as HDR. Set to 0 if only perfect alignments to the donor template marked as HDR, unadvised due to error rate of the sequencers.

donor\_strict

(logical) Applies more strict algorithm for HDR detection. Only these reads that have all of the donor events will count as HDR. Tolerates 'donor\_mismatch' level of noise, but no indels are allowed. Use this when your reads should span over the whole window of the donor events. Might be more time consuming.

## 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\(\)](#), [amplicanPipelineConservative\(\)](#), [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 | <i>Extract consensus out of forward and reverse events.</i> |
|-------------------|---|

---

## 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

(boolean 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\(\)](#), [amplicanPipelineConservative\(\)](#), [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 | <i>Filter Events Overlapping Primers, PRIMER DIMERS and Low Alignment Score Events.</i> |
|----------------|---|

---

**Description**

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

```
ACTGAAAA----- <- 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:<br>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\(\)](#), [amplicanPipelineConservative\(\)](#), [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 | <i>Map events to their respective relative coordinates specified with UPPER case.</i> |
|-------------|---|

---

## 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 expected cut sites.

## See Also

Other analysis steps: [amplicanAlign\(\)](#), [amplicanConsensus\(\)](#), [amplicanFilter\(\)](#), [amplicanNormalize\(\)](#), [amplicanOverlap\(\)](#), [amplicanPipeline\(\)](#), [amplicanPipelineConservative\(\)](#), [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 | <i>Remove events that can be found in Controls.</i> |
|-------------------|---|

---

## 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()`, `amplicanPipelineConservative()`, `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)
```

---

|                              |   |
|------------------------------|---|
| <code>amplicanOverlap</code> | <i>Check which events overlap expected cut sites.</i> |
|------------------------------|---|

---

**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**

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

**Value**

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

**See Also**

Other analysis steps: `amplicanAlign()`, `amplicanConsensus()`, `amplicanFilter()`, `amplicanMap()`, `amplicanNormalize()`, `amplicanPipeline()`, `amplicanPipelineConservative()`, `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 | <i>Wraps main package functionality into one function.</i> |
|------------------|--|

---

**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 = "None",
  average_quality = 30,
  min_quality = 0,
  filter_n = FALSE,
  batch_size = 1e+07,
  use_parallel = FALSE,
  scoring_matrix = Biostrings::nucleotideSubstitutionMatrix(match = 5, mismatch = -4,
    baseOnly = FALSE, type = "DNA"),
  gap_opening = 25,
  gap_extension = 0,
  fastqfiles = 0.5,
  primer_mismatch = 2,
  donor_mismatch = 3,
  donor_strict = FALSE,
  PRIMER_DIMER = 30,
  event_filter = TRUE,
  cut_buffer = 5,
```

```

promiscuous_consensus = TRUE,
normalize = c("guideRNA", "Group"),
min_freq = min_freq_default,
continue = TRUE
)

```

## Arguments

- |                         |   |
|-------------------------|---|
| config                  | (string) The path to your configuration file. For example: <code>system.file("extdata", "config.txt", package = "amplican")</code> . 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 | <p>(character vector) Whether amplicanPipeline should write alignments results to separate files. Alignments are also always saved as .rds object of <a href="#">AlignmentsExperimentSet</a> class. Possible options are:</p> <ul style="list-style-type: none"> <li>• "fasta" outputs alignments in fasta format where header indicates experiment ID, read id and number of reads</li> <li>• "txt" simple format, read information followed by forward read and amplicon sequence followed by reverse read with its amplicon sequence eg.:</li> </ul> <pre style="margin-left: 40px;"> ID: ID_1 Count: 7 ACTGAAAAA----- ACTG-----ACTGACTG  -----G-ACTG ACTGACTGACTG </pre> <ul style="list-style-type: none"> <li>• "None" Don't write any alignments to files.</li> <li>• c("fasta", "txt") There are also possible combinations of above formats, pass a vector to get alignments in multiple formats.</li> </ul> |
| average_quality         | (numeric) The FASTQ file have a quality for each nucleotide, depending on sequencing technology there exist many formats. This package uses <a href="#">readFastq</a> 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 BELOW min_quality, then the sequence is filtered. Default is 20.  |
| filter_n                | (boolean) Whether to filter out reads containing N base.  |

|                       |   |
|-----------------------|---|
| batch_size            | (numeric) How many reads to analyze at a time? Needed for filtering of large fastq files.   |
| use_parallel          | (boolean) Set to TRUE, if you have registered multicore back-end.   |
| scoring_matrix        | (matrix) Default is 'NUC44'. Pass desired matrix using <a href="#">nucleotideSubstitutionMatrix</a> .   |
| 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: <ul style="list-style-type: none"> <li>• 0 Use both FASTQ files.</li> <li>• 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.</li> <li>• 1 Use only the forward FASTQ file.</li> <li>• 2 Use only the reverse FASTQ file.</li> </ul> |
| 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 unassigned read.   |
| donor_mismatch        | (numeric) How many events of length 1 (mismatches, deletions and insertions of length 1) are allowed when aligning toward the donor template. This parameter is only used when donor template is specified. The higher the parameter the less strict will be algorithm accepting read as HDR. Set to 0 if only perfect alignments to the donor template marked as HDR, unadvised due to error rate of the sequencers.   |
| donor_strict          | (logical) Applies more strict algorithm for HDR detection. Only these reads that have all of the donor events will count as HDR. Tolerates 'donor_mismatch' level of noise, but no indels are allowed. Use this when your reads should span over the whole window of the donor events. Might be more time consuming.  |
| 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:<br>length of amplicon - (lengths of PRIMERS + PRIMER_DIMER value)   |
| event_filter          | (logical) Whether detection of offtarget reads, should be enabled.  |
| 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 <a href="#">amplicanConsensus</a> 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.                                    |
| min_freq | (numeric) All events below this frequency are treated as sequencing errors and rejected. This parameter is used during normalization through <a href="#">amplicanNormalize</a> . |
| continue | (boolean) Default TRUE, decides whether to continue failed ampliCan runs. In case of FALSE, all contents in 'results' folder will be removed.                                    |

## Value

(invisible) results\_folder path

## See Also

Other analysis steps: [amplicanAlign\(\)](#), [amplicanConsensus\(\)](#), [amplicanFilter\(\)](#), [amplicanMap\(\)](#), [amplicanNormalize\(\)](#), [amplicanOverlap\(\)](#), [amplicanPipelineConservative\(\)](#), [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)
```

---

amplicanPipelineConservative

*Wraps main package functionality into one function.*

---

## Description

amplicanPipelineIndexHopping is identical as amplicanPipeline except that default min\_freq threshold is set to 0.15. Setting this threshold higher will decrease risks of inadequate normalization in cases of potential Index Hopping, potentially decreasing precision of true editing rate calling. Index Hopping can be mitigated with use of unique dual indexing pooling combinations. However, in cases when you might expect Index Hopping to occur you should use this function instead of amplicanPipeline.

**Usage**

```
amplicanPipelineConservative(
  config,
  fastq_folder,
  results_folder,
  knit_reports = TRUE,
  write_alignments_format = "None",
  average_quality = 30,
  min_quality = 0,
  filter_n = FALSE,
  batch_size = 1e+07,
  use_parallel = FALSE,
  scoring_matrix = Biostrings::nucleotideSubstitutionMatrix(match = 5, mismatch = -4,
    baseOnly = FALSE, type = "DNA"),
  gap_opening = 25,
  gap_extension = 0,
  fastqfiles = 0.5,
  primer_mismatch = 2,
  donor_mismatch = 3,
  donor_strict = FALSE,
  PRIMER_DIMER = 30,
  event_filter = TRUE,
  cut_buffer = 5,
  promiscuous_consensus = TRUE,
  normalize = c("guideRNA", "Group"),
  min_freq = min_freq_default,
  continue = TRUE
)
```

**Arguments**

- |                         |  |
|-------------------------|--|
| config                  | (string) The path to your configuration file. For example: <code>system.file("extdata", "config.txt", package = "amplican")</code> . 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 | <p>(character vector) Whether amplicanPipeline should write alignments results to separate files. Alignments are also always saved as <code>.rds</code> object of <code>AlignmentsExperimentSet</code> class. Possible options are:</p> <ul style="list-style-type: none"> <li>• "fasta" outputs alignments in fasta format where header indicates experiment ID, read id and number of reads</li> </ul> |

- "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 BELOW min\_quality, then the sequence is filtered. Default is 20.

#### filter\_n

(boolean) Whether to filter out reads containing N base.

#### batch\_size

(numeric) How many reads to analyze at a time? Needed for filtering of large fastq files.

#### use\_parallel

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

#### 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 unassigned read.

#### donor\_mismatch

(numeric) How many events of length 1 (mismatches, deletions and insertions of length 1) are allowed when aligning toward the donor template. This parameter is only used when donor template is specified. The higher the parameter the



|                       |  |
|-----------------------|--|
|                       | less strict will be algorithm accepting read as HDR. Set to 0 if only perfect alignments to the donor template marked as HDR, unadvised due to error rate of the sequencers.   |
| donor_strict          | (logical) Applies more strict algorithm for HDR detection. Only these reads that have all of the donor events will count as HDR. Tolerates 'donor_mismatch' level of noise, but no indels are allowed. Use this when your reads should span over the whole window of the donor events. Might be more time consuming.   |
| 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:<br>length of amplicon - (lengths of PRIMERS + PRIMER_DIMER value)  |
| event_filter          | (logical) Whether detection of offtarget reads, should be enabled.   |
| 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 <a href="#">amplicanConsensus</a> 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. |
| min_freq              | (numeric) All events below this frequency are treated as sequencing errors and rejected. This parameter is used during normalization through <a href="#">amplicanNormalize</a> .   |
| continue              | (boolean) Default TRUE, decides whether to continue failed amplican runs. In case of FALSE, all contents in 'results' folder will be removed.  |

## Details

result\_folder and also knit prepared reports into 'reports' folder.

## Value

(invisible) results\_folder path

## See Also

Other analysis steps: [amplicanAlign\(\)](#), [amplicanConsensus\(\)](#), [amplicanFilter\(\)](#), [amplicanMap\(\)](#), [amplicanNormalize\(\)](#), [amplicanOverlap\(\)](#), [amplicanPipeline\(\)](#), [amplicanReport\(\)](#), [amplicanSummarize\(\)](#)

---

|                |                                       |
|----------------|---------------------------------------|
| amplicanReport | <i>Prepare reports as .Rmd files.</i> |
|----------------|---------------------------------------|

---

## 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 <a href="#">amplicanAlign</a> 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\(\)](#), [amplicanPipelineConservative\(\)](#), [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 | <i>Summarize how many reads have frameshift and how many reads have deletions.</i> |
|-------------------|--|

---

**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()`, `amplicanPipelineConservative()`, `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**

|                      |  |
|----------------------|--|
| <code>forward</code> | (character or vector of characters) Forward reads.                                 |
| <code>reverse</code> | (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")
```

---

|                |   |
|----------------|---|
| cigarsToEvents | <i>Transform extended CIGAR strings into GRanges.</i> |
|----------------|---|

---

## Description

Transform extended CIGAR strings into [GRanges](#) representation with events of deletions, insertions and mismatches.

## Usage

```
cigarsToEvents(
  cigars,
  aln_pos_start,
  query_seq,
  ref,
  read_id,
  mapq,
  seqnames,
  strands,
  counts
)
```

## Arguments

|               |   |
|---------------|---|
| cigars        | (character) Extended CIGARS.  |
| aln_pos_start | (integer) Pos of CIGARS.  |
| query_seq     | (character) Aligned query sequences.  |
| ref           | (character) Reference sequences used for alignment.                             |
| read_id       | (numeric) Read id for assignment for each of the CIGARS.                        |
| mapq          | (numeric) Mapping scores.   |
| seqnames      | (character) Names of the sequences, potentially ids of the reference sequences. |
| strands       | (character) Strands to assign.  |
| counts        | (integer) Vector of cigar counts, if data collapsed.                            |

## Value

([GRanges](#)) Same as events.

---

|            |  |
|------------|--|
| comb_along | <i>Generate all combinations along string exchanging m characters at a time with dictionary letters.</i> |
|------------|--|

---

### 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 permute                            |
| m       | (integer) number of elements to permute at each step              |
| letters | (character vector) dictionary source for combinations of elements |

### Value

(character vector) all unique combinations of permuted string

### Examples

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

---

|         |   |
|---------|---|
| findEOP | <i>Find Events Overlapping Primers.</i> |
|---------|---|

---

### Description

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

```
ACTGAAAA----- <- 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 | <i>Find PRIMER DIMER reads.</i> |
|--------|---------------------------------|

---

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)

```

geom\_bezier

*Create quadratic or cubic bezier curves [copied from ggforce]***Description**

This set of functionality is copied from ggforce package due to dependency issues on Bioconductor and is used internally (not exported) only. This set of geoms makes it possible to connect points creating either quadratic or cubic beziers. bezier works by calculating points along the bezier and connecting these to draw the curve.

**Usage**

```

stat_bezier(
  mapping = NULL,
  data = NULL,
  geom = "path",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  n = 100,
  inherit.aes = TRUE,
  ...
)

```

```

geom_bezier(
  mapping = NULL,
  data = NULL,
  stat = "bezier",
  position = "identity",
  arrow = NULL,
  lineend = "butt",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  n = 100,
  ...
)

```

**Arguments**

|             |  |
|-------------|--|
| mapping     | Set of aesthetic mappings created by <a href="#">aes()</a> . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.  |
| data        | <p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to <a href="#">ggplot()</a>.</p> <p>A <code>data.frame</code>, or other object, will override the plot data. All objects will be fortified to produce a data frame. See <a href="#">fortify()</a> for which variables will be created.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</p>  |
| geom        | <p>The geometric object to use to display the data for this layer. When using a <code>stat_*()</code> function to construct a layer, the <code>geom</code> argument can be used to override the default coupling between stats and geoms. The <code>geom</code> argument accepts the following:</p> <ul style="list-style-type: none"> <li>• A <code>Geom</code> ggproto subclass, for example <code>GeomPoint</code>.</li> <li>• A string naming the geom. To give the geom as a string, strip the function name of the <code>geom_</code> prefix. For example, to use <code>geom_point()</code>, give the geom as "point".</li> <li>• For more information and other ways to specify the geom, see the <a href="#">layer geom</a> documentation.</li> </ul>  |
| position    | <p>A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The <code>position</code> argument accepts the following:</p> <ul style="list-style-type: none"> <li>• The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position.</li> <li>• A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as "jitter".</li> <li>• For more information and other ways to specify the position, see the <a href="#">layer position</a> documentation.</li> </ul> |
| na.rm       | If <code>FALSE</code> , the default, missing values are removed with a warning. If <code>TRUE</code> , missing values are silently removed.  |
| show.legend | logical. Should this layer be included in the legends? <code>NA</code> , the default, includes if any aesthetics are mapped. <code>FALSE</code> never includes, and <code>TRUE</code> always includes. It can also be a named logical vector to finely select the aesthetics to display.   |
| n           | The number of points to create for each segment  |
| inherit.aes | If <code>FALSE</code> , overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <a href="#">borders()</a> .   |
| ...         | Other arguments passed on to <a href="#">layer()</a> 's <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the   |

position argument, or aesthetics that are required can *not* be passed through ... Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through ... This can be one of the functions described as [key glyphs](#), to change the display of the layer in the legend.

|                      |  |
|----------------------|--|
| <code>stat</code>    | The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following: <ul style="list-style-type: none"> <li>• A Stat ggproto subclass, for example <code>StatCount</code>.</li> <li>• A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as <code>"count"</code>.</li> <li>• For more information and other ways to specify the stat, see the <a href="#">layer stat</a> documentation.</li> </ul> |
| <code>arrow</code>   | Arrow specification, as created by <code>grid::arrow()</code> .  |
| <code>lineend</code> | Line end style (round, butt, square).  |

## Details

Input data is understood as a sequence of data points the first being the start point, then followed by one or two control points and then the end point. More than 4 and less than 3 points per group will throw an error.

## Aesthetics

`geom_link`, `geom_link2` and `geom_lino` understand the following aesthetics (required aesthetics are in bold):

- **`x`** - **`y`** - `color` - `linewidth` - `linetype` - `alpha` - `lineend`

**Computed variables**

**x, y** The interpolated point coordinates

**index** The progression along the interpolation mapped between 0 and 1

**Author(s)**

Thomas Lin Pedersen

**Examples**

```
beziers <- data.frame(
  x = c(1, 2, 3, 4, 4, 6, 6),
  y = c(0, 2, 0, 0, 2, 2, 0),
  type = rep(c('cubic', 'quadratic'), c(3, 4)),
  point = c('end', 'control', 'end', 'end', 'control', 'control', 'end')
)
help_lines <- data.frame(
  x = c(1, 3, 4, 6),
  xend = c(2, 2, 4, 6),
  y = 0,
  yend = 2
)
ggplot2::ggplot() + ggplot2::geom_segment(
  ggplot2::aes(x = x, xend = xend, y = y, yend = yend),
  data = help_lines,
  arrow = ggplot2::arrow(length = ggplot2::unit(c(0, 0, 0.5, 0.5), 'cm')),
  colour = 'grey') +
  amplican::geom_bezier(ggplot2::aes(x= x, y = y, group = type, linetype = type),
    data = beziers) +
  ggplot2::geom_point(ggplot2::aes(x = x, y = y, colour = point), data = beziers)
```

---

getEvents

---

*Transform aligned strings into GRanges representation of events.*


---

**Description**

Transforms aligned strings into GRanges representation with events of deletions, insertions and mismatches. Subject should come from one amplicon sequence, after alignment to many sequences (patterns).

**Usage**

```
getEvents(
  pattern,
  subject,
  scores,
  ID = "NA",
```

```

    ampl_shift = 1L,
    ampl_start = 1L,
    strand_info = "+"
  )

```

### Arguments

|             |  |
|-------------|--|
| pattern     | (character) Aligned pattern.   |
| subject     | (character) Aligned subject.   |
| scores      | (integer) Alignment scores of the pattern and subject.   |
| ID          | (character) Will be used as seqnames of output GRanges.  |
| ampl_shift  | (numeric) Possible shift of the amplicons.   |
| ampl_start  | (numeric) Real amplicon starts. <a href="#">pairwiseAlignment</a> clips alignments, therefore to output GRanges relative to the amplicon sequence (subject) ranges have to be shifted. |
| strand_info | (character) Strands to assign.   |

### Value

([GRanges](#)) Same as events.

---

|               |   |
|---------------|---|
| is_hdr_strict | <i>Figure out which reads conform to the HDR using the donor.</i> |
|---------------|---|

---

### Description

This is strict detection as compared to 'is\_hdr' which was designed to be less specific and allow for all kinds of donors. This method requires that you have exactly the same events (mismatches, insertions, deletions) as the difference between amplicon and donor sequences. It ignores everything else, so other mismatches and small indels etc. as noise are allowed here for valid HDR.

### Usage

```
is_hdr_strict(aln, cfgT, scoring_matrix, gap_opening = 25, gap_extension = 0)
```

### Arguments

|                |   |
|----------------|---|
| aln            | (data.table) This are events that contain already consensus column, they are also shifted and normalized. |
| cfgT           | (data.table) Config data.table with columns for amplicon and donor.                                       |
| scoring_matrix | (scoring matrix)  |
| gap_opening    | (integer)   |
| gap_extension  | (integer)   |

### Value

(aln) same as aln on entry, but readType is updated to TRUE when read is recognized as HDR

---

metaplot\_deletions      *MetaPlots deletions using ggplot2.*


---

## 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 <a href="#">amplicanOverlap</a>                         |

## 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.*


---

## 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.*


---

## 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,
                    config, "Group", "Betty")
```



---

|              |  |
|--------------|--|
| pairToEvents | <i>Read "pair" format of EMBOSS needle into GRanges as events.</i> |
|--------------|--|

---

### Description

Parse EMBOSS needle (or needleall) "pair" format into GRanges representation with events of deletions, insertions and mismatches. Make sure that each file corresponds to single subject (single amplicon). Assumes that bottom sequence "-bsequence" corresponds to the "subject" and full sequence alignment is returned.

### Usage

```
pairToEvents(file, ID = "NA", strand_info = "+")
```

### Arguments

|             |   |
|-------------|---|
| file        | (character) File path.  |
| ID          | (character) ID of the experiment, will be used as seqnames of the reutner ranges. |
| strand_info | (character) Strand to assign.   |

### Value

([GRanges](#)) Same as events.

---

|           |                                  |
|-----------|----------------------------------|
| plot_cuts | <i>Plots cuts using ggplot2.</i> |
|-----------|----------------------------------|

---

### 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) gtable 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 | <i>Plots deletions using ggplot2.</i> |
|----------------|---------------------------------------|

---

**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 <a href="#">amplicanOverlap</a>  |

**Value**

(deletions plot) gtable 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)
p <- 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 | <i>Plots heterogeneity of the reads using ggplot2.</i> |
|--------------------|--|

---

**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 | <i>Plots insertions using ggplot2.</i> |
|-----------------|--|

---

## 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) gtable 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)
p <- plot_insertions(alignments, config, c('ID_1','ID_3'))
```

---

|                 |  |
|-----------------|--|
| plot_mismatches | <i>Plots mismatches using ggplot2.</i> |
|-----------------|--|

---

## 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) gtable 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)
id <- c('ID_1', 'ID_3'); cut_buffer = 5; xlab_spacing = 4;
p <- plot_mismatches(alignments, config, c('ID_1', 'ID_3'))
ggplot2::ggsave("~/test.png", p, width = 25, units = "in")
```

---

plot\_variants

*Plots most frequent variants using ggplot2.*

---

## 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 = if (amplican::get_seq(config, id, "Donor") == "") "cov" else NA,
  summary_plot = amplican::get_seq(config, id, "Donor") == "",
  frameshift = amplican::get_seq(config, id, "Donor") == ""
)
```

## 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. If Donor is available we will try to add the first donor and mark it on the 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 <a href="#">plot_heterogeneity</a> to see how many reads will be enough to give good overview of your variants.  |

|   |   |
|---|---|
| annot   | ("codon" or "cov" or NA) What to display for annotation top plot. When NA will not display anything, also not display total summary. Codon plot is all reading frames for a given window, and the default "cov" is coverage of all indels and mismatches over a given window. |
| summary_plot                                    | (boolean) Whether small summary plot in the upper right corner should be displayed. Top bar summarizes total reads with frameshift (F), reads with Edits without Frameshift (Edits) and reads without Edits (Match).  |
| frameshift                                      | (boolean) Whether to include Frameshift column in the table.  |
| annot                      on               off |   |

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) gtable 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)

alignments <- alignments[alignments$consensus & alignments$overlaps, ]
p <- plot_variants(alignments[alignments$consensus & alignments$overlaps, ],
                  config, c('ID_1','ID_3'))

# with Donor we dont plot summary and the annot, summary plot and frameshift
p <- plot_variants(alignments[alignments$consensus & alignments$overlaps, ],
                  config, c('ID_5'))
```

# Index

## \* analysis steps

- [amplicanAlign](#), 3
- [amplicanConsensus](#), 6
- [amplicanFilter](#), 7
- [amplicanMap](#), 8
- [amplicanNormalize](#), 9
- [amplicanOverlap](#), 10
- [amplicanPipeline](#), 11
- [amplicanPipelineConservative](#), 14
- [amplicanReport](#), 18
- [amplicanSummarize](#), 19

## \* filters

- [findEOP](#), 22
- [findLQR](#), 23
- [findPD](#), 24

## \* specialized plots

- [metaplot\\_deletions](#), 30
- [metaplot\\_insertions](#), 31
- [metaplot\\_mismatches](#), 32
- [plot\\_cuts](#), 33
- [plot\\_deletions](#), 34
- [plot\\_heterogeneity](#), 36
- [plot\\_insertions](#), 37
- [plot\\_mismatches](#), 38
- [plot\\_variants](#), 39

[aes\(\)](#), 26

[AlignmentsExperimentSet](#), 3, 5, 12, 15

[amplican](#), 3

[amplican-package \(amplican\)](#), 3

[amplican\\_print\\_reads](#), 20

[amplicanAlign](#), 3, 6–8, 10, 11, 14, 17–20

[amplicanConsensus](#), 5, 6, 7, 8, 10, 11, 13, 14, 17, 19, 20

[amplicanFilter](#), 5, 6, 7, 8, 10, 11, 14, 17, 19, 20

[amplicanMap](#), 5–7, 8, 10, 11, 14, 17, 19, 20, 30–32

[amplicanNormalize](#), 5–8, 9, 11, 14, 17, 19, 20

[amplicanOverlap](#), 5–8, 10, 10, 14, 17, 19, 20, 30, 35

[amplicanPipeline](#), 5–8, 10, 11, 11, 17, 19, 20

[amplicanPipelineConservative](#), 5–8, 10, 11, 14, 14, 19, 20

[amplicanReport](#), 5–8, 10, 11, 14, 17, 18, 20

[amplicanSummarize](#), 5–8, 10, 11, 14, 17, 19, 19

[borders\(\)](#), 26

[cigarsToEvents](#), 21

[comb\\_along](#), 22

[findEOP](#), 7, 22, 24

[findLQR](#), 23, 23, 24

[findPD](#), 7, 23, 24, 24

[fortify\(\)](#), 26

[geom\\_bezier](#), 25

[getEvents](#), 28

[ggplot\(\)](#), 26

[GRanges](#), 8, 21, 24, 29, 33

[grid::arrow\(\)](#), 27

[is\\_hdr\\_strict](#), 29

[key\\_glyphs](#), 27

[layer geom](#), 26

[layer position](#), 26

[layer stat](#), 27

[layer\(\)](#), 26, 27

[lookupAlignment](#), 5

[metaplot\\_deletions](#), 30, 31, 32, 34–38, 40

[metaplot\\_insertions](#), 30, 31, 32, 34–38, 40

[metaplot\\_mismatches](#), 30, 31, 32, 34–38, 40

[nucleotideSubstitutionMatrix](#), 4, 13, 16

[pairToEvents](#), 33

pairwiseAlignment, [29](#)  
plot\_cuts, [30–32](#), [33](#), [35–38](#), [40](#)  
plot\_deletions, [30–32](#), [34](#), [34](#), [36–38](#), [40](#)  
plot\_height, [35](#)  
plot\_heterogeneity, [30–32](#), [34](#), [35](#), [36](#),  
[37–40](#)  
plot\_insertions, [30–32](#), [34–36](#), [37](#), [38](#), [40](#)  
plot\_mismatches, [30–32](#), [34–37](#), [38](#), [40](#)  
plot\_variants, [30–32](#), [34–38](#), [39](#)  
plotAlignments, [40](#)  
  
readFastq, [4](#), [12](#), [16](#)  
  
stat\_bezier (geom\_bezier), [25](#)