Package ‘minSNPs’

August 8, 2023

Title Resolution-Optimised SNPs Searcher

Version 0.1.0


Depends R (>= 3.4.0)

License MIT + file LICENSE

Imports BiocParallel, data.table

Encoding UTF-8

RoxygenNote 7.2.3

Suggests knitr, testthat, pkgdown, rmarkdown, withr

VignetteBuilder knitr

URL https://github.com/ludwigHoon/minSNPs

NeedsCompilation no

Author Ludwig Kian Soon Hoon [aut, cre]

Maintainer Ludwig Kian Soon Hoon <ldwgkshoon@gmail.com>

Repository CRAN

Date/Publication 2023-08-08 08:40:02 UTC

R topics documented:

```
calculate_percent ........................................................ 3
calculate_simpson .......................................................... 3
calculate_variant_within_group ......................................... 4
cal_fn ................................................................. 4
cal_fp ................................................................. 5
```

R topics documented:

cal_met_snp ......................................................... 5
check_fasta_meta_mapping .................................... 6
check_multistate .................................................. 6
check_percent ..................................................... 7
combine_fastq_search_result .................................... 7
combine_search_string_result .................................... 8
combine_search_string_result_from_files ...................... 8
combine_search_string_result_from_list ....................... 9
estimate_coverage ............................................... 10
extend_length .................................................... 11
find_optimised_snps ............................................. 12
full_merge ........................................................ 13
full_merge_1 ....................................................... 14
generate_kmers ................................................... 15
generate_kmer_search_string .................................... 15
generate_pattern .................................................. 16
generate_snp_search_string ...................................... 16
get_all_process_methods ......................................... 17
get_metric_fun ..................................................... 18
get_snps_set ....................................................... 18
identify_overlaps ................................................ 19
infer_from_combined ............................................. 19
iterate_merge ..................................................... 20
iterate_through ................................................... 21
match_count ....................................................... 21
merge fasta ........................................................ 21
output_result ...................................................... 22
output_to_files .................................................... 23
process_allele ...................................................... 24
process_kmer_result .............................................. 25
process_result_file .............................................. 25
process_snp_result .............................................. 26
read_fasta ........................................................ 26
read_sequences_from_fastq ....................................... 27
remove_snp_conflict ............................................ 27
resolve_IUPAC_missing .......................................... 28
reverse_complement .............................................. 28
search_from_fastq_reads ........................................ 30
sequence_reads_match_count .................................... 31
view_percent ....................................................... 31
view_simpson ....................................................... 32
write_fasta ......................................................... 32

Index 33
**calculate_percent**

Description

calculate_percent is used to calculate dissimilarity index, proportion of isolates not in goi that have been discriminated against. 1 being all and 0 being none.

Usage

calculate_percent(pattern, goi)

Arguments

- **pattern**: list of sequences’ pattern (profile)
- **goi**: group of interest

Value

Will return the dissimilarity index of the list of patterns.

**calculate_simpson**

Description

calculate_simpson is used to calculate Simpson’s index. Which is in the range of 0-1, where the greater the value, the more diverse the population.

Usage

calculate_simpson(pattern)

Arguments

- **pattern**: list of sequences’ pattern (profile)

Value

Will return the Simpson’s index of the list of patterns.
**calculate_variant_within_group**

**Usage**

`calculate_variant_within_group(pattern, meta, target, get_count = FALSE)`

**Arguments**

- `pattern`: list of sequences’ pattern (profile)
- `meta`: metadata of the sequences
- `target`: column name of the target group
- `get_count`: whether to return the count of samples rather than the raw number, default to FALSE.

**Value**

Will return the Simpson’s index of the list of patterns.

---

**cal_fn**

**Description**

`cal_fn` is used to check if the proportion of false negative fastas and metas are compatible.

**Usage**

`cal_fn(pattern, goi, target)`

**Arguments**

- `pattern`: the pattern from `generate_pattern`
- `goi`: the group of interest (names of isolates)
- `target`: the target sequence(s)

**Value**

proportion: no. false negative/number of isolates
**cal_fp**

Description

*cal_fp* is used to check if the proportion of false positive fastas and metas are compatible.

Usage

```
cal_fp(pattern, goi, target)
```

Arguments

- **pattern**: the pattern from `generate_pattern`
- **goi**: the group of interest (names of isolates)
- **target**: the target sequence(s)

Value

- proportion: no. false positive/number of isolates

**cal_met_snp**

Description

*cal_met_snp* is used to calculate the metric at each position

Usage

```
cal_met_snp(position, metric, seqc, prepend_position = c(), ...)
```

Arguments

- **position**: position to check
- **metric**: either 'simpson' or 'percent'
- **seqc**: list of sequences, either passed directly from `process_allele` or `read_fasta` or equivalence
- **prepend_position**: is the position to be added to the
- **...**: other parameters as needed

Value

- return the value at that position, as well as base pattern for next iteration.
**check_fasta_meta_mapping**

**Description**

check_fasta_meta_mapping is used to check if fastas and metas are compatible.

**Usage**

check_fasta_meta_mapping(fasta, meta)

**Arguments**

- **fasta**
  - the fasta read into memory to join
- **meta**
  - the meta read into memory to join

**Value**

TRUE/FALSE if the fasta and meta are compatible

**check_multistate**

**Description**

check_multistate is used to remove positions where there are more than 1 state within the group of interest.

**Usage**

check_multistate(position, sequences)

**Arguments**

- **position**
  - position to check
- **sequences**
  - sequences from group of interest

**Value**

return ‘TRUE’ if the position contains multistate otherwise ‘FALSE’
**Description**

`check_percent` is used to check if parameters needed by `calculate_percent` are all present.

**Usage**

```r
check_percent(list_of_parameters)
```

**Arguments**

- `list_of_parameters` is a list of parameter passed to functions that will perform the calculation

**Value**

TRUE if goi exists, else FALSE

---

**Description**

`combine_fastq_search_result` combines the search results from `search_from_fastq_reads`

**Usage**

```r
combine_fastq_search_result(
  results,
  search_table,
  previous_result = NULL,
  bp = MulticoreParam()
)
```

**Arguments**

- `results` the result (fastq_search_result) from `search_from_fastq_reads` to combine.
- `search_table` a dataframe with the following columns: - "id", "type", "sequence", "strand", "result", "extra", "match_ref_seq"
- `previous_result` the result (fastq_search_result) to append to
- `bp` BiocParallel backend to use for parallelization
Value

will return a dataframe containing: - 'sequence', 'search_id', 'reads', 'raw_match', 'mean_qualities', 'indexes', 'id', 'type', 'strand', 'result', 'extra', 'match_ref_seq', 'n_reads'

---

**combine_search_string_result_from_files**

Description

combine_search_string_result_from_files combines the search results from temp file generated from search_from_fastq_reads

Usage

```r
combine_search_string_result_from_files(
  results,
  search_table,
  append_to_current_result = data.frame(),
  bp = MulticoreParam()
)
```

Arguments

- **results**: the dataframes to collapse.
- **search_table**: a dataframe with the following columns: - "id", "type", "sequence", "strand", "result", "extra", "match_ref_seq"
- **append_to_current_result**: the dataframe of previous result to append to
- **bp**: BiocParallel backend to use for parallelization

Value

will return a dataframe containing: - 'sequence', 'search_id', 'reads', 'raw_match', 'mean_qualities', 'indexes', 'id', 'type', 'strand', 'result', 'extra', 'match_ref_seq', 'n_reads'

---

**combine_search_string_result**

Description

combine_search_string_result combines the search results from search_from_fastq_reads

Usage

```r
combine_search_string_result(
  results,
  search_table,
  append_to_current_result = data.frame(),
  bp = MulticoreParam()
)
```

Arguments

- **results**: the dataframes to collapse.
- **search_table**: a dataframe with the following columns: - 'sequence', 'search_id', 'reads', 'raw_match', 'mean_qualities', 'indexes', 'id', 'type', 'strand', 'result', 'extra', 'match_ref_seq', 'n_reads'

Value

will return a dataframe containing: - 'sequence', 'search_id', 'reads', 'raw_match', 'mean_qualities', 'indexes', 'id', 'type', 'strand', 'result', 'extra', 'match_ref_seq', 'n_reads'
Usage

combine_search_string_result_from_files(
  result_files,
  search_table,
  read_length_files = c(),
  append_to_current_result = NULL,
  bp = MulticoreParam()
)

Arguments

result_files  the output files from search_from_fastq_reads to combine
search_table  a dataframe with the following columns: - "id","type","sequence","strand","result","extra","match_ref_seq"
read_length_files  the read_length output files from search_from_fastq_reads
append_to_current_result  the fastq_search_result of result to append to
bp  BiocParallel backend to use for parallelization

Value


Description

combine_search_string_result_from_list combines the search results from search_from_fastq_reads

Usage

combine_search_string_result_from_list(
  results,
  search_table,
  append_to_current_result = data.frame(),
  bp = MulticoreParam()
)
**estimate_coverage**

**Arguments**

- **results**  
  the dataframes from `search_from_fastq_reads` to combine.

- **search_table**  
  a dataframe with the following columns: - "id", "type", "sequence", "strand", "result", "extra", "match_ref_seq"

- **append_to_current_result**  
  the dataframe of previous result to append to

- **bp**  
  BioCParallel backend to use for parallelization

**Value**

will return a dataframe containing: - 'sequence', 'search_id', 'reads', 'raw_match', 'mean_qualities', 'indexes', 'id', 'type', 'strand', 'result', 'extra', 'match_ref_seq', 'n_reads'

---

**Description**

`estimate_coverage` estimate the average coverage by comparing number of bases from reads to genome size

**Usage**

`estimate_coverage(read_lengths, genome_size)`

**Arguments**

- **read_lengths**  
  the lengths of the reads

- **genome_size**  
  the genome size

**Value**

will return an estimated average coverage
extend_length

Description

extend_length extend the search sequence such that there will always be (prev) bases before the SNPs and (after) bases after the SNPs.

Usage

extend_length(
    overlaps,
    position_reference,
    genome_position,
    prev,
    after,
    ori_string_start,
    ori_string_end,
    ori.snp_pos,
    genome_max
)

Arguments

overlaps Overlapping
position_reference the mapping of position in SNP matrix to reference genome
genome_position the position of the SNP in the reference genome
prev number of bases before the SNP included in the search string
after number of bases after the SNP included in the search string
ori_string_start original starting point of search string
ori_string_end original ending point of the search string
ori.snp_pos original SNP position in search string
genome_max length of the reference genome

Value

da list containing the new ‘string_start’, ‘string_end’, ‘snp_pos’, ‘snps_in_string’.
**Description**

`find_optimised_snps` is used to find optimised SNPs set.

**Usage**

```r
find_optimised_snps(
  seqc,
  metric = "simpson",
  goi = c(),
  accept_multiallelic = TRUE,
  number_of_result = 1,
  max_depth = 1,
  included_positions = c(),
  excluded_positions = c(),
  search_from = NULL,
  iterate_included = FALSE,
  completely_unique = FALSE,
  bp = SerialParam(),
  ...
)
```

**Arguments**

- `seqc`: list of sequences, either passed directly from `processallele` or `read_fasta` or equivalence
- `metric`: either 'simpson' or 'percent'
- `goi`: group of interest, if criterion is percent, must be specified, ignored otherwise
- `accept_multiallelic`: whether include positions with > 1 state in goi
- `number_of_result`: number of results to return, 0 will be coerced to 1
- `max_depth`: maximum depth to go before terminating, 0 means it will only calculate the metric for included position
- `included_positions`: included positions
- `excluded_positions`: excluded positions
- `search_from`: search only from these positions, i.e., any positions not in here are excluded, default to NULL
- `iterate_included`: whether to calculate index at each level of the included SNPs
**full_merge**

`completely_unique`  
whether to identify completely unique SNPs set, default to FALSE, only the 1st SNP must be different

`bp`  
BiocParallel backend. Rule of thumbs: use MulticoreParam(workers = ncpus - 2)

`...`  
other parameters as needed

**Value**

Will return the resolution-optimised SNPs set, based on the metric.

---

**full_merge**

**full_merge**

**Description**

`full_merge` is used to merge 2 fasta, where a position exist only in 1 of the fasta, the fasta without allele in that positions are given reference genome’s allele at that position. **Doesn’t work for large dataset, hence the need for full_merge_1**

**Usage**

```r
full_merge(
  fasta_1,
  fasta_2,
  meta_1,
  meta_2,
  ref,
  bp = BiocParallel::MulticoreParam(),
  ...
)
```

**Arguments**

- `fasta_1`: fasta read into memory to join
- `fasta_2`: fasta read into memory to join
- `meta_1`: meta file for ‘fasta_1’ denoting all positions of SNPs and position in reference genome
- `meta_2`: meta file for ‘fasta_2’ denoting all positions of SNPs and position in reference genome
- `ref`: name of the reference genome (needs to be in both fasta files)
- `bp`: the BiocParallel backend
- `...`: all other arguments

**Value**

merged fasta and meta
**Description**

`full_merge_1` is used to merge 2 fasta, where a position exist only in 1 of the fasta, the fasta without allele in that positions are given reference genome’s allele at that position.

**Usage**

```r
full_merge_1(
    fasta_1,
    fasta_2,
    meta_1,
    meta_2,
    ref,
    bp = BiocParallel::SerialParam(),
    ...
)
```

**Arguments**

- `fasta_1` : fasta read into memory to join
- `fasta_2` : fasta read into memory to join
- `meta_1` : meta file for ‘fasta_1’ denoting all positions of SNPs and position in reference genome
- `meta_2` : meta file for ‘fasta_2’ denoting all positions of SNPs and position in reference genome
- `ref` : name of the reference genome (needs to be in both fasta files)
- `bp` : the BiocParallel backend
- `...` : all other arguments

**Value**

merged fasta and meta
generate_kmers

Description

generate_kmers generate the kmer sequences of the given length

Usage

generate_kmers(final_string, k)

Arguments

final_string the string to generate kmers
k the length of the kmer

Value

a vector of kmers

generate_kmer_search_string

Description

generate_kmer_search_string generate the search strings to detect genes’ presence

Usage

generate_kmer_search_string(
    gene_seq,
    k,
    id_prefix = NULL,
    bp = MulticoreParam()
)

Arguments

gene_seq sequences to generate k_mers from
k kmer length
id_prefix prefix for the gene id
bp BiocParallel backend to use

Value

a dataframe containing the search strings
generate_pattern

description

generate_pattern is used to generate pattern for calculation.

usage

generate_pattern(seqc, ordered_index = c(), append_to = list())

arguments

seqc list of sequences
ordered_index list of indexes for the pattern in the order
append_to existing patterns to append to

value

will return concatenated list of string for searching.

generate_snp_search_string

description

generate_snp_search_string identify the SNPs that will overlap the search strings generated from the targeted SNPs

usage

generate_snp_search_string(
    selected_snps,
    position_reference,
    ref_seq,
    snp_matrix,
    prev,
    after,
    position_type = "fasta",
    extend_length = TRUE,
    fasta_name_as_result = TRUE,
    bp = MulticoreParam()
)
Arguments

- `selected_snps` list of targeted SNPs
- `position_reference` the mapping between reference genome positions and orthologous SNP matrix positions
- `ref_seq` the reference genome sequence
- `snp_matrix` the orthologous SNP matrix
- `prev` number of characters before the SNP
- `after` number of characters after the SNP
- `position_type` type of SNPs input, "fasta" (orthologous SNP matrix based) or "genome" (reference genome based); Default to "fasta"
- `extend_length` whether to extend the search string before and after the SNP and ignore overlapping SNPs
- `fasta_name_as_result` Whether the result should use the fasta matching sequence name or the fasta position and allele, default to using fasta sequence name (TRUE)
- `bp` BiocParallel backend to use

Value

a dataframe containing the search strings

Description

`get_all_process_methods` is used to get the metrics function and required parameters. Additional metric may be set by assigning it to `MinSNPs_process_methods` variable.

Usage

`get_all_process_methods(process_name = "")`

Arguments

- `process_name` name of the metric, "" to return all, ‘SNP’ or ‘KMER’ are provided as default.

Value

a list, including the function to process the search sequence result
Description

get_metric_fun is used to get the metrics function and required parameters. Additional metric may set by assigning to 'MinSNPs_metrics' variable.

Usage

get_metric_fun(metric_name = '')

Arguments

metric_name name of the metric, by default percent/simpson

Value

a list, including the function to calculate the metric based on a position ('calc'), and function to check for additional parameters the function need ('args')

Description

get_snps_set extract the SNP sets from the output of 'find_optimised_snps'.

Usage

get_snps_set(results, as = "data.frame")

Arguments

results output from 'find_optimised_snps'

as output format, either 'data.frame' or 'list'.

Value

will return either 1. a dataframe containing SNPs_set (SNP position separated by ",") and score 2. a list containing SNPs_set (SNP position as numeric vector) and score (attr of the list)
**identify_overlaps**

Description

*identify_overlaps* identify the overlapping SNPs in the sequences

Usage

*identify_overlaps*(position_reference, genome_position, prev, after)

Arguments

- **position_reference**: the mapping of position in SNP matrix to reference genome
- **genome_position**: the position of the SNP in the reference genome
- **prev**: number of bases before the SNP included in the search string
- **after**: number of bases after the SNP included in the search string

Value

a list containing 2 dataframes listing the bystander SNPs in the flanking sequence before and after the SNPs

**infer_from_combined**

Description

*infer_from_combined* infers the results (presence/absence of genes & CC) from the combined result

Usage

*infer_from_combined*(combined_result, search_table, genome_size, ...)

Arguments

- **combined_result**: the combined result from *combine_fastq_search_result* or equivalent, with a list containing:
  - result: a dataframe containing the following columns: `sequence`, `search_id`, `reads`, `raw_match`, `mean_qualities`, `indexes`, `id`, `type`, `strand`, `result`, `extra`, `match_ref_seq`, `n_reads` - read_length: `reads_id`, `reads_length`
iterate_merge

search_table  a dataframe with the following columns: - "id", "type", "sequence", "strand", "result", "extra", "match_ref_seq"

genome_size  estimated genome size for coverage calculation

...  additional arguments to pass to the process methods

Value

a dataframe containing the following columns: - type, rank, result, reads_count, proportion_matched, pass_filter

Description

iterate_merge is used to combine > 2 fastas iteratively.

Usage

iterate_merge(
  fastas,
  metas,
  ref,
  method = "full",
  bp = BiocParallel::SerialParam(),
  ...
)

Arguments

  fastas  list of fastas read into memory to join
  metas  list of metas read into memory to join
  ref  name of the reference genome (needs to be in both fasta files)
  method  how to join the 2 fasta, currently supported methods are: inner, full
  bp  the BiocParallel backend
  ...  all other arguments

Value

Will return a list containing a merged FASTA and a meta.
iterate_through

Description
iterate_through is used to calculate the metric at each position

Usage
iterate_through(metric, seqc, bp = MulticoreParam(), ...)

Arguments

metric either 'simpson' or 'percent'
seqc list of sequences, either passed directly from process_allele or read_fasta or equivalence
bp BiocParallel backend. Rule of thumbs: use MulticoreParam(workers = ncpus - 2)
...
other parameters as needed

Value
return a dataframe containing the position and result.

match_count

Description
match_count return the number of matches of the target string in the given sequence

Usage
match_count(target, search_from)

Arguments
target the search target
search_from the sequence to search from

Value
number of matches
merge_fasta

Description

merge_fasta is used to combine 2 fasta.

Usage

merge_fasta(
    fasta_1,
    fasta_2,
    meta_1,
    meta_2,
    ref,
    method = "full",
    bp = BiocParallel::SerialParam(),
    ...
)

Arguments

fasta_1  fasta read into memory to join
fasta_2  fasta read into memory to join
meta_1   meta file for ‘fasta_1’ denoting all positions of SNPs and position in reference genome
meta_2   meta file for ‘fasta_2’ denoting all positions of SNPs and position in reference genome
ref      name of the reference genome (needs to be in both fasta files)
method   how to join the 2 fasta, currently supported methods are: inner, full
bp       the BiocParallel backend
...      all other arguments

Value

Will return a list containing a merged FASTA and a meta.
output_result

Description

output_result is used to present the result and save the result as tsv.

Usage

output_result(result, view = "", ...)

Arguments

result is the result from find_optimised_snps
view how to present the output, "csv" or "tsv" will be saved as a file. Otherwise, printed to console.
... if view is "tsv" or "csv", file name can be passed, e.g., file_name = "result.tsv", otherwise, file is saved as <timestamp>.tsv.

Value

NULL, result either printed or saved as tsv.

output_to_files

Description

output_to_files is write the result to files.

Usage

output_to_files(merged_result, filename = "merged")

Arguments

merged_result a list containing the merged fasta and meta.
filename filename to write to, will output <filename>.fasta and <filename>.csv.

Value

NULL, files written to filesystem
**process_allele**

---

**process_allele**

**Description**

`process_allele` is used to return the processed allelic profiles, by removing the allele profile with duplicate name and length different from most. 1st allele profile with the duplicated name is returned, the longer length is taken as normal should there be 2 modes.

**Usage**

```r
process_allele(
  seqc,
  bp = BiocParallel::SerialParam(),
  check_length = TRUE,
  check_bases = TRUE,
  dash_ignore = TRUE,
  accepted_char = c("A", "C", "T", "G"),
  ignore_case = TRUE,
  remove_invariant = FALSE,
  biallelic_only = FALSE
)
```

**Arguments**

- **seqc**: a list containing list of nucleotides. To keep it simple, use provided `read_fasta` to import the fasta file.
- **bp**: is the biocparallel backend, default to `serialParam`, most likely sufficient in most scenario.
- **check_length**: Check the length of each sample in the matrix, default to `TRUE`.
- **check_bases**: Check the bases of each sample in the matrix, default to `TRUE`.
- **dash_ignore**: whether to treat `'-` as another type.
- **accepted_char**: character to accept, default to `c("A", "C", "T", "G")`.
- **ignore_case**: whether to be case insensitive, default to `TRUE`.
- **remove_invariant**: whether to remove invariant positions, default to `FALSE`.
- **biallelic_only**: whether to remove positions with more than 2 alleles, default to `FALSE`.

**Value**

Will return the processed allelic profiles.
**process_kmer_result**

Description

process_kmer_result processes the KMER result from infer_from_combined

Usage

process_kmer_result(partial_result, search_table, min_match_per_read = 1, ...)

Arguments

- partial_result: the result from infer_from_combined with only KMER
- search_table: a dataframe with the following columns: "id","type","sequence","strand","result","extra","match_ref_seq"
- min_match_per_read: the minimum number of kmer matches in a read, discarding reads with less than this number
- ...: ignored

Value

a dataframe containing the following columns: type, rank, result, reads_count, proportion_matched, pass_filter, proportion_scheme_found, details

**process_result_file**

Description

process_result_file extract the SNP sets from the saved output file.

Usage

process_result_file(result_filepath)

Arguments

- result_filepath: is the path of the saved output file.

Value

will return a list containing SNPs_set (SNP position as numeric vector).
**process_snp_result** process_snp_result

Description

*process_snp_result* processes the SNP result from *infer_from_combined*

Usage

```r
process_snp_result(
    partial_result,
    search_table,
    count_measure = "n_reads",
    ...
)
```

Arguments

- **partial_result**: the result from *infer_from_combined* with only SNP
- **search_table**: a dataframe with the following columns: "id","type","sequence","strand","result","extra","match_ref_seq"
- **count_measure**: the column name of the count measure to use for removing the conflicts
- **...**: ignored

Value

A list containing:
- **result**: a dataframe containing the following columns: type, rank, result, reads_count, proportion_matched, pass_filter, proportion_scheme_found, details
- **snps_found**: a vector containing the SNPs ID that have been identified without conflict
- **proportion_snps_found**: the proportion of SNPs found without conflict

**read_fasta**

Description

*read_fasta* is used to read fasta file, implementation similar to seqinr, but much simpler and allow for spaces in sample name.

Usage

```r
read_fasta(file, force_to_upper = TRUE, bp = SerialParam())
```
read_sequences_from_fastq

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>file path</td>
</tr>
<tr>
<td>force_to_upper</td>
<td>whether to transform sequences to upper case, default to TRUE</td>
</tr>
<tr>
<td>bp</td>
<td>is the biocparallel backend, default to serialParam, most likely sufficient in most scenario</td>
</tr>
</tbody>
</table>

Value

Will return list of named character vectors.

Description

read_sequences_from_fastq get the sequences from a fastq file, it completely ignores the quality scores

Usage

read_sequences_from_fastq(
  fastq_file,
  force_to_upper = TRUE,
  skip_n_reads = 0,
  max_n_reads = -1,
  output_quality = TRUE,
  quality_offset = 33,
  bp = MulticoreParam()
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fastq_file</td>
<td>location of the fastq file</td>
</tr>
<tr>
<td>force_to_upper</td>
<td>whether to transform sequences to upper case, default to TRUE</td>
</tr>
<tr>
<td>skip_n_reads</td>
<td>number of reads to skip, default to 0</td>
</tr>
<tr>
<td>max_n_reads</td>
<td>maximum number of reads to read, default to -1 (all)</td>
</tr>
<tr>
<td>output_quality</td>
<td>whether to output the quality scores, default to TRUE</td>
</tr>
<tr>
<td>quality_offset</td>
<td>the quality offset to use, default to 33</td>
</tr>
<tr>
<td>bp</td>
<td>BiocParallel backend to use for parallelization</td>
</tr>
</tbody>
</table>

Value

will return a list of sequences, with qualities as attribute
remove_snp_conflict  remove_snp_conflict

Description

remove_snp_conflict removes the reads with SNPs conflicts from the result

Usage

remove_snp_conflict(result, count_measure = "n_reads")

Arguments

result  the result from infer_from_combined

count_measure  the column name of the count measure to use for removing the conflicts

Value

a dataframe containing the same columns as the input result with row containing conflicts removed

resolve_IUPAC_missing  resolve_IUPAC_missing

Description

resolve_IUPAC_missing is used to replace the ambiguity codes found in the sequences.

Usage

resolve_IUPAC_missing(
    seqc,
    log_operation = TRUE,
    log_file = "replace.log",
    max_ambiguity = -1,
    replace_method = "random",
    N_is_any_base = FALSE,
    output_progress = TRUE,
    bp = MulticoreParam()
)
**reverse_complement**

**Arguments**

- `seq`: the sequences to be processed
- `log_operation`: whether to log the operation
- `log_file`: log file to write the operations
- `max_ambiguity`: proportion of ambiguity codes to tolerate, -1 = ignore. Default to -1
- `replace_method`: how to substitute the ambiguity codes, current supported methods: `random` and `most_common`, default to "random".
- `N_is_any_base`: whether to treat N as any base or substitute it with one of the alleles found at the position.
- `output_progress`: whether to output progress
- `bp`: the BiocParallel backend

**Value**

Will return the processed sequences.

---

**reverse_complement**

**Description**

`reverse_complement` returns the reverse complement of the given sequence

**Usage**

`reverse_complement(seq)`

**Arguments**

- `seq`: the sequence to reverse complement

**Value**

reverse complemented sequence
search_from_fastq_reads

Description

search_from_fastq_reads identify the matches from a list of search strings

Usage

search_from_fastq_reads(  
  fastq_file,  
  search_tables,  
  skip_n Reads = 0,  
  progress = TRUE,  
  max_n_reads = -1,  
  quality_offset = 33,  
  output_temp_result = TRUE,  
  temp_result_folder = "/temp_results",  
  simplify_id = TRUE,  
  output_read_length = TRUE,  
  bp = MulticoreParam()
)

Arguments

fastq_file     fastq file containing the runs to search from
search_tables  a dataframe with the following columns: - ["id"], "type", ["sequence"], "strand", "result", "extra", "match_ref_seq"
skip_n_reads   number of reads to skip, default is 0
progress       whether to show the progress bar
max_n_reads    maximum number of reads to read, default to -1 (all)
quality_offset the quality offset to use, default to 33
output_temp_result whether to output the temporary results
temp_result_folder directory to output the temporary results
simplify_id    simplify and shorten the read id to the first part
output_read_length whether to output the read length, NULL - do not output; csv - output to csv file; data - output to result
bp             BiocParallel backend to use for parallelization

Value

will return a list of dataframe containing: - 'search_id', 'sequence', 'reads', 'raw_match', 'mean_qualities', 'indexes'.

sequence_reads_match_count

Description

sequence_reads_match_count look for the search sequences in reads and return the matches indexes and mean qualities

Usage

sequence_reads_match_count(search_sequence, reads, qualities)

Arguments

search_sequence
  the search sequence to look for where '.' stands for any character.
reads
  the sequences reads to search for.
qualities
  the qualities of each bases in the reads.

Value

will return a list containing for each read: - count, mean_quality, indexes

view_percent

Description

view_percent is used to present the result of selected SNPs set based on Simpson’s Index.

Usage

view_percent(result, ...)

Arguments

result
  is the result from find_optimised_snps
...
  other optional parameters

Value

formatted result list to be saved or presented.
Description
view_simpson is used to present the result of selected SNPs set based on Simpson's Index.

Usage
view_simpson(result, ...)

Arguments
result is the result from find_optimised_snps
...
other optional parameters

Value
formatted result list to be saved or presented.

Description
write_fasta is used to write the named character vectors to fasta file.

Usage
write_fasta(seqc, filename)

Arguments
seqc a list containing list of nucleotides. To keep it simple, use provided read_fasta to import the fasta file.
filename filename of the output file

Value
will write the alignments to file
Index

cal_fn, 4
cal_fp, 5
cal_met_snp, 5
calculate_percent, 3
calculate_simpson, 3
calculate_variant_within_group, 4
check_fasta_meta_mapping, 6
check_multistate, 6
check_percent, 7
combine_fastq_search_result, 7
combine_search_string_result, 8
combine_search_string_result_from_files, 8
combine_search_string_result_from_list, 9
estimate_coverage, 10
extend_length, 11
find_optimised_snps, 12
full_merge, 13
full_merge_1, 14
generate_kmer_search_string, 15
generate_kmers, 15
generate_pattern, 16
generate_snp_search_string, 16
get_all_process_methods, 17
get_metric_fun, 18
get_snps_set, 18
identify_overlaps, 19
infer_from_combined, 19
iterate_merge, 20
iterate_through, 21
match_count, 21
merge_fasta, 22
output_result, 23
output_to_files, 23
process_allele, 24
process_kmer_result, 25
process_result_file, 25
process_snp_result, 26
read_fasta, 26
read_sequences_from_fastq, 27
remove_snp_conflict, 28
resolve_IUPAC_missing, 28
reverse_complement, 29
search_from_fastq_reads, 30
sequence_reads_match_count, 31
view_percent, 31
view_simpson, 32
write_fasta, 32