Package 'minSNPs'

May 8, 2026

Title Resolution-Optimised SNPs Searcher

Version 0.2.0

Description This is a R implementation of ``Minimum SNPs" software as described in ``Price E.P., Inman-Bamber, J., Thiruvenkataswamy, V., Huygens, F and Giffard, P.M." (2007) <doi:10.1186/1471-2105-8-278> ``Computer-aided identification of polymorphism sets diagnostic for groups of bacterial and viral genetic variants."

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] (ORCID: <https://orcid.org/0000-0002-2310-3403>), Peter Shaw [aut, ctb] (ORCID: <https://orcid.org/0000-0002-3187-8938>), Phil Giffard [aut, ctb] (ORCID: <https://orcid.org/0000-0002-3030-9127>)

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

Repository CRAN

Date/Publication 2024-03-05 14:30:02 UTC

Contents

`binomial_naive_bayes` `binomial_naive_bayes`

Description

`binomial_naive_bayes` is an implementation of the binomial naive bayes algorithm. modified from bernoulli_naive_bayes function in the naivebayes package

Usage

```
binomial_naive_bayes(x, y, prior = NULL, laplace = 1, ...)
```

Arguments

`x`	a matrix with numeric: 0,1, up to binomial_n columns
`y`	a factor or character or logical vector
`prior`	a vector of prior probabilities
`laplace`	a numeric value for Laplace smoothing
`...`	additional arguments

Value

return a binomial_naive_bayes object

calculate_mcc calculate_mcc

Description

calculate_mcc is used to calculate the MCC score given the SNP profile.

Usage

```
calculate_mcc(pattern, goi, MUST_HAVE_TARGET = TRUE)
```

Arguments

pattern	the SNP profile for each samples
goi	the samples belonging to the group of interest
MUST_HAVE_TARGET	whether to force the profile to have at least 1 target profile (the profile containing the most goi)

Value

the MCC score

calculate_mcc_multi calculate_mcc_multi

Description

calculate_mcc_multi Calculate the multi-class MCC score for the SNPs. It assigns each SNP profile to a class, based on the majority of the samples having the profile, targets with the less samples are prioritised first, breaking ties by alphabetical order.

Usage

```
calculate_mcc_multi(pattern, meta, target = "target", priority = NULL)
```

Arguments

pattern	the SNP profile for each samples
meta	A data.table containing the meta data
target	the column name of the target in the meta data, default to target
priority	A data.table of the targets and priority, either supplied by user, or by default generated by generate_prioritisation.

Value

multiclass-MCC score

`calculate_percent` `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` `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_simpson_by_group` `calculate_simpson_by_group`

Description

`calculate_simpson_by_group` 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_by_group(pattern, meta, target)
```

Arguments

`pattern`	list of sequences' pattern (profile)
`meta`	the metadata
`target`	the target column name

Value

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

`calculate_state` `calculate_state`

Description

`calculate_state` calculate the number of states given the SNP(s)

Usage

```
calculate_state(pattern)
```

Arguments

`pattern`	list of sequences' pattern (profile)

Value

number of states

`calculate_variant_within_group` `identify_group_variant_breakdown`

Description

`calculate_variant_within_group` is used to identify proportion of different samples having the same profile.

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` `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 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 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
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_meta_target check_meta_target

Description

check_meta_target is used to check if parameters needed by calculate_mcc_multi and simpson_by_group are all present.

Usage

```
check_meta_target(list_of_parameters)
```

Arguments

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

Value

TRUE if the parameters exists, else FALSE

check_multistate 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'

check_percent check_percent

Description

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

Usage

```
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

coef.binomial_naive_bayes coef.binomial_naive_bayes

Description

coef.binomial_naive_bayes is an implementation of the coef method for the binomial naive bayes algorithm. modified from bernoulli_naive_bayes function in the naivebayes package

Usage

```
## S3 method for class 'binomial_naive_bayes'
coef(object, ...)
```

Arguments

object	a binomial_naive_bayes object
...	additional arguments

Value

return a data frame of coefficients

combine_fastq_search_result combine_fastq_search_result

Description

combine_fastq_search_result combines the search results from search_from_fastq_reads

Usage

```
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_se
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`
`combine_search_string_result`

Description

`combine_search_string_result` combines the search results from `search_from_fastq_reads`

Usage

```
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: - "id","type","sequence","strand","result","extra","match_ref_se
`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_from_files`
`combine_search_string_result_from_files`

Description

`combine_search_string_result_from_files` `combine_search_string_result` combines the search results from temp file generated from `search_from_fastq_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_se
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

will return a fastq_search_result object containing read_lengths and 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_list

combine_search_string_result_from_list

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()
)
```

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_se
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‘

---

estimate_coverage estimate_coverage

Description

estimate_coverage 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 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	Overlappings
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

a list containing the new ‘string_start‘, ‘string_end‘, ‘snp_pos‘, ‘snps_in_string‘.

`find_optimised_snps` `find_optimised_snps`

Description

`find_optimised_snps` is used to find optimised SNPs set.

Usage

```
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 `process_allele` or `read_fasta` or equivalence
`metric`	either 'simpson' or 'percent'
`goi`	group of interest, if creteria 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

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

```
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

full_merge_1 full_merge_1

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

```
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 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
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 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_prioritisation generate_prioritisation

Description

generate_prioritisation create a vector of the targets in order of priority. Targets with the less samples are prioritised first, breaking ties by alphabetical order.

Usage

```
generate_prioritisation(meta)
```

Arguments

meta	A data.table containing the meta data, expect the

Value

A data.table of the targets in order of priority

generate_snp_search_string
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

get_all_process_methods get_all_process_methods

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

get_binomial_tables get_binomial_tables

Description

get_binomial_tables is an internal function that returns a table of probability for binomial naive bayes. modified from bernoulli_naive_bayes function in the naivebayes package

Usage

```
get_binomial_tables(prob1)
```

Arguments

prob1	a matrix of probabilities

Value

return table of probability for binomial naive bayes

get_metric_fun get_metric_fun

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')

get_snps_set get_snps_set

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 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 infer_from_combined

Description

infer_from_combined infers the results (presence/absense 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'

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

---

iterate_merge iterate_merge

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 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.

map_profile_to_target map_profile_to_target

Description

map_profile_to_target creates a mapping of the profile to the target, breaking the ties by the priority.

Usage

```
map_profile_to_target(meta, patterns, priority, sensitive_to_1 = FALSE)
```

Arguments

meta	A data.table containing the meta data
patterns	A list of the patterns from generate_pattern
priority	A data.table of the targets and priority, either generated by generate_prioritisation or supplied by user
sensitive_to_1	whether to be completely sensitive to the first target (percent default), set to TRUE for percent

Value

A vector of the targets in order of priority

`match_count` `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

`mcc_calculation` `mcc_calculation`

Description

`mcc_calculation` calculate the MCC score given the truth and predicted target.

Usage

```
mcc_calculation(
  result_with_truth,
  is_multi = TRUE,
  return_all_intermediate = FALSE
)
```

Arguments

`result_with_truth`	the dataframe containing the truth and predicted target
`is_multi`	Whether to use MCC-multi or MCC
`return_all_intermediate`	whether to return all intermediate values, only possible for binary class

Value

Will return the mcc score

`merge_fasta` `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 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 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

parse_group_mcc parse_group_mcc

Description

parse_group_mcc is used to group the sample according to SNPs profile and present in a table format

Usage

```
parse_group_mcc(pattern, goi, MUST_HAVE_TARGET = TRUE)
```

Arguments

pattern	the SNP profile for each samples
goi	the samples belonging to the group of interest
MUST_HAVE_TARGET	whether to force the profile to have at least 1 target profile (the profile containing the most goi)

Value

the parsed group views

parse_group_mcc_multi parse_group_mcc_multi

Description

parse_group_mcc_multi is used to put samples according to SNP profile, and put them into a table format.

Usage

```
parse_group_mcc_multi(result, as_string = TRUE)
```

Arguments

result	result from find_optimised_snps
as_string	whether to return the result as string or data.frame

Value

Will return the grouped samples.

predict.binomial_naive_bayes predict.binomial_naive_bayes

Description

predict.binomial_naive_bayes is an implementation of the predict method for the binomial naive bayes algorithm. modified from bernoulli_naive_bayes function in the naivebayes package

Usage

```
## S3 method for class 'binomial_naive_bayes'
predict(object, newdata = NULL, type = c("class", "prob"), ...)
```

Arguments

object	a binomial_naive_bayes object
newdata	a matrix with numeric: 0,1,up to binomial_n columns
type	a character string specifying the type of output: "class" or "prob"
...	additional arguments

Value

return a factor or matrix of class probabilities

predict_balk predict_balk

Description

predict_balk is a function that predicts the class of new data using a binomial naive bayes classifier

Usage

```
predict_balk(object, newdata = NULL, snp_id = NULL, type = "prob")
```

Arguments

object	The classifier object
newdata	A list of sequences
snp_id	A vector of SNP IDs
type	The type of prediction, either "prob" or "class"

Value

A vector of either the class or the probability of the class

print.binomial_naive_bayes print.binomial_naive_bayes

Description

print.binomial_naive_bayes is an implementation of the print method for the binomial naive bayes algorithm. modified from bernoulli_naive_bayes function in the naivebayes package

Usage

```
## S3 method for class 'binomial_naive_bayes'
print(x, ...)
```

Arguments

x	a binomial_naive_bayes object
...	additional arguments

Value

return a binomial_naive_bayes object

process_allele process_allele

Description

process_allele is used to returned 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

```
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 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_se
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 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

```
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_se
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

---

`profile_to_group_result` `profile_to_group_result`

Description

`profile_to_group_result` given profile target, return the result

Usage

```
profile_to_group_result(patterns, profile_target)
```

Arguments

`patterns`	the SNP profile for each samples
`profile_target`	the profile target - should be from samples previously seen, generate with `map_profile_to_target`

Value

Will return the result, given the SNP profile.

`read_fasta` `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

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

Arguments

`file`	file path
`force_to_upper`	whether to transform sequences to upper case, default to TRUE
`bp`	is the biocparallel backend, default to serialParam, most likely sufficient in most scenario

Value

Will return list of named character vectors.

`read_sequences_from_fastq` `read_sequences_from_fastq`

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

`fastq_file`	location of the fastq file
`force_to_upper`	whether to transform sequences to upper case, default to TRUE
`skip_n_reads`	number of reads to skip, default to 0
`max_n_reads`	maximum number of reads to read, default to -1 (all)
`output_quality`	whether to output the quality scores, default to TRUE
`quality_offset`	the quality offset to use, default to 33
`bp`	BiocParallel backend to use for parallelization

Value

will return a list of sequences, with qualities as attribute

remove_snp_conflict remove_snp_conflic

Description

remove_snp_conflic 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()
)
```

Arguments

seqc	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 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

scramble_sequence scramble_sequence

Description

scramble_sequence scramble the orthologous matrix based on a seed

Usage

```
scramble_sequence(seqc, seed)
```

Arguments

seqc	the sequence to scramble
seed	the seed to use for scrambling

Value

a named list, containing the scambled sequence and the new positions

search_from_fastq_reads 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
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‘.

---

search_from_reads search_from_reads

Description

search_from_reads identify the matches from a list of search strings

Usage

```
search_from_reads(
  all_reads,
  search_tables,
  progress = TRUE,
  ID = "S1",
  all_qualities = NULL,
  output_temp_result = TRUE,
  temp_result_folder = "./temp_results",
  output_read_length = TRUE,
  bp = MulticoreParam()
)
```

Arguments

all_reads	The reads containing the runs to search from
search_tables	a dataframe with the following columns: - ["id"],"type",["sequence"],"strand","result","extra","match_ref
progress	whether to show the progress bar
ID	the ID to use, default to S1
all_qualities	quality data, default to NULL
output_temp_result	whether to output the temporary results
temp_result_folder	directory to output the temporary results
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
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

summarise_result summarise_result

Description

summarise_result calculate the MCC score given the SNP sets, training, validation and metadata(s).

Usage

```
summarise_result(
  snp_sets,
  training_seqs,
  validation_seqs,
  training_metadata,
  validation_metadata,
  priority,
  is_multi = TRUE,
  return_all_intermediate = FALSE,
  is_percent = FALSE
)
```

Arguments

snp_sets	the dataframe containing the truth and predicted target
training_seqs	the training sequences
validation_seqs	the validation sequences
training_metadata	the training metadata
validation_metadata	the validation metadata
priority	the priority of the target, generated by generate_prioritisation
is_multi	Whether to use MCC-multi or MCC
return_all_intermediate	whether to return all intermediate values, only possible for binary class
is_percent	whether to return result by considering all the profiles having a GOI as target profile

Value

Will return the summarised result

summary.binomial_naive_bayes summary.binomial_naive_bayes

Description

summary.binomial_naive_bayes is an implementation of the summary method for the binomial naive bayes algorithm. modified from bernoulli_naive_bayes function in the naivebayes package

Usage

```
## S3 method for class 'binomial_naive_bayes'
summary(object, ...)
```

Arguments

object	a binomial_naive_bayes object
...	additional arguments

Value

return a summary of the binomial_naive_bayes object

train_balk train_balk

Description

train_balk is a function that trains a binomial naive bayes classifier for sequence data

Usage

```
train_balk(
  seqc,
  snps_pos,
  meta,
  binomial_n = 1,
  laplace = 1,
  snp_id = NULL,
  prior = NULL,
  fit_prior = FALSE
)
```

Arguments

seqc	A list of sequences
snps_pos	A vector of SNP positions
meta	A data frame containing the metadata, require isolate and target columns
binomial_n	The number of classes for the binomial naive bayes, default to 1 - bernoulli, 2 - binomial (support heterozygous SNPs)
laplace	The Laplace smoothing parameter
snp_id	A vector of SNP IDs, if not provided, it will be inferred from the SNP positions
prior	The prior probabilities of the classes
fit_prior	Whether to learn class prior probabilities

Value

A list containing the classifier and the transformation levels

---

transform_snp transform_snp

Description

transform_snp is a function that transforms a SNP into a matrix for binomial naive bayes.

Usage

```
transform_snp(pat, binomial_n, levels = c(), get = c("levels", "transformed"))
```

Arguments

pat	A string of a SNP
binomial_n	The number of classes for the binomial naive bayes, default to 1 - bernoulli, 2 - binomial (support heterozygous SNPs)
levels	Existing transformation levels, if not provided, it will be inferred from the SNP
get	What to return, either "levels" or "transformed", or both

Value

A vector of either the transformation levels or the transformed SNP or a list containing both

translate_position translate_position

Description

translate_position translate the scambled position in the alignment to the original position or vice versa

Usage

```
translate_position(position, positions_table, to = "original")
```

Arguments

position	the position to translate
positions_table	the table containing the original and scrambled positions
to	the direction to translate, either "original" or "scrambled"

Value

the translated position

---

view_mcc view_mcc

Description

view_mcc is used to present the result of selected SNPs set based on the MCC score

Usage

```
view_mcc(result, ...)
```

Arguments

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

Value

formatted result list to be saved or presented.

view_mcc_multi view_mcc_multi

Description

view_mcc_multi is used to present the result of selected SNPs set based on the multi-MCC score

Usage

```
view_mcc_multi(result, ...)
```

Arguments

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

Value

formatted result list to be saved or presented.

view_percent 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.

view_simpson view_simpson

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.

write_fasta write_fasta

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