Package 'moire'

September 26, 2026

Title Multiplicity of Infection and Allele Frequency Recovery from Noisy Polyallelic Genetics Data

Version 3.7.0

Description A Markov Chain Monte Carlo (MCMC) based approach to Bayesian estimation of individual level multiplicity of infection, within host relatedness, and population allele frequencies from polyallelic genetic data. Implements the model described in Murphy and Greenhouse (2024) <doi:10.1093/bioinformatics/btae619>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

LazyDataCompression bzip2

SystemRequirements C++17, GNU make

LinkingTo Rcpp, RcppProgress, RcppParallel, BH

Imports Rcpp, RcppProgress, RcppParallel, dplyr, tidyr, stats, parallel, purrr, rlang, ggplot2

URL https://github.com/EPPIcenter/moire, https://eppicenter.github.io/moire/, https://eppicenter.ucsf.edu/resources

BugReports https://github.com/EPPIcenter/moire/issues

RoxygenNote 7.3.3

Suggests knitr, rmarkdown, forcats, testthat (>= 3.0.0), parallelly

VignetteBuilder knitr

Depends R (>= 4.1.0)

Config/testthat/edition 3

NeedsCompilation yes

Author Maxwell Murphy [aut, cre] (ORCID: <https://orcid.org/0000-0003-0332-4388>), Bryan Greenhouse [aut, ths] (ORCID: <https://orcid.org/0000-0003-0287-9111>)

Maintainer Maxwell Murphy <mm@maxmurphy.dev>

Repository CRAN

Date/Publication 2026-09-26 16:50:02 UTC

Contents

`calculate_he` *Calculate the expected heterozygosity from allele frequencies*

Description

Calculate the expected heterozygosity from allele frequencies

Usage

```
calculate_he(allele_freqs)
```

Arguments

allele_freqs	Simplex of allele frequencies

Value

Numeric scalar, the expected heterozygosity `1 - sum(p^2)`.

Examples

```
calculate_he(c(0.5, 0.3, 0.2))
```

---

calculate_med_allele_freqs

Calculate the geometric median of the posterior distribution of allele frequencies

---

Description

Calculate the geometric median of the posterior distribution of allele frequencies

Usage

```
calculate_med_allele_freqs(mcmc_results, merge_chains = TRUE)
```

Arguments

mcmc_results	Result of calling run_mcmc()
merge_chains	boolean indicating that all chain results should be merged

Details

Returns the geometric median of the posterior distribution, defined as the point minimizing the L2 distance from each sampled point.

Value

If `merge_chains` is `TRUE`, a named list with one numeric vector of allele frequencies per locus. Otherwise a list with one such list per chain.

Examples

```
med_freqs <- calculate_med_allele_freqs(mcmc_results)
med_freqs[[1]]
```

calculate_naive_allele_frequencies *Calculate naive allele frequencies*

Description

Calculate naive allele frequencies

Usage

```
calculate_naive_allele_frequencies(data)
```

Arguments

data	List of lists of numeric vectors, where each list element is a collection of observations across samples at a single genetic locus

Details

Estimate naive allele frequencies from the empirical distribution of alleles

Value

List with one numeric vector of allele frequencies per locus.

Examples

```
naive_freqs <- calculate_naive_allele_frequencies(simulated_data$data)
naive_freqs[[1]]
```

calculate_naive_coi *Calculate naive COI*

Description

Calculate naive COI

Usage

```
calculate_naive_coi(data)
```

Arguments

data	List of lists of numeric vectors, where each list element is a collection of observations across samples at a single genetic locus.

Details

Estimates the complexity of infection using a naive approach that chooses the highest number of observed alleles.

Value

Numeric vector with one naive COI estimate per sample.

Examples

```
calculate_naive_coi(simulated_data$data)
```

`calculate_naive_coi_offset`
Calculate naive COI offset

Description

Calculate naive COI offset

Usage

```
calculate_naive_coi_offset(data, offset)
```

Arguments

`data`	List of lists of numeric vectors, where each list element is a collection of observations across samples at a single genetic locus.
`offset`	Numeric offset – n'th highest number of observed alleles

Details

Estimates the complexity of infection using a naive approach that chooses the n'th highest number of observed alleles.

Value

Numeric vector with one naive COI estimate per sample.

Examples

```
calculate_naive_coi_offset(simulated_data$data, offset = 2)
```

load_delimited_data *Load delimited data*

Description

Load delimited data

Usage

```
load_delimited_data(data, sep = ";", warn_uninformative = TRUE)
```

Arguments

data	data.frame containing the described data
sep	string used to separate alleles
warn_uninformative	boolean whether or not to print message when removing uninformative loci

Details

Load `data.frame` with a `sample_id` column and the remaining columns are `loci`. Each cell contains a separator delimited string representing the observed alleles at that locus for that sample. Returned data contains vectors `sample_ids` and `loci` that are ordered as the results will be ordered from running the MCMC algorithm.

Value

List structured for `run_mcmc()`; see `load_long_form_data()`.

Examples

```
df <- data.frame(
  sample_id = c("S1", "S2", "S3"),
  L1 = c("A;B", "A", "B"),
  L2 = c("A", "A;C", "C")
)
dat <- load_delimited_data(df, sep = ";")
dat$sample_ids
dat$data[["L2"]]
```

load_long_form_data *Load long form data*

Description

Load long form data

Usage

```
load_long_form_data(df, warn_uninformative = TRUE)
```

Arguments

df	data frame with 3 columns: sample_id, locus, allele. Each row is a single observation of an allele at a particular locus for a given sample.
warn_uninformative	boolean whether or not to print message when removing uninformative loci

Details

Long form data is a data frame with 3 columns: sample_id, locus, allele. Returned data contains vectors sample_ids and loci that are ordered as the results will be ordered from running the MCMC algorithm.

Value

List structured for run_mcmc() with elements sample_ids, data (list of per-locus lists of binary allele vectors), loci, is_missing (loci by samples logical matrix), and uninformative_loci (loci removed for having a single allele).

Examples

```
df <- data.frame(
  sample_id = c("S1", "S1", "S1", "S2", "S2"),
  locus = c("L1", "L1", "L2", "L1", "L2"),
  allele = c("A", "B", "A", "A", "B")
)
dat <- load_long_form_data(df)
dat$loci
dat$data[["L1"]]

# A subset of the bundled Namibia data
ids <- unique(namibia_data$sample_id)[1:20]
dat <- load_long_form_data(namibia_data[namibia_data$sample_id %in% ids, ])
str(dat, max.level = 1)
```

mcmc_results *MCMC results from using the packaged simulated data and calling* `run_mcmc()`

Description

MCMC results from using the packaged simulated data and calling `run_mcmc()`

Usage

```
mcmc_results
```

Format

A list as returned by `run_mcmc()` on simulated_data: a single chain with 1000 burnin and 1000 sampling iterations, using 80 parallel tempering replicas with adaptive temperatures. The `mcmc_demo` vignette shows the call.

namibia_data *Genetic and epidemiological data from Namibia*

Description

A dataset containing the genetic and epidemiological data from Namibia

Usage

```
namibia_data
```

Format

A data frame with 7 columns and 97214 rows:

sample_id Sample ID

HealthFacility Health facility

HealthDistrict Health district

Region Region

Country Country

locus Genetic locus

allele Allele observed

Source

doi:10.7554/eLife.43510.018

plot_chain_swaps *Plot chain swap acceptance rates*

Description

Plot chain swap acceptance rates

Usage

```
plot_chain_swaps(mcmc_results)
```

Arguments

mcmc_results	list of results from run_mcmc

Details

Plot the swap acceptance rates for each chain. The x-axis is the temperature, and the y-axis is the swap acceptance rate. The dashed lines indicate the temperatures used for parallel tempering.

Value

list of ggplot objects, one per chain

Examples

```
plots <- plot_chain_swaps(mcmc_results)
plots[[1]]
```

rdirichlet *Dirichlet distribution*

Description

Dirichlet distribution

Usage

```
rdirichlet(n, alpha)
```

Arguments

n	total number of draws
alpha	vector controlling the concentration of simplex

Details

Implementation of random sampling from a Dirichlet distribution

Value

Numeric matrix with `n` rows and `length(alpha)` columns; each row is a draw from the simplex.

Examples

```
rdirichlet(3, alpha = c(1, 1, 1))
```

---

regional_allele_frequencies *Allele frequencies for different regions*

Description

A list of allele frequencies for different regions, estimated from the pf7k dataset.

Usage

```
regional_allele_frequencies
```

Format

A list of lists, where each list element is a list of allele frequencies for a specific region.

---

run_mcmc *Sample from the target distribution using MCMC*

Description

Sample from the target distribution using MCMC

Usage

```
run_mcmc(
  data,
  is_missing = FALSE,
  allow_relatedness = TRUE,
  thin = 1,
  burnin = 10000,
  samples_per_chain = 1000,
  verbose = TRUE,
  use_message = FALSE,
```

```
  eps_pos_alpha = 1,
  eps_pos_beta = 1,
  eps_neg_alpha = 1,
  eps_neg_beta = 1,
  r_alpha = 1,
  r_beta = 1,
  mean_coi_shape = 0.1,
  mean_coi_scale = 10,
  max_eps_pos = 2,
  max_eps_neg = 2,
  max_coi = 40,
  record_latent_genotypes = FALSE,
  num_chains = 1,
  num_cores = 1,
  pt_chains = 1,
  pt_grad = 1,
  pt_num_threads = 1,
  adapt_temp = TRUE,
  pre_adapt_steps = 25,
  temp_adapt_steps = 25,
  max_initialization_tries = 10000,
  max_runtime = Inf,
  seed = NULL
)
```

Arguments

data	Data to be used in MCMC, as generated by the load_*_data functions
is_missing	Boolean matrix indicating whether the observation should be treated as missing data and ignored. Number of rows equals the number of loci, number of columns equals the number samples. Alternatively, the user may pass in FALSE if no data should be considered missing.
allow_relatedness	Bool indicating whether or not to allow relatedness within host
thin	Positive Integer. How often to sample from mcmc, 1 means do not thin
burnin	Positive Integer. Number of MCMC samples to discard as burnin
samples_per_chain	Positive Integer. Number of samples to take after burnin
verbose	Logical indicating if progress is printed
use_message	Logical indicating if progress is printed using message or print
eps_pos_alpha	Positive Numeric. Alpha parameter in Beta distribution for eps_pos prior
eps_pos_beta	Positive Numeric. Beta parameter in Beta distribution for eps_pos prior
eps_neg_alpha	Positive Numeric. Alpha parameter in Beta distribution for eps_neg prior
eps_neg_beta	Positive Numeric. Beta parameter in Beta distribution for eps_neg prior
r_alpha	Positive Numeric. Alpha parameter in Beta distribution for relatedness prior

r_beta Positive Numeric. Beta parameter in Beta distribution for relatedness prior

mean_coi_shape shape parameter for gamma hyperprior on mean COI

mean_coi_scale scale parameter for gamma hyperprior on mean COI

max_eps_pos Numeric. Maximum allowed value for eps_pos

max_eps_neg Numeric. Maximum allowed value for eps_neg

max_coi Positive Numeric. Maximum allowed complexity of infection

record_latent_genotypes
Logical indicating whether or not to record the latent genotypes at each step of the MCMC. WARNING: This will increase the size of the output object significantly.

num_chains Total number of chains to run, possibly simultaneously

num_cores How many independent chains to run at once, passed to `parallel::mclapply()` as `mc.cores`. `num_cores * pt_num_threads` should not exceed the number of cores available on your system.

pt_chains Total number of chains to run with parallel tempering or a vector containing the temperatures that should be used for parallel tempering.

pt_grad Power to raise parallel tempering chains to. A value of 1 results in evenly distributed temperatures between [0,1], below 1 will bias towards 1 and above 1 will bias towards 0. Only used if pt_chains is a single value (i.e. not a vector).

pt_num_threads Total number of OMP parallel threads to be used to process parallel tempered chains num_cores * pt_num_threads should not exceed the number of cores available on your system.

adapt_temp Logical indicating whether or not to adapt the parallel tempering temperatures. If TRUE, the temperatures will be adapted during the `burnin` period, starting after `pre_adapt_steps` steps. The adaptation will occur every `temp_adapt_steps` steps until burnin is complete. The range of temperatures will remain the same as specified by `pt_chains`.

pre_adapt_steps
Number of steps to take before starting to adapt the parallel tempering temperatures. Only used if `adapt_temp` is TRUE.

temp_adapt_steps
Number of steps to take between temperature adaptation steps. Only used if `adapt_temp` is TRUE.

max_initialization_tries
Number of times to try to initialize the chain before giving up

max_runtime Maximum runtime in minutes. If the MCMC is running for more than this amount of time, the function will stop and return the current state of the MCMC.

seed Integer seed. `NULL` draws from the current R RNG a value that still fits after per-chain offsets. Independent MCMCs are spaced by the number of parallel-tempering replicas. The value used is returned as `$seed` and printed if the run errors; per-chain offsets are `$chain_seeds`. Replay needs the same sampler settings; `num_cores` does not affect the draws.

Value

List with elements:

chains List with one entry per chain. Each holds the sampled `coi`, `allele_freqs`, `eps_pos`, `eps_neg`, and `relatedness`, along with log-likelihood traces, acceptance rates, and parallel tempering diagnostics. Pass the whole result to the `summarize_*()` functions.

args The arguments the sampler was called with, including `data`.

seed Integer seed used for the run.

chain_seeds Integer seed used for each chain.

runtime Wall-clock time of the run as a `difftime`.

Initialization failure

If every Initialization retry yields a non-finite genotyping log-likelihood, `run_mcmc` stops with a classed error (`moire_initialization_failure`) whose message includes rule-based guidance and whose `$diagnostics` field has Failure-locus counts and classification (Consistent failure cause vs Hard starting set). Inspect with `tryCatch(run_mcmc(...), error = function(e) e$diagnostics)`.

Examples

```
sim <- simulate_data(
  mean_coi = 2,
  num_samples = 10,
  epsilon_pos = 0.01,
  epsilon_neg = 0.1,
  locus_freq_alphas = list(rep(1, 4), rep(1, 4), rep(1, 4))
)

# A short run for illustration. Real analyses should use far more burnin
# and samples (see the defaults) and parallel tempering (pt_chains) to
# improve mixing.
res <- run_mcmc(
  sim,
  is_missing = sim$is_missing,
  burnin = 100,
  samples_per_chain = 100,
  verbose = FALSE,
  seed = 1
)
res$seed
summarize_coi(res)
```

---

simulated_data *Simulated genotyping data*

Description

A simulated dataset created using `simulate_data()`

Usage

```
simulated_data
```

Format

A list as returned by `simulate_data()`, with 100 samples at 30 loci (15 with 5 alleles, 15 with 10) and a mean COI of 5: observed `data`, `sample_ids`, `loci`, `is_missing`, and the simulated truth (`allele_freqs`, `sample_cois`, `sample_relatedness`, `true_genotypes`).

---

simulate_allele_frequencies

Simulate allele frequencies

Description

Simulate allele frequencies

Usage

```
simulate_allele_frequencies(alpha, num_loci)
```

Arguments

`alpha`	vector parameter controlling the Dirichlet distribution
`num_loci`	total number of loci to draw

Details

Simulate allele frequency vectors as a draw from a Dirichlet distribution

Value

Numeric matrix with `length(alpha)` rows and `num_loci` columns; each column is an allele frequency vector for one locus.

Examples

```
simulate_allele_frequencies(alpha = c(1, 1, 1, 1), num_loci = 3)
```

`simulate_data` *Simulate data generated according to the assumed model*

Description

Simulate data generated according to the assumed model

Usage

```
simulate_data(
  mean_coi = NULL,
  num_samples,
  epsilon_pos,
  epsilon_neg,
  sample_cois = NULL,
  locus_freq_alphas = NULL,
  allele_freqs = NULL,
  internal_relatedness_alpha = 0,
  internal_relatedness_beta = 1,
  internal_relatedness = NULL,
  missingness = 0
)
```

Arguments

`mean_coi`	Mean multiplicity of infection drawn from a Poisson
`num_samples`	Total number of biological samples to simulate
`epsilon_pos`	False positive rate, expected number of false positives
`epsilon_neg`	False negative rate, expected number of false negatives
`sample_cois`	List of sample COIs to be used instead of simulating
`locus_freq_alphas`	List of alpha vectors to be used to simulate from a Dirichlet distribution to generate allele frequencies.
`allele_freqs`	List of allele frequencies to be used instead of simulating allele frequencies
`internal_relatedness_alpha`	alpha parameter of beta distribution controlling the random relatedness draws for each sample
`internal_relatedness_beta`	beta parameter of beta distribution controlling the random relatedness draws for each sample
`internal_relatedness`	List of internal relatedness values to be used instead of simulating
`missingness`	probability of data being missing

Value

List structured for `run_mcmc()`, with elements `data`, `sample_ids`, `loci`, and `is_missing`, along with the simulated truth (`allele_freqs`, `sample_cois`, `sample_relatedness`, `true_genotypes`) and the `input` arguments.

Examples

```
sim <- simulate_data(
  mean_coi = 2,
  num_samples = 10,
  epsilon_pos = 0.01,
  epsilon_neg = 0.1,
  locus_freq_alphas = list(rep(1, 4), rep(1, 4), rep(1, 4))
)
sim$sample_cois
str(sim$data, max.level = 1)
```

---

simulate_observed_allele

Simulates the observation process

Description

Simulates the observation process

Usage

```
simulate_observed_allele(alleles, epsilon_pos, epsilon_neg, missingness)
```

Arguments

`alleles`	A numeric vector representing the number of strains contributing each allele
`epsilon_pos`	expected number of false positives
`epsilon_neg`	expected number of false negatives
`missingness`	probability that the data is missing

Details

Takes a numeric value representing the number of strains contributing an allele and returns a binary vector indicating the presence or absence of the allele.

Value

Binary numeric vector the same length as `alleles` indicating which alleles were observed. All zeros if the observation is missing.

Examples

```
simulate_observed_allele(
  alleles = c(2, 0, 1),
  epsilon_pos = 0.01, epsilon_neg = 0.1, missingness = 0
)
```

`simulate_observed_genotype` *Simulate observed genotypes*

Description

Simulate observed genotypes

Usage

```
simulate_observed_genotype(
  true_genotypes,
  epsilon_pos,
  epsilon_neg,
  missingness
)
```

Arguments

`true_genotypes`	a list of numeric vectors that are input to sim_observed_allele
`epsilon_pos`	expected number of false positives
`epsilon_neg`	expected number of false negatives
`missingness`	probability of data being missing

Details

Simulate the observation process across a list of observation vectors

Value

List the same length as `true_genotypes`, each element a binary vector of observed alleles as returned by `simulate_observed_allele()`.

Examples

```
true_genotypes <- list(c(1, 0, 1), c(0, 2, 0))
simulate_observed_genotype(
  true_genotypes,
  epsilon_pos = 0.01, epsilon_neg = 0.1, missingness = 0
)
```

simulate_sample_coi *Simulate sample COI*

Description

Simulate sample COI

Usage

```
simulate_sample_coi(num_samples, mean_coi)
```

Arguments

num_samples	the total number of biological samples to simulate
mean_coi	mean multiplicity of infection

Details

Simulate sample COIs from a zero-truncated Poisson distribution

Value

Integer vector of length num_samples with a COI of at least 1 for each sample.

Examples

```
simulate_sample_coi(num_samples = 10, mean_coi = 2)
```

simulate_sample_genotype *Simulate sample genotype*

Description

Simulate sample genotype

Usage

```
simulate_sample_genotype(sample_cois, locus_allele_dist, internal_relatedness)
```

Arguments

`sample_cois`	Numeric vector indicating the multiplicity of infection for each biological sample
`locus_allele_dist`	Allele frequencies – simplex parameter of a multinomial distribution
`internal_relatedness`	numeric 0-1 indicating the probability for a strain's allele to come from an existing lineage within host

Details

Simulates sampling the genetics at a single locus given an allele frequency distribution and a vector of sample COIs

Value

List with one element per sample, each a 1-row integer matrix counting the number of distinct strains carrying each allele.

Examples

```
simulate_sample_genotype(
  sample_cois = c(1, 2, 3),
  locus_allele_dist = c(0.5, 0.3, 0.2),
  internal_relatedness = c(0, 0, 0.5)
)
```

`summarize_allele_freqs` *Summarize allele frequencies*

Description

Summarize allele frequencies

Usage

```
summarize_allele_freqs(
  mcmc_results,
  lower_quantile = 0.025,
  upper_quantile = 0.975,
  merge_chains = TRUE
)
```

Arguments

mcmc_results	Result of calling run_mcmc()
lower_quantile	The lower quantile of the posterior distribution to return
upper_quantile	The upper quantile of the posterior distribution to return
merge_chains	boolean indicating that all chain results should be merged

Details

Summarize individual allele frequencies from the posterior distribution of sampled allele frequencies

Value

Data frame with one row per allele (per chain if `merge_chains` is `FALSE`) giving posterior quantiles and mean of its frequency, along with `locus` and `allele` identifiers.

Examples

```
allele_freq_summary <- summarize_allele_freqs(mcmc_results)
head(allele_freq_summary)
```

---

summarize_allele_freq_fn

Summarize Function of Allele Frequencies

Description

Summarize Function of Allele Frequencies

Usage

```
summarize_allele_freq_fn(
  mcmc_results,
  fn,
  lower_quantile = 0.025,
  upper_quantile = 0.975,
  merge_chains = TRUE
)
```

Arguments

mcmc_results	Result of calling run_mcmc()
fn	Function that takes as input a simplex to apply to each allele frequency vector
lower_quantile	The lower quantile of the posterior distribution to return
upper_quantile	The upper quantile of the posterior distribution to return
merge_chains	boolean indicating that all chain results should be merged

Details

General function to summarize the posterior distribution of functions of the sampled allele frequencies

Value

Data frame with one row per locus (per chain if `merge_chains` is `FALSE`) giving posterior quantiles and mean of `fn` applied to the sampled allele frequencies.

Examples

```
# Posterior summary of the major allele frequency at each locus
major_allele <- summarize_allele_freq_fn(mcmc_results, fn = max)
head(major_allele)
```

---

summarize_coi *Summarize COI*

Description

Summarize COI

Usage

```
summarize_coi(
  mcmc_results,
  lower_quantile = 0.025,
  upper_quantile = 0.975,
  naive_offset = 2,
  merge_chains = TRUE
)
```

Arguments

`mcmc_results`	Result of calling run_mcmc
`lower_quantile`	The lower quantile of the posterior distribution to return
`upper_quantile`	The upper quantile of the posterior distribution to return
`naive_offset`	Offset used in calculate_naive_coi_offset
`merge_chains`	boolean indicating that all chain results should be merged

Details

Summarize complexity of infection results from MCMC. Returns a dataframe that contains summaries of the posterior distribution of COI for each biological sample, as well as naive estimates of COI.

Value

Data frame with one row per sample (per chain if `merge_chains` is `FALSE`) giving posterior quantiles and mean of COI, naive estimates, and the posterior probability that the sample is polyclonal.

Examples

```
coi_summary <- summarize_coi(mcmc_results)
head(coi_summary)
```

`summarize_effective_coi` *Summarize effective COI*

Description

Summarize effective COI

Usage

```
summarize_effective_coi(
  mcmc_results,
  lower_quantile = 0.025,
  upper_quantile = 0.975,
  merge_chains = TRUE
)
```

Arguments

`mcmc_results`	Result of calling run_mcmc()
`lower_quantile`	The lower quantile of the posterior distribution to return
`upper_quantile`	The upper quantile of the posterior distribution to return
`merge_chains`	boolean indicating that all chain results should be merged

Details

Summarize effective COI from MCMC. Returns a dataframe that contains summaries of the posterior distribution of effective COI for each biological sample.

Value

Data frame with one row per sample (per chain if `merge_chains` is `FALSE`) giving posterior quantiles and mean of effective COI, `(1 - relatedness) * (COI - 1) + 1`.

Examples

```
effective_coi_summary <- summarize_effective_coi(mcmc_results)
head(effective_coi_summary)
```

summarize_epsilon_neg *Summarize epsilon_neg*

Description

Summarize epsilon_neg

Usage

```
summarize_epsilon_neg(
  mcmc_results,
  lower_quantile = 0.025,
  upper_quantile = 0.975,
  merge_chains = TRUE
)
```

Arguments

mcmc_results	Result of calling run_mcmc()
lower_quantile	The lower quantile of the posterior distribution to return
upper_quantile	The upper quantile of the posterior distribution to return
merge_chains	boolean indicating that all chain results should be merged

Details

Summarize epsilon negative results from MCMC. Returns a dataframe that contains summaries of the posterior distribution of epsilon negative for each biological sample.

Value

Data frame with one row per sample (per chain if `merge_chains` is `FALSE`) giving posterior quantiles and mean of the false negative rate.

Examples

```
eps_neg_summary <- summarize_epsilon_neg(mcmc_results)
head(eps_neg_summary)
```

summarize_epsilon_pos *Summarize epsilon_pos*

Description

Summarize epsilon_pos

Usage

```
summarize_epsilon_pos(
  mcmc_results,
  lower_quantile = 0.025,
  upper_quantile = 0.975,
  merge_chains = TRUE
)
```

Arguments

mcmc_results	Result of calling run_mcmc()
lower_quantile	The lower quantile of the posterior distribution to return
upper_quantile	The upper quantile of the posterior distribution to return
merge_chains	boolean indicating that all chain results should be merged

Details

Summarize epsilon positive results from MCMC. Returns a dataframe that contains summaries of the posterior distribution of epsilon positive for each biological sample.

Value

Data frame with one row per sample (per chain if merge_chains is FALSE) giving posterior quantiles and mean of the false positive rate.

Examples

```
eps_pos_summary <- summarize_epsilon_pos(mcmc_results)
head(eps_pos_summary)
```

summarize_he *Summarize locus heterozygosity*

Description

Summarize locus heterozygosity

Usage

```
summarize_he(
  mcmc_results,
  lower_quantile = 0.025,
  upper_quantile = 0.975,
  merge_chains = TRUE
)
```

Arguments

mcmc_results	Result of calling run_mcmc()
lower_quantile	The lower quantile of the posterior distribution to return
upper_quantile	The upper quantile of the posterior distribution to return
merge_chains	Merge the results of multiple chains into a single summary

Details

Summarize locus heterozygosity from the posterior distribution of sampled allele frequencies.

Value

Data frame with one row per locus (per chain if merge_chains is FALSE) giving posterior quantiles and mean of expected heterozygosity.

Examples

```
he_summary <- summarize_he(mcmc_results)
head(he_summary)
```

summarize_relatedness *Summarize relatedness*

Description

Summarize relatedness

Usage

```
summarize_relatedness(
  mcmc_results,
  lower_quantile = 0.025,
  upper_quantile = 0.975,
  merge_chains = TRUE
)
```

Arguments

mcmc_results	Result of calling run_mcmc()
lower_quantile	The lower quantile of the posterior distribution to return
upper_quantile	The upper quantile of the posterior distribution to return
merge_chains	boolean indicating that all chain results should be merged

Details

Summarize relatedness results from MCMC. Returns a dataframe that contains summaries of the posterior distribution of relatedness for each biological sample.

Value

Data frame with one row per sample (per chain if `merge_chains` is `FALSE`) giving posterior quantiles and mean of within-host relatedness, computed over draws where COI is greater than 1.

Examples

```
relatedness_summary <- summarize_relatedness(mcmc_results)
head(relatedness_summary)
```

Index