Package `BeviMed`

March 28, 2020

Type Package
Title Bayesian Evaluation of Variant Involvement in Mendelian Disease
Version 5.7
Date 2020-03-19
Author Daniel Greene, Sylvia Richardson and Ernest Turro
Maintainer Daniel Greene <dg333@cam.ac.uk>

Description A fast integrative genetic association test for rare diseases based on a model for disease status given allele counts at rare variant sites. Probability of association, mode of inheritance and probability of pathogenicity for individual variants are all inferred in a Bayesian framework - 'A Fast Association Test for Identifying Pathogenic Variants Involved in Rare Diseases', Greene et al 2017 <doi:10.1016/j.ajhg.2017.05.015>.

License GPL (>= 2)
Imports Rcpp (>= 0.12.3), methods
LinkingTo Rcpp
Depends R (>= 3.0.0)
Suggests knitr
VignetteBuilder knitr
RoxygenNote 6.1.1
NeedsCompilation yes
Repository CRAN
Date/Publication 2020-03-28 13:20:05 UTC

R topics documented:

BeviMed-package .................................................. 2
bevimed .......................................................... 3
bevimed_m ......................................................... 4
bevimed_polytomous ............................................. 7
call_cpp ......................................................... 8
CI_gamma1_evidence ............................................ 10
Description

A fast integrative genetic association test for rare diseases.

Details

BeviMed estimates a probability of association between a case/control label and allele counts at rare variant sites in a genomic locus and also, given that there is an association, the probabilities that each variant is involved in the disease. It does so by estimating the evidence for a model where the case/control label is independent of the allele configurations, and a model in which the probability of the case/control label depends on the corresponding allele configuration and a latent partition of variants into pathogenic and non-pathogenic groups.
Author(s)
Daniel Greene.
Maintainer: Daniel Greene <dg333@cam.ac.uk>

References
Greene et al., A Fast Association Test for Identifying Pathogenic Variants Involved in Rare Diseases, The American Journal of Human Genetics (2017), http://dx.doi.org/10.1016/j.ajhg.2017.05.015.

See Also
bevimed

Description
Infer probabilities of association between disease label and locus and posterior parameter values under BeviMed model.

Usage
bevimed(y, G, ploidy = rep(2L, length(y)),
prior_prob_association = 0.01, prior_prob_dominant = 0.5,
dominant_args = NULL, recessive_args = NULL, ...)

Arguments

y Logical vector of case (TRUE) control (FALSE) status.
G Integer matrix of variant counts per individual, one row per individual and one column per variant.
ploidy Integer vector giving ploidy of samples.
prior_prob_association The prior probability of association.
prior_prob_dominant The prior probability of dominant inheritance given that there is an association.
dominant_args Arguments to pass to bevimed_m conditioning on dominant inheritance.
recessive_args Arguments to pass to bevimed_m conditioning on recessive inheritance.
... Arguments to be passed to bevimed_m for both modes of inheritance.

Value
BeviMed object containing results of inference.
References

Greene et al., A Fast Association Test for Identifying Pathogenic Variants Involved in Rare Diseases, The American Journal of Human Genetics (2017), http://dx.doi.org/10.1016/j.ajhg.2017.05.015.

See Also

prob_association, bevimed_m, summary.BeviMed, bevimed_polytomous

---

**bevimed_m**

*Perform inference under model gamma = 1 conditional on mode of inheritance*

---

**Description**

Sample from posterior distribution of parameters under model gamma = 1 and conditional on mode of inheritance, set via the min_ac argument.

**Usage**

bevimed_m(y, G, min_ac = 1L, tau_shape = c(1, 1), pi_shape = c(6, 1), omega_shape = if (max(min_ac) == 1L) c(2, 8) else c(2, 2), samples_per_chain = 1000, stop_early = FALSE, blocks = 5, burn = as.integer(samples_per_chain/10), temperatures = (0:6/6)^2, tune_temps = 0, return_z_trace = TRUE, return_x_trace = TRUE, raw_only = FALSE, swaps = as.integer(length(temperatures)/2), optimise_z0 = FALSE, tune_omega_and_phi_proposal_sd = FALSE, tune_block_size = 100, variant_weights = NULL, standardise_weights = TRUE, log_phi_mean = -0.15, log_phi_sd = sqrt(0.3), tandem_variant_updates = if (max(min_ac) == 1) 0 else min(sum(y), ncol(G)), ...)

**Arguments**

- **y** Logical vector of case (TRUE) control (FALSE) status.
- **G** Integer matrix of variant counts per individual, one row per individual and one column per variant.
- **min_ac** Integer vector with a length equalling the number of individuals or length 1 (in which case the given value is used for all individuals) giving the minimum number of alleles at pathogenic variant sites each individual requires in order to classify as having a 'pathogenic allele configuration'. Thus, this parameter encodes the mode of inheritance. For instance, setting this parameter to 1 corresponds to dominant inheritance. If there are differences in ploidy between individuals in the locus, it is necessary to set it on an sample level basis - e.g. to ensure sex is accounted for if the locus lies on the X chromosome.
- **tau_shape** Beta shape hyper-priors for prior on rate of affection (i.e. being a case) amongst individuals with non-pathogenic variant combinations (i.e. they have less than min_ac variants).
**pi_shape**  
Beta shape hyper-priors for prior on rate of affection (i.e. being a case) amongst individuals with pathogenic variant combinations (i.e. they have at least `min_ac` variants).

**omega_shape**  
Beta shape hyper-priors for prior on rate of pathogenicity amongst variants.

**samples_per_chain**  
Number of samples to draw from each chain.

**stop_early**  
Logical value determining whether to attempt to stop the sampling as soon as certain conditions are met (i.e. either the estimated marginal log likelihood lies within a certain confidence interval, or we are sufficiently confident that the log Bayes factor against of model gamma = 1 over model gamma = 0 is sufficiently low).

**blocks**  
Maximum number of blocks of `samples_per_chain` samples to draw before either the confidence interval for the marginal likelihood under the model gamma = 1 is sufficiently small or terminating the sampling. This parameter is ignored if `stop_early==TRUE`.

**burn**  
Number of samples to drop from the start of the chain.

**temperatures**  
Numeric vector of temperatures of power posteriors. One chain will be created for each element of the vector at the corresponding temperature.

**tune_temps**  
Integer value - if greater than 0, the `temperatures` argument is ignored, and instead `tune_temps` tuned temperatures are used instead.

**return_z_trace**  
Logical value determining whether to store the z-vectors for each chain, which uses a lot of memory, particularly if `samples_per_chain`, `k` and `length(temperatures)` are large.

**return_x_trace**  
Logical value determining whether to store the x variable determined by success samples of z. Potentially uses a lot of memory, particularly if `samples_per_chain`, `k` and `length(temperatures)` are large.

**raw_only**  
Logical value determining whether to return raw output of MCMC routine only.

**swaps**  
Number of swaps between adjacent tempered chains to perform per update cycle.

**optimise_z0**  
Logical value determining whether to use a simulated annealing optimisation run to tune the initial values of z.

**tune_omega_and_phiProposal_sd**  
Logical value determining whether the proposal SDs of the Metropolis-Hastings estimated parameters should be tuned for a target acceptance range.

**tune_block_size**  
Integer value giving number of samples to draw when estimating the acceptance rate of the omega/phi proposals.

**variant_weights**  
Vector of log-odds off-sets for rates of pathogenicity of individual variants relative to the global rate, omega.

**standardise_weights**  
Boolean value determining whether weights should be standardised by subtracting their mean and dividing by their sample standard deviation. If `FALSE`, weights are untransformed.
log_phi_mean  Mean for normal prior on scaling factor phi.
log_phi_sd  SD for normal prior on scaling factor phi. Setting to 0 causes the weights to be
            fixed and not estimated.
tandem_variant_updates  Number of tandem variant updates to make per update cycle.

...  Other arguments to be passed to stop_chain and/or tune_proposal_sds.

Details

A BeviMed_m object is a list containing elements:

- ‘parameters’: a list containing arguments used in the function call, including the adjusted
  weights used in the inference in the ‘c_weights’ slot,
- ‘traces’: a list of traces of model parameters from all MCMC chains for each parameter.
  Parameters sampled are z, omega, phi and x (the indicator of having a pathogenic configuration
  of alleles). The list of traces is named by parameter name, and each is a matrix where the rows
  correspond to samples. $z$ has $k$ columns for each temperature, with the samples from the
  true posterior (i.e. with temperature equal to 1) of $z$ corresponding to the final $k$ columns.
  Likewise, the true posterior is given by the final column for the traces of phi and omega. The
  trace of $x$ is only given for temperature equal to 1 to reduce memory usage.
- ‘final’: a list named by model parameter giving the final sample of each,
- ‘swaps’: a list with an element named ‘accept’ which is a logical vector whose ith element
  indicates whether the ith swap between adjacent tempered chains was accepted or not, and an
  element named ‘at_temperature’, an integer vector whose ith element indicates which pair of
  consecutive temperatures was the ith to be proposed for swapping (giving the lowest one).

Value

An object of class BeviMed_m.

References

Greene et al., A Fast Association Test for Identifying Pathogenic Variants Involved in Rare Diseases,

See Also

bevimed_m, prob_association_m
bevimed_polytomous

Model selection for multiple association models

Description

Apply bevimed to the no association model (gamma = 0) and multiple association models for different sets of variants, for instance, corresponding to different functional consequences.

Usage

bevimed_polytomous(y, G, ploidy = rep(2L, length(y)), variant_sets, prior_prob_association = rep(0.01/length(variant_sets), length(variant_sets)), tau0_shape = c(1, 1), moi = rep("dominant", length(variant_sets)), model_specific_args = vector(mode = "list", length = length(variant_sets)), ...)

Arguments

y Logical vector of case (TRUE) control (FALSE) status.

G Integer matrix of variant counts per individual, one row per individual and one column per variant.

ploidy Integer vector giving ploidy of samples.

variant_sets List of integer vectors corresponding to sets of indices of G, each of which is to be considered in a model explaining the phenotype, y.

prior_prob_association The prior probability of association.

tau0_shape Beta shape hyper-priors for prior on rate of case labels.

moi Character vector giving mode of inheritance for each model.

model_specific_args List of named lists of parameters to use in bevimed_m applications for specific models.

... Other arguments to pass to bevimed_m.

References

Greene et al., A Fast Association Test for Identifying Pathogenic Variants Involved in Rare Diseases, The American Journal of Human Genetics (2017), http://dx.doi.org/10.1016/j.ajhg.2017.05.015.

See Also

bevimed_m, bevimed
call_cpp

R interface to BeviMed c++ MCMC procedure

Description

Allows other functions in the package to call the c++ function passing arguments more succinctly and by name.

Usage

call_cpp(samples_per_chain, y, block_starts, block_ends, cases, counts, min_ac, tau_shape, pi_shape, omega_shape, temperatures, z0_matrix, estimate_omega, logit_omegas, logit_omega_proposal_sds, variant_weights, estimate_phi, log_phis, log_phi_mean, log_phi_sd, log_phi_proposal_sds, chain_swaps_per_cycle, annealing, tandem_variant_updates, comphet_variant_block_starts, comphet_variant_block_ends, comphet_variants, return_z_trace, return_x_trace, burn = 0, check = TRUE)

Arguments

samples_per_chain
Number of samples to draw from each chain.

y
Logical vector of subject affectedness status.

block_starts
Integer vector of k 0-indexed start positions (with respect to cases and counts) for contiguous blocks relating to the k variants.

block_ends
Integer vector of (exclusive) k 0-indexed end positions.

cases
0 based vector of case indices with respect to y.

counts
Vector of variant counts.

min_ac
Integer vector with a length equalling the number of individuals or length 1 (in which case the given value is used for all individuals) giving the minimum number of alleles at pathogenic variant sites each individual requires in order to classify as having a ‘pathogenic allele configuration’. Thus, this parameter encodes the mode of inheritance. For instance, setting this parameter to 1 corresponds to dominant inheritance. If there are differences in ploidy between individuals in the locus, it is necessary to set it on an sample level basis - e.g. to ensure sex is accounted for if the locus lies on the X chromosome.

tau_shape
Beta distribution parameterisation of benign variant configuration rate of affection, q.

pi_shape
Beta distribution parameterisation of pathogenic variant configuration rate of affection, p.

omega_shape
Beta distribution of global rate of pathogenicity of variants in gene given pathogenicity of gene, omega.
temperatures  Numeric vector of temperatures of power posteriors. One chain will be created for each element of the vector at the corresponding temperature.

z0_matrix  Matrix of logicals, where the rows are used as an initial zs for the chains.

estimate_omega  Logical value determining whether to estimate the parameter omega.

logit_omegas  Numeric vector of logit omega values, one value per chain.

logit_omega_proposal_sds  Numeric vector of proposal standard deviations for Metropolis-Hastings sampling of logit omega parameter, one value per chain.

variant_weights  Vector of log-odds off-sets for rates of pathogenicity of individual variants relative to the global rate, omega.

estimate_phi  Logical value determining whether to estimate a scaling factor of variant_weights.

log_phis  Numeric vector of log phi values, one value per chain.

log_phi_mean  Mean for normal prior on scaling factor phi.

log_phi_sd  SD for normal prior on scaling factor phi.

log_phi_proposal_sds  Numeric vector of proposal standard deviations for Metropolis-Hastings sampling of log phi parameter, one value per chain.

chain_swaps_per_cycle  Number of chain swaps to propose per update cycle.

annealing  Logical value determining whether to anneal the chains, e.g. for optimisation.

tandem_variant_updates  Number of tandem variant updates to make per update cycle.

comphet_variant_block_starts  0-indexed start positions for contiguous blocks of variants in comphet_variants.

comphet_variant_block_ends  As comphet_variant_block_starts for (exclusive) stop positions.

comphet_variants  Integer vector giving variant numbers (0-based, i.e. between 0 and k-1). Used to pick pairs of variants for tandem updates from.

return_z_trace  Logical value determining whether to store the z-vectors for each chain, which uses a lot of memory, particularly if samples_per_chain, k and length(temperatures) are large.

return_x_trace  Logical value determining whether to store the x variable determined by success samples of z. Potentially uses a lot of memory, particularly if samples_per_chain, k and length(temperatures) are large.

burn  Number of samples to drop from the start of the chain.

check  Logical value indicating whether to perform validation on the arguments before calling the c++ function.

Value

Object of class BeviMed_raw, containing the output of the MCMC sampling.
CI_gamma1_evidence  
Estimate confidence interval for estimated marginal likelihood

Description

Central limit theorem not applicable so use simulation to estimate confidence interval for evidence.

Usage

CI_gamma1_evidence(temperatures, y_loglik_t_equals_1_traces, 
               confidence = 0.95, simulations = 1000)

Arguments

temperatures  Numeric vector of temperatures of power posteriors. One chain will be created 
              for each element of the vector at the corresponding temperature.

y_loglik_t_equals_1_traces  Numeric matrix of log probabilities of y at different temperatures (columns) in 
                            different iterations (rows).

confidence  Numeric value of statistical confidence with which returning interval should 
            contain the true value.

simulations  Integer value of number of simulations to use in estimation of the confidence 
             interval.

Value

Confidence interval as numeric vector of length 2.

conditional_prob_pathogenic

Calculate probability of pathogenicity for variants conditional on mode of inheritance.

Description

Calls bevimed_m and extract_conditional_prob_pathogenic to obtain probabilities of pathogenicity.

Usage

conditional_prob_pathogenic(...)  

Arguments

...  Arguments to pass to bevimed_m.
expected_explained

Value

Probabilities of pathogenicity.

See Also

extract_conditional_prob_pathogenic, bevimed_m

expected_explained  

Calculate expected number of explained cases

Description

Use bevimed_m to perform inference under model gamma = 1 and return only the expected number of cases explained by pathogenic allele configurations.

Usage

expected_explained(...)

Arguments

...  

Arguments to pass to bevimed_m.

Value

Numeric value.

See Also

bevimed_m, extract_expected_explained

explaining_variants  

Calculate expected number of pathogenic variants in cases

Description

Use bevimed_m to perform inference under model gamma = 1 and return only the expected number of pathogenic variants in cases.

Usage

explaining_variants(...)

Arguments

...  

Arguments to pass to bevimed_m.
Value

Numeric value.

See Also

extract_explaining_variants, bevimed_m

description

Extract the probability of pathogenicity for individual variants from a BeviMed_m object.

Usage

eextract_conditional_prob_pathogenic(x)

Arguments

x
Object of class x_BeviMed_m. See function bevimed_m.

Value

Vector of probabilities of pathogenicity for individual variants.

See Also

conditional_prob_pathogenic, bevimed_m

description

Extract expected number of explained cases

Description

Extract expected number of cases explained by pathogenic configurations of alleles from BeviMed_m object.

Usage

eextract_expected_explained(x)
extract_explaining_variants

Arguments
x Object of class x_BeviMed_m. See function bevimed_m.

Value
Numeric value.

See Also
expected_explained, bevimed_m

Description
Extract expected number of variants involved in cases explained by pathogenic configurations of alleles from BeviMed_m object.

Usage
extract_explaining_variants(x)

Arguments
x Object of class x_BeviMed_m. See function bevimed_m.

Value
Numeric value.

See Also
explaining_variants, bevimed_m
**extract_gammapl_evidence**

*Extract evidence for model gamma = 1*

**Description**

Extract evidence from BeviMed_m object.

**Usage**

```r
extract_gammapl_evidence(x)
```

**Arguments**

- `x` Object of class `x_BeviMed_m`. See function `bevimed_m`.

**Value**

Log marginal likelihood.

**See Also**

`gammapl_evidence`, `bevimed_m`

---

**extract_prob_association**

*Extract the posterior probability of association*

**Description**

Get posterior probability of association as numeric value, or optionally as numeric vector of length two with probabilities broken down by mode of inheritance (by passing `by_model=TRUE`), from a BeviMed object.

**Usage**

```r
extract_prob_association(x, by_model = FALSE)
```

**Arguments**

- `x` Object of class `BeviMed`.
- `by_model` Logical value determining whether to return probabilities broken down by mode of inheritance.
**extract_prob_pathogenic**

**Value**

Probability values.

**See Also**

`prob_association, bevimed`

---

**extract_prob_pathogenic**

*Extract variant marginal probabilities of pathogenicity*

---

**Description**

Extract the marginal probability of pathogenicity for individual variants from BeviMed object, optionally broken down by mode of inheritance/model.

**Usage**

```r
extract_prob_pathogenic(x, by_model = TRUE)
```

**Arguments**

- `x` Object of class BeviMed.
- `by_model` Logical value determining whether to return probabilities broken down by mode of inheritance.

**Value**

A vector of probabilities of pathogenicity for individual variants, or if by_model is TRUE, then a matrix of probabilities, with rows corresponding to modes of inheritance and columns to variants.

**See Also**

`prob_pathogenic, bevimed`
**gamma0_evidence**

Calculate marginal probability of observed case-control status $y$ under model $\gamma = 0$

**Description**

Marginal probability calculated exactly by integration.

**Usage**

```
gamma0_evidence(y, tau0_shape = c(1, 1))
```

**Arguments**

- `y`: Logical vector of case (TRUE) control (FALSE) status.
- `tau0_shape`: Beta shape hyper-priors for prior on rate of case labels

**Value**

Log marginal likelihood.

**See Also**

`bevimed`, `gamma1_evidence`

---

**gamma1_evidence**

Calculate evidence under model $\gamma = 1$

**Description**

Use `bevimed_m` to perform inference under model $\gamma = 1$ and return only the log evidence/integrated likelihood.

**Usage**

```
gamma1_evidence(...)```

**Arguments**

- `...`: Arguments to pass to `bevimed_m`.

**Value**

Log marginal likelihood.

**See Also**

`bevimed_m`, `extract_gamma1_evidence`
Calculate log Bayes factor between an association model with a given mode of inheritance and model gamma = 0

Description

Compute log Bayes factor of an association model and model gamma = 0.

Usage

log_BF(y, tau0_shape = c(1, 1), ...)

Arguments

y Logical vector of case (TRUE) control (FALSE) status.
tau0_shape Beta shape hyper-priors for prior on rate of case labels.
... Arguments to pass to bevimed_m.

Value

Log Bayes factor.

See Also

bevimed_m, prob_association_m

Print readable summary of BeviMed object

Description

Print summary statistics of BeviMed inference, including probability of association, probability of dominant inheritance given association and probability of pathogenicity of each variant under dominant and recessive inheritance.

Usage

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

Arguments

x BeviMed object.
... Arguments passed to summary.BeviMed
Value
Prints a summary.

See Also
summary.BeviMed

print.BeviMed_m  Print BeviMed_m object

Description
Print summary statistics for BeviMed_m object.

Usage
## S3 method for class 'BeviMed_m'
print(x, ...)

Arguments
x  Object of class BeviMed_m. See function bevimed_m.
...
Unused arguments.

Value
Prints a summary.

See Also
summary.BeviMed_m


Description
Print summary statistics of BeviMed inference, including probability of association, probability
of dominant inheritance given association and probability of pathogenicity of each variant under
dominant and recessive inheritance.

Usage
## S3 method for class 'BeviMed_summary'
print(x, print_prob_pathogenic = TRUE, ...)

Arguments
x  Object of class BeviMed_summary
print_prob_pathogenic  Logical indicating whether to print
probabilities of pathogenicity.
...  Additional arguments.
**prob_association**

**Arguments**

- `x` BeviMed_summary object.
- `print_prob_pathogenic` Logical value indicating whether to print list of marginal probabilities of $z_j = 1$ for all variants $j$ under each mode of inheritance.
- `...` Unused arguments

**Value**

Prints a summary

---

**prob_association**  *Calculate probability of association*

**Description**

Calculate probability of an association between case/control label and allele configuration, optionally broken down by mode of inheritance/model.

**Usage**

`prob_association(by_model = FALSE, ...)`

**Arguments**

- `by_model` Logical value determining whether to return probabilities broken down by mode of inheritance.
- `...` Arguments to pass to `bevimed`.

**Value**

Probability of association.

**See Also**

`bevimed`, `extract_prob_association`
**prob_association_m**    
*Calculate probability of association for one mode of inheritance*

**Description**
Equivalent to **prob_association** where the prior probability of one mode of inheritance is 1. This function is faster, as it only calls **bevimed_m** once.

**Usage**
```
prob_association_m(y, min_ac = 1L, prior_prob_association = 0.01, ...)
```

**Arguments**
- **y**
  Logical vector of case (TRUE) control (FALSE) status.
- **min_ac**
  Integer vector with a length equalling the number of individuals or length 1 (in which case the given value is used for all individuals) giving the minimum number of alleles at pathogenic variant sites each individual requires in order to classify as having a 'pathogenic allele configuration'. Thus, this parameter encodes the mode of inheritance. For instance, setting this parameter to 1 corresponds to dominant inheritance. If there are differences in ploidy between individuals in the locus, it is necessary to set it on an sample level basis - e.g. to ensure sex is accounted for if the locus lies on the X chromosome.
- **prior_prob_association**
  The prior probability of association.
- **...**
  Other arguments to pass to **log_BF**.

**Value**
Probability value.

**See Also**
- **log_BF**, **prob_association**, **bevimed_m**

**prob_pathogenic**    
*Calculate variant marginal probabilities of pathogenicity*

**Description**
Calls **bevimed** and **extract_prob_pathogenic** to obtain marginal probabilities of pathogenicity.

**Usage**
```
prob_pathogenic(by_model = FALSE, ...)
```
**Arguments**

- **by_model** Logical value determining whether to return probabilities broken down by mode of inheritance.
- ... Arguments to pass to `bevimed`.

**Value**

If `by_model` is FALSE, a vector of probabilities of pathogenicity for each variant, otherwise a list of vectors of probabilities of pathogenicity conditional on each compared association model.

**See Also**

`extract_prob_pathogenic`, `bevimed`

---

**Description**

This function could be used to stitch together consecutive chains to create one larger sampled set of states from the MCMC procedure.

**Usage**

```r
stack_BeviMeds(objects)
```

**Arguments**

- **objects** list of `BeviMed_raw` objects.

**Value**

`BeviMed` object.
stop_chain  

Apply the MCMC algorithm in blocks until conditions are met

Description

Sample blocks of a given size until either the estimated log marginal likelihood falls within a given confidence interval, there is sufficient confidence that the evidence model gamma = 1 is at most a certain quantity, or a certain number of blocks have been sampled.

Usage

```r
stop_chain(y, blocks_remaining, start_zs, start_logit_omegas, start_log_phis, temperatures, tolerance = 1, confidence = 0.95, simulations = 1000, log_evidence_threshold = -Inf, y_log_lik_t_equals_1_traces = matrix(ncol = length(temperatures), nrow = 0), full_block_traces = list(), verbose = FALSE, ...)
```

Arguments

- `y` Logical vector of case (TRUE) control (FALSE) status.
- `blocks_remaining` Maximum number of blocks left before termination.
- `start_zs` Initial (logical) z-matrix.
- `start_logit_omegas` Initial values of logit_omega (numeric vector - one value per chain).
- `start_log_phis` Initial values of log_phi (numeric vector - one value per chain).
- `temperatures` Numeric vector of temperatures of power posteriors. One chain will be created for each element of the vector at the corresponding temperature.
- `tolerance` Maximum width for confidence_interval of log marginal likelihood to allow before stopping the chain.
- `confidence` Numeric value of statistical confidence with which returning interval should contain the true value.
- `simulations` Integer value of number of simulations to use in estimation of the confidence interval.
- `log_evidence_threshold` Numeric value used to determine whether to stop the sampling procedure after successive blocks. If we are confident (to the level of confidence) that the evidence for model gamma = 1 is under this value, sampling is halted.
- `y_log_lik_t_equals_1_traces` Numeric matrix of log probabilities of y at different temperatures (columns) in different iterations (rows).
- `full_block_traces` List of outputs of calls to MCMC routine.
- `verbose` To print execution progress or not.
- `...` Other arguments passed to `call_cpp`
Value

An object of class BeviMed.

Description

Subset an allele count matrix given a minimum allele count threshold for pathogenicity per individual so that only variants for which data relevant to pathogenicity are retained. This is useful to apply before running `bevimed` as it reduces the size of the parameter space used in the inference.

Usage

```r
subset_variants(G, min_ac = 1L, return_variants = FALSE)
```

Arguments

- `G` : Integer matrix of variant counts per individual, one row per individual and one column per variant.
- `min_ac` : Integer vector with a length equalling the number of individuals or length 1 (in which case the given value is used for all individuals) giving the minimum number of alleles at pathogenic variant sites each individual requires in order to classify as having a 'pathogenic allele configuration'. Thus, this parameter encodes the mode of inheritance. For instance, setting this parameter to 1 corresponds to dominant inheritance. If there are differences in ploidy between individuals in the locus, it is necessary to set it on a sample level basis - e.g. to ensure sex is accounted for if the locus lies on the X chromosome.
- `return_variants` : Logical value determining whether to return an integer vector of indices of retained variants or the subsetted allele count matrix

Description

Create a summary of inference over model gamma = 0 and association models.

Usage

```r
## S3 method for class 'BeviMed'
summary(object, ...)
```
Arguments

object Object of class BeviMed.
... Arguments passed to summary.BeviMed_m.

Details

Returns a BeviMed_summary object, which is a list containing elements:

- 'prob_association': the probability of association under each association model,
- 'prior_prob_association': the prior probability of association for each association model,
- 'gamma0_evidence': the log evidence under model gamma = 0,
- 'models': a list of summaries of model conditional inferences, i.e. objects of class BeviMed_m_summary. See summary.BeviMed_m for more details.

Value

Object of class BeviMed_summary.

See Also

summary.BeviMed_m

summary.BeviMed_m
Summarise a BeviMed_m object

Description

Create a summary of inference conditional on mode of inheritance.

Usage

## S3 method for class 'BeviMed_m'
summary(object, confidence = 0.95,
simulations = 1000, ...)

Arguments

object Object of class BeviMed_m. See function bevimed_m.
confidence Numeric value of statistical confidence with which returning interval should contain the true value.
simulations Integer value of number of simulations to use in estimation of the confidence interval.
... Unused arguments.
Details

Returns a `BeviMed_m_summary` object, which is a list containing elements:

- ‘gamma1_evidence’: the log evidence under model gamma = 1,
- ‘gamma1_evidence_confidence_interval’: a confidence interval for the log evidence under model gamma = 1,
- ‘conditional_prob_pathogenic’: vector of marginal probabilities of pathogenicity for individual variants,
- ‘expected_explained’: the expected number of cases with a pathogenic configuration of alleles,
- ‘explaining_variants’: the expected number of variants present for which cases harbour a rare allele,
- ‘number_of_posterior_samples’: the number of samples from the posterior distribution of the model parameters which upon which the summary is based,
- ‘omega_estimated’: logical value indicating whether the parameter omega was estimated,
- ‘omega’: the posterior mean of omega,
- ‘omega_acceptance_rate’: if omega was estimated, the rate of acceptance of proposed omega values in the Metropolis-Hastings sampling routine,
- ‘phi_estimated’: logical value indicating whether the parameter phi was estimated,
- ‘phi’: the posterior mean of phi,
- ‘phi_acceptance_rate’: if phi was estimated, the rate of acceptance of proposed phi values in the Metropolis-Hastings sampling routine,
- ‘N’: number of samples in the analysis,
- ‘k’: number of variants in the analysis,
- ‘variant_counts’: list of counts of each variant for cases and controls,
- ‘temperatures’: numeric vector of temperatures used as temperatures for tempered MCMC chains

Value

Object of class `BeviMed_m_summary`.

See Also

`summary.BeviMed_m_summary`
**sum_ML_over_PP**  
*Calculate marginal likelihood from power posteriors output*

**Description**
Calculate the Marginal Likelihood by summation over power posterior likelihood expectances

**Usage**
```
sum_ML_over_PP(y_log_lik_t_equals_1_traces, temperatures)
```

**Arguments**
- `y_log_lik_t_equals_1_traces`
  Numeric matrix of log probabilities of `y` at different temperatures (columns) in different iterations (rows).
- `temperatures`
  Numeric vector of temperatures used to produce `y_log_lik_t_equals_1_traces`.

**Value**
Numeric value of estimated log marginal likelihood.

---

**tune_proposal_sds**  
*Tune proposal standard deviation for MH sampled parameters*

**Description**
Tune the proposal standard deviations for the Metropolis-Hastings updates of either `phi` or `omega`

**Usage**
```
tune_proposal_sds(tune_for = c("logit_omega"), initial_proposal_sds,  
target_acceptance_range = c(0.3, 0.7), other_param_proposal_sd = 0.7,  
max_tuning_cycles = 10, initial_rate = 1, rate_decay = 1.2,  
verbose = FALSE, ...)
```

**Arguments**
- `tune_for`
  Character vector of length one, naming which variable to tune the proposal SDs for: either "logit_omega" or "log_phi".
- `initial_proposal_sds`
  Numeric vector with the initial values of the proposal SDs.
- `target_acceptance_range`
  Numeric vector of length 2 where the first element is the lower bound for the acceptance interval and the second is the upper bound.
other_param_proposal_sd
The proposal SD to use for log phi when tuning logit_omega or vice versa.

max_tuning_cycles
Maximum number of tuning cycles to perform before returning the proposal
SDs as they are.

initial_rate
Initial rate at which to mutate the proposal SDs.

rate_decay
Geometric rate of decay for size of proposal SD mutation with each successive
tuning cycle.

verbose
To print execution progress or not.

...
Other arguments to be passed to call_cpp.

Value
Numeric vector of proposal SDs for the different temperature chains.

Description
Tune temperatures using interval bisection to minimise Kullback-Liebler divergence between
adjacent power posteriors.

Usage
tune_temperatures(number_of_temperatures, return_temperatures = FALSE,
...

Arguments

number_of_temperatures
Integer value giving number of tuned temperatures (including 0 and 1) to obtain.

return_temperatures
Logical value determining whether to return just the numeric vector of tuned
temperatures or to return the BeviMed_m-classed object containing the output of
the MCMC sampling.

...
Other arguments to pass to call_cpp.

Value
If return_temperatures == TRUE, a numeric vector of tuned temperatures, otherwise an object of
class BeviMed_m.
Index

BeviMed (BeviMed-package), 2
bevimed, 3, 3, 7, 15, 16, 19–21, 23
BeviMed-package, 2
bevimed_m, 3, 4, 4, 6, 7, 10–14, 16–18, 20, 24
bevimed_polytomous, 4, 7
call_cpp, 8, 22, 27
CI_gamma1_evidence, 10
conditional_prob_pathogenic, 10, 12
expected_explained, 11, 13
explaining_variants, 11, 13
extract_conditional_prob_pathogenic,
   10, 11, 12
extract_expected_explained, 11, 12
extract_explaining_variants, 12, 13
extract_gamma1_evidence, 14, 16
extract_prob_association, 14, 19
extract_prob_pathogenic, 15, 20, 21
gamma0_evidence, 16
gamma1_evidence, 14, 16, 16
log_BF, 17, 20
print.BeviMed, 17
print.BeviMed_m, 18
print.BeviMed_summary, 18
prob_association, 4, 15, 19, 20
prob_association_m, 6, 17, 20
prob_pathogenic, 15, 20
stack_BeviMeds, 21
stop_chain, 6, 22
subset_variants, 23
sum_ML_over_PP, 26
summary.BeviMed, 4, 17, 18, 23, 25
summary.BeviMed_m, 18, 24, 24
tune_proposal_sds, 6, 26
tune_temperatures, 27