Package 'bhetGP'

July 19, 2025

Type Package

Title Bayesian Heteroskedastic Gaussian Processes

Version 1.0.1

Maintainer Parul V. Patil <parulvijay@vt.edu>

Description Performs Bayesian posterior inference for heteroskedastic Gaussian processes. Models are trained through MCMC including elliptical slice sampling (ESS) of latent noise processes and Metropolis-Hastings sampling of kernel hyperparameters. Replicates are handled efficientyly through a Woodbury formulation of the joint likelihood for the mean and noise process (Binois, M., Gramacy, R., Ludkovski, M. (2018) <doi:10.1080/10618600.2018.1458625>) For large data, Vecchia-approximation for faster computation is leveraged (Sauer, A., Cooper, A., and Gramacy, R., (2023), <doi:10.1080/10618600.2022.2129662>). Incorporates 'OpenMP' and SNOW parallelization and utilizes 'C'/'C++' under the hood.

License LGPL

Encoding UTF-8

NeedsCompilation yes

Imports grDevices, graphics, stats, doParallel, foreach, parallel, GpGp, GPvecchia, Matrix, Rcpp, mvtnorm, FNN, hetGP, laGP

LinkingTo Rcpp, RcppArmadillo

Suggests interp

RoxygenNote 7.3.2

Author Parul V. Patil [aut, cre]

Repository CRAN

Date/Publication 2025-07-18 22:50:02 UTC

Contents

---

bhetGP-package *Package bhetGP*

Description

Performs Bayesian posterior inference for heteroskedastic Gaussian processes. Models are trained through MCMC including elliptical slice sampling (ESS) of latent noise processes and Metropolis-Hastings sampling of kernel hyperparameters. Replicates are handled efficientyly through a Woodbury formulation of the joint likelihood for the mean and noise process (Binois, M., Gramacy, R., Ludkovski, M. (2018) <doi:10.1080/10618600.2018.1458625>) For large data, Vecchia-approximation for faster computation is leveraged (Sauer, A., Cooper, A., and Gramacy, R., (2023), <doi:10.1080/10618600.2022.2129662>) Incorporates 'OpenMP' and SNOW parallelization and utilizes 'C'/'C++' under the hood.

Important Functions

- `bhetGP`: conducts MCMC sampling of hyperparameters and latent noise layer for a heteroskedatic GP.
- `bhomGP`: conducts MCMC sampling of hyperparameters for a homoskedastic GP.
- `trim`: cuts off burn-in and optionally thins samples
- `predict`: calculates posterior mean and variance over a set of input locations (optionally calculates EI or entropy)
- `plot`: produces trace plots, hidden layer plots, and posterior predictive plots

Author(s)

Parul V. Patil <parulvijay@vt.edu>

References

M. Binois, Robert B. Gramacy, M. Ludkovski (2018), Practical heteroskedastic Gaussian process modeling for large simulation experiments, Journal of Computational and Graphical Statistics, 27(4), 808–821.

Katzfuss, Matthias, Joseph Guinness, and Earl Lawrence. Scaled Vecchia approximation for fast computer-model emulation. SIAM/ASA Journal on Uncertainty Quantification 10.2 (2022): 537-554.

Sauer, A., Cooper, A., & Gramacy, R. B. (2023). Vecchia-approximated deep Gaussian processes for computer experiments. Journal of Computational and Graphical Statistics, 32(3), 824-837.

Examples

```
# See ?bhetGP, or ?bhomGP for examples
```

bhetGP *MCMC sampling for Heteroskedastic GP*

Description

Conducts MCMC sampling of hyperparameters and latent noise process `llam` for a hetGP. Separate length scale parameters `theta_lam` and `theta_y` govern the correlation strength of the hidden layer and outer layer respectively. `lam` layer may have a non-zero nugget `g` which governs noise for the latent noise layer. `tau2_y` and `tau2_lam` control the amplitude of the mean and noise process respectively. In Matern covariance, `v` governs smoothness.

Usage

```
bhetGP(
  x = NULL,
  y = NULL,
  reps_list = NULL,
  nmcmc = 1000,
  sep = TRUE,
  inits = NULL,
  priors = NULL,
  reps = TRUE,
  cov = c("exp2", "matern", "ARD matern"),
  v = 2.5,
  stratergy = c("default", "flat"),
  vecchia = FALSE,
  m = min(25, length(y) - 1),
  ordering = NULL,
  verb = TRUE,
  omp_cores = 4
)
```

Arguments

`x`	vector or matrix of input locations
`y`	vector of response values
`reps_list`	list object from hetGP::find_reps
`nmcmc`	number of MCMC iterations
`sep`	logical indicating whether to fit isotropic GP (`sep = FALSE`) or seperable GP (`sep = TRUE`)
`inits`	set initial values for hyparameters: `llam`, `theta_y`, `theta_lam`, `g`, `mean_y`, `mean_lam`, `scale_y`, `scale_lam`. Additionally, set initial conditions for tuning: • `theta_check`: logical; if `theta_check = TRUE`, then ensures that theta_lam > theta_y i.e., decay of correlation for noise process is slower than mean process.

- prof_ll_lam: logical; if prof_ll_lam = TRUE, infers tau2_lam i.e., scale parameter for latent noise process
- noise: logical; if noise = TRUE, infers nugget g throught M-H for latent noise process.

priors	hyperparameters for priors and proposals (see details)
reps	logical; if reps = TRUE uses Woodbury inference adjusting for replication of design points and reps = FALSE does not use Woodbury inference
cov	covariance kernel, either Matern, ARD Matern or squared exponential ("exp2")
v	Matern smoothness parameter (only used if cov = "matern")
stratergy	choose initialization stratergy; "default" uses hetGP for vecchia = FALSE settings and sVecchia for vecchia = TRUE. See details.
vecchia	logical indicating whether to use Vecchia approximation
m	size of Vecchia conditioning sets (only used if vecchia = TRUE)
ordering	optional ordering for Vecchia approximation, must correspond to rows of x, defaults to random, is applied to x (only used if vecchia = TRUE)
verb	logical indicating whether to print progress
omp_cores	logical; if vecchia = TRUE, user may specify the number of cores to use for OpenMP parallelization. Uses min(4, limit) where limit is max openMP cores available on the machine.

Details

Maps inputs x to mean response y and noise levels llam. Conducts sampling of the latent noise process using Elliptical Slice sampling. Utilizes Metropolis Hastings sampling of the length scale and nugget parameters with proposals and priors controlled by priors. g for the noise process is set to a specific value, and by default, is not estimated. When vecchia = TRUE, all calculations leverage the Vecchia approximation with specified conditioning set size m. tau2_y is always inferred from likelihood; tau2_lam is inferred by default but may be pre-specified and fixed.

NOTE on OpenMP: The Vecchia implementation relies on OpenMP parallelization for efficient computation. This function will produce a warning message if the package was installed without OpenMP (this is the default for CRAN packages installed on Apple machines). To set up OpenMP parallelization, download the package source code and install using the gcc/g++ compiler.

Proposals for g and theta follow a uniform sliding window scheme, e.g.

theta_star <- runif(1, l * theta_t / u, u * theta_t / l),

with defaults l = 1 and u = 2 provided in priors. To adjust these, set priors = list(l = new_l, u = new_u). Priors on g, theta_y, and theta_lam follow Gamma distributions with shape parameters (alpha) and rate parameters (beta) controlled within the priors list object. Defaults are

- priors$alpha$theta_lam <- 1.5
- priors$beta$theta_lam <- 4
- priors$alpha$theta_y <- 1.5
- priors$beta$theta_y <- 4
- priors$alpha$g <- 1.5

- priors$beta$g <- 4

tau2_y and tau2_lam are not sampled; rather directly inferred under conjugate Inverse Gamma prior with shape (alpha) and scale parameters (beta) within the priors list object

- priors$a$tau2_y <- 10
- priors$a$tau2_y <- 4
- priors$a$tau2_lam <- 10
- priors$a$tau2_lam <- 4

These priors are designed for x scaled to [0, 1] and y having mean mean_y. These may be adjusted using the priors input.

Initial values for theta_y, theta_lam, llam may be specified by the user. If no initial values are specified, stratergy will determine the initialization method. stratergy = "default" leverages mleHetGP for initial values of hyper-parameters if vecchia = FALSE and Scaled-Vecchia with Stochastic Kriging (Sk-Vec) hybrid approach if vecchia = TRUE.

For SK-Vec hybrid approach, scaled Vecchia code from https://github.com/katzfuss-group/scaledVecchia/blob/master/vecchia_scaled.R is used to fit two GPs using the Vecchia approximation. The first for (x, y) pairs, which result in estimated residual sums of squares based on predicted y values. Another GP on (x, s) to obtain latent noise estimates which are smoothed. A script is leveraged internally within this package that fits this method.

Optionally, choose stratergy = "flat" which which will start at uninformative initial values; llam = log(var(y) * 0.1) or specify initial values.

The output object of class bhetgp or bhetgp_vec is designed for use with trim, predict, and plot.

Value

a list of the S3 class bhetgp or bhetgp_vec with elements:

- x: copy of input matrix
- y: copy of response mean at inputs (x)
- Ylist: list of all responses observed per location (x)
- A: number of replicates at each location
- nmcmc: number of MCMC iterations
- priors: copy of proposal/priors
- settings: setting for predictions using bhetgp or bhetgp_vec object
- theta_y: vector of MCMC samples for theta_y (length scale of mean process)
- theta_lam: matrix of MCMC samples for theta_lam (length scale of latent noise process)
- llam_samples: matrix of ESS samples for log lambda (latent noise process samples)
- g: vector of MCMC samples for g if infered
- tau2: vector of MAP estimates for tau2 (scale parameter of mean process)
- tau2_lam: vector of MAP estimates for tau2_lam (scale parameter of latent noise process)
- llik_y: vector of MVN log likelihood of Y for reach Gibbs iteration

- `llik_lam`: vector of MVN log likelihood of `llam` i.e. the latent noise process for reach Gibbs iteration
- `x_approx`: conditioning set, NN and ordering for `vecchia = TRUE`
- `m`: copy of size of conditioning set for `vecchia = TRUE`
- `time`: computation time in seconds

References

Binois, Mickael, Robert B. Gramacy, and Mike Ludkovski. "Practical heteroscedastic Gaussian process modeling for large simulation experiments." Journal of Computational and Graphical Statistics 27.4 (2018): 808-821.

Katzfuss, Matthias, Joseph Guinness, and Earl Lawrence. "Scaled Vecchia approximation for fast computer-model emulation." SIAM/ASA Journal on Uncertainty Quantification 10.2 (2022): 537-554.

Sauer, Annie Elizabeth. "Deep Gaussian process surrogates for computer experiments." (2023).

Examples

```
# 1D function with 1D noise

# Truth
fx <- function(x){
result <- (6 * x - 2)^2* sin(12 * x - 4)
}

# Noise
rx <- function(x){
result <- (1.1 + sin(2 * pi * x))^2
return(result)
}

# Training data
r <- 10 # replicates
xn <- seq(0, 1, length = 25)
x <- rep(xn, r)

rn <- drop(rx(x))
noise <- as.numeric(t(mvtnorm::rmvnorm(r, sigma = diag(rn, length(xn)))))

f <- fx(x)
y <- f + noise

# Testing data
xx <- seq(0, 1, length = 100)
yy <- fx(xx)

#-------------------------------------------------------------------------
# Example 1: Full model, no Vecchia
#-------------------------------------------------------------------------
```

```
# Fitting a bhetGP model using all the data
fit <- bhetGP(x, y, nmcmc = 100, verb = FALSE)

# Trimming the object to remove burn in and thin samples
fit <- trim(fit, 50, 10)

# Predition using the bhetGP object (indepedent predictions)
fit <- predict(fit, xx, cores = 2)

# Visualizing the mean predictive surface.
# Can run plot(fit, trace = TRUE) to view trace plots
plot(fit)


#------------------------------------------------------------------------
# Example 2: Vecchia approximated model
#------------------------------------------------------------------------

# Fitting a bhetGP model with vecchia approximation. Two cores for OpenMP
fit <- bhetGP(x, y, nmcmc = 100, vecchia = TRUE, m = 5, omp_cores = 2, verb = FALSE)

# Trimming the object to remove burn in and thin samples
fit <- trim(fit, 50, 10)

# Predition using the bhetGP_vec object with joint predictions (lite = FALSE)
# Two cores for OpenMP, default setting (omp_cores = 2). No SNOW
fit <- predict(fit, xx, lite = FALSE, vecchia = TRUE)

# Visualizing the mean predictive surface
plot(fit)


#------------------------------------------------------------------------
# Example 3: Vecchia inference, non-vecchia predictions
#------------------------------------------------------------------------

# Fitting a bhetGP model with vecchia approximation. Two cores for OpenMP
fit <- bhetGP(x, y, nmcmc = 200, vecchia = TRUE, m = 5, omp_cores = 2)

# Trimming the object to remove burn in and thin samples
fit <- trim(fit, 100, 10)

# Predition using the bhetGP object with joint predictions (lite = FALSE)
# Two cores for OpenMP which is default setting (omp_cores = 2)
# Two cores for SNOW (cores = 2)
fit <- predict(fit, xx, vecchia = FALSE, cores = 2, lite = FALSE)

# Visualizing the mean predictive surface
plot(fit)
```

bhomGP *MCMC sampling for Homoskedastic GP*

Description

Conducts MCMC sampling of hyperparameters for a homGP. Separate length scale parameters `theta_y` govern the correlation strength of the response. `g` governs noise for the noise. `tau2_y` control the amplitude of the mean process. In Matern covariance, `v` governs smoothness.

Usage

```
bhomGP(
  x = NULL,
  y = NULL,
  reps_list = NULL,
  nmcmc = 1000,
  sep = TRUE,
  inits = NULL,
  priors = NULL,
  cov = c("exp2", "matern", "ARD matern"),
  v = 2.5,
  stratergy = c("default", "flat"),
  vecchia = FALSE,
  m = min(25, length(y) - 1),
  ordering = NULL,
  reps = TRUE,
  verb = TRUE,
  omp_cores = 4
)
```

Arguments

`x`	vector or matrix of input locations
`y`	vector of response values
`reps_list`	list object from hetGP::find_reps
`nmcmc`	number of MCMC iterations
`sep`	logical indicating whether to fit isotropic GP (`sep = FALSE`) or seperable GP (`sep = TRUE`)
`inits`	set initial values for hyparameters: `theta_y`, `g`, `mean_y`, `scale_y`, Additionally, set initial conditions for tuning: • `prof_ll`: logical; if `prof_ll = TRUE`, infers tau2_y i.e., scale parameter for homGP. • `noise`: logical; if `noise = TRUE`, infers nugget `g` throught M-H for latent noise process.
`priors`	hyperparameters for priors and proposals (see details)

cov	covariance kernel, either Matern, ARD Matern or squared exponential ("exp2")
v	Matern smoothness parameter (only used if cov = "matern")
stratergy	choose initialization stratergy; "default" uses hetGP for vecchia = FALSE settings and sVecchia for vecchia = TRUE. See details.
vecchia	logical indicating whether to use Vecchia approximation
m	size of Vecchia conditioning sets (only used if vecchia = TRUE)
ordering	optional ordering for Vecchia approximation, must correspond to rows of x, defaults to random, is applied to x
reps	logical; if reps = TRUE uses Woodbury inference adjusting for replication of design points and reps = FALSE does not use Woodbury inference
verb	logical indicating whether to print progress
omp_cores	if vecchia = TRUE, user may specify the number of cores to use for OpenMP parallelization. Uses min(4, limit) where limit is max openMP cores available on the machine.

Details

Maps inputs x to mean response y. Utilizes Metropolis Hastings sampling of the length scale and nugget parameters with proposals and priors controlled by priors. g is estimated by default but may be specified and fixed. When vecchia = TRUE, all calculations leverage the Vecchia approximation with specified conditioning set size m. tau2_y is inferred by default but may be pre-specified and fixed.

NOTE on OpenMP: The Vecchia implementation relies on OpenMP parallelization for efficient computation. This function will produce a warning message if the package was installed without OpenMP (this is the default for CRAN packages installed on Apple machines). To set up OpenMP parallelization, download the package source code and install using the gcc/g++ compiler.

Proposals for g and theta follow a uniform sliding window scheme, e.g.

theta_star <- runif(1, l * theta_t / u, u * theta_t / l),

with defaults l = 1 and u = 2 provided in priors. To adjust these, set priors = list(l = new_l, u = new_u). Priors on g, theta_y follow Gamma distributions with shape parameters (alpha) and rate parameters (beta) controlled within the priors list object. Defaults are

- priors$alpha$theta_lam <- 1.5
- priors$beta$theta_lam <- 4
- priors$alpha$theta_y <- 1.5
- priors$beta$theta_y <- 4
- priors$alpha$g <- 1.5
- priors$beta$g <- 4

tau2_y is not sampled; rather directly inferred under conjugate Inverse Gamma prior with shape (alpha) and scale parameters (beta) within the priors list object

- priors$a$tau2_y <- 10
- priors$a$tau2_y <- 4

These priors are designed for x scaled to [0, 1] and y having mean mean_y. These may be adjusted using the priors input.

Initial values for theta_y, and g may be specified by the user. If no initial values are specified, stratergy will determine the initialization method. stratergy = "default" leverages mleHomGP for initial values of hyper-parameters if vecchia = FALSE and scaled vecchia approach if vecchia = TRUE.

For scaled Vecchia code from https://github.com/katzfuss-group/scaledVecchia/blob/master/vecchia_scaled.R is used to fit a vecchia approximated GP to (x, y). A script is leveraged internally within this package that fits this method.

Optionally, choose stratergy = "flat" which will start at uninformative initial values or specify initial values.

The output object of class bhomgp or bhomgp_vec is designed for use with trim, predict, and plot.

Value

a list of the S3 class bhomgp or bhomgp_vec with elements:

- x: copy of input matrix
- y: copy of response vector
- Ylist: list of all responses observed per location (x)
- A: number of replicates at each location
- nmcmc: number of MCMC iterations
- priors: copy of proposal/prior settings
- settings: setting for predictions using bhetgp or bhetgp_vec object
- g_y: vector of MCMC samples for g_y
- theta_y: vector of MCMC samples for theta_y (length scale of mean process)
- tau2: vector of MAP estimates for tau2 (scale parameter of outer layer)
- llik_y: vector of MVN log likelihood of Y for reach Gibbs iteration
- time: computation time in seconds
- x_approx: conditioning set, NN and ordering for vecchia = TRUE
- m: copy of size of conditioning set for vecchia = TRUE

References

Binois, Mickael, Robert B. Gramacy, and Mike Ludkovski. "Practical heteroscedastic Gaussian process modeling for large simulation experiments." Journal of Computational and Graphical Statistics 27.4 (2018): 808-821.

Katzfuss, Matthias, Joseph Guinness, and Earl Lawrence. "Scaled Vecchia approximation for fast computer-model emulation." SIAM/ASA Journal on Uncertainty Quantification 10.2 (2022): 537-554.

Sauer, Annie Elizabeth. "Deep Gaussian process surrogates for computer experiments." (2023).

Examples

```
# 1D example with constant noise

# Truth
fx <- function(x){
result <- (6 * x - 2)^2* sin(12 * x - 4)
}

# Training data
r <- 10
xn <- seq(0, 1, length = 25)
x <- rep(xn, r)

f <- fx(x)
y <- f + rnorm(length(x)) # adding constant noise

# Testing data
xx <- seq(0, 1, length = 100)
yy <- fx(xx)

# Example 1: Full model, no Vecchia

# Fitting a bhomGP model using all the data
fit <- bhomGP(x, y, nmcmc = 100, verb = FALSE)

# Trimming the object to remove burn in and thin samples
fit <- trim(fit, 50, 10)

# Predition using the bhomGP object (indepedent predictions)
fit <- predict(fit, xx, lite = TRUE, cores = 2)

#' # Visualizing the mean predictive surface.
# Can run plot(fit, trace = TRUE) to view trace plots
plot(fit)

# Example 2: Vecchia approximated model

# Fitting a bhomGP model using vecchia approximation
fit <- bhomGP(x, y, nmcmc = 100, vecchia = TRUE, m = 5, omp_cores = 2, verb = FALSE)

# Trimming the object to remove burn in and thin samples
fit <- trim(fit, 50, 10)

# Predition using the bhomGP_vec object with Vecchia (indepedent predictions)
fit <- predict(fit, xx, vecchia = TRUE, cores = 2)

# Visualizing the mean predictive surface.
plot(fit)
```

plot *Plots object from* bhetGP *package*

Description

Acts on a bhetgp, bhetgp_vec, bhomgp or, bhomgp_vec object. Generates trace plots for log likelihood of mean and noise process, length scales of corresponding processes, scale parameters and the nuggets. Generates plots of hidden layers for one-dimensional inputs. Generates plots of the posterior mean and estimated 90% prediction intervals for one-dimensional inputs; generates heat maps of the posterior mean and point-wise variance for two-dimensional inputs.

Usage

```
## S3 method for class 'bhetgp'
plot(x, trace = NULL, predict = NULL, verb = TRUE, ...)

## S3 method for class 'bhomgp'
plot(x, trace = NULL, predict = NULL, verb = TRUE, ...)

## S3 method for class 'bhetgp_vec'
plot(x, trace = NULL, predict = NULL, verb = TRUE, ...)

## S3 method for class 'bhomgp_vec'
plot(x, trace = NULL, predict = NULL, verb = TRUE, ...)
```

Arguments

x	object of class bhetgp, bhetgp_vec, bhomgp, or bhomgp_vec
trace	logical indicating whether to generate trace plots (default is TRUE if the object has not been through predict)
predict	logical indicating whether to generate posterior predictive plot (default is TRUE if the object has been through predict)
verb	logical indicating whether to print plot.
...	N/A

Details

Trace plots are useful in assessing burn-in. If there are too many hyperparameters to plot them all, then it is most useful to visualize the log likelihood (e.g., plot(fit$ll, type = "l")).

Value

...N/A

predict *Predict posterior mean and variance/covariance*

Description

Acts on a bhetgp, bhetgp_vec, bhomgp or, bhomgp_vec object. Calculates posterior mean and variance/covariance over specified input locations. Optionally utilizes SNOW parallelization.

Usage

```
## S3 method for class 'bhetgp'
predict(
  object,
  x_new,
  lite = TRUE,
  return_all = FALSE,
  interval = c("pi", "ci", "both"),
  lam_ub = TRUE,
  cores = 1,
  samples = TRUE,
  ...
)

## S3 method for class 'bhetgp_vec'
predict(
  object,
  x_new,
  lite = TRUE,
  return_all = FALSE,
  interval = c("pi", "ci", "both"),
  lam_ub = TRUE,
  vecchia = FALSE,
  m = object$m,
  ordering_new = NULL,
  cores = 1,
  omp_cores = 2,
  samples = TRUE,
  ...
)

## S3 method for class 'bhomgp'
predict(
  object,
  x_new,
  lite = TRUE,
  return_all = FALSE,
  interval = c("pi", "ci", "both"),
```

```
  cores = 1,
  samples = TRUE,
  ...
)

## S3 method for class 'bhomgp_vec'
predict(
  object,
  x_new,
  lite = TRUE,
  return_all = FALSE,
  interval = c("pi", "ci", "both"),
  vecchia = FALSE,
  m = object$m,
  ordering_new = NULL,
  cores = 1,
  omp_cores = 2,
  samples = TRUE,
  ...
)
```

Arguments

object	object from bhetGP or bhomGP, with burn-in already removed
x_new	matrix of predictive input locations
lite	logical indicating whether to calculate only point-wise variances (lite = TRUE) or full covariance (lite = FALSE)
return_all	logical indicating whether to return mean and point-wise variance prediction for ALL samples (only available for lite = TRUE)
interval	returns predictive variances by default interval = "pi". interval = "ci" returns variances for only mean process and interval = "both" returns both variances.
lam_ub	logical uses upper 95 quantile for latent noise to obtain predictive variances for the response. If lam_ub = FALSE, the mean latent noise is used for inference.
cores	number of cores to utilize for SNOW parallelization
samples	logical indicating if you want all posterior samples returned including latent layer.
...	N/A
vecchia	logical uses vecchia approximation for prediction if vecchia = TRUE.
m	size of Vecchia conditioning sets (only for fits with vecchia = TRUE), defaults to the m used for MCMC
ordering_new	optional ordering for Vecchia approximation, must correspond to rows of x_new, defaults to random, is applied to all layers in deeper models.
omp_cores	sets cores used for OpenMP if vechhia = TRUE and lite = FALSE.

Details

All iterations in the object are used for prediction, so samples should be burned-in. Thinning the samples using `trim` will speed up computation. Posterior moments are calculated using conditional expectation and variance. As a default, only point-wise variance is calculated. Full covariance may be calculated using `lite = FALSE`.

The posterior predictive variances are returned by default. The variance for the mean process may be obtained by specifying `interval = "ci"`. `interval = "both"` will return both variances.

SNOW parallelization reduces computation time but requires more memory storage.

Value

object of the same class with the following additional elements:

- `x_new`: copy of predictive input locations
- `m_pred`: size of predictive conditioning set if `vecchia = TRUE`
- `mean_y`: predicted posterior mean, indices correspond to `x_new` locations
- `s2_y`: predicted point-wise variances, indices correspond to `x_new` locations (only returned when `lite = TRUE` & `interval = c("pi", "both")`)
- `s2_y_ci`: predicted point-wise variances for the mean process, indices correspond to `x_new` locations (only returned when `lite = TRUE` & `interval = c("ci", "both")`)
- `mean_all`: predicted posterior mean for each sample (column indices), only returned when `return_all = TRUE`
- `s2_all` predicted point-wise variances each sample (column indices), only returned when `return-all = TRUE` & `interval = c("pi", "both")`
- `s2_all_ci` predicted point-wise variances for each sample (column indices), only returned when `return-all = TRUE` & `interval = c("ci", "both")`
- `Sigma`: predicted posterior covariance, indices correspond to `x_new` locations (only returned when `lite = FALSE` & `interval = c("pi", "both")`)
- `Sigma_ci`: predicted posterior covariance for mean process, indices correspond to `x_new` locations (only returned when `lite = FALSE` & `interval = c("ci", "both")`)

Additionally, if object belongs to class `bhetGP` or `bhetGP_vec`, the log-noise process is also predicted for new locations `x_new`. The following are returned:

- `mean_lnugs`: predicted posterior mean for log noise process, indices correspond to `x_new` locations
- `s2_lnugs`: predicted point-wise variances for log noise process, indices correspond to `x_new` locations (only returned when `lite = TRUE` & `interval = c("pi", "both")`)
- `s2_lnugs_ci`: predicted point-wise variances for the log noise process, indices correspond to `x_new` locations (only returned when `lite = TRUE` & `interval = c("ci", "both")`)
- `mean_lnugs_all`: predicted posterior mean for each sample for log noise process (column indices), only returned when `return_all = TRUE`
- `s2_lnugs_all` predicted point-wise variances each sample (column indices) for log noise process, only returned when `return-all = TRUE` & `interval = c("pi", "both")`

- s2_lnugs_all_ci predicted point-wise variances for each sample (column indices) for log noise process, only returned when return-all = TRUE & interval = c("ci", "both")
- Sigma_lnugs: predicted posterior covariance for log noise process, indices correspond to x_new locations (only returned when lite = FALSE & interval = c("pi", "both"))
- Sigma_lnugs_ci: predicted posterior covariance for log noise process, indices correspond to x_new locations (only returned when lite = FALSE & interval = c("ci", "both"))

Computation time is added to the computation time of the existing object.

---

trim *Trim/Thin MCMC iterations*

Description

Acts on a bhetgp, bhetgp_vec, bhomgp, or bhomgp_vec object. Removes the specified number of MCMC iterations (starting at the first iteration). After these samples are removed, the remaining samples are optionally thinned.

Usage

```
trim(object, burn, thin)
```

Arguments

object	object from bhetGP, or bhomGP
burn	integer specifying number of iterations to cut off as burn-in
thin	integer specifying amount of thinning (thin = 1 keeps all iterations, thin = 2 keeps every other iteration, thin = 10 keeps every tenth iteration, etc.)

Details

The resulting object will have nmcmc equal to the previous nmcmc minus burn divided by thin. Removing burn-ins are necessary following convergence. Thinning is recommended as it can eliminate highly correlated consecutive samples. Additionally, the size of the object reduces and ensures faster prediction.

Value

object of the same class with the selected iterations removed

Index