Package 'geommc'

May 8, 2026

Title Geometric Markov Chain Sampling

Type Package

Version 1.3.1

Maintainer Vivekananda Roy <vroy@iastate.edu>

Date 2026-2-27

Description Simulates from discrete and continuous target distributions using geometric Metropolis-Hastings (MH) algorithms. Users specify the target distribution by an R function that evaluates the log un-normalized pdf or pmf. The package also contains a function implementing a specific geometric MH algorithm for performing high-dimensional Bayesian variable selection.

Encoding UTF-8

RoxygenNote 7.3.3

Imports Rcpp, cubature, Matrix, numDeriv, progress

LinkingTo Rcpp, RcppArmadillo

URL https://github.com/vroys/geommc

License GPL (>= 3)

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Author Vivekananda Roy [aut, cre] (ORCID:
<https://orcid.org/0000-0002-2964-9503>)

Repository CRAN

Date/Publication 2026-02-27 18:30:02 UTC

Contents

geomc *Markov chain Monte Carlo for discrete and continuous distributions using geometric MH algorithms.*

Description

geomc produces Markov chain samples from a user-defined target, which may be either a probability density function (pdf) or a probability mass function (pmf). The target is provided as an R function returning the log of its unnormalized pdf or pmf.

Usage

```
geomc(
  logp,
  initial,
  n.iter,
  eps = 0.5,
  ind = FALSE,
  gaus = TRUE,
  imp = list(enabled = FALSE, n.samp = 100, samp.base = TRUE),
  a = 1,
  show.progress = TRUE,
  ...
)
```

Arguments

`logp` Either a function that evaluates the logarithm of the un-normalized target density (pdf or pmf) to sample from, or a list containing at least one element named `log.target`. The list may optionally include any of the following elements specifying additional functions for the base and approximate target densities: `mean.base`, `var.base`, `dens.base`, `samp.base`, `mean.ap.tar`, `var.ap.tar`, `dens.ap.tar`, `samp.ap.tar`, and `bhat.coef`. See below for details of these functions. Any optional elements not provided are treated as missing, and default behavior is applied.

- `log.target` A function that evaluates the logarithm of the un-normalized target density (pdf or pmf). Its first argument must be the current state vector x, and it must return a numeric scalar. If the target density requires additional parameters, the user-supplied `log.target` must be written to accept them through `...`
- `mean.base` is the mean of the base density f, needs to be written as a function of the current state x.
- `var.base` is the covariance matrix of the base density f, needs to be written as a function of the current state x.
- `dens.base` is the density function of the base density f, needs to be written as a function (y, x) (in this order) of the proposed state y and the current state x, although it may not depend on x.

- `samp.base` is the function to draw from the base density f, needs to be written as a function of the current state x.
- `mean.ap.tar` is the vector of means of the densities $g_i(y|x), i = 1, \ldots, k$. It needs to be written as a function of the current state x. It must have the same dimension as k times the length of initial.
- `var.ap.tar` is the matrix of covariance matrices of the densities $g_i(y|x), i = 1, \ldots, k$ formed by column concatenation. It needs to be written as a function of the current state x. It must have the same dimension as the length of initial by k times the length of initial.
- `dens.ap.tar` is the vector of densities $g_i(y|x), i = 1, \ldots, k$. It needs to be written as a function (y, x) (in this order) of the proposed state y and the current state x, although it may not depend on x.
- `samp.ap.tar` is the function to draw from the densities $g_i(y|x), i = 1, \ldots, k$. It needs to be written as a function of (current state x, the indicator of component kk). It must return a vector of the length of that of the initial.
- `bhat.coef` is the vector of Bhattacharyya coefficients $\langle \sqrt{f(\cdot|x)}, \sqrt{g_i(\cdot|x)} \rangle, i = 1, \ldots, k$. It needs to be written as a function of the current state x. When `gaus= TRUE`, the `bhat.coef` argument is ignored and the built-in function is used.

`initial` is the initial state.

`n.iter` is the no. of samples needed.

`eps` is the value for epsilon perturbation. Default is 0.5.

`ind` is False if either the base density, f or the approximate target density, g depends on the current state x. Default is False.

`gaus` is True if both f and g are normal distributions. Default is True.

`imp` A list of three elements controlling optional importance sampling. This list has components:

- `enabled` Logical. If `FALSE` (default), numerical integration is used to compute the Bhattacharyya coefficient $\langle \sqrt{f}, \sqrt{g_i} \rangle$. If `TRUE`, importance sampling is used instead.
- `n.samp` A positive integer giving the number of Monte Carlo samples used when `enabled = TRUE`.
- `samp.base` Logical. If `TRUE`, the samples in the importance sampler are drawn from the base density f; otherwise they are drawn from the approximate target density g. Default is `FALSE`.

When `gaus = TRUE`, the `imp` argument is ignored. Also, when `bhat.coef` is provided, the `imp` argument is ignored.

`a` is the probability vector for the mixture proposal density. Default is the uniform distribution.

`show.progress` Logical. Whether to show progress during sampling. Default: TRUE.

`...` additional arguments passed to `log.target`

Details

The function geomc implements the geometric approach of Roy (2024) to move an initial (possibly uninformed) base density f toward one or more approximate target densities $g_i, i = 1, \ldots, k$, thereby constructing efficient proposal distributions for MCMC. The details of the method can be found in Roy (2024).

The base density f can be user-specified through its mean, covariance matrix, density function, and sampling function. One or more approximate target densities $g_i, i = 1, \ldots, k$ can also be supplied. Just like the base density, each approximate target may be specified through its mean, covariance, density, and sampling functions.

A Gaussian (f or g_i) density must be specified in terms of its mean and covariance; optional density and sampling functions may also be supplied.

Non-Gaussian densities, including discrete pmfs for either the base or approximate target, are specified solely via their density and sampling functions. If for either the base or the approximate target, the user specifies both a density function and a sampling function but omits either the mean function or the covariance function, then `gaus = FALSE` is automatically enforced.

If either or both of the mean and variance arguments and any of the density and sampling functions is omitted for the base density, geomc automatically constructs a base density corresponding to a random-walk proposal with an appropriate scale. If either or both of the mean and variance arguments and any of the density and sampling functions is missing for the approximate target density, then geomc automatically constructs a diffuse multivariate normal distribution as the approximate target.

If the target distribution is a pmf (i.e., a discrete distribution) then the user must provide the base pmf f and one or more g_i's. The package vignette `vignette(package="geommc")` provides several useful choices for f and g_i. In addition, the user must set `gaus=FALSE` and either supply `bhat.coef` or set `imp$enabled=TRUE` (neither of which is the default for `gaus` or `imp`). This ensures that the algorithm uses either the user defined `bhat.coef` function or the importance sampling, rather than numerical integration or closed-form Gaussian expressions, for computing the Bhattacharyya coefficient $\langle \sqrt{f}, \sqrt{g_i} \rangle$ for discrete distributions.

For a discussion and several illustrative examples of the geomc function, see the package vignette `vignette(package="geommc")`.

Value

The function returns a list with the following components:

`samples`	A matrix containing the MCMC samples. Each row is one sample.
`acceptance.rate`	The Metropolis-Hastings acceptance rate.
`log.p`	A vector with the logarithm of the (un-normalized) target density for each MCMC sample.
`gaus`	The value of the logical gaus.
`ind`	The value of the logical ind.
`model.case`	Describes whether default settings are used for both functions f and g, for only one of them, or for neither.

var.base	The default variance used for the random-walk base density if not provided by the user.
mean.ap.tar	The default mean vector used for the approximate target density if not provided by the user.
var.ap.tar	The default covariance matrix used for the approximate target density if not provided by the user.

Author(s)

Vivekananda Roy vroy@iastate.edu

References

Roy, V.(2024) A geometric approach to informative MCMC sampling https://arxiv.org/abs/2406.09010

Examples

```
#target is multivariate normal, sampling using geomc with default choices
log_target_mvnorm <- function(x, target.mean, target.Sigma) {d  <- length(x)
xc <- x - target.mean; Q  <- solve(target.Sigma); -0.5 * drop(t(xc) %*% Q %*% xc)}
result <- geomc(logp=log_target_mvnorm,initial= c(0, 0),n.iter=500, target.mean=c(1, -2),
               target.Sigma=matrix(c(1.5, 0.7,0.7, 2.0), 2, 2))
               # additional arguments passed via ...
result$samples # the MCMC samples.
result$acceptance.rate # the acceptance rate.
result$log.p # the value of logp at the MCMC samples.
#Additional returned values are
result$var.base; result$mean.ap.tar; result$var.ap.tar; result$model.case; result$gaus; result$ind
#target is the posterior of (\eqn{\mu}, \eqn{\sigma^2}) with iid data from
#N(\eqn{\mu}, \eqn{\sigma^2}) and N(mu0, \eqn{tau0^2}) prior on \eqn{\mu}
#and an inverse-gamma (alpha0, beta0) prior on \eqn{\sigma^2}
log_target <- function(par, x, mu0, tau0, alpha0, beta0) {
mu  <- par[1];sigma2 <- par[2]
if (sigma2 <= 0) return(-Inf)
n  <- length(x); SSE <- sum((x - mu)^2)
val <- -(n/2) * log(sigma2) - SSE / (2 * sigma2) - (mu - mu0)^2 / (2 * tau0^2)
-(alpha0 + 1) * log(sigma2) -beta0 / sigma2
return(val)}
# sampling using geomc with default choices
result=geomc(logp=log_target,initial=c(0,1),n.iter=1000,x=1+rnorm(100),mu0=0,
tau0=1,alpha0=2.01,beta0=1.01)
colMeans(result$samples)
#target is mixture of univariate normals, sampling using geomc with default choices
set.seed(3);result<-geomc(logp=list(log.target=function(y)
log(0.5*dnorm(y)+0.5*dnorm(y,mean=10,sd=1.4))),
initial=0,n.iter=1000)
hist(result$samples)
#target is mixture of univariate normals, sampling using geomc with a random walk base density,
# and an informed choice for dens.ap.tar
result<-geomc(logp=list(log.target=function(y) log(0.5*dnorm(y)+0.5*dnorm(y,mean=10,sd=1.4)),
mean.base = function(x) x,
```

```
var.base= function(x) 4, dens.base=function(y,x) dnorm(y, mean=x,sd=2),
samp.base=function(x) x+2*rnorm(1),
mean.ap.tar=function(x) c(0,10),var.ap.tar=function(x) c(1,1.4^2),
dens.ap.tar=function(y,x) c(dnorm(y),dnorm(y,mean=10,sd=1.4)),
samp.ap.tar=function(x,kk=1){if(kk==1){return(rnorm(1))} else{return(10+1.4*rnorm(1))}}),
initial=0,n.iter=500)
hist(result$samples)
#target is mixture of univariate normals, sampling using geomc with a random walk base density,
# and another informed choice for dens.ap.tar
samp.ap.tar=function(x,kk=1){s.g=sample.int(2,1,prob=c(.5,.5))
if(s.g==1){return(rnorm(1))
}else{return(10+1.4*rnorm(1))}}
result<-geomc(logp=list(log.target=function(y) log(0.5*dnorm(y)+0.5*dnorm(y,mean=10,sd=1.4)),
dens.base=function(y,x) dnorm(y, mean=x,sd=2),samp.base=function(x) x+2*rnorm(1),
dens.ap.tar=function(y,x) 0.5*dnorm(y)+0.5*dnorm(y,mean=10,sd=1.4), samp.ap.tar=samp.ap.tar),
initial=0,n.iter=500,gaus=FALSE,imp=list(enabled=TRUE,n.samp=100,samp.base=TRUE))
hist(result$samples)
#target is mixture of bivariate normals, sampling using geomc with random walk base density,
# and an informed choice for dens.ap.tar
log_target_mvnorm_mix <- function(x, mean1, Sigma1, mean2, Sigma2) {
return(log(0.5*exp(geommc:::ldens_mvnorm(x, mean1,Sigma1))+
0.5*exp(geommc:::ldens_mvnorm(x,mean2,Sigma2))))}
result <- geomc(logp=list(log.target=log_target_mvnorm_mix, mean.base = function(x) x,
var.base= function(x) 2*diag(2), mean.ap.tar=function(x) c(0,0,10,10),
var.ap.tar=function(x) cbind(diag(2),2*diag(2))),initial= c(5, 5),n.iter=500, mean1=c(0, 0),
Sigma1=diag(2), mean2=c(10, 10), Sigma2=2*diag(2))
colMeans(result$samples)
#While the geomc with random walk base successfully moves back and forth between the two modes,
# the random walk Metropolis is trapped in a mode, as run below
result<-geommc:::rwm(log_target= function(x) log_target_mvnorm_mix(x,c(0, 0), diag(2),
c(10, 10), 2*diag(2)), initial= c(5, 5), n_iter = 500, sig = 2,return_sample = TRUE)
colMeans(result$samples)
 #target is binomial, sampling using geomc
size=5
result <- geomc(logp=list(log.target = function(y) dbinom(y, size, 0.3, log = TRUE),
dens.base=function(y,x) 1/(size+1), samp.base= function(x) sample(seq(0,size,1),1),
dens.ap.tar=function(y,x) dbinom(y, size, 0.7),samp.ap.tar=function(x,kk=1) rbinom(1, size, 0.7)),
initial=0,n.iter=500,ind=TRUE,gaus=FALSE,imp=list(enabled=TRUE,n.samp=1000,samp.base=TRUE))
 table(result$samples)
```

`geomc.vs` *Markov chain Monte Carlo for Bayesian variable selection using a geometric MH algorithm.*

Description

geomc.vs uses a geometric approach to MCMC for performing Bayesian variable selection. It produces MCMC samples from the posterior density of a Bayesian hierarchical feature selection model.

Usage

```
geomc.vs(
  X,
  y,
  initial = NULL,
  n.iter = 50,
  burnin = 1,
  eps = 0.5,
  symm = TRUE,
  move.prob = c(0.4, 0.4, 0.2),
  lam0 = 0,
  a0 = 0,
  b0 = 0,
  lam = nrow(X)/ncol(X)^2,
  w = sqrt(nrow(X))/ncol(X),
  model.summary = FALSE,
  model.threshold = 0.5,
  show.progress = TRUE
)
```

Arguments

X	The $n \times p$ covariate matrix without intercept. The following classes are supported: matrix and dgCMatrix. No need to center or scale this matrix manually. Scaling is performed implicitly and regression coefficients are returned on the original scale.
y	The response vector of length n. No need to center or scale.
initial	is the initial model (the set of active variables). Default: Null model.
n.iter	is the no. of samples needed. Default: 50.
burnin	is the value of burnin used to compute the median probability model. Default: 1.
eps	is the value for epsilon perturbation. Default: 0.5.
symm	indicates if the base density is of symmetric RW-MH. Default: True.
move.prob	is the vector of ('addition', 'deletion', 'swap') move probabilities. Default: (0.4,0.4,0.2). move.prob is used only when symm is set to False.
lam0	The precision parameter for β_0. Default: 0 (corresponding to the improper uniform prior).
a0	The shape parameter for prior on σ^2. Default: 0.
b0	The scale parameter for prior on σ^2. Default: 0.
lam	The slab precision parameter. Default: n/p^2.
w	The prior inclusion probability of each variable. Default: $\sqrt{n}/p$.
model.summary	If true, additional summaries are returned. Default: FALSE.

`model.threshold`
: The threshold probability to select the covariates for the median model (`median.model`) and the weighted average model (`wam`). A covariate will be included in median.model (`wam`) if its marginal inclusion probability (weighted marginal inclusion probability) is greater than the threshold. Default: 0.5.

`show.progress`
: Logical. Whether to show progress during sampling. Default: TRUE.

Details

`geomc.vs` provides MCMC samples using the geometric MH algorithm of Roy (2024) for variable selection based on a hierarchical Gaussian linear model with priors placed on the regression coefficients as well as on the model space as follows:

$$y|X, \beta_0, \beta, \gamma, \sigma^2, w, \lambda \sim N(\beta_0 1 + X_\gamma \beta_\gamma, \sigma^2 I_n)$$

$$\beta_i|\beta_0, \gamma, \sigma^2, w, \lambda \overset{indep.}{\sim} N(0, \gamma_i \sigma^2/\lambda),\ i = 1, \ldots, p,$$

$$\beta_0|\gamma, \sigma^2, w, \lambda \sim N(0, \sigma^2/\lambda_0)$$

$$\sigma^2|\gamma, w, \lambda \sim Inv - Gamma(a_0, b_0)$$

$$\gamma_i|w, \lambda \overset{iid}{\sim} Bernoulli(w)$$

where X_γ is the $n \times |\gamma|$ submatrix of X consisting of those columns of X for which $\gamma_i = 1$ and similarly, β_γ is the $|\gamma|$ subvector of β corresponding to γ. The density $\pi(\sigma^2)$ of $\sigma^2 \sim Inv - Gamma(a_0, b_0)$ has the form $\pi(\sigma^2) \propto (\sigma^2)^{-a_0-1} \exp(-b_0/\sigma^2)$. The functions in the package also allow the non-informative prior $(\beta_0, \sigma^2)|\gamma, w \sim 1/\sigma^2$ which is obtained by setting $\lambda_0 = a_0 = b_0 = 0$. `geomc.vs` provides the empirical MH acceptance rate and MCMC samples from the posterior pmf of the models $P(\gamma|y)$, which is available up to a normalizing constant. `geomc.vs` also provides the log-posterior values (up to an additive constant) at the observed MCMC samples. If `model.summary` is set TRUE, `geomc.vs` also returns other model summaries. In particular, it returns the marginal inclusion probabilities (`mip`) computed by the Monte Carlo average as well as the weighted marginal inclusion probabilities (`wmip`) computed with weights

$$w_i = P(\gamma^{(i)}|y) / \sum_{k=1}^{K} P(\gamma^{(k)}|y), i = 1, 2, ..., K$$

where $\gamma^{(k)}, k = 1, 2, ..., K$ are the distinct models sampled. Thus, if N_k is the no. of times the kth distinct model $\gamma^{(k)}$ is repeated in the MCMC samples, the `mip` for the jth variable is

$$mip_j = \sum_{k=1}^{K} N_k I(\gamma_j^{(k)} = 1)/n.iter$$

and `wmip` for the jth variable is

$$wmip_j = \sum_{k=1}^{K} w_k I(\gamma_j^{(k)} = 1).$$

The `median.model` is the model containing variables j with $mip_j >$ `model.threshold` and the wam is the model containing variables j with $wmip_j >$ `model.threshold`. Note that $E(\beta|\gamma, y)$,

the conditional posterior mean of β given a model γ is available in closed form (see Li, Dutta, Roy (2023) for details). `geomc.vs` returns two estimates (beta.mean, beta.wam) of the posterior mean of β computed as

$$beta.mean = \sum_{k=1}^{K} N_k E(\beta|\gamma^{(k)}, y)/n.iter$$

and

$$beta.wam = \sum_{k=1}^{K} w_k E(\beta|\gamma^{(k)}, y),$$

respectively.

For a discussion and some illustrative examples of the `geomc.vs` function, see the package vignette `vignette(package="geommc")`.

Value

A list with components

`samples`	MCMC samples from $P(\gamma\|y)$ returned as a n.iter$\times p$ sparse `lgCMatrix`.
`acceptance.rate`	The acceptance rate based on all samples.
`log.p`	The n.iter vector of log of the unnormalized marginal posterior pmf $P(\gamma\|y)$ evaluated at the samples.
`mip`	The p vector of marginal inclusion probabilities of all variables based on post burnin samples.
`median.model`	The median probability model based on post burnin samples.
`beta.mean`	The Monte Carlo estimate of posterior mean of β (the $p+1$ vector c(intercept, regression coefficients)) based on post burnin samples.
`wmip`	The p vector of weighted marginal inclusion probabilities of all variables based on post burnin samples.
`wam`	The weighted average model based on post burnin samples.
`beta.wam`	The model probability weighted estimate of posterior mean of β (the $p+1$ vector c(intercept, regression coefficients)) based on post burnin samples.

Author(s)

Vivekananda Roy

References

Roy, V.(2024) A geometric approach to informative MCMC sampling https://arxiv.org/abs/2406.09010

Li, D., Dutta, S., Roy, V.(2023) Model Based Screening Embedded Bayesian Variable Selection for Ultra-high Dimensional Settings, Journal of Computational and Graphical Statistics, 32, 61-73

Examples

```
n=50; p=100; nonzero = 3
trueidx <- 1:3
nonzero.value <- 4
TrueBeta <- numeric(p)
TrueBeta[trueidx] <- nonzero.value
rho <- 0.5
xone <- matrix(rnorm(n*p), n, p)
X <- sqrt(1-rho)*xone + sqrt(rho)*rnorm(n)
y <- 0.5 + X %*% TrueBeta + rnorm(n)
result <- geomc.vs(X=X, y=y,model.summary = TRUE)
result$samples # the MCMC samples
result$acceptance.rate #the acceptance.rate
result$mip #marginal inclusion probabilities
result$wmip #weighted marginal inclusion probabilities
result$median.model #the median.model
result$wam #the weighted average model
result$beta.mean #the posterior mean of regression coefficients
result$beta.wam #another estimate of the posterior mean of regression coefficients
result$log.p #the log (unnormalized) posterior probabilities of the MCMC samples.
```

`logp.vs` *The log-unnormalized posterior probability of a model for Bayesian variable selection.*

Description

Calculates the log-unnormalized posterior probability of a model.

Usage

```
logp.vs(model, X, y, lam0 = 0, a0 = 0, b0 = 0, lam, w)
```

Arguments

`model`	The indices of active variables.
`X`	The $n \times p$ covariate matrix without intercept.
`y`	The response vector of length n.
`lam0`	The precision parameter for β_0. Default: 0 (corresponding to the improper uniform prior).
`a0`	The shape parameter for prior on σ^2. Default: 0.
`b0`	The scale parameter for prior on σ^2. Default: 0.
`lam`	The slab precision parameter.
`w`	The prior inclusion probability of each variable.

Value

The log-unnormalized posterior probability of the model.

Author(s)

Vivekananda Roy

References

Roy, V.(2024) A geometric approach to informative MCMC sampling https://arxiv.org/abs/2406.09010

Examples

```
n=50; p=100; nonzero = 3
trueidx <- 1:3
nonzero.value <- 4
TrueBeta <- numeric(p)
TrueBeta[trueidx] <- nonzero.value
rho <- 0.5
xone <- matrix(rnorm(n*p), n, p)
X <- sqrt(1-rho)*xone + sqrt(rho)*rnorm(n)
y <- 0.5 + X %*% TrueBeta + rnorm(n)
result <- geomc.vs(X=X, y=y)
logp.vs(result$median.model,X,y,lam = nrow(X)/ncol(X)^2,w = sqrt(nrow(X))/ncol(X))
```

Index