Package ‘atmcmc’

September 29, 2014

Type Package

Title Automatically Tuned Markov Chain Monte Carlo

Version 1.0

Date 2014-09-16

Author Jinyoung Yang

Maintainer Jinyoung Yang <jinyoung.yang@mail.utoronto.ca>

Description Uses adaptive diagnostics to tune and run a random walk Metropolis MCMC algorithm, to converge to a specified target distribution and estimate means of functionals.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-09-29 23:00:15

R topics documented:

`atmcmc-package` *Automatically Tuned Markov Chain Monte Carlo*

Description

This package tunes a symmetric random walk Metropolis algorithm with Gaussian proposals via finite MCMC adaption combined with new adaptive diagnostics, and it runs the tuned algorithm to converge to a target distribution and estimate the stationary mean of a functional.

Details

Package:	atmcmc
Type:	Package
Version:	1.0
Date:	2014-09-16
License:	GPL (>= 2)

The main function is 'atmcmc', which is to run a MCMC algorithm with adaptive schemes and diagnostics embedded. 'plotmcmc' function is to get traceplots and histograms of output from an 'atmcmc' run. 'summarymcmc' function displays summary statistics from the output.

Author(s)

Jinyoung Yang

Maintainer: Jinyoung Yang <jinyoung.yang@mail.utoronto.ca>

References

Jinyoung Yang and Jeffrey S. Rosenthal. Automatically Tuned General-Purpose MCMC via New Convergence Diagnostics. *Preprint,* 2014.

`atmcmc` *Runs a MCMC algorithm tuned via adaption and corresponding diagnostics*

Description

A symmetric random walk Metropolis algorithm with Gaussian proposals is automatically tuned and run and diagnosed to converge to a target distribution and estimate the stationary mean of a functional

Usage

```
atmcmc(g, dim, X0, support = cbind(rep(-Inf, dim), rep(Inf, dim)),
  multimodal = F, functional = function(X) {     X }, maxiter = 2e+06,
  mult = (2.38)^2/(dim), mrep = 10, nrep = 10, batchwidth = 200,
  holdup = 10, batchwidth.adp1 = 100, scale.adj = 0.05,
  endbatch.adp1 = 2, minaccpt = 0.28, maxaccpt = 0.6, nreg = 5,
  startdist = 1.5, minR = 0.9, maxR = 1.1, CI.alpha = 0.05,
  nimprob.X = 100, minaccpt.adp2 = 0.02, batchwidth.adp2 = 200,
  jumpprob = 0.05, displayfreq = 100, plot = T, m = 10)
```

Arguments

`g`	log of target density function
`dim`	dimension of target density function
`X0`	initial value of Markov chain
`support`	support of target density function. Argument takes a 'dim' x 2 matrix
`multimodal`	whether to assume target density function is strongly multimodal. Argument takes T or F
`functional`	function to estimate an expectation with respect to target density function
`maxiter`	maximum iteration number for a full MCMC run
`mult`	constant multiplied to the covariance matrix of the proposal distribution in the 2nd adaption phase and the sampling phase
`mrep`	number of chains created to find multiple modes if multimodal=T
`nrep`	number of replicative chains for Gelman-Rubin diagnostics (convergence diagnostics)
`batchwidth`	number of iterations in a batch. Is used to find the average X values in the transient phase and the average squared jumping distances in the 2nd adaption phase.
`holdup`	holdup*batchwidth is the number of iterations between the start of the sampling phase and the first computation of Gelman-Rubin diagnostics (to check for convergence of the chain) in the sampling phase
`batchwidth.adp1`	number of iterations in a batch initially used to check for the acceptance rate in the 1st adaption phase
`scale.adj`	amount added to or subtracted from the log of standard deviation of the univariate Gaussian proposal for each coordinate in the 1st adaption phase
`endbatch.adp1`	batchwidth.adp1 x 2^(endbatch.adp1) iterations needs to have the desired level of acceptance rate to end the 1st adaption phase
`minaccpt`	minimum acceptable acceptance rate to end the 1st adaption phase
`maxaccpt`	maximum acceptable acceptance rate to end the 1st adaption phase
`nreg`	number of distinct points in the simple linear regression to check if there is a linear trend in average X values in the transient phase or in average squared jumping distances in the 2nd adaption phase

`startdist`	number multiplied to 'max X value - min X value' in the 2nd adaption + flat part of transient phase (or just 2nd adaption phase if multimodal=T). Is used to determine over-disposed starting distribution for the sampling phase
`minR`	minimum acceptable R value to end the sampling phase
`maxR`	maximum acceptable R value to end the sampling phase
`CI.alpha`	(1-CI.alpha)x100% confidence interval is constructed for R_interval in the sampling phase
`nimprob.X`	number of consecutive g(X)=-Inf iterations required to break off the full algorithm. This is to prevent the chain from drifting to a wrong direction
`minaccpt.adp2`	minimum acceptable acceptance rate in the 2nd adaption phase to keep the value of 'mult' as it is. If the acceptance rate is less than 'minaccpt.adp2', 'mult' is cut down to 'mult'/max(2,'dim')
`batchwidth.adp2`	first n number of iterations used in the 2nd adaption phase to check for the acceptance rate to decide whether to decrease 'mult'
`jumpprob`	probability of a jump between modes at each iteration in the sampling phase
`displayfreq`	how frequently to display the iteration number as 'atmcmc' runs. Every 'displayfreq'th iteration number is printed on the screen
`plot`	whether to display traceplots of coordinate 1 (& mode 1 for multimodal=T) as each phase ends. Takes argument T or F
`m`	every 'm'th iteration is plotted. Has to be a factor of 'batchwidth'/2

Details

The algorithm automatically tunes the covariance matrix of the proposal distribution $N(X_n, \Sigma_p)$ of a symmetric random walk Metropolis algorithm. The algorithm can be broken down into four main phases: a 1st adaption phase, transient phase, 2nd adaption phase and sampling phase. The 1st adaption phase employs the Adaptive Metropolis-within-Gibbs algorithm from Roberts and Rosenthal (2009), and the diagnostics to end this phase is to check whether the acceptance rate for every coordinate comes into the desired range. The transient phase runs a Metropolis-within-Gibbs algorithm, and this runs until the chain reaches the mode of the target distribution. The purpose of the transient phase is to avoid including bad X values when tuning for $\Sigma_p = mult * \Sigma_n$, where Σ_n is the empirical covariance matrix of the target distribution calculated from the values generated by the Markov chain. The diagnostics to end this phase is to fit a simple linear regression to see if the chain values are trending. The 2nd adaption phase employs a slightly modified version of the Adaptive Metropolis algorithm from Haario et al. (2001) or Roberts and Rosenthal (2009). This phase updates Σ_p at every iteration by calculating the empirical covariance matrix of the target distribution from the chain values. The diagnostics to confirm whether this phase is indeed improving the chain is to see if the squared jumping distance between every consecutive iteration is increasing. Again, a simple linear regression is used to see if the squared jumping distances are increasing. After all this, a symmetric random walk Metropolis algorithm is run and Gelman-Rubin diagnostics is used to verify convergence of the Markov chain. Note that 2nd half of the sampling phase is what we take as a sample.

For a target distribution that is considered to be 'strongly multimodal', the basic structure of the algorithm is still the same, but multiple chains are run in the 1st adaption phase and transient phase until each chain reaches different mode. The algorithm leaves only one chain for each unique local mode and deletes others. It considers two modes are different when, in at least one coordinate, the absolute value of the difference of two means is lesser than the smaller of the standard deviation of two. A 2nd adaption phase is run for each remaining chain, and after the 2nd adaption phase, the algorithm confirms whether each chain has different mode. For the sampling phase, at each iteration, the chain either moves inside one local mode or jumps to another mode at a fixed probability. Again, Gelman-Rubin diagnostics is used to check for convergence.

Value

A list consisting of

`Xveclistdim1, Xveclistdim2,...`
: matrix of X values saved from the sampling phase. Each matrix contains X values of each coordinate

`Xveclistbase`
: matrix of X values saved from the 1st adaption ,transient, and 2nd adaption phase. Includes only one chain for each unique local mode for multimodal=T

`nummodes`
: number of unique local modes found for multimodal=T

`dim`
: dimension of target density function

`batchwidth`
: number of iterations in a batch. Is used to find the average X values in the transient phase and the average squared jumping distances in the 2nd adaption phase.

`means`
: average of X values from the 2nd half of sampling phase

`functionalmeans`
: average of functional X values from the 2nd half of sampling phase

`sumchain`
: string of iteration numbers to show when each phase has ended. For the sampling phase, this shows when the 1st half of sampling phase has ended also

`acceptrate`
: acceptance rate of the 2nd half of sampling phase

`runtime`
: runtime of the full MCMC

It also prints values of estimates, estimates_of_functional, acceptance_rate, time_elapsed, and phase_length. For details, see 'summarymcmc' section

References

Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive metropolis algorithm. *Bernoulli,* 7(2):223-242, 2001.

Gareth O. Roberts and Jeffrey S. Rosenthal. Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics,* 18(2):349-367, 2009.

Andrew Gelman and Donald B. Rubin. Inference from iterative simulation using multiple sequences. *Statistical science,* 7(4):457-472, 1992.

Stephen P. Brooks and Andrew Gelman. General methods for monitoring convergence of iterative simulations. *Journal of computational and graphical statistics,* 7(4):434-455, 1998.

Examples

```
dim = 3  #dimension of target density function
X0 = rep(0.1,dim)  #initial X value

tmpmat = rbind(c(-0.7, 1.2, 1.6),c(0.9, 1.1, -0.1),c(0.2, 0.3, -1.5))
targSigma = t(tmpmat) %*% tmpmat
targMean = c(22, -10,  15)
#log of target density function
g = function(X){-0.5*t(X-targMean)%*%solve(targSigma)%*%(X-targMean)}

output = atmcmc(g, dim, X0)
plotmcmc(output, name = "project1")
summarymcmc(output, name = "project1")
```

---

plotmcmc   *plots output from 'atmcmc'*

Description

Produces traceplots and histograms of output from an 'atmcmc' run

Usage

```
plotmcmc(output, name = "MCMC", multimodal = F, plottype = "trace",
  format = "default", m = 10, phase.start = 1, phase.end = 5,
  nrow.trace = 3, ncol.trace = 1, nrow.hist = 3, ncol.hist = 3)
```

Arguments

`output`	output from an 'atmcmc' run
`name`	name of the project
`multimodal`	whether to assume target density function is strongly multimodal. Argument takes T or F
`plottype`	whether to produce traceplots or histograms, or both. Takes argument 'trace', 'hist', or 'all'
`format`	file format. R graphics, pdf, or png. Takes argument 'default', 'pdf', or 'png'
`m`	every 'm'th iteration is plotted. Has to be a factor of 'batchwidth'/2 from the 'atmcmc' run
`phase.start`	first phase to be plotted in traceplots. Takes argument 1, 2, 3, 4, or 5. 1 = 1st adaption phase, 2 = transient phase, 3 = 2nd adaption phase, 4 = 1st half of sampling phase, and 5 = 2nd half of sampling phase
`phase.end`	last phase to be plotted in traceplots. Takes argument 1, 2, 3, 4, or 5. 1 = 1st adaption phase, 2 = transient phase, 3 = 2nd adaption phase, 4 = 1st half of sampling phase, and 5 = 2nd half of sampling phase
`nrow.trace`	number of rows for traceplots per page

`ncol.trace` number of columns for traceplots per page

`nrow.hist` number of rows for histograms per page

`ncol.hist` number of columns for histograms per page

Details

Histograms and traceplots are drawn for each coordinate separately. For the sampling phase, traceplots show only one replicative chain (chain that takes the last value of the 2nd adaption phase as the starting value of the sampling phase).

Value

Plots including

`'name'_histogram` histograms of the sample obtained from the 'atmcmc' run

`'name'_traceplot` traceplots from the 'atmcmc' run. Every 'm'th iteration is plotted

Examples

```
## see examples in `atmcmc'
```

`summarymcmc` *summary of output from 'atmcmc'*

Description

Shows a summary of output from an 'atmcmc' run. It includes project name, sample mean, functional sample mean, acceptance rate, runtime, and iteration numbers of end of all phases.

Usage

```
summarymcmc(output, name = "MCMC")
```

Arguments

`output` output from an 'atmcmc' run

`name` name of the project

Value

Displays the following items:

- `name` — name of the project
- `estimates` — sample mean
- `estimates_of_functional` — functional sample mean
- `acceptance_rate` — acceptance rate of the 2nd half of sampling phase
- `time_elapsed` — runtime of the full MCMC
- `phase_length` — string of iteration numbers to show when each phase has ended. For the sampling phase, this shows when the 1st half of sampling phase has ended also

Examples

```
## see examples in `atmcmc'
```

Index