

The mcmc Package

February 16, 2008

Version 0.5-1

Date 2 Apr 2005

Title Markov Chain Monte Carlo

Author Charles J. Geyer <charlie@stat.umn.edu>.

Maintainer Charles J. Geyer <charlie@stat.umn.edu>

Depends R (>= 1.7.0)

Suggests xtable

Description functions for Markov chain Monte Carlo (MCMC).

License X11 (http://www.x.org/Downloads_terms.html)

URL <http://www.stat.umn.edu/geyer/mcmc/>

R topics documented:

metrop	1
olbm	3

Index	5
--------------	----------

metrop	<i>Metropolis Algorithm</i>
--------	-----------------------------

Description

Markov chain Monte Carlo for continuous random vector using a Metropolis algorithm.

Usage

```
metrop(obj, initial, nbatch, blen = 1, nspac = 1, scale = 1, outfun,  
       debug = FALSE, ...)
```

Arguments

<code>obj</code>	an R function that evaluates the log unnormalized probability density of the desired equilibrium distribution of the Markov chain. First argument is the state vector of the Markov chain. Other arguments arbitrary and taken from the <code>...</code> arguments of this function. Should return <code>- Inf</code> for points of the state space having probability zero under the desired equilibrium distribution. Alternatively, an object of class <code>"metropolis"</code> from a previous run can be supplied, in which case any missing arguments (including the log unnormalized density function) are taken from this object.
<code>initial</code>	a real vector, the initial state of the Markov chain.
<code>nbatch</code>	the number of batches.
<code>blen</code>	the length of batches.
<code>nspac</code>	the spacing of iterations that contribute to batches.
<code>scale</code>	controls the proposal step size. If scalar or vector, the proposal is $x + scale * z$ where x is the current state and z is a standard normal random vector. If matrix, the proposal is $x + scale \%*\% z$.
<code>outfun</code>	controls the output. If a function, then the batch means of <code>outfun(state, ...)</code> are returned. If a numeric or logical vector, then the batch means of <code>state[outfun]</code> (if this makes sense). If missing, the the batch means of <code>state</code> are returned.
<code>debug</code>	if <code>TRUE</code> extra output useful for testing.
<code>...</code>	additional arguments for <code>obj</code> or <code>outfun</code> .

Details

Runs a “random-walk” Metropolis algorithm with multivariate normal proposal producing a Markov chain with equilibrium distribution having a specified unnormalized density. Distribution must be continuous. Support of the distribution is the support of the density specified by argument `obj`. The initial state must satisfy `obj(state, ...) > 0`.

Value

an object of class `"mcmc"`, subclass `"metropolis"`, which is a list containing at least the following components:

<code>accept</code>	fraction of Metropolis proposals accepted.
<code>batch</code>	<code>nbatch</code> by <code>p</code> matrix, the batch means, where <code>p</code> is the dimension of the result of <code>outfun</code> if <code>outfun</code> is a function, otherwise the dimension of <code>state[outfun]</code> if that makes sense, and the dimension of <code>state</code> when <code>outfun</code> is missing.
<code>initial</code>	value of argument <code>initial</code> .
<code>final</code>	final state of Markov chain.
<code>initial.seed</code>	value of <code>.Random.seed</code> before the run.
<code>final.seed</code>	value of <code>.Random.seed</code> after the run.
<code>time</code>	running time of Markov chain from <code>system.time()</code> .

lud	the function used to calculate log unnormalized density, either <code>obj</code> or <code>obj\$lud</code> from a previous run.
nbatch	the argument <code>nbatch</code> or <code>obj\$nbatch</code> .
blen	the argument <code>blen</code> or <code>obj\$blen</code> .
nspac	the argument <code>nspac</code> or <code>obj\$nspac</code> .
outfun	the argument <code>outfun</code> or <code>obj\$outfun</code> .

Examples

```
h <- function(x) if (all(x >= 0) && sum(x) <= 1) return(1) else return(-Inf)
out <- metrop(h, rep(0, 5), 1000)
out$accept
# acceptance rate too low
out <- metrop(out, scale = 0.1)
out$accept
# acceptance rate o. k. (about 25 percent)
plot(out$batch[, 1])
# but run length too short (few excursions from end to end of range)
out <- metrop(out, nbatch = 1e4)
out$accept
plot(out$batch[, 1])
hist(out$batch[, 1])
```

olbm

Overlapping Batch Means

Description

Variance of sample mean of time series calculated using overlapping batch means.

Usage

```
olbm(x, batch.length, demean = TRUE)
```

Arguments

<code>x</code>	a matrix or time series object. Each column of <code>x</code> is treated as a scalar time series.
<code>batch.length</code>	length of batches.
<code>demean</code>	when <code>demean = TRUE</code> (the default) the sample mean is subtracted from each batch mean when estimating the variance. Using <code>demean = FALSE</code> would essentially assume the true mean is known to be zero, which might be useful in a toy problem where the answer is known.

Value

The estimated variance of the sample mean.

See Also[ts](#)**Examples**

```
h <- function(x) if (all(x >= 0) && sum(x) <= 1) return(1) else return(-Inf)
out <- metrop(h, rep(0, 5), 1000)
out <- metrop(out, scale = 0.1)
out <- metrop(out, nbatch = 1e4)
olbm(out$batch, 150)
# monte carlo estimates (true means are same by symmetry)
apply(out$batch, 1, mean)
# monte carlo standard errors (true s. d. are same by symmetry)
sqrt(diag(olbm(out$batch, 150)))
# check that batch length is reasonable
acf(out$batch, lag.max = 200)
```

Index

*Topic **misc**
metrop, 1
*Topic **ts**
olbm, 3

metrop, 1

olbm, 3

ts, 4