Package ‘MHadaptive’

February 19, 2015

Type  Package
Title  General Markov Chain Monte Carlo for Bayesian Inference using adaptive Metropolis-Hastings sampling
Version 1.1-8
Date  2012-04-05
Author Corey Chivers
Maintainer Corey Chivers <corey.chivers@mail.mcgill.ca>
Depends MASS, R (>= 2.14.0)
Description Performs general Metropolis-Hastings Markov Chain Monte Carlo sampling of a user defined function which returns the un-normalized value (likelihood times prior) of a Bayesian model. The proposal variance-covariance structure is updated adaptively for efficient mixing when the structure of the target distribution is unknown. The package also provides some functions for Bayesian inference including Bayesian Credible Intervals (BCI) and Deviance Information Criterion (DIC) calculation.
License GPL (>= 3)
LazyLoad yes
Repository CRAN
Date/Publication 2012-03-24 17:49:17
NeedsCompilation no

R topics documented:

MHadaptive-package ......................................................... 2
BCI ........................................................................ 2
mcmc_r ................................................................. 3
mcmc_thin ............................................................. 4
Metro_Hastings ......................................................... 5
plotMH .................................................................. 7
positiveDefinite ....................................................... 8
**Index**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MHadaptive</td>
<td>General Markov Chain Monte Carlo for Bayesian Inference using adaptive Metropolis-Hastings sampling</td>
</tr>
</tbody>
</table>

**Description**

Performs general Metropolis-Hastings Markov Chain Monte Carlo sampling of a user defined function which returns the un-normalized value (likelihood times prior) of a Bayesian model. The proposal variance-covariance structure is updated adaptively for efficient mixing when the structure of the target distribution is unknown. The package also provides some functions for Bayesian inference including Bayesian Credible Intervals (BCI) and Deviance Information Criterion (DIC) calculation.

**Details**

<table>
<thead>
<tr>
<th>Package:</th>
<th>MHadaptive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Version:</td>
<td>1.1-6</td>
</tr>
<tr>
<td>Date:</td>
<td>2011-12-20</td>
</tr>
<tr>
<td>License:</td>
<td>GPL (&gt;= 3)</td>
</tr>
<tr>
<td>LazyLoad:</td>
<td>yes</td>
</tr>
</tbody>
</table>

This package provides a simple Metropolis-Hastings algorithm with an adaptive proposal distribution for estimating posterior distributions of Bayesian models. The user need only define the model as a function which returns the un-normalized posterior distribution (ie. $\log[L(\theta|x)P(\theta)]$).

**Author(s)**

Corey Chivers <corey.chivers@mail.mcgill.ca>

**References**


**Description**

Calculate the Bayesian Credible Intervals for an mcmcMH object.
Usage

`BCI(mcmc_object, interval = c(0.025, 0.975))`

Arguments

- `mcmc_object`: object returned by a call to `Metro_Hastings`
- `interval`: vector containing the percentiles over which to calculate the credible interval. The default of `c(0.025, 0.975)` corresponds to a 95% BCI.

Value

matrix of BCI values. Each row contains the marginal BCI for each parameter, as well as the marginal posterior means. Columns correspond to the percentiles given by `interval`.

Author(s)

Corey Chivers <corey.chivers@mail.mcgill.ca>

References


See Also

`Metro_Hastings`, `mcmc_thin`, `plotMH`

Examples

```r
data(mcmc_r)
BCI(mcmc_r) ## 95% BCIs of a simple Bayesian linear regression
```

Description

Result of a Markov Chain Monte Carlo run on a simple Bayesian linear regression model. For demonstrating `BCI`, `plotMH`, and `mcmc_thin`
mcmc_thin

Thin an MCMC object to reduce autocorrelation.

Description
This function reduces the autocorrelation of an MCMC run from Metro_Hastings() by retaining only every <thin> iterations of the chain.

Usage
mcmc_thin(mcmc_object, thin = 5)

Arguments
mcmc_object object returned by a call to Metro_Hastings()
thin integer: retain only every <thin> iterations of the MCMC.

Value
object (list) of the same type as that returned by a call to Metro_Hastings()

Author(s)
Corey Chivers <corey.chivers@mail.mcgill.ca>

See Also
Metro_Hastings, BCI, plotMH

Examples

data(mcmc_r)
## Thin the results of a simple Bayesian linear regression
mcmc_rTHINNED<-mcmc_thin(mcmc_r)
plotMH(mcmc_rTHINNED)
Description

The function `metro_hastings` performs a general Metropolis-Hastings sampling of a user defined function which returns the un-normalized value (likelihood times prior) of a Bayesian model. The proposal variance-covariance structure is updated adaptively for efficient mixing when the structure of the target distribution is unknown.

Usage

```r
metro_hastings(li_func, pars, prop_sigma = NULL,
par_names = NULL, iterations = 50000, burn_in = 1000,
adapt_par = c(100, 20, 0.5, 0.75), quiet = FALSE,...)
```

Arguments

- **li_func**: user defined function (target distribution) which describes a Bayesian model to be estimated. The function should return the un-normalized log-density function (i.e. $\log[L(\theta|x)P(\theta)]$). The first argument to this function should be a vector of parameter values at which to evaluate the function.
- **pars**: vector of initial parameter values defining the starting position of the Markov Chain.
- **prop_sigma**: covariance matrix giving the covariance of the proposal distribution. This matrix need not be positive definite. If the covariance structure of the target distribution is known (approximately), it can be given here. If not given, the diagonal will be estimated via the Fisher information matrix.
- **par_names**: character vector providing the names of each parameter in the model.
- **iterations**: integer: number of iterations to run the chain for. Default 50000.
- **burn_in**: integer: discard the first burn_in values. Default 100.
- **adapt_par**: vector of tuning parameters for the proposal covariance adaptation. Default is c(100, 20, 0.5, 0.75). The first element determines after which iteration to begin adaptation. The second gives the frequency with which updating occurs. The third gives the proportion of the previous states to include when updating (by default 1/2). Finally, the fourth element indicates when to stop adapting (default after 75% of the iterations).
- **quiet**: logical: set to TRUE to suppress printing of chain status.
- **...**: additional arguments to be passed to li_func.
Value

trace matrix containing the Markov Chain  
prop_sigma adapted covariance matrix of the proposal distribution  
par_names character vector of the parameter names  
DIC Deviance Information Criteria  
acceptance_rate proportion of times proposed jumps were accepted  

Note

While Metro_Hastings has an adaptive proposal structure built in, if prop_sigma differs greatly from the covariance structure of the target distribution, stationarity may not be achieved.

Author(s)

Corey Chivers <corey.chivers@mail.mcgill.ca>

See Also

mcmc_thin, plotMH, BCI

Examples

```r
### A LINEAR REGRESSION EXAMPLE ###
## Define a Bayesian linear regression model
li_reg <- function(pars, data) {
  a <- pars[1]  # intercept
  b <- pars[2]  # slope
  sd_e <- pars[3]  # error (residuals)
  if(sd_e <= 0) return(NaN)
  pred <- a + b * data[,1]
  log_likelihood <- sum(dnorm(data[,2], pred, sd_e, log=TRUE))
  prior <- prior_reg(pars)
  return(log_likelihood + prior)
}

## Define the Prior distributions
prior_reg <- function(pars) {
  a <- pars[1]  # intercept
  b <- pars[2]  # slope
  epsilon <- pars[3]  # error

  prior_a <- dnorm(a, 0, 100, log=TRUE)  # non-informative (flat) priors on all
  prior_b <- dnorm(b, 0, 100, log=TRUE)  # parameters.
  prior_epsilon <- dgamma(epsilon, 1, 1/100, log=TRUE)

  return(prior_a + prior_b + prior_epsilon)
}
```
```r
# simulate data
x<-runif(30,5,15)
y<-x+rnorm(30,0,5)
d<-cbind(x,y)

mcmc_r<-Metro_Hastings(li_func=li_reg,pars=c(0,1,1),
            par_names=c('a','b','epsilon'),data=d)

## For best results, run again with the previously
## adapted variance-covariance matrix.

mcmc_r<-Metro_Hastings(li_func=li_reg,pars=c(0,1,1),
            prop_sigma=mcmc_r$prop_sigma,par_names=c('a','b','epsilon'),data=d)

mcmc_r<-mcmc_thin(mcmc_r)
plotMH(mcmc_r)
```

---

**plotMH**

*Plot MCMC results of a call to Metro_Hastings().*

**Description**

This function plots histograms and traces of each parameter of the Bayesian model.

**Usage**

```r
plotMH(mcmc_object, correlogram = TRUE)
```

**Arguments**

- `mcmc_object` an object returned by a call to `Metro_Hastings()`
- `correlogram` logical: if TRUE, plots a pairwise correlogram of each parameter in the model.

**Value**

NULL

**Author(s)**

Corey Chivers <corey.chivers@mail.mcgill.ca>

**See Also**

`Metro_Hastings`, `BCI`, `mcmc_thin`
positiveDefinite

Examples

data(mcmc_r)
plotMH(mcmc_r)  ## Plot the results of a simple Bayesian linear regression


positiveDefinite  Positive Definite Matrixes

Description

Checks if a matrix is positive definite and/or forces a matrix to be positive definite.

Usage

isPositiveDefinite(x)
makePositiveDefinite(x)

Arguments

x  a square numeric matrix.

Details

The function isPositiveDefinite checks if a square matrix is positive definite.
The function makePositiveDefinite forces a matrix to be positive definite.

These functions were originally implemented in fUtilities Copyright: (c) 1999-2008 Diethelm Wuertz
and Rmetrics Foundation URL: http://www.rmetrics.org

Author(s)

Korbinian Strimmer.

Examples

## isPositiveDefinite -
# is the 3x3 identity matrix positive definite?
isPositiveDefinite(diag(c(1,1,1)))
Index

*Topic datasets
  mcmc_r, 3
*Topic math
  positiveDefinite, 8

BCI, 2, 4, 6, 7

generatingstart.MHadaptive
  (Metro_Hastings), 5

isPositiveDefinite (positiveDefinite), 8

makePositiveDefinite
  (positiveDefinite), 8
mcmc_r, 3
mcmc_thin, 3, 4, 6, 7
Metro_Hastings, 3, 4, 5, 7
MHadaptive (MHadaptive-package), 2
MHadaptive-package, 2

plotMH, 3, 4, 6, 7
positiveDefinite, 8