Package ‘ConsensusMCMC’

December 1, 2016

Type Package

Title Consensus MCMC

Version 1.0

Date 2016-11-24

Author Virginia Aglietti, Tamar Loach, Emilia Pompe

Maintainer Tamar Loach <tamar.loach@spc.ox.ac.uk>

Description Package to perform consensus MCMC for some example models.

NeedsCompilation yes

RoxygenNote 5.0.1

Depends ggplot2, parallel

R topics documented:

`ConsensusMCMC-package` *ConsensusMCMC package*

Description

An package performing the ConsensusMCMC algorithm for different univariate and multivariate target distributions.

Details

Package:	ConsensusMCMC
Type:	Package
Version:	1.1
Date:	2016-12-01
License:	GPL-2

Author(s)

Virginia Aglietti, Tamar Loach, Emilia Pompe

Maintainer: Virginia Aglietti <virginia.aglietti@stats.ox.ac.uk>, Tamar Loach <tamar.loach@stats.ox.ac.uk>, Emilia Pompe <emilia.pompe@stats.ox.ac.uk>.

References

@articlescott2016bayes, title=Bayes and big data: The consensus Monte Carlo algorithm, author=Scott, Steven L and Blocker, Alexander W and Bonassi, Fernando V and Chipman, Hugh A and George, Edward I and McCulloch, Robert E, journal=International Journal of Management Science and Engineering Management, volume=11, number=2, pages=78–88, year=2016, publisher=Taylor \& Francis

---

`ACFPlot` *Plot for the autocorrelation function of a given Markov chain.*

Description

The ACFPlot function returns the plot for the autocorrelation function (acf) of a given Markov chain.

Usage

```
ACFPlot(chain, lag.max = 10, ...)
```

Arguments

`chain`	A vector of values from the Markov chain.
`lag.max`	Maximum number of lags to be displayed in the plot. Dafault is 10.
`...`	Additional graphical parameters to be passed to ggplot.

Value

The function returns the acf plot for a specified Markov chain.

Examples

```
# Generate two sets of values
chain = rnorm(100)

# Produce the plot
ACFPlot(chain, lag.max=10, ggtitle('acf'))
```

BetaMH *MH-Algorithm for a Univariate Beta target distribution.*

Description

The BetaMH function returns the Markov chain generated by a random walk Metropolis-Hasting algorithm having a univariate beta distribution as target distribution.

Usage

```
BetaMH(data, n, sigma, alpha_prior, beta_prior, s, x_0)
```

Arguments

data	Observations distributed as an Binomial distribution.
n	The required number of iterations for the MH-algorithm.
sigma	The value of the standard deviation for the Gaussian proposal distribution of the random-walk MH-algorithm.
alpha_prior	The prior value of the first non-negative parameter of the Beta prior distribution of the probability of success.
beta_prior	The prior value of second non-negative parameter of the Beta prior distribution of the probability of success.
s	The number of machines on which to run the MH-algorithm. The value ranges between 1 and 4. Must be set different to 1 if running in parallel.
x_0	The initial value for the unknown parameter of the Beta target distribution.

Value

The function returns a vector with the values of the Markov chain generated by the MH-algorithm.

Examples

```
# Generate data
observations = rbinom(10000, size = 1, prob = 0.5)

# Set the parameters for the function
nr_servers = 1
n_iter = 1000
burn_in = 0.1*n_iter
sigma = 0.01
alpha_prior = 1.0
beta_prior = 1.0
x_0 = 0.1


# Run the function for the chosen parameters
markov_chain = BetaMH(data=observations,
               n = n_iter,
               sigma = sigma,
               alpha_prior = alpha_prior,
               beta_prior = beta_prior,
```

```
s = 1,
x_0 = x_0)
```

GammaMH *MH-Algorithm for a Univariate Gamma target distribution.*

Description

The GammaMH function returns the Markov chain generated by a random walk Metropolis-Hasting algorithm having a univariate gamma distribution as target distribution.

Usage

```
GammaMH(data, n, sigma, k_prior, theta_prior, s, x_0)
```

Arguments

data	Observations distributed as a Poisson distribution.
n	The required number of iterations for the MH-algorithm.
sigma	The value of the standard deviation for the Gaussian proposal distribution of the random-walk MH-algorithm.
k_prior	The prior value of the shape parameter for the Gamma prior distribution of the unknown parameter.
theta_prior	The prior value of the rate parameter for the Gamma prior distribution of the unknown parameter.
s	The number of machines on which to run the MH-algorithm. The value ranges between 1 and 4. Must be set different to 1 if running in parallel.
x_0	The initial value for the unknown parameter of the Beta target distribution.

Value

The function returns a vector with the values of the Markov chain generated by the MH-algorithm.

Examples

```
# Generate data
observations = rpois(10000, 4)

# Set the parameters for the function
nr_servers = 1
n_iter = 100000
burn_in = 0.1*n_iter
sigma = 0.01
k_prior = 1.0
theta_prior = 4.0
x_0 = 4

# Run the function for the chosen parameters
markov_chain = GammaMH(data=observations,
                       n = n_iter,
```

```
                          sigma = sigma,
                          k_prior = k_prior,
                          theta_prior = theta_prior,
                          s = 1,
                          x_0 = x_0)
```

---

gslrng *Random number generation*

Description

Generate uniform random numbers

Usage

```
gslrng(n)
```

Arguments

n	The number of random numbers to generate.

Details

Generates n random numbers using GSL in C code. The random number generator state is automatically maintained between calls via a static C variable. The high quality Mersenne Twister algorithm due to Makoto Matsumoto and Takuji Nishimura is used, with dimensionality 623.

Value

Returns n random numbers between 0 and 1.

Author(s)

Louis J. M. Aslett <aslett@stats.ox.ac.uk>

References

OxWaSP module 7 slides

Makoto Matsumoto and Takuji Nishimura, "Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator". *ACM Transactions on Modeling and Computer Simulation*, Vol. 8, No. 1 (Jan. 1998), Pages 3-30

Examples

```
gslrng(5)
gslrng(20)
```

HistPlot *Density estimates of the histograms for a list of Markov chains.*

Description

The HistPlot function returns the density estimates for the histograms of a given list of Markov chains.

Usage

```
HistPlot(list_of_vectors, method = NULL, burn_in = 0.1, size_line = 1,
  ...)
```

Arguments

`list_of_vectors`	A list of Markov chains to be plotted. The number of element in the list can be greater or equal than one.
`method`	A vector giving the different algorithms used to produce the Markov chains. The dafault value is NULL.
`burn_in`	A proportion of the generated sample that need to be discarded in the plot. Need to be given as a percentage in decimal number of the algorithm iterations. The default value is 0.1.
`size_line`	The size of the line in the plot. The default value is 1.
`...`	Additional graphical parameters to be passed to ggplot.

Value

The function returns the density estimates for the histograms of the specified Markov chains.

Examples

```
# Generate two sets of values
chain1 = rnorm(100)
chain2 = rnorm(100)

# Produce the density plots
HistPlot(list(chain1, chain2),
         method = c('Single machine', 'Multiple machines'),
         burn_in = 0.2,
         size_line = 2)
```

NormalMH *MH-Algorithm for a Univariate Normal target distribution.*

Description

The NormalMH function returns the Markov chain generated by a random walk Metropolis-Hasting algorithm having a univariate normal distribution as target distribution.

Usage

```
NormalMH(data, n, sigma, mean_prior, sigma_prior, sigma_known, s, x_0)
```

Arguments

data	Observations distributed as an univariate normal distribution with unknown mean and known variance.
n	The required number of iterations for the MH-algorithm.
sigma	The value of the satandard deviation for the Gaussian proposal distribution of the random-walk MH-algorithm.
mean_prior	The prior mean value for the unknown parameter.
sigma_prior	The prior standard deviation value for the unknown parameter.
sigma_known	The value of the known parameter.
s	The number of machines on which to run the MH-algorithm. The value ranges between 1 and 4. Must be set different to 1 if running in parallel.
x_0	The initial value for the unknown parameter.

Value

The function returns a vector with the values of the Markov chain generated by the MH-algorithm.

Examples

```
# Set the parameters for the function
sigma_known = 1
nr_servers = 1
n_iter = 1000
burn_in = 0.1*n_iter
sigma = 0.01
mean_prior = 0.0
sigma_prior = 1.0
x_0 = 0.0

# Generate data
observations = rnorm(10000, 1 , sigma_known)

# Run the function for the chosen parameters
markov_chain = NormalMH(data= observations,
               n = n_iter,
               sigma = sigma,
               mean_prior = mean_prior,
```

```
sigma_prior = sigma_prior,
sigma_known = sigma_known,
s = nr_servers,
x_0 = x_0)
```

QQPlot *QQPlot for two Markov chains.*

Description

The QQPlot function returns the qqplot for two given Markov chains.

Usage

```
QQPlot(chain1, chain2, line = TRUE, size_point = 2, size_line = 1, ...)
```

Arguments

`chain1`	The first Markov-chain to plot.
`chain2`	The second Markov-chain to plot.
`line`	Logical parameter. If TRUE, the line chain1 = chain2 is added to the qqplot. Default value is TRUE.
`size_point`	The desired size of the points in the plot. Default value is 2.
`size_line`	The desired size of the line in the plot. Default value is 1.
`...`	Additional graphical parameters to be passed to ggplot.

Value

The function returns the qqplot for two specified Markov chains.

Examples

```
# Generate two sets of values
chain1=rnorm(100)
chain2=rnorm(100)

# Produce the qqplot
QQPlot(chain1=rnorm(100), chain2=rnorm(100))
```

`TracePlot` *Trace plots for a list of Markov chains.*

Description

The TracePlot function returns the trace plots for a given list of Markov chains.

Usage

```
TracePlot(list_of_vectors, method = NULL, burn_in = 0.1, size_line = 1,
  ...)
```

Arguments

`list_of_vectors`	A list of Markov chains to be plotted. The number of element in the list can be greater or equal than one.
`method`	A vector giving the different algorithms used to produce the Markov chains. The dafault value is NULL.
`burn_in`	A proportion of the generated sample that need to be discarded in the plot. Need to be given as a percentage in decimal number of the algorithm iterations. The default value is 0.1.
`size_line`	The size of the line in the plot. The default value is 1.
`...`	Additional graphical parameters to be passed to ggplot.

Value

The function returns the qqplot for two specified Markov chains.

Examples

```
# Generate two sets of values
chain1 = rnorm(100)
chain2 = rnorm(100)

# Produce the trace plots
TracePlot(list(chain1, chain2),
        method = c('Single machine', 'Multiple machines'),
        burn_in = 0.2)
```

`weightsComputation` *Weights computation for Consensus MCMC algorithm*

Description

The function computes the weights used for the Consensus MCMC algorithm. The computed weights are used to aggregate the Markov Chains obtained parallelizing the data on different machines.

Usage

```
weightsComputation(df, method)
```

Arguments

`df`	A dataframe derived from running `BetaMH` or `NormalMH` or `GammaMH` on the relative data.
`method`	The method that needs to be used to aggregate the markov chains obtained from different machines. Need to be used when running in parallel.

Value

The functions returns the aggregated Markov chain from the Consensus MCMC.

Examples

```
df = data.frame(lapply(lambda, function(y) y))
parallel_markov_chain = weightsComputation(df, method = "sample variance")
```

Index