

Package ‘mcsm’

April 28, 2009

Type Package

Title Functions for Monte Carlo Methods with R

Version 1.0

Date 2009-02-26

Depends stats, MASS, coda

Author Christian P. Robert, Universite Paris Dauphine

Maintainer Christian P. Robert <xian@ceremade.dauphine.fr>

Description mcsm contains a collection of functions that allows the reenactment of the R programs used in the book *EnteR Monte Carlo Methods* without further programming. Programs being available as well, they can be modified by the user to conduct one’s own simulations.

License GPL (>= 2)

Repository CRAN

Date/Publication 2009-04-28 10:10:13

R topics documented:

adapump	2
betagen	3
Braking	4
challenge	5
challenger	6
dmunorm	7
dyadic	8
EMcenso	8
Energy	9
gibbsmix	10
hastings	10
jamestein	11
kscheck	12

logima	13
maximple	14
mhmix	15
mochoice	16
mump	17
normbyde	18
pimamh	19
pimax	20
randogibs	21
randogit	21
randomeff	22
rdirichlet	23
reparareff	24
rnorm	25
SAmix	26
sqar	27
sqaradap	28
test1	29
test2	30
test3	31
test4	32

Index	33
--------------	-----------

adapump	<i>Illustration of the danger of adaptive MCMC for the pump failure data</i>
---------	--

Description

This function constructs an update of the location and scale matrix of a normal proposal in a Metropolis-Hastings algorithm, based on earlier simulations in order to show the danger of online adaptivity.

Usage

```
adapump(T = 10^2, MM = 10^2)
```

Arguments

T	Number of steps between updates
MM	Number of updates, leading to a chain of length $MM \times T$

Value

The function simply plots the sequence of β 's along iterations, which should collapse, as well as the range of the variability of the proposed values.

Author(s)

Christian P. Robert and George Casella

References

Chapter 8 of **EnteR Monte Carlo Statistical Methods**

See Also

[kscheck](#)

Examples

```
adapump(T=50,MM=50)
```

betagen

Plot explaining accept-reject on a Beta(2.7,6.3) target

Description

This function of *Nsim* represents *Nsim* points simulated from either a uniform or a Beta(2,6) proposal in terms of their location above versus below the density of the Beta(2.7,6.3) target in order to explain accept-reject methods.

Usage

```
betagen(Nsim = 10^3)
```

Arguments

`Nsim` Number of points to be represented on the subgraph, the default value being 10^3

Details

The R code can be modified for further use, but note that the bounds *M* in the program are hardcoded towards the current values of *a* and *b*.

Value

The function returns a graph, either on an existing graphical window or by creating a new graphical window, with two plots side-by-side.

Warning

This function is only intended to illustrate the principle at work behind accept-reject. For Beta simulation, use `rbeta`.

Author(s)

Christian P. Robert and George Casella

References

Chapter 3 of **EnteR Monte Carlo Statistical Methods**

See Also

rbeta

Examples

```
betagen(Nsim=10^4)
```

Braking

Quadratic regression on the car braking dataset

Description

This function uses MCMC to analyse the car braking dataset *cars* via a quadratic regression model. The underlying MCMC algorithm is an independent Metropolis-Hastings algorithm centered at the MLE.

Usage

```
Braking(nsim = 10^3)
```

Arguments

`nsim` Number of iterations in the MCMC algorithm

Value

Returns a plot of some simulated regression lines along with their average and the original data.

Author(s)

Christian P. Robert and George Casella

References

Chapter 3 of **EnteR Monte Carlo Statistical Methods**

Examples

```
Braking(10^4)
```

`challenge`*Slice sampler analysis of the challenger dataset*

Description

This function illustrates a slice sampling implementation of the simulation from the posterior distribution associated with a logistic regression model

$$P(y = 1|x) = 1/[1 + \exp\{-\alpha - \beta x\}]$$

when applied to the `challenger` dataset.

Usage

```
challenge(Nsim = 10^4)
```

Arguments

`Nsim` Number of slice sampling iterations

Value

The output is a `list` made of

`a` Sequence of values of the intercept α produced by the slice sampler
`b` Sequence of values of the regression coefficient β produced by the slice sampler

Warning

The function `challenge` uses a function `rtrun` that is replicated from a function used in the package `bayesm`. In the current case, the simulation of the truncated normal distribution is done by a simple cdf inversion and may thus be fragile in the tails.

Author(s)

Christian P. Robert and George Casella

References

Chapter 6 of **EnteR Monte Carlo Statistical Methods**

See Also

[challenger](#)

Examples

```
data(challenger)
chares=challenge(10^4)
plot(chares$a, chares$b, type="l", xlab="a", ylab="b", pch=19, cex=.4)
```

`challenger`*O-ring failures against temperature for shuttle launches*

Description

This dataset records the occurrence of failures in the O-rings of space shuttles against the temperature at the time of the launches. One (1) stands for a failure and zero for a success.

Usage

```
data(challenger)
```

Format

A data frame with 23 observations on the following 2 variables.

oring Occurrence or not of an O-ring failure

temp Temperature at the time of the launch

Details

In 1986, the space shuttle Challenger exploded during take off, killing the seven astronauts aboard. The explosion was the result of an O-ring failure, a splitting of a ring of rubber that seals the parts of the ship together. The accident was believed to be caused by the unusually cold weather (31 degrees F or 0 degrees C) at the time of launch, as there is reason to believe that the O-ring failure probabilities increase as temperature decreases.

Source

Dalal, S.R., Fowlkes, E.B. and Hoadley, B. (1989) Risk analysis of the space shuttle: pre-Challenger prediction of failure. J. American Statistical Association, 84, 945–957.

See Chapters 6 and 7 of **EnteR Monte Carlo Statistical Methods** by Christian P. Robert and George Casella

Examples

```
data(challenger)
plot(challenger$temp, challenger$oring)
```

`dmunorm`*Density function of the multivariate normal distribution*

Description

This function computes the density of the multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$ in either natural or logarithmic scales.

Usage

```
dmunorm(x, mu, sig, log = FALSE)
```

Arguments

<code>x</code>	Running argument of the density
<code>mu</code>	Mean μ of the normal distribution
<code>sig</code>	Covariance matrix Σ of the normal distribution
<code>log</code>	Boolean describing whether or not the output is in logarithmic scales

Value

This function returns a real number corresponding to the density in x in either natural or logarithmic scales.

Warning

This function is *fragile* in that it does not test for

1. the fact that `sig` is a square matrix,
2. the compatibility of the dimensions of `x`, `mu`, `sig`
3. the symmetry nor the invertibility of the matrix `sig`

It is therefore prone to fail if those conditions are not satisfied! If the package `mvtnorm` is installed, the function `dmvnorm` should be used instead.

Author(s)

Christian P. Robert and George Casella

References

Chapter 8 of **EnteR Monte Carlo Statistical Methods**

Examples

```
dmunorm(c(1,2), c(1,2), diag(rep(1,2)))-1/(2*pi)
# Should be equal to zero!
```

dyadic

A dyadic antithetic improvement for a toy problem

Description

Using dyadic replicas of a uniform sample, when evaluating the mean of $h(x) = [\cos(50x) + \sin(20x)]^2$, this function shows the corresponding improvement in variance. The parameter q is used to break the unit interval into 2^q blocks where a transform of the original uniform sample is duplicated.

Usage

```
dyadic(N = 10^4, q=4)
```

Arguments

N	final number of simulations
q	number of dyadic levels

Value

Produces a plot comparing the q th and $(2q)$ th dyadic improvements with the original.

Author(s)

Christian P. Robert and George Casella

References

From Chapter 4 of **EnteR Monte Carlo Statistical Methods**

Examples

```
dyadic(N=10^2, q=5)
```

EMcenso

EM paths for a censored normal model

Description

This function produces a series of EM paths for a censored normal model.

Usage

```
EMcenso(repp = 10)
```

Arguments

`repp` Number of paths

Value

The outcome of this function is a plot.

Author(s)

Christian P. Robert and George Casella

References

From Chapter 5 of **EnteR Monte Carlo Statistical Methods**

Examples

```
EMcenso(45)
```

Energy

Energy intake

Description

Energy intake, measured in megajoules, over a 24 hour period of 15 year-old males and females obtained in a study on metabolism.

Usage

```
data(Energy)
```

Format

A data frame with 16 observations on the following 2 variables.

Girls Energy intake of 16 girls

Boys Energy intake of 16 boys

Source

McPherson, G. (1990). Statistics in Scientific Investigation. New York: Springer-Verlag
Chapter 7 of **EnteR Monte Carlo Statistical Methods** by Christian P. Robert and George Casella

Examples

```
data(Energy)  
str(Energy)
```

`gibbsmix`*Implementation of a Gibbs sampler on a mixture posterior*

Description

This function runs a standard Gibbs sampling algorithm on a posterior distribution associated with a mixture model and 500 datapoints.

Usage

```
gibbsmix(Niter = 10^4, v = 1)
```

Arguments

<code>Niter</code>	Number of MCMC iterations
<code>v</code>	Scale of the normal prior

Value

The function returns a plot of the log-posterior surface, along with the MCMC sample represented both by points and lines linking one value to the next. Evaluating the log-posterior surface on a 250×250 grid takes some time.

Author(s)

Christian P. Robert and George Casella

References

Chapter 6 of **Enter Monte Carlo Statistical Methods**

Examples

```
## Not run: gibbsmix(Nit=10^3)
```

`hastings`*Reproduction of Hastings' experiment*

Description

This function compares three scales of uniform proposals when the target is a standard normal distribution and the algorithm is a Metropolis-Hastings algorithm. This is an example from the original Hastings' (1970) paper.

Usage

```
hastings(nsim = 10^3)
```

Arguments

nsim Number of Metropolis-Hastings steps

Value

The outcome of the function is a graph with nine panels comparing the three uniform proposals in terms of shape, fit and autocorrelation.

Author(s)

Christian P. Robert and George Casella

References

From Chapter 6 of **EnteR Monte Carlo Statistical Methods**

Examples

```
hastings(10^4)
```

jamestein

Monte Carlo plots of the risks of James-Stein estimators

Description

This is a Monte-Carlo representation of the risks of some James-Stein estimators of the mean θ of a p -dimensional $\mathcal{N}_p(\theta, I)$ distribution, taking advantage of a variance reduction principle based on recycling random variates.

Usage

```
jamestein(N = 10^3, p = 5)
```

Arguments

N Number of simulations
p Dimension of the problem

Details

Given that the risk is computed for all values of the mean θ , using a different normal sample for each value of θ creates an extraneous noise that is unnecessary. Using the same sample produces a smooth and well-ordered (in the shrinkage parameter α) set of graphs.

Value

Returns a plot with 10 different values of the shrinkage factor α between 1 and $2(p - 2)$, which is the maximal possible value for minimaxity.

Warning

Because of the multiple loops used in the code, this program takes quite a while to produce its outcome. Note that there is a James-Stein effect only when $p > 2$ but that it may not be visible for a small value of N .

Author(s)

Christian P. Robert and George Casella

References

Chapter 4 of **EnteR Monte Carlo Statistical Methods**

Examples

```
jamestein(N=2*10^2)      #N is too small to show minimaxity
```

kscheck

Convergence assessment for the pump failure data

Description

This function produces graphs and one or several samples from the Gibbs sampler applied to the benchmark of the pump failure dataset (given within the function). It also provides convergence assessments in connection with the `coda` package.

Usage

```
kscheck(T = 10^3, heidel = FALSE)
```

Arguments

T	Number of Gibbs iterations
heidel	Boolean indicating whether or not several parallel chains need to be simulated

Value

Depending on the value of `heidel`, the function returns two plots of Kolmogorov-Smirnov statistics comparing two different scales or the outputs of `heidel.diag` and `geweke.diag` functions. It also returns as a `list` the sequence(s) of β and λ .

Author(s)

Christian P. Robert and George Casella

References

Chapter 8 of **EnteR Monte Carlo Statistical Methods**

See Also

[adapump](#)

Examples

```
outum=kscheck(heidel=TRUE)
```

logima

Logistic analysis of the Pima.tr dataset with control variates

Description

Provides a logistic analysis of the ‘Pima.tr’ by comparing the standard Bayesian estimate with another one based on log-cumulant control variates. The effect is visible if not huge.

Usage

```
logima(N = 10^3)
```

Arguments

N Number of Monte Carlo iterations

Value

Returns a graph for both the estimate of the intercept and the estimate of the coefficient of the body mass index.

Author(s)

Christian P. Robert and George Casella

References

Chapter 4 of **EnteR Monte Carlo Statistical Methods**

See Also

Pima

Examples

```
logima()
```

```
maximple
```

Graphical representation of a toy example of simulated annealing

Description

For the toy function $h(x) = (\cos(50x) + \sin(20x))^2$, this function represents simulated annealing sequences converging to a local or global maxima as paths on top of the function h itself. The simulated annealing parameters *ratemp* and *powemp* can be changed, as well as the stopping rule *tolerance*.

Usage

```
maximple(tolerance = 10^(-5), ratemp = 5, powemp = 2)
```

Arguments

<code>tolerance</code>	maximal difference in the target value needed to stop the simulated annealing algorithm
<code>ratemp</code>	scale factor of $\sqrt{T_t}$ that determines the scale of the random walk
<code>powemp</code>	power of $\frac{1}{1+t}$ used to set the temperature schedule

Value

The value of this function is a list, with components

<code>x</code>	coordinates of the successive values of the random walk produced by the simulated annealing algorithm
<code>y</code>	corresponding values of $h(x)$, of the same length as <code>x</code>

Author(s)

Christian P. Robert and George Casella

References

From Chapter 5 of **EnteR Monte Carlo Statistical Methods**

See Also

[dyadic](#)

Examples

```
# Section 5.2.2, artificial function example 5.8

h=function(x){(cos(50*x)+sin(20*x))^2}
par(mar=c(4,4,1,1),mfrow=c(2,2))
for(tt in 1:4){

  curve(h,from=0,to=1,n=10001,col="grey",lwd=2)
  sam=maximble()
  xgo=sam$x
  hgo=sam$y
  lines(xgo,hgo,col="steelblue4",lwd=2)
  points(xgo,hgo,col="steelblue4",cex=.5,pch=19)
}
```

mhmix

*Implement two Metropolis-Hastings algorithms on a mixture posterior***Description**

This function runs a Metropolis-Hastings algorithm on a posterior distribution associated with a mixture model and 500 datapoints. Depending on the value of the boolean indicator `lange`, the function uses a regular Gaussian random-walk proposal or a Langevin alternative.

Usage

```
mhmix(Niter = 10^4, lange = FALSE, scale = 1)
```

Arguments

<code>Niter</code>	Number of MCMC iterations
<code>lange</code>	Boolean variable indicating the use of the Langevin alternative
<code>scale</code>	Scale factor of the Gaussian perturbation

Value

The function returns a plot of the log-posterior surface, along with the MCMC sample represented both by points and lines linking one value to the next.

Author(s)

Christian P. Robert and George Casella

References

Chapter 6 of **EnteR Monte Carlo Statistical Methods**

Examples

```
## Not run: mhmix(Nit=10^3,scale=2)
#you can also try mhmix(lange=TRUE,scale=.1)
```

mochoice

An MCMC model choice illustration for the linear model

Description

Using a Gibbs sampling strategy of changing one indicator at a time, this function explores the space of models and returns the most likely models among those visited. The data used in this example is swiss, with four explanatory variables.

Usage

```
mochoice(Niter = 10^4)
```

Arguments

Niter	Number of MCMC iterations
-------	---------------------------

Value

model	Sequence of model indicators visited by the MCMC algorithm
top	Five most likely models

Note

For more details, see Chapter 3 of **Bayesian Core** (2007, Springer-Verlag) by J.-M. Marin and C.P. Robert, since the procedure is derived from the developments in this chapter.

Author(s)

Christian P. Robert and George Casella

References

From Chapter 6 of **Enter Monte Carlo Statistical Methods**

Examples

```
mochoice(10^3)
```

Description

This function produces several Gibbs chains for the same pump failure data as in the function [kscheck](#) in order to illustrate Gelman and Rubin's diagnostic `gelman.diag` and `gelman.plot` provided in `coda`.

Usage

```
mump(Nsim = 10^4, MM = 10)
```

Arguments

<code>Nsim</code>	Number of Gibbs iterations
<code>MM</code>	Number of parallel chains (minimum value of 10)

Value

The function returns plots and diagnostic values connected with `gelman.diag` and `gelman.plot`, as well as the autocorrelation analysis.

Author(s)

Christian P. Robert and George Casella

References

Chapter 8 of **EnteR Monte Carlo Statistical Methods**

See Also

[kscheck](#)

Examples

```
mump(10^3)
```

`normbyde`*Compare two double-exponentials approximations to a normal distribution*

Description

This simple program compares a double-exponential distribution with parameter $\alpha = 1$ and a double-exponential distribution with parameter $\alpha \neq 1$ in their approximation to the standard normal distribution. Quite obviously, this function is not to be used when compared when `rnorm`.

Usage

```
normbyde(nsim = 10^3, a = 3)
```

Arguments

<code>nsim</code>	Number of simulations
<code>a</code>	Scale of the second double-exponential scale

Value

The function produces a tryptich graph with the comparison of the cumulated averages, and both acf graphs.

Author(s)

Christian P. Robert and George Casella

References

From Chapter 6 of **EnteR Monte Carlo Statistical Methods**

See Also

`rnorm`

Examples

```
normbyde(10^4, 5)
```

pimamh

Langevin MCMC algorithm for the probit posterior

Description

This function implements a Langevin version of the Metropolis-Hastings algorithm on the posterior of a probit model, applied to the `Pima.tr` dataset.

Usage

```
pimamh(Niter = 10^4, scale = 0.01)
```

Arguments

<code>Niter</code>	Number of MCMC iterations
<code>scale</code>	Scale of the Gaussian noise in the MCMC proposal

Value

The function produces an `image` plot of the log-posterior, along with the simulated values of the parameters represented as dots.

Warning

This function is fragile since, as described in the book, too large a value of `scale` may induce divergent behaviour and crashes with error messages

```
Error in if (log(runif(1)) > like(prop[1], prop[2]) - likecur - sum(dnorm(prop,..))
  missing value where TRUE/FALSE needed
```

Author(s)

Christian P. Robert and George Casella

References

Chapter 6 of **EnteR Monte Carlo Statistical Methods**

See Also

`Pima.tr`, [pimax](#)

Examples

```
## Not run: pimamh(10^4, scale=.01)
```

pimax

Monte Carlo approximation of a probit posterior marginal

Description

This function represents the variability of a Monte Carlo approximation to the posterior distribution of the intercept of a probit model. The data used in this example is the Pima indian `Pima.tr` dataset. The function produces three plots, the top one being based on a single simulated sample for all values of the intercept along with the Monte Carlo variability estimated by 100 repeated calls, the medium one being based on iid simulated samples for all values of the intercept along with the corresponding Monte Carlo variability and the bottom one being obtained by numerical integration.

Usage

```
pimax(Nsim = 10^3)
```

Arguments

`Nsim` Number of simulations in all Monte Carlo experiments

Value

Return three plots

Author(s)

Christian P. Robert and George Casella

References

From Chapter 5 of **EnteR Monte Carlo Statistical Methods**

See Also

[Pima.tr](#), [pimamh](#)

Examples

```
## Not run: pimax()
```

randogibs	<i>First illustrations of coda's output for the one-way random effect model</i>
-----------	---

Description

This function provides graphical illustrations of `coda`'s output for the one-way random effect model processed in Chapters 7 and 8 of **EnteR Monte Carlo Statistical Methods**.

Usage

```
randogibs(T = 10^3)
```

Arguments

T	Number of MCMC iterations
---	---------------------------

Value

The function produces two plots separated by a `browser()` prompt where the user needs to press the Return key to see the second graph.

Author(s)

Christian P. Robert and George Casella

References

Chapter 8 of **EnteR Monte Carlo Statistical Methods**

Examples

```
resagibs=randogibs()
```

randogit	<i>MCEM resolution for a probit maximum likelihood</i>
----------	--

Description

Based on Booth and Hobert (*JRSS B*, 1999), this function evaluates the maximum likelihood estimate of a simulated probit model with random effects. The random effects are simulated by a MCMC algorithm.

Usage

```
randogit(Tem = 10^3, Tmc = 10^2)
```

Arguments

Tem	starting number of MCEM iterations
Tmc	number of Monte Carlo points in the likelihood approximations

Value

The function returns two plots, one of (β, σ) and one of the true likelihood $L(\beta, \sigma, u_0)$, where u_0 is the true vector of random effects.

Author(s)

Christian P. Robert and George Casella

References

From Chapter 2 of **EnteR Monte Carlo Statistical Methods**

Examples

```
## Not run: randogit(20,10)
#very small values to let the example run
```

randomeff

Gibbs sampler for a one-way random effect model

Description

This function describes a Gibbs sampler for the one-way random effect model for the [Energy](#) dataset provided with the `mcsM` package. The prior is made of conjugate distributions, i.e. normal and inverted gamma distributions.

Usage

```
randomeff(nsim = 10^3, a = 10, b = 30)
```

Arguments

nsim	Number of Gibbs iterations
a	Common shape parameter for the inverted variances
b	Common scale parameter for the inverted variances

Value

This function produces a plot made of six histograms corresponding to the six parameters of the model $\mu, \theta_1, \theta_2, \sigma_\mu, \tau,$ and σ , eliminating the first 10% of the simulations as burnin.

Author(s)

Christian P. Robert and George Casella

References

Chapter 7 of **EnteR Monte Carlo Statistical Methods**

Examples

```
data(Energy)
randomeff(10^4, a=1, b=1)
```

rdirichlet	<i>Dirichlet generator</i>
------------	----------------------------

Description

This is a random generator for Dirichlet $\mathcal{D}(\alpha_1, \dots, \alpha_d)$ distributions, based on the normalisation of Gamma $\mathcal{G}(\alpha_i)$ random variables.

Usage

```
rdirichlet(n, shape)
```

Arguments

n	Number of generations requested
shape	Vector of α_i 's

Value

A $n \times d$ matrix, with one simulated vector per row.

Author(s)

Christian P. Robert and George Casella

References

Used in Chapter 7 **EnteR Monte Carlo Statistical Methods**

See Also

rgamma

Examples

```
A=rdirichlet(10^3,rep(.5,5))
hist(A[,1],fre=FALSE,col="grey",nclass=123,xlim=c(0,1),
main="",xlab=expression(p[1]))
curve(dbeta(x,.5,4*.5),add=TRUE,col="sienna",lwd=2)
```

reparareff

Reparameterized version of the one-way random effect model

Description

This function proposes a reanalysis of the one-way random effect model processed in [randomeff](#) via a Gibbs sampler using another parameterization of the model in order to exhibit a different convergent behavior.

Usage

```
reparareff(nsim = 10^4, a = 10, b = 10)
```

Arguments

nsim	Number of Gibbs iterations
a	Common shape parameter for the inverted variances
b	Common scale parameter for the inverted variances

Value

This function produces a plot made of six [acf](#) output corresponding to the parameters μ , θ_1 , and θ_2 , both in the original parameterization and in the new one.

Author(s)

Christian P. Robert and George Casella

References

Chapter 7 of **EnteR Monte Carlo Statistical Methods**

See Also

[randomeff](#)

Examples

```
reparareff(10^4,a=1,b=1)
```

`rmunorm`*Random generator for the multivariate normal distribution*

Description

This function produces one random vector distributed from the multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$.

Usage

```
rmunorm(mu, sig)
```

Arguments

<code>mu</code>	Mean μ of the normal distribution
<code>sig</code>	Covariance matrix Σ of the normal distribution

Value

This function returns a real vector of the same dimension as μ .

Warning

Similar to `dmunorm`, this function is fragile in that it does not test for

1. the fact that `sig` is a square matrix,
2. the compatibility of the dimensions of `x`, `mu`, `sig`
3. the symmetry nor the invertibility of the matrix `sig`

It is therefore prone to fail if those conditions are not satisfied! If the package `bayesm` can be installed, `rmvnorm` is to be preferred to `rmunorm`.

Author(s)

Christian P. Robert and George Casella

References

Chapter 8 of **Enter Monte Carlo Statistical Methods**

See Also

`rnorm`, `dmunorm`, `rmvnorm`(`bayesm`)

Examples

```
test=NULL
for (t in 1:10^4) test=rbind(test,rmunorm(rep(1,2),matrix(c(1,-2,-2,10),ncol=2)))
cor(test[,1],test[,2])*sqrt(10) # should be close to -2
```

SAmix

Graphical representation of the simulated annealing sequence for the mixture posterior

Description

This function implements a simulated annealing algorithm to optimize the posterior distribution of a normal mixture with two components and only the means unknown,

$$.25\mathcal{N}(\mu_1, 1) + .75\mathcal{N}(\mu_2, 1)$$

with a schedule $T_t = 1/\log(1 + t)$.

Usage

```
SAmix(x, tolerance = 10^(-4), factor = 1)
```

Arguments

x	two-dimensional vector, starting point of the simulated annealing algorithm
tolerance	maximal difference in the target value needed to stop the simulated annealing algorithm
factor	scale factor of $\sqrt{T_t}$ that determines the scale of the random walk

Value

theta	sequence of points explored by the simulated annealing algorithm
like	corresponding sequence of posterior values
ite	number of iterations to reach stable value

Author(s)

Christian P. Robert and George Casella

References

From Chapter 5 of **EnteR Monte Carlo Statistical Methods**

Examples

```
da=sample(rbind(rnorm(10^2), 2.5+rnorm(3*10^2)))
SAres=SAmix(x=c(-.3, .6), tol=10^(-2), fac=.1)
```

`sqar`*Illustration of some of coda's criterions on the noisy squared AR model*

Description

This function illustrates some of `coda`'s criterions on the noisy squared AR model, using a Metropolis-Hastings algorithm based on a random walk. Depending on the value of the boolean `multies`, those criterions are either the `geweke.diag` and `heidel.diag` diagnostics, along with a Kolmogorov-Smirnov test of our own, or `plot(mcmc.list())` if several parallel chains are produced together.

Usage

```
sqar(T = 10^4, multies = FALSE, outsave = FALSE, npara = 5)
```

Arguments

<code>T</code>	Number of MCMC iterations
<code>multies</code>	Boolean variable determining whether or not parallel chains are simulated
<code>outsave</code>	Boolean variable determining whether or not the MCMC output is saved
<code>npara</code>	Number of parallel chains

Value

This function produces plots and, if `outsave` is `TRUE`, it produces a `list` with value the MMC sample(s).

Author(s)

Christian P. Robert and George Casella

References

Chapter 8 of **EnteR Monte Carlo Statistical Methods**

See Also

[sqaradap](#)

Examples

```
ousqar=sqar(outsave=TRUE)
```

`sqaradap`*Illustration of the dangers of doing adaptive MCMC on a noisy squared AR model*

Description

This function constructs a non-parametric proposal after each iteration of the MCMC algorithm, based on the earlier simulations. It shows how poorly this "natural" solution fares.

Usage

```
sqaradap(T = 10^4, TT = 10^4, scale = 0.5, factor = 1)
```

Arguments

<code>T</code>	Number of primary MCMC iterations
<code>TT</code>	Number of further adaptive MCMC iterations
<code>scale</code>	Scale of the normal random walk during the first T iterations
<code>factor</code>	Factor of the <code>bw.nrd0(xmc)</code> bandwidth estimation

Value

The function produces two graphs showing the lack of proper fit of the resulting sample.

Author(s)

Christian P. Robert and George Casella

References

Chapter 8 of **EnteR Monte Carlo Statistical Methods**

See Also

[sqar](#)

Examples

```
sqaradap()
```

test1	<i>Poor chi-square generator</i>
-------	----------------------------------

Description

This function is a direct implementation of the transform principle for the chi-square distribution, with poor performances even though the output is correct.

Usage

```
test1(Nsim = 10^4, df = 6)
```

Arguments

Nsim	Number of simulations
df	Degrees of freedom

Details

This function replicates `rchisq(Nsim, df)` and thus should not be used.

Value

This function produces a numerical vector of size *Nsim*.

warning

For efficient chi-square simulation, make sure to use `rchisq`

Author(s)

Christian P. Robert and George Casella

References

From Chapter 2 of **EnteR Monte Carlo Statistical Methods**

See Also

[test2](#), [rchisq](#)

Examples

```
mean(test1())  
# produces [1] 6.02526
```

test2	<i>Generic chi-square generator</i>
-------	-------------------------------------

Description

This function is a front-end for `rchisq`, designed for comparison with [test1](#).

Usage

```
test2(Nsim = 10^4, df = 6)
```

Arguments

<code>Nsim</code>	Number of simulations
<code>df</code>	Degrees of freedom

Details

This function replicates `rchisq(Nsim, df)` and thus should not be used.

Author(s)

Christian P. Robert and George Casella

References

From Chapter 2 of **EnteR Monte Carlo Statistical Methods**

See Also

[test1](#), `rchisq`

Examples

```
mean(test2())  
# produces [1] 5.955972
```

`test3`*Approximate Poisson generator*

Description

This function considers the possible values of a Poisson random variable to lay within $3 \cdot \sqrt{\lambda}$ of λ and is proposed for comparison with the regular `rpois` generator.

Usage

```
test3(Nsim = 10^4, lambda = 100)
```

Arguments

<code>Nsim</code>	Number of simulations
<code>lambda</code>	Poisson parameter

Details

This function replicates `rpois(Nsim, lambda)` and thus should not be used.

Value

This function produces a integer vector of size `Nsim`

warning

The output is not an exact Poisson sample because of the truncation on the support. Use `rpois` instead.

Author(s)

Christian P. Robert and George Casella

References

From Chapter 2 of **EnteR Monte Carlo Statistical Methods**

See Also

[test4](#), `rpois`

Examples

```
mean(test3(Nsim=10^4))  
# providing [1] 114
```

`test4`*Replicate Poisson generator*

Description

This function is a front-end for the standard `{rpois}` Poisson generator.

Usage

```
test4(Nsim = 10^4, lambda = 100)
```

Arguments

<code>Nsim</code>	Number of simulations
<code>lambda</code>	Poisson parameter

Details

This function replicates `rpois(Nsim, lambda)` and thus should not be used.

Value

This function produces a integer vector of size `Nsim`.

Author(s)

Christian P. Robert and George Casella

References

From Chapter 2 of **EnteR Monte Carlo Statistical Methods**

See Also

[test3](#), `rpois`

Examples

```
mean(test4())  
# produces [1] 100.0273
```

Index

*Topic **datagen**

betagen, 2
challenge, 4
gibbsmix, 9
mhmix, 14
mochoice, 15
pimamh, 18
test1, 28
test2, 29
test3, 30
test4, 31

*Topic **datasets**

challenger, 5
Energy, 8
logima, 12

*Topic **distribution**

betagen, 2
dmunorm, 6
dyadic, 7
hastings, 9
normbyde, 17
rdirichlet, 22
rmunorm, 24
test1, 28
test2, 29
test3, 30
test4, 31

*Topic **hplot**

adapump, 1
betagen, 2
Braking, 3
challenge, 4
gibbsmix, 9
hastings, 9
jamestein, 10
kscheck, 11
mhmix, 14
mump, 16
pimamh, 18

randogibs, 20
randomeff, 21
reparareff, 23
SAmix, 25
sqar, 26
sqaradap, 27

*Topic **optimize**

EMcenso, 7
maximple, 13
pimamh, 18
pimax, 19
randogit, 20
SAmix, 25

acf, 23
adapump, 1, 12

betagen, 2
Braking, 3

challenge, 4
challenger, 4, 5, 5

dmunorm, 6, 24
dyadic, 7, 13

EMcenso, 7
Energy, 8, 21

gibbsmix, 9

hastings, 9

jamestein, 10

kscheck, 2, 11, 16

logima, 12

maximple, 13
mhmix, 14
mochoice, 15

mump, 16

normbyte, 17

pimamh, 18, 19

pimax, 18, 19

randogibs, 20

randogit, 20

randomeff, 21, 23

rbeta (*betagen*), 2

rchisq (*test1*), 28

rdirichlet, 22

reparareff, 23

rmunorm, 24

SAmix, 25

sqar, 26, 27

sqaradap, 26, 27

test1, 28, 29

test2, 28, 29

test3, 30, 31

test4, 30, 31