R-package FME : MCMC tests

Karline Soetaert
NIOO-CEME
The Netherlands

Abstract

This vignette tests the markov chain monte carlo (MCMC) implementation of Rpackage **FME** (Soetaert 2009).

It includes the delayed rejection and adaptive metropolis algorithm (Haario, Laine, Mira, and Saksman 2006)

Keywords: markov chain monte carlo, delayed rejection, adapative metropolis, MCMC, DRAM, R.

1. Introduction

Here some functions from R-package **FME** are tested. As it is a test of the function implementation, this does not mean that the results obtained are meaningful.

It includes two tests of the implemented MCMC method:

- The "banana" function, sampling from a curvilinear function ((Laine 2008))
- A simple chemical model, fitted to a data series ((Haario *et al.* 2006))

Other tests are in the following vignettes:

- "FMEsteady", a steady-state solution of a partial differential equation
- "FMEdyna", a dynamic ordinary differential equation model and
- "FMEother", a monod function, fitted to a data-series.

2. The banana

2.1. the model

This example is from Laine (2008).

A banana-shaped function is created by distorting a two-dimensional Gaussian distribution, with mean = 0 and a covariance matrix τ) with unity variances and covariance of 0.9.

$$\tau = \begin{bmatrix} 0.9 & 0 \\ 0 & 0.9 \end{bmatrix}$$

The distortion is along the second-axis only and given by:

$$y_1 = x_1$$
$$y_2 = x_2 + x_1^2 + 1$$

2.2. R-implementation

First the banana function is defined.

```
> Banana <- function (x1,x2)
+ {
+   return(x2 - (x1^2+1))
+ }
```

We also need a function that estimates the probability of a multinormally distributed vector

```
> pmultinorm <- function(vec,mean,Cov)
+
+ {
+  diff <- vec - mean
+  ex   <- -0.5*t(diff) %*% solve(Cov) %*% diff
+  rdet   <- sqrt(det(Cov))
+  power  <- -length(diff)*0.5
+  return((2.*pi)^power / rdet * exp(ex))
+ }
```

The target function returns -2 *log (probability) of the value

```
> BananaSS <- function (p)
+ {
+   P <- c(p[1],Banana(p[1],p[2]))
+   Cov <- matrix(nr=2,data=c(1,0.9,0.9,1))
+  -2*sum(log(pmultinorm(P,mean=0,Cov=Cov)))
+ }
```

The initial proposal covariance (`jump`) is the identity matrix with a variance of 5. The simulated chain is of length 1000 (`niter`). The `modMCMC` function prints the % of accepted runs. More information is in item `count` of its return element.

The First Markov chain is generated with the simple metropolis hastings (MH) algorithm

```
> MCMC <- modMCMC(f=BananaSS, p=c(0,0.5), jump=diag(nrow=2,x=5),
+                 niter=1000)
```

```
number of accepted runs: 99 out of 1000 (9.9%)

> MCMC$count

     dr_steps     Alfasteps  num_accepted num_covupdate
            0             0            99             1
```

Next we use the adaptive metropolis (AM) algorithm and update the proposal every 100 runs (`updatecov`)

```
> MCMC2 <- modMCMC(f=BananaSS, p=c(0,0.5), jump=diag(nrow=2,x=5),
+                  updatecov=100,niter=1000)

number of accepted runs: 200 out of 1000 (20%)

> MCMC2$count

     dr_steps     Alfasteps  num_accepted num_covupdate
            0             0           200            10
```

Then the metropolis algorithm with delayed rejection (DR) is applied; upon rejection one next parameter cadidate is tried (`ntrydr`). (note `ntrydr=1` means no delayed rejection steps).

```
> MCMC3 <- modMCMC(f=BananaSS, p=c(0,0.5), jump=diag(nrow=2,x=5),
+                  ntrydr=2,niter=1000)

number of accepted runs: 547 out of 1000 (54.7%)

> MCMC3$count

     dr_steps     Alfasteps  num_accepted num_covupdate
          891          2673           547             1
```

Finally the adaptive metropolis with delayed rejection (DRAM) is used. (Here we also estimate the elapsed CPU time - `print(system.time())` does this)

```
> print(system.time(
+ MCMC4 <- modMCMC(f=BananaSS, p=c(0,0.5), jump=diag(nrow=2,x=5),
+                  updatecov=100,ntrydr=2,niter=1000)
+ ))

number of accepted runs: 558 out of 1000 (55.8%)
   user  system elapsed
   0.90    0.00    0.91

> MCMC4$count
```

```
     dr_steps     Alfasteps  num_accepted num_covupdate
          891          2673           558            10
```

We plot the generated chains for both parameters and for the four runs in one plot. Calling `plot` with `mfrow=NULL` prevents the plotting function to overrule these settings.

```
> par(mfrow=c(4,2))
> par(mar=c(2,2,4,2))
> plot(MCMC ,mfrow=NULL,main="MH")
> plot(MCMC2,mfrow=NULL,main="AM")
> plot(MCMC3,mfrow=NULL,main="DR")
> plot(MCMC4,mfrow=NULL,main="DRAM")
> mtext(outer=TRUE,side=3,line=-2,at=c(0.05,0.95),c("y1","y2"),cex=1.25)
> par(mar=c(5.1,4.1,4.1,2.1))
```

The 2-D plots show the banana shape:

```
> par(mfrow=c(2,2))
> xl <- c(-3,3)
> yl <- c(-1,8)
> plot(MCMC$pars,main="MH",xlim=xl,ylim=yl)
> plot(MCMC2$pars,main="AM",xlim=xl,ylim=yl)
> plot(MCMC3$pars,main="DR",xlim=xl,ylim=yl)
> plot(MCMC4$pars,main="DRAM",xlim=xl,ylim=yl)
```

Finally, we test convergence to the original distribution. This can best be done by estimating means and covariances of the transformed parameter values.

```
> trans <- cbind(MCMC4$pars[,1],Banana(MCMC4$pars[,1],MCMC4$pars[,2]))
> colMeans(trans)    # was:c(0,0)

[1] -0.2146383 -0.1918152

> sd(trans)          # was:1

[1] 1.120227 1.129278

> cor(trans)         # 0.9 off-diagonal

          [,1]      [,2]
[1,] 1.0000000 0.9085867
[2,] 0.9085867 1.0000000
```

3. A simple chemical model

This is an example from (Haario *et al.* 2006). We fit two parameters that describe the dynamics in the following reversible chemical reaction:

$$\mathrm{A} \rightleftharpoons [k_2]k_1\mathrm{B}$$

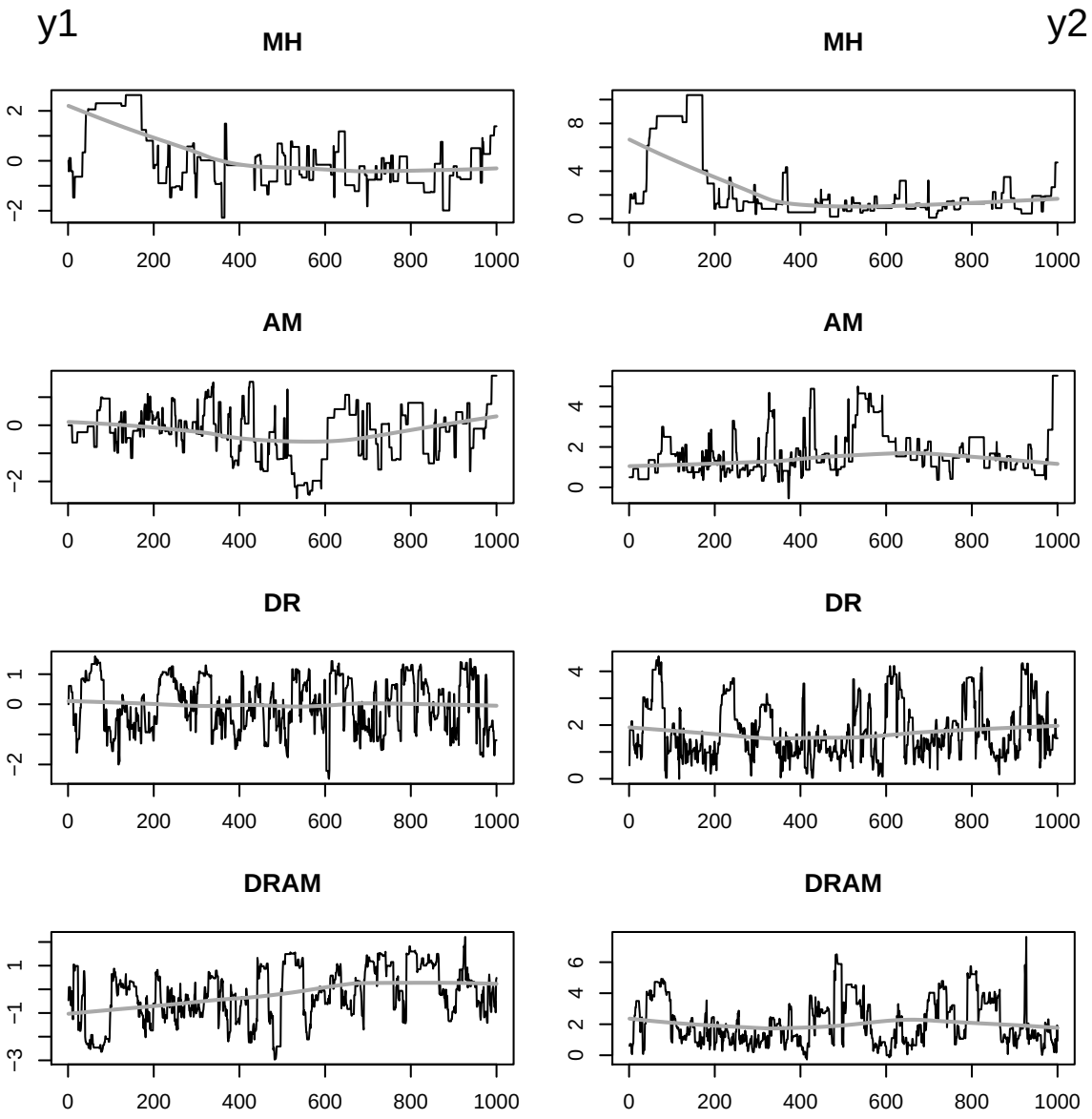

Figure 1: The MCMC chains for the four methods - see text for R-code

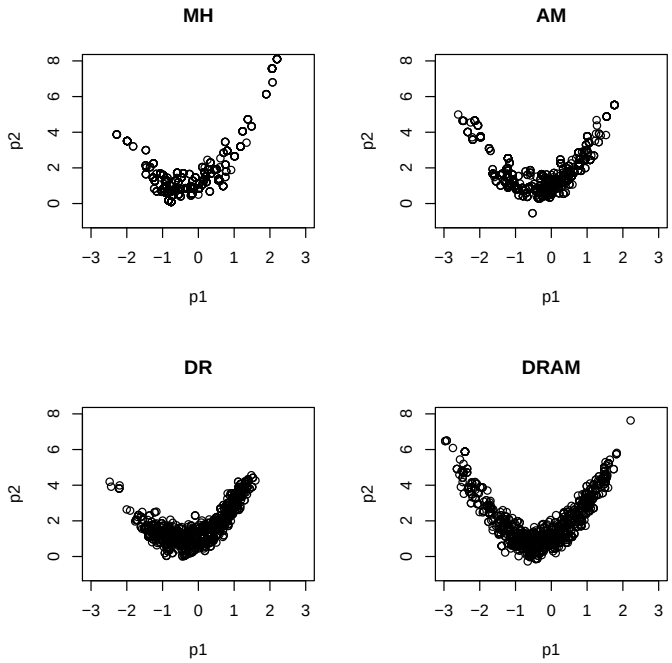

Figure 2: The bananas - see text for R-code

Here k_1 is the forward, k_2 the backward rate coefficient.
The ODE system is written as:

$$\frac{dA}{dt} = -k_1 \cdot A + k_2 \cdot B$$
$$\frac{dB}{dt} = +k_1 \cdot A - k_2 \cdot B$$

with initial values $A_0 = 1$, $B_0 = 0$.
The analytical solution for this system of differential equations is given in (Haario *et al.* 2006).
First a function is defined that takes as input the parameters and that returns the values of the concentrations A and B, at selected output times.

```
> Reaction <- function (k, times)
+ {
+   fac <- k[1]/(k[1]+k[2])
+   A   <- fac + (1-fac)*exp(-(k[1]+k[2])*times)
+   return(data.frame(t=times,A=A))
+ }
```

All the concentrations were measured at the time the equilibrium was already reached. The data are the following:

```
> Data     <- data.frame(
+   times = c(2,     4,     6,     8,     10   ),
+   A     = c(0.661, 0.668, 0.663, 0.682, 0.650))
> Data

  times     A
1     2 0.661
2     4 0.668
3     6 0.663
4     8 0.682
5    10 0.650
```

We need parameter priors to prevent the model parameters from drifting to infinite values. The prior is taken to be a broad Gaussian distribution with mean (2,4) and standard deviation = 200 for both.

The prior function returns the weighted sum of squared residuals of the parameter values with the expected value.

```
> Prior <- function(p)
+     return( sum(((p-c(2,4))/200)^2 ))
```

First the model is fitted to the data; we restrict the parameter values to be in the interval [0,1].

```
> residual <- function(k) return(Data$A - Reaction(k,Data$times)$A)
> Fit <- modFit(p=c(k1=0.5,k2=0.5),f=residual,lower=c(0,0),upper=c(1,1))
> (sF <- summary(Fit))

Parameters:
   Estimate Std. Error t value Pr(>|t|)
k1   1.0000     0.3944   2.536   0.0850 .
k2   0.5123     0.1928   2.657   0.0765 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.01707 on 3 degrees of freedom

Parameter correlation:
      k1    k2
k1 1.000 0.996
k2 0.996 1.000
```

The residual error of the fit is used as initial model variance, the scaled covariance matrix of the fit is used as the proposal distribution (to generate new parameter values). As the covariance matrix is nearly singular this is not a very good approximation. The initial MCMC method, using the Metropolis-Hastings method does not converge. The MCMC is initiated with the best-fit parameters; the parameters are restricted to be positive numbers (`lower`).

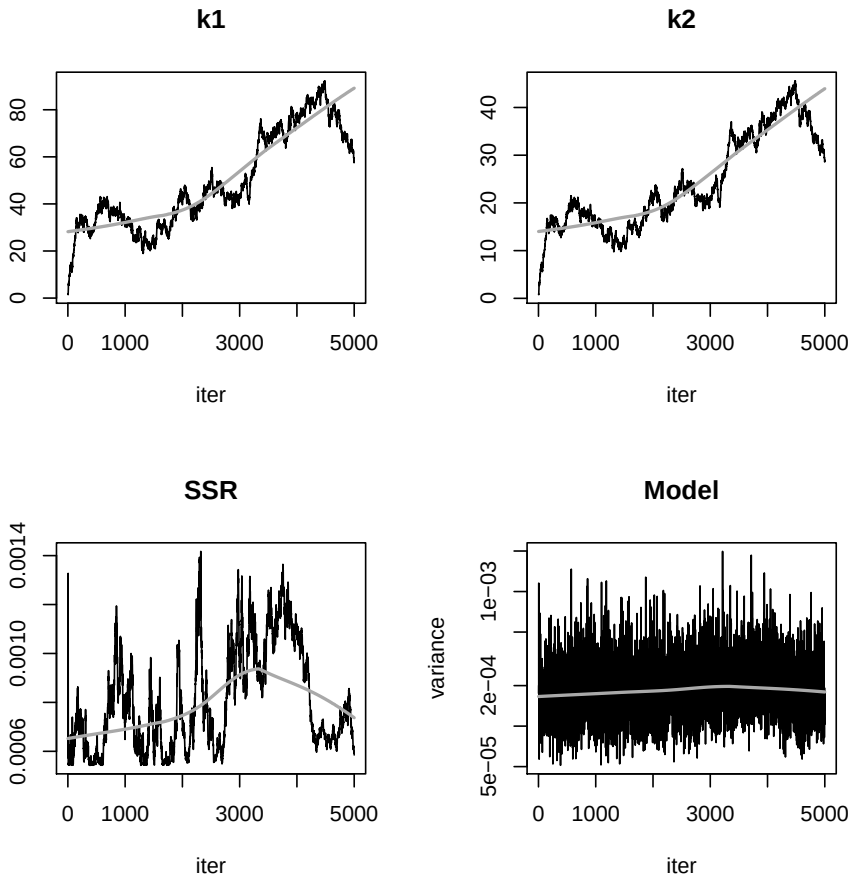

Figure 3: Metropolis-Hastings MCMC of the chemical model - see text for R-code

```
> mse <- sF$modVariance
> Cov <- sF$cov.scaled * 2.4^2/2
> print(system.time(
+ MCMC <- modMCMC(f=residual, p=Fit$par, jump=Cov, lower=c(0,0),
+                  var0=mse, wvar0=1, prior=Prior, niter=5000)
+ ))

number of accepted runs: 4893 out of 5000 (97.86%)
   user  system elapsed
   2.97    0.00    2.98
```

The number of accepted runs is much too high, and indeed the MCMC has not at all converged...

```
> plot(MCMC,Full=TRUE)
```

Better convergence is achieved by the adaptive metropolis, updating the proposal every 100 runs

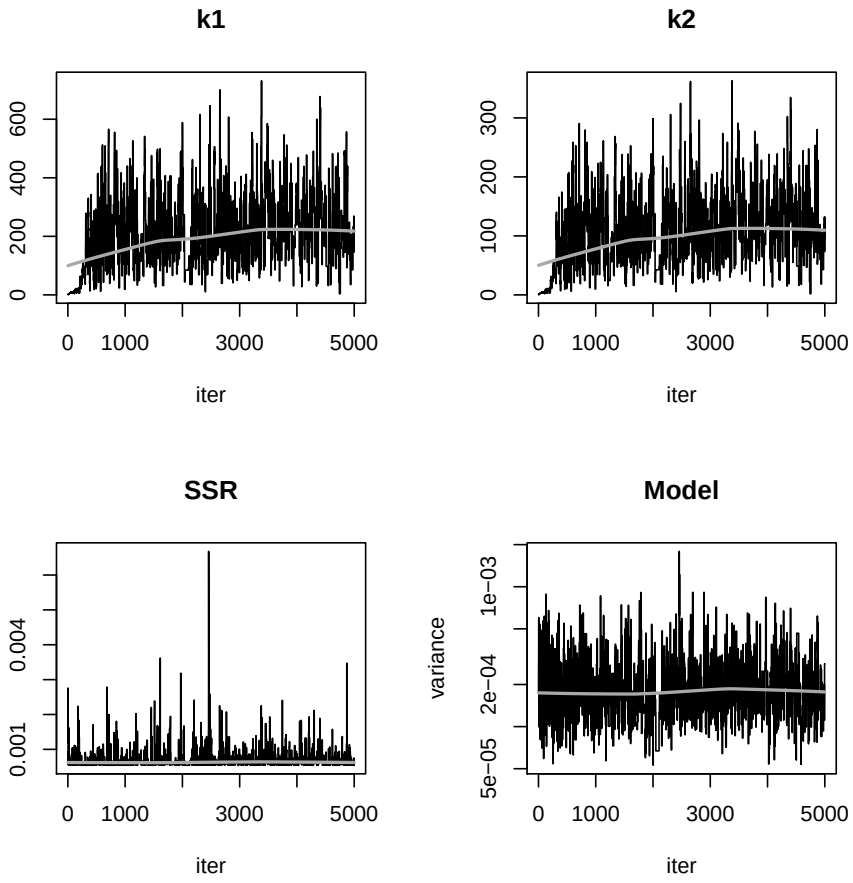

Figure 4: Adaptive Metropolis MCMC of the chemical model - see text for R-code

```
> MCMC2<- modMCMC(f=residual, p=Fit$par, jump=Cov, updatecov=100, lower=c(0,0),
+                 var0=mse, wvar0=1, prior=Prior,niter=5000)    #

number of accepted runs: 1720 out of 5000 (34.4%)

> plot(MCMC2,Full=TRUE)
```

The correlation between the two parameters is clear:

```
> pairs(MCMC2)
```

4. finally

This vignette is a Sweave (Leisch 2002) translation of part of the **FME** examples.

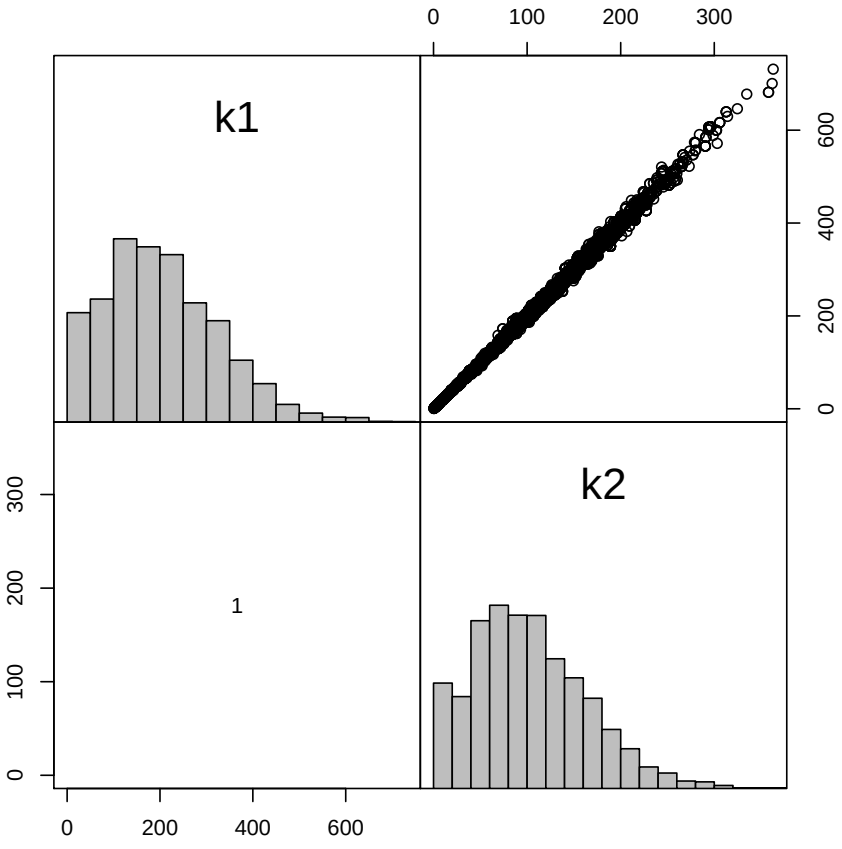

Figure 5: Pairs plot of the Adaptive Metropolis MCMC of the chemical model - see text for R-code

References

Haario H, Laine M, Mira A, Saksman E (2006). "DRAM: efficient adaptive MCMC." *Statistical Computing*, **16**, 339–354.

Laine M (2008). *Adaptive MCMC methods with applications in environmental and geophysical models*. Finnish meteorological institute contributions n0 69 -ISBN 978-951-697-662-7.

Leisch F (2002). "Sweave: Dynamic Generation of Statistical Reports Using Literate Data Analysis." In W Härdle, B Rönz (eds.), "Compstat 2002 - Proceedings in Computational Statistics," pp. 575–580. Physica Verlag, Heidelberg. ISBN 3-7908-1517-9, URL http://www.stat.uni-muenchen.de/~leisch/Sweave.

Soetaert K (2009). *FME: A Flexible Modelling Environment for inverse modelling, sensitivity, identifiability, monte carlo analysis.* R package version 1.0.

Affiliation:

Karline Soetaert
Centre for Estuarine and Marine Ecology (CEME)
Netherlands Institute of Ecology (NIOO)
4401 NT Yerseke, Netherlands
E-mail: k.soetaert@nioo.knaw.nl
URL: http://www.nioo.knaw.nl/ppages/ksoetaert