evolMC demo

Grady Weyenberg

November 8, 2013

evolMC is a framework for doing Monte-Carlo simulations.
We wish to use a metropolis sampler to draw from a distribution with density

$$f(x) \propto \frac{sin(x)}{x} \cdot 1_{(0,\pi)}(x).$$

```
library(evolMC)
fn <- function(x) sin(x)/x * (0 < x) * (x < pi)
```

We can use a uniform distribution on $(-1, 1)$ to propose distances to jump from the current location.

```
propose <- function(x) x + runif(length(x), -1, 1)
```

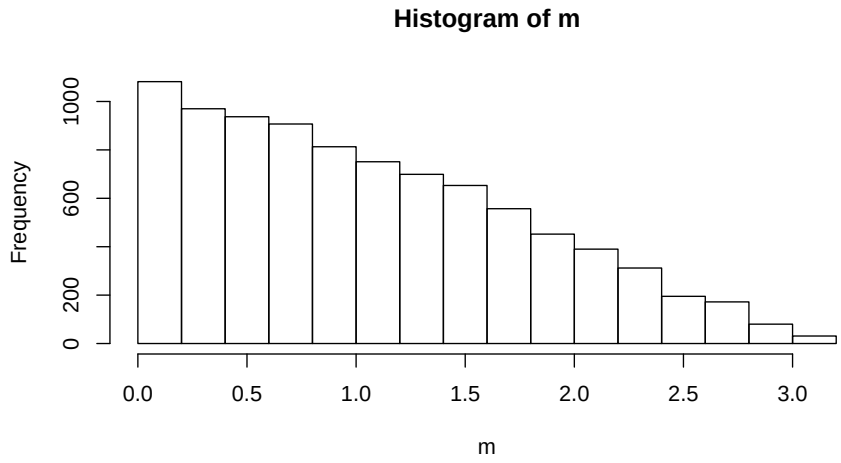
Since the proposal distribution is symmetric, this is enough information to implement a Metropolis updater.

```
updater <- metropolis(fn, propose)
```

A Markov chain is formed by iteratively calling the updating function starting with some initial value.

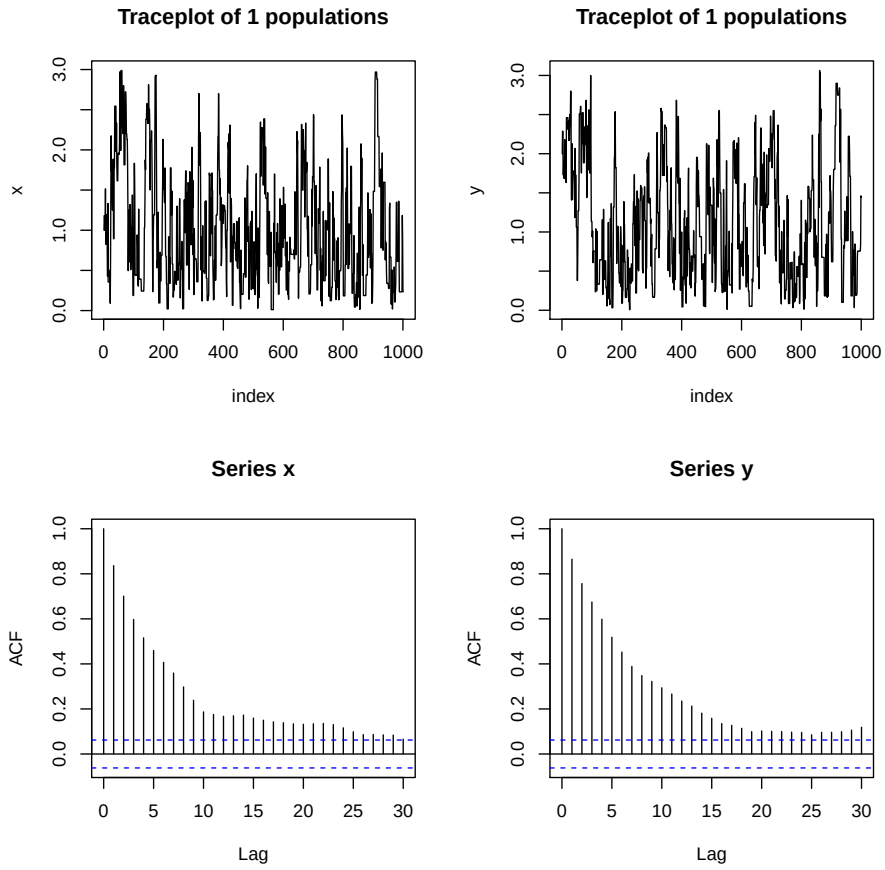
```
chain <- iterate(10000, updater, init = matrix(1))
summary(chain)

## Discarding first 1000 states.
##   mean     se   2.5%  97.5%
## 1.0606 0.7365 0.0399 2.6534
```

```
hist(chain, breaks = "fd")

## Discarding first 1000 states.
```



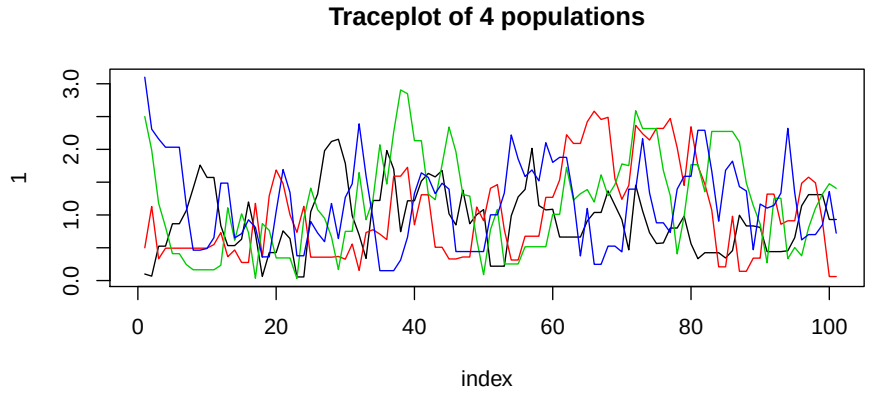
Of course, multivariate distributions may also be sampled.

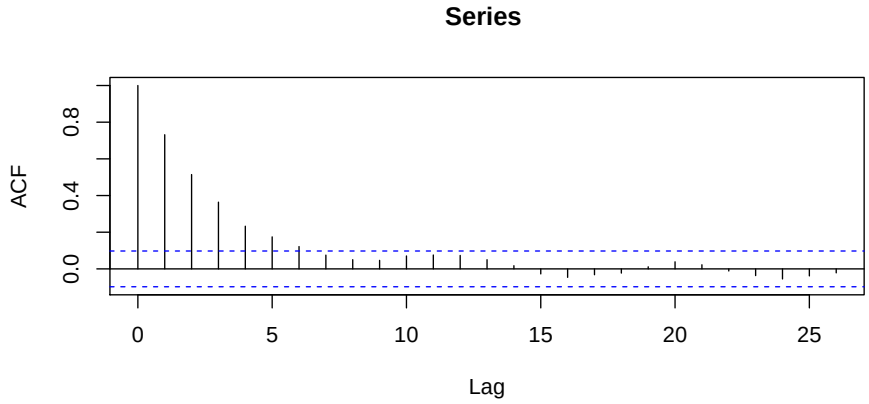
```
mvtarget <- function(x) prod(sin(x)/x) * all(x > 0) * all(x < pi)
mvupdate <- metropolis(mvtarget, propose)
chain2 <- iterate(1000, mvupdate, cbind(x = 1, y = 2))
summary(chain2)

## Discarding first 100 states.
##             x      y
## mean  0.92918 0.9995
## se    0.67649 0.7103
## 2.5%  0.04376 0.0510
## 97.5% 2.43684 2.5511

plot(chain2)
```



If the target density and proposal generator are "carefully selected" then multiple chains can be run in parallel.

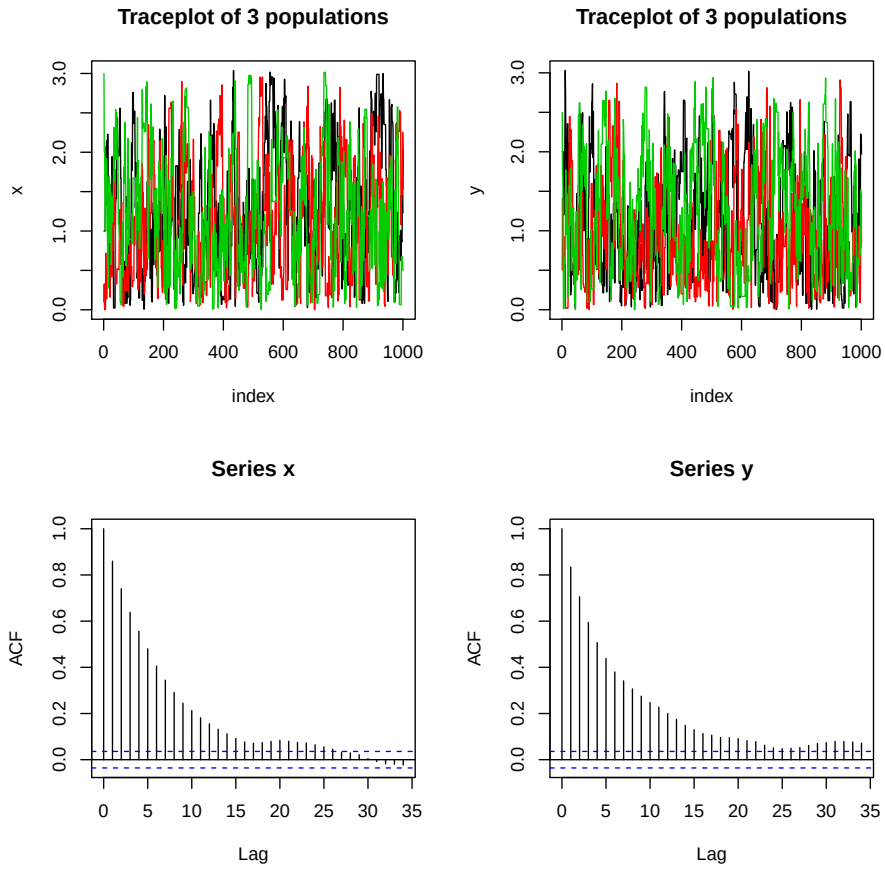
```
chain2 <- iterate(100, updater, init = rbind(0.1, 0.5, 2.5, 3.1))
plot(chain2)
```





Multiple chains of multivariate samples are also possible

```
mvt2 <- function(x) apply(x, 1, mvtarget)
mvup2 <- metropolis(mvt2, propose)
init <- rbind(c(x = 1, y = 2), c(0.1, 0.5), c(3, 2.5))
chain3 <- iterate(1000, mvup2, init)
plot(chain3)
```



The chain methods, such as summary, plot, and hist, assume that a state is a matrix with individuals in rows, and multiple variables in columns, however this is not required by the functions which generate the chain. The main requirement is that the following expression is valid and that the resulting set of logical values can be used to "correctly" subset the state object. (Where "target" is the density function, and "state" is the current state of the chain.)

```
target(state) < 1
```