hMC example

** Note: Adapted from code in Appendix C.4 of Bayesian Data Analysis by Gelman et al **

preliminaries

```r
# clear workspace
rm(list=ls())

# set random seed
set.seed(5873)
```

define simulation objects

```r
# log-likelihood function
log_p_theta = function(theta) {
  theta_1 = theta[1]
  theta_2 = theta[2]
  radius = sqrt(theta_1 ^ 2 + theta_2 ^ 2)
  log_like = dnorm(radius, mu, sigma, log = TRUE)
  return(log_like)
}

# gradient of log-likelihood function
gradient_theta = function(theta) {
  theta_1 = theta[1]
  theta_2 = theta[2]
  radius = sqrt(theta_1 ^ 2 + theta_2 ^ 2)
  d_theta_1 = - (theta_1 * (radius - 1)) / (sigma^2 * radius)
  d_theta_2 = - (theta_2 * (radius - 1))/ (sigma^2 * radius)
  return( c(d_theta_1, d_theta_2) )
}

# hmc_iteration
hmc_iteration = function(theta, epsilon, L, M) {
  M_inv = 1/M
  phi = rnorm(2, 0, sqrt(M))
  theta_old = theta
  log_p_old = log_p_theta(theta) - 0.5 * sum(M_inv * phi^2)
  phi = phi + 0.5 * epsilon * gradient_theta(theta)
  for (t in 1:L) {
    theta = theta + epsilon * M_inv * phi
    phi = phi + (if (t==L) 0.5 else 1) * epsilon * gradient_theta(theta)
  }
  log_p_star = log_p_theta(theta) - 0.5 * sum(M_inv * phi^2)
  r = exp(log_p_star - log_p_old)
  if (is.nan(r)) r = 0
  p_jump = min(r,1)
  if (runif(1) < p_jump) {
    theta_new = theta
    accept = 1 }
  else {
    theta_new = theta_old
    accept = 0
  }
  return(list(theta = theta_new, p_jump = p_jump, accept = accept))
}



# hmc run wrapper
hmc_run = function(starting_values, iter, epsilon, L, M) {
  chains = nrow(starting_values)
  d = ncol(starting_values)
  sims = array(NA, c(iter, chains, d),
               dimnames = list(NULL, NULL, colnames(starting_values)))
  p_jump = array(NA, c(iter, chains))
  accept = array(NA, c(iter, chains))
    warmup = 0.5 * iter
  for (j in 1:chains) {
    theta = starting_values[j,]
    for (t in 1:iter) {
      temp = hmc_iteration(theta, epsilon, L, M)
      p_jump[t,j] = temp$p_jump
      sims[t,j,] = temp$theta
      accept[t,j] = temp$accept
      theta = temp$theta #### KEY MISSING
    }
  }
  cat("Avg acceptance probs:",
      round(colMeans(p_jump[(warmup+1):iter,]),2), "\n")
  return(list(sims = sims[(warmup+1):iter,,],
              p_jump = p_jump[(warmup+1):iter,],
              accept = accept[(warmup+1):iter,]))
}
```

run simulations

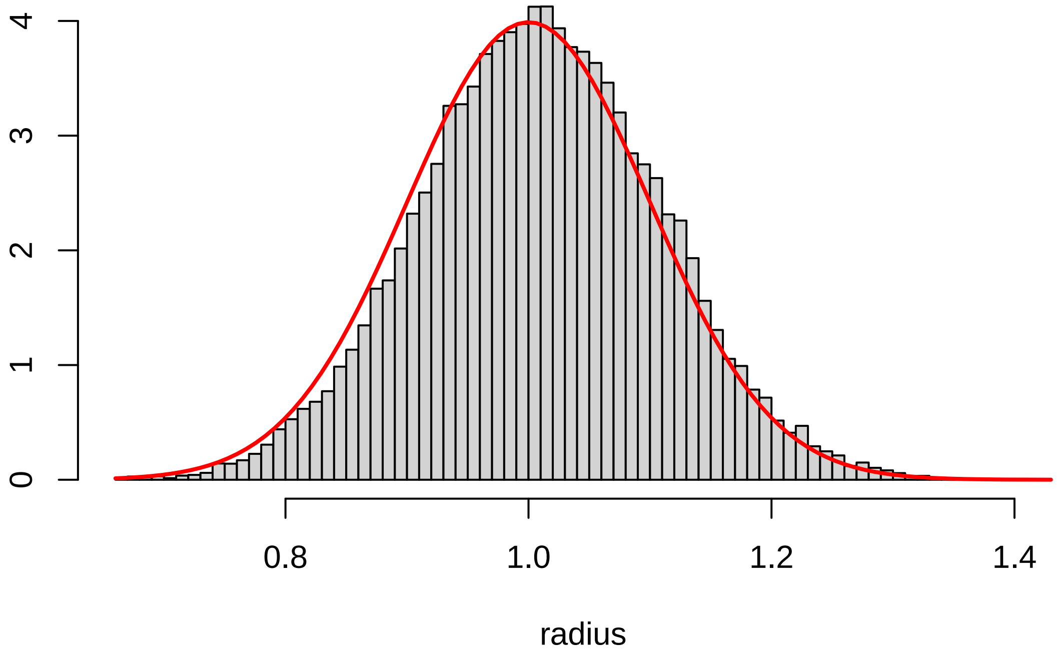
```
# example parameters
sigma = 0.1
mu = 1

epsilon = 0.05
L = 10


parameter_names = c("theta_1", "theta_2")
d = length(parameter_names)
```

```
chains = 2

mass_vector = c(sigma, sigma)

starts = array(NA, c(chains, d),
               dimnames = list(NULL, parameter_names))
starts[1,] = c(1,0)
starts[2,] = - starts[1,]


# sample
M1 = hmc_run(starting_values = starts,
             iter = 50000,
             epsilon = epsilon,
             L = L,
             M = mass_vector)
```

```
## Avg acceptance probs: 0.86 0.86
```

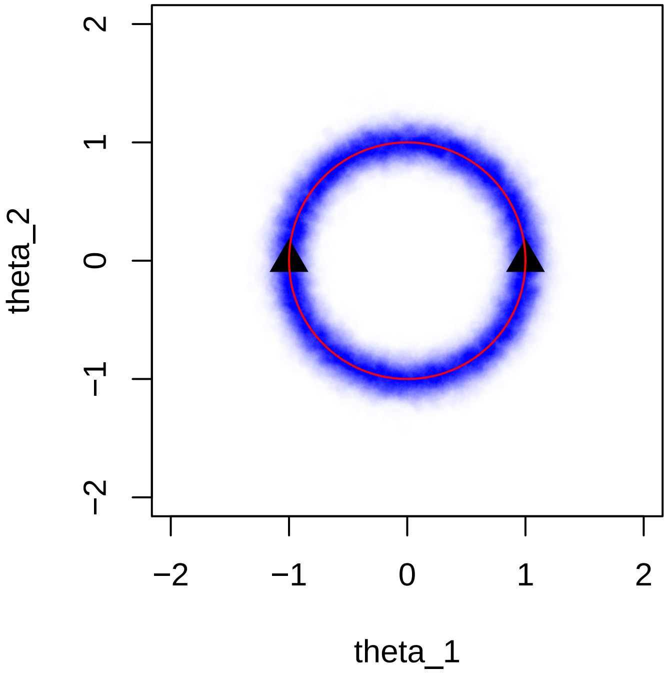
visualize

```
xSamples = as.vector(M1$sims[,,1])
ySamples = as.vector(M1$sims[,,2])
radiusSamples = sqrt(xSamples ^ 2 + ySamples ^ 2)

#plot radius histogram
hist(radiusSamples, breaks = 100, prob= TRUE, main = "", xlab="radius", ylab="")
curve(dnorm(x, mu, sigma), col = "red", add=TRUE, lwd=2)
```



```
#plot samples with small fraction of actual jumps
library("plotrix")
par(pty="s")
```

```r
plot(1, type="n",
     xlab="theta_1", ylab="theta_2",
     xlim=c(-2,2), ylim=c(-2,2))

#points(M1$sims[1:10,1,1],M1$sims[1:10,1,2], type="l")

points(xSamples,ySamples,
       col=rgb(red=0.0, green=0.0, blue=1.0, alpha=0.005),
       pch = 20)
for (n in 1:nrow(starts)) {
  points(starts[n,1], starts[n,2],pch = 17,col="black", cex = 2, lty=3)
}
draw.circle(0,0,mu, border="red")
```


theta_2
theta_1