Hamiltonian Monte Carlo

Dr. Jarad Niemi

Iowa State University

September 4, 2017

Parameter augmentation

Suppose we are interested in sampling from a posterior distribution for $\theta \in \mathbb{R}^d$

$$p(\theta|y) \propto p(y|\theta)p(\theta).$$

Now augment θ with moment variable $\omega \sim N_d(0, D)$ independent of $\theta|y$ such that

$$p(\theta|y) = \int p(\theta|\omega, y)p(\omega)d\omega = \int p(\theta|y)p(\omega)d\omega$$

To compare with Neal (2010), we have $q = \theta$, $p = \omega$,

$$U(\theta) = -\log[p(y|\theta)p(\theta)] = -\log p(\theta|y) - \log p(\theta),$$

and

$$K(\omega) = -\log p(\omega).$$

Hamiltonian Monte Carlo algorithm

Set tuning parameters

- L: the number of steps
- e: stepsize
- $D = \{d_i\}$: covariance matrix for ω

Let $\theta^{(i)}$ be the current value of the parameter θ. The leap-frog Hamiltonian Monte Carlo algorithm is

1. Sample $\omega \sim N_d(0, D)$.
2. Simulate Hamiltonian dynamics on location $\theta^{(i)}$ and momentum ω via the leapfrog method (or any reversible method that preserves volume). Call these updated values θ^* and $-\omega^*$.
3. Set $\theta^{(i+1)} = \theta^*$ with probability $\min\{1, \rho(\theta^{(i)}, \theta^*)\}$ where

$$\rho(\theta^{(i)}, \theta^*) = \frac{p(\theta^*|y)}{p(\theta^{(i)}|y)} \frac{p(\omega^*)}{p(\omega^{(i)})} = \frac{p(y|\theta^*)p(\theta^*)}{p(y|\theta^{(i)})p(\theta^{(i)})} \frac{N_d(\omega^*; 0, D)}{N_d(\omega^{(i)}; 0, D)}$$

   otherwise set $\theta^{(i+1)} = \theta^{(i)}$.

Leap-frog simulation of Hamiltonian dynamics

Given a current location $\theta(0)$ and momentum $\omega(0)$ at time 0, the leap-frog method can be used to approximate simulating Hamiltonian dynamics up to time Le using a series of L steps each of time e.

The algorithm is

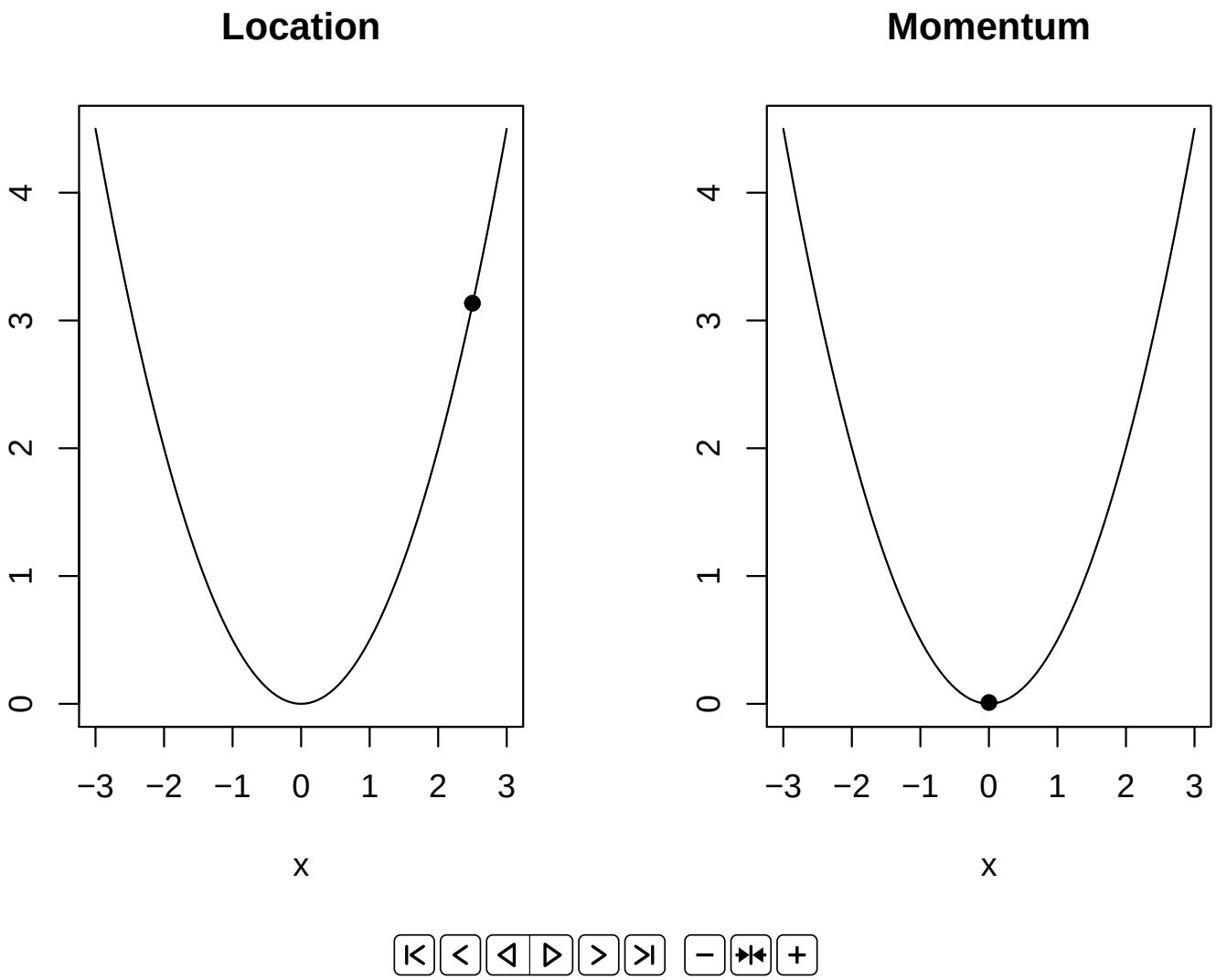
1. For $\ell = 1, \ldots, L$,
   a. For $i = 1, \ldots, d$, $\omega_i\left(\left[\ell - \frac{1}{2}\right] e\right) = \omega_i([\ell - 1]e) - \frac{e}{2}\frac{\partial U}{\partial \theta_i}(\theta([\ell - 1]e))$
   b. For $i = 1, \ldots, d$, $\theta_i(\ell e) = \theta_i([\ell - 1]e) + e\frac{\omega_i\left(\left[\ell - \frac{1}{2}\right]e\right)}{d_i}$
   c. For $i = 1, \ldots, d$, $\omega_i(\ell e) = \omega_i\left(\left[\ell - \frac{1}{2}\right] e\right) - \frac{e}{2}\frac{\partial U}{\partial \theta_i}(\theta(\ell e))$

where θ_i and ω_i are the i^{th} element of the location and momentum, respectively.

Leap-frog simulator

```
leap_frog = function(U, grad_U, e, L, theta, omega) {
  omega = omega - e/2 * grad_U(theta)

    for (l in 1:L) {
      theta = theta + e * omega
      if (l<L) omega = omega - e * grad_U(theta)
    }
    omega = omega - e/2 * grad_U(theta)
  return(list(theta=theta,omega=omega))
}
```

Leap-frog simulator



Reversibility

A reversible simulation means that

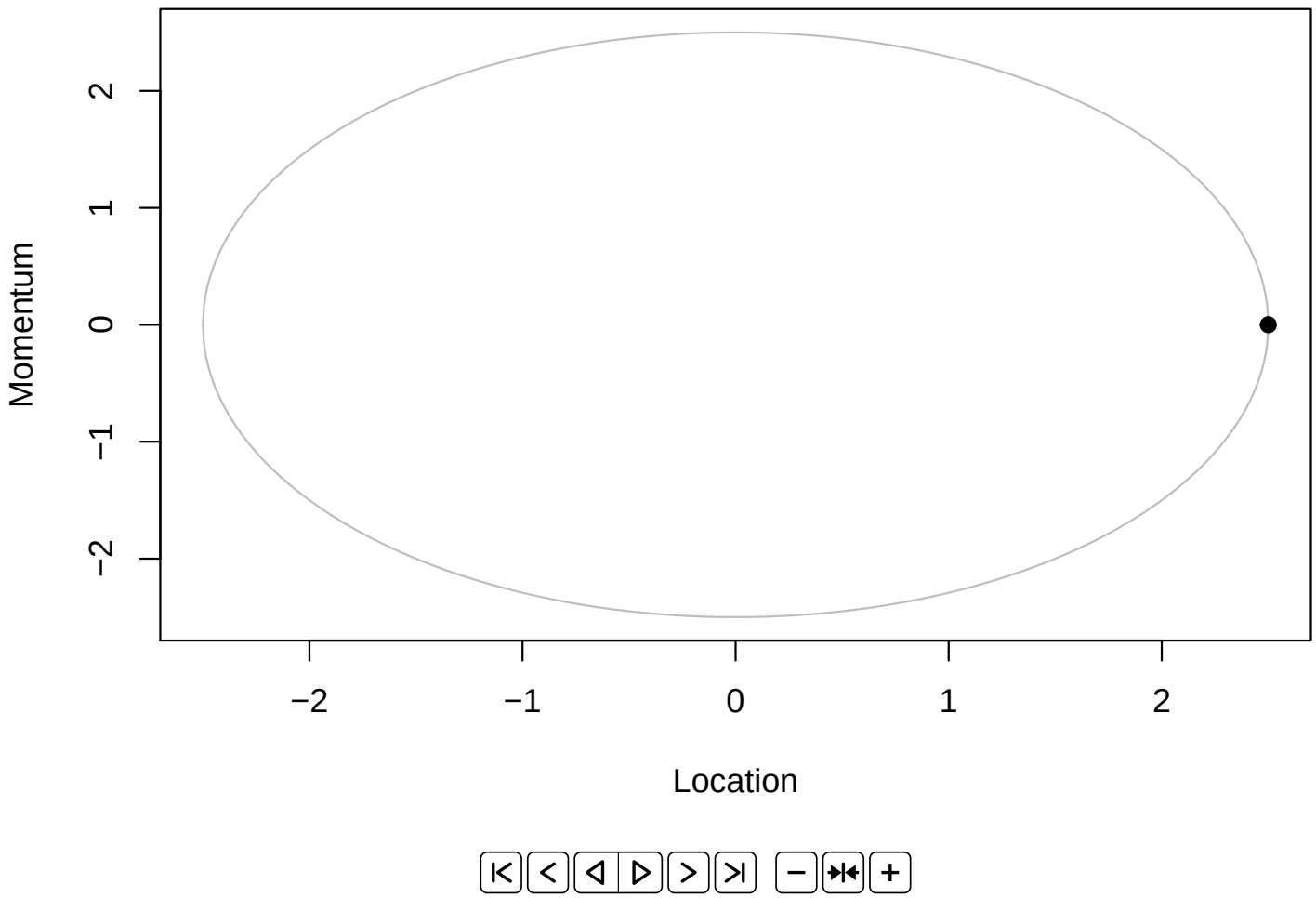
- if you simulate from (θ, ω) to (θ', ω') for some step size e and number of steps L then
- if you simulate from (θ', ω') for some step size e and number of steps L, you will end up at (θ, ω).

If we use q to denote our simulation "density", then reversibility means

$$q(\theta', \omega' | \theta, \omega) = q(\theta, \omega | \theta', \omega')$$

and thus in the Metropolis-Hastings calculation, the proposal is symmetric. In order to ensure reversibility of our proposal, we need to negate momentum after we complete the leap-frog simulation, but so long as $p(\omega) = p(-\omega)$ this will not affect our acceptance probability.

Leap-frog simulator



Volume preserving results in perfect acceptance

Recall that we accept with probability $\min\{1, \rho(\theta^{(i)}, \theta^*)\}$ where

$$\rho(\theta^{(i)}, \theta^*) = \frac{p(\theta^*|y)}{p(\theta^{(i)}|y)} \frac{p(\omega^*)}{p(\omega^{(i)})}$$

Volume is preserved if

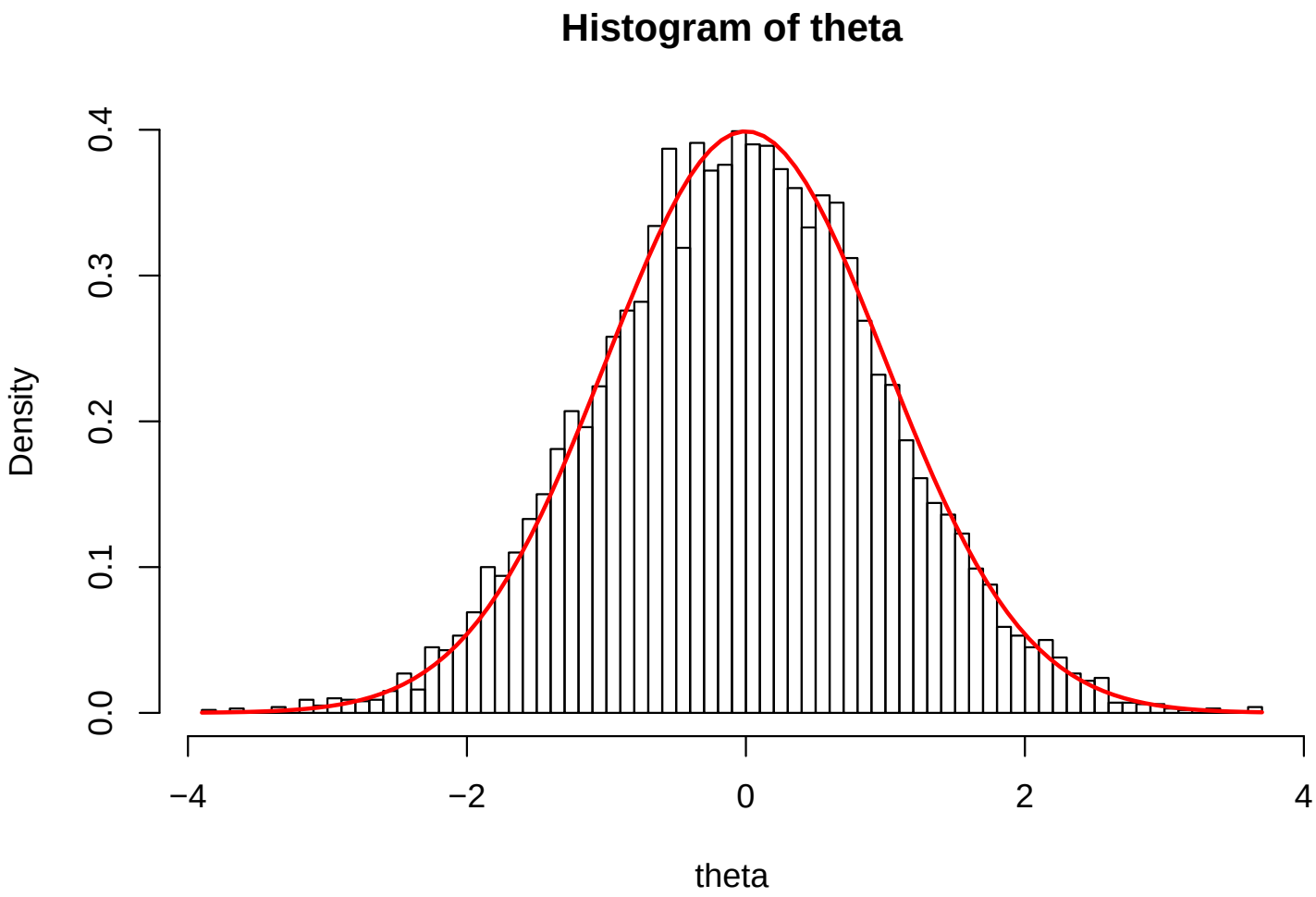
$$p(\theta^{(i)}|y)p(\omega^{(i)}) = p(\theta^*|y)p(\omega^*) \implies \frac{p(\theta^*|y)}{p(\theta^{(i)}|y)} \frac{p(\omega^*)}{p(\omega^{(i)})} = 1$$

This will only be the case if the simulation is perfect! But we have discretization error. The acceptance probability accounts for this error.

```
HMC_neal = function(U, grad_U, epsilon, L, current_q) {
  q = current_q
  p = rnorm(length(q),0,1)
  current_p = p

  p = p-epsilon*grad_U(q)/2

  for (i in 1:L) {
    q = q+epsilon*p
    if (i!=L) p = p -epsilon * grad_U(q)
  }
  p = p-epsilon * grad_U(q)/2

  p = -p

  current_U = U(current_q)
  current_K = sum(current_p^2)/2
  proposed_U = U(q)
  proposed_K = sum(p^2)/2

  if (runif(1) < exp(current_U-proposed_U+current_K-proposed_K))
  {
    return(q)
  }
  else {
    return(current_q)
  }
}
```

```
HMC = function(n_reps, log_density, grad_log_density, tuning, initial) {
  theta = rep(0, n_reps)
  theta[1] = initial$theta

  for (i in 2:n_reps) theta[i] = HMC_neal(U = function(x) -log_density(x),
                                          grad_U = function(x) -grad_log_density(x),
                                          e = tuning$e,
                                          L = tuning$L,
                                          theta[i-1])
  theta
}
```

```r
theta = HMC(1e4, function(x) -x^2/2, function(x) -x, list(e=1,L=1), list(theta=0))
hist(theta, freq=F, 100)
curve(dnorm, add=TRUE, col='red', lwd=2)
```



Tuning parameters

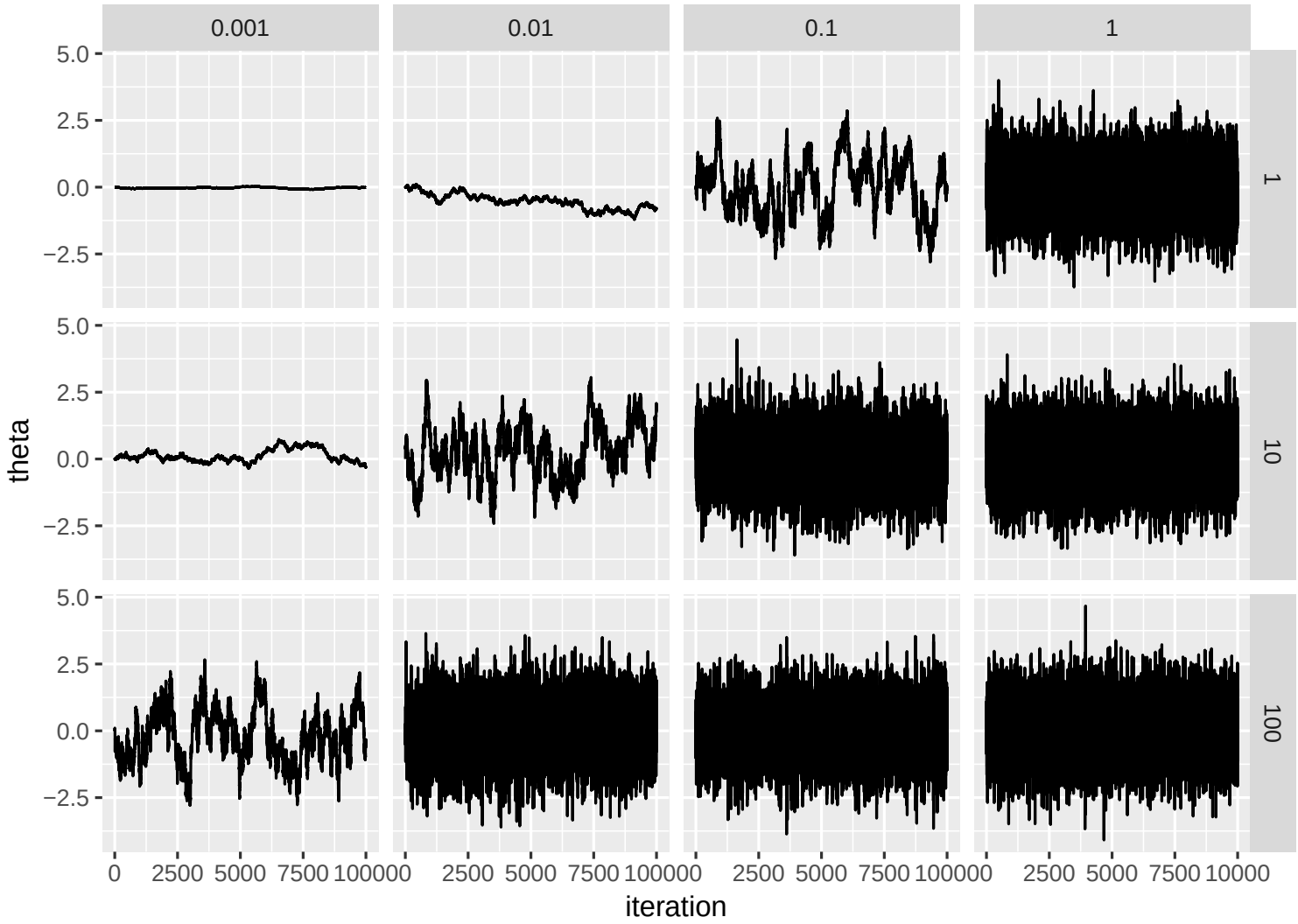
There are three tuning parameters:

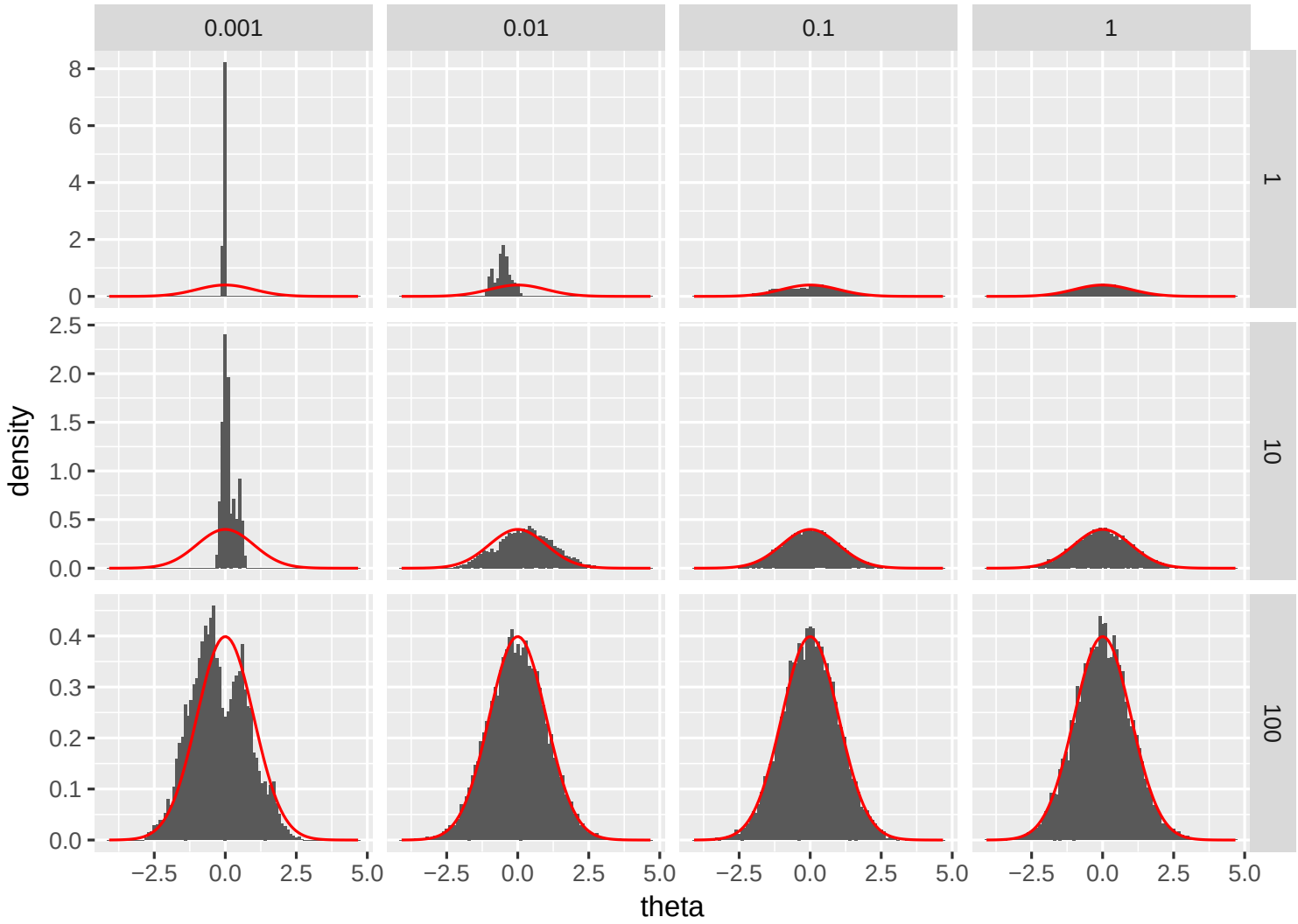
- e: step size
- L: number of steps
- D: covariance matrix for momentum

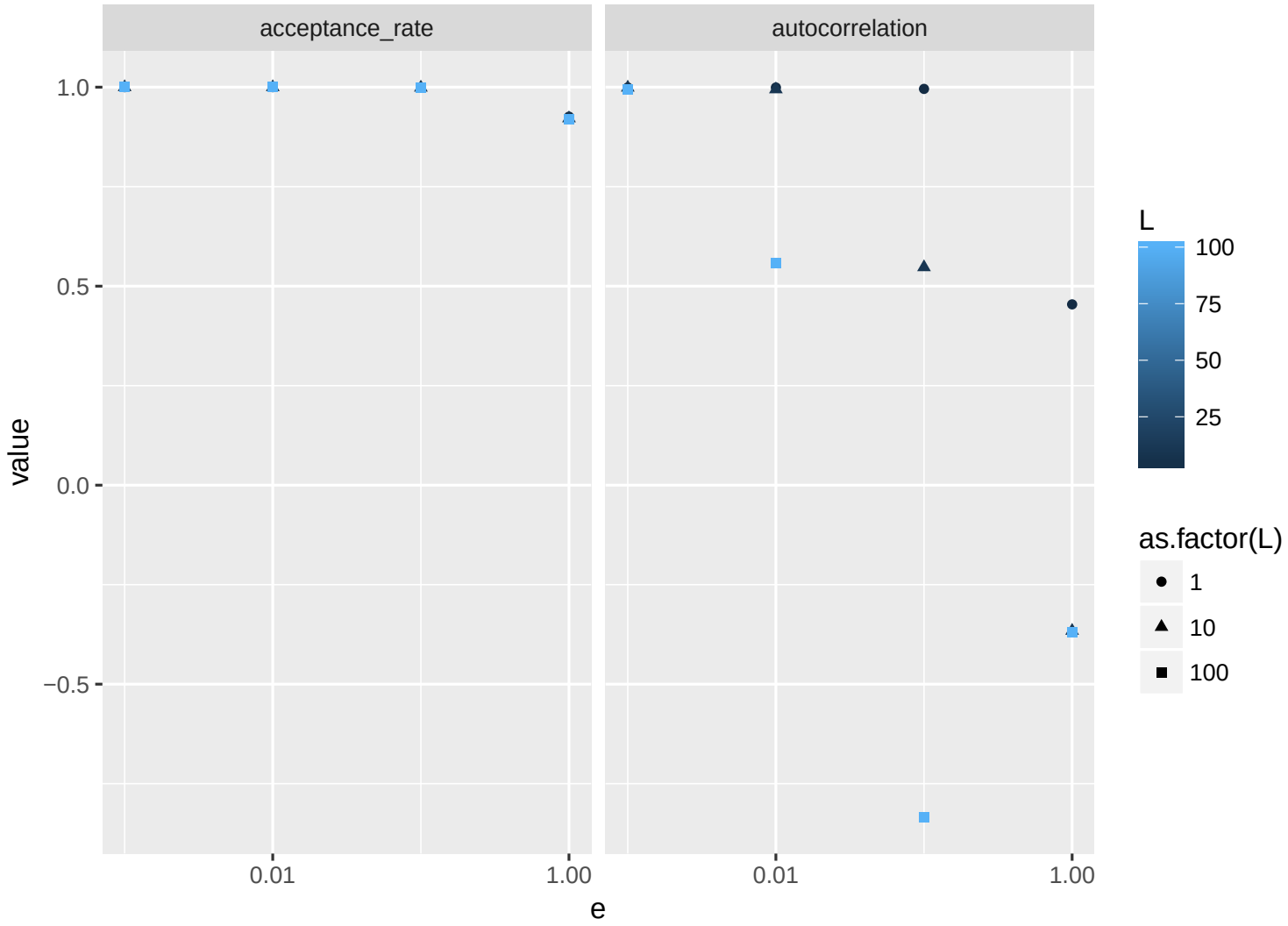
Let $\Sigma = V(\theta|y)$, then an optimal normal distribution for ω is $N(0, \Sigma^{-1})$. Typically, we do not know Σ, but we can estimate it using posterior samples. We can update this estimate throughout burn-in (or warm-up).

Effect of

```
n_reps = 1e4
d = expand.grid(e=10^seq(-3,0,by=1), L=10^seq(0,2))
r = ddply(d, .(e,L), function(xx) {
  data.frame(
    iteration = 1:n_reps,
    theta = HMC(n_reps, function(x) -x^2/2, function(x) -x, list(e=xx$e,L=xx$L), list(theta=0)))
})
```


0.001
0.01
0.1
1
1
10
100
theta
5.0
2.5
0.0
−2.5
0
2500
5000
7500
10000
iteration


0.001
0.01
0.1
1
1
10
100
density
theta
8
6
4
2
0
2.5
2.0
1.5
1.0
0.5
0.0
0.4
0.3
0.2
0.1
0.0
−2.5
0.0
2.5
5.0


acceptance_rate
autocorrelation
1.0
0.5
0.0
−0.5
value
0.01
1.00
0.01
1.00
e
L
100
75
50
25
as.factor(L)
1
10
100

Random-walk vs HMC

https://www.youtube.com/watch?v=Vv3f0QNWvWQ

Summary

Hamiltonian Monte Carlo (HMC) is a Metropolis-Hastings method using parameter augmentation and a sophisticated proposal distribution based on Hamiltonian dynamics such that

- the acceptance probability can be kept near 1
- while still efficiently exploring the posterior.

HMC still requires us to set tuning parameters

- e: step size
- L: number of steps
- D: covariance matrix for momentum

and can only be run in models with continuous parameters in $\mathbb{R}^d$ (or transformed to $\mathbb{R}^d$).