Hamiltonian Monte Carlo, Stan & brms

Edps 590BAY

Carolyn J. Anderson

Department of Educational Psychology

ILLINOIS

Fall 2019

Overview

- Advantages of Hamiltonian MCMC (HMC)
- General idea of how HMC works
- Stan language and examples
- Interfaces that make Bayesian model fit almost routine (e.g, brms)

Depending on the book that you select for this course, read either Gelman et al. Chapter 12 or Kruschke Chapter 14. I also read various other material from books, papers, and internet.

Why Hamiltonian MCMC?

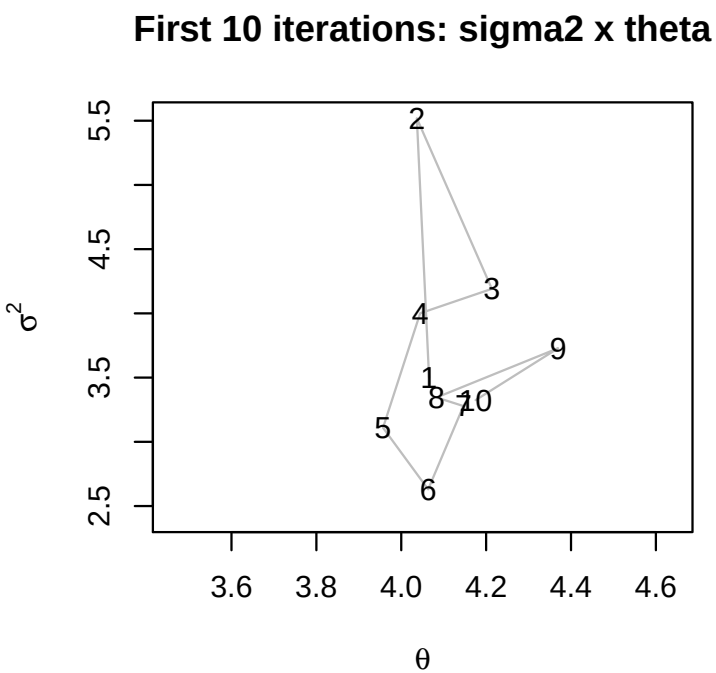
"Not all MCMC algorithms are created equal"
(Hoffman & Gelman, 2011)

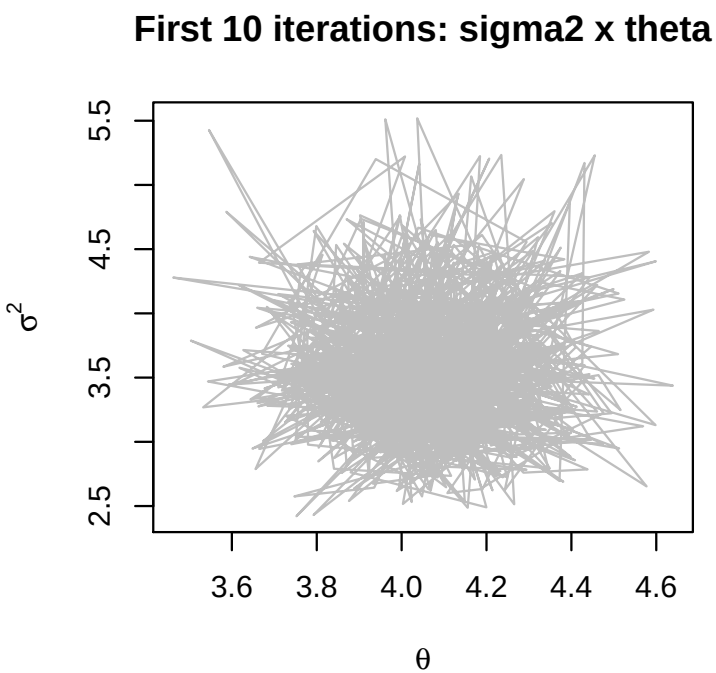
Problems with Metropolis and Gibbs algorithms:

- Random walk behavior can take a long time to achieve convergence, especially for complex, high dimensional models.
- In other words, they are inefficient at exploring the parameter space.
- Can get stuck along wall when have highly correlated parameters.

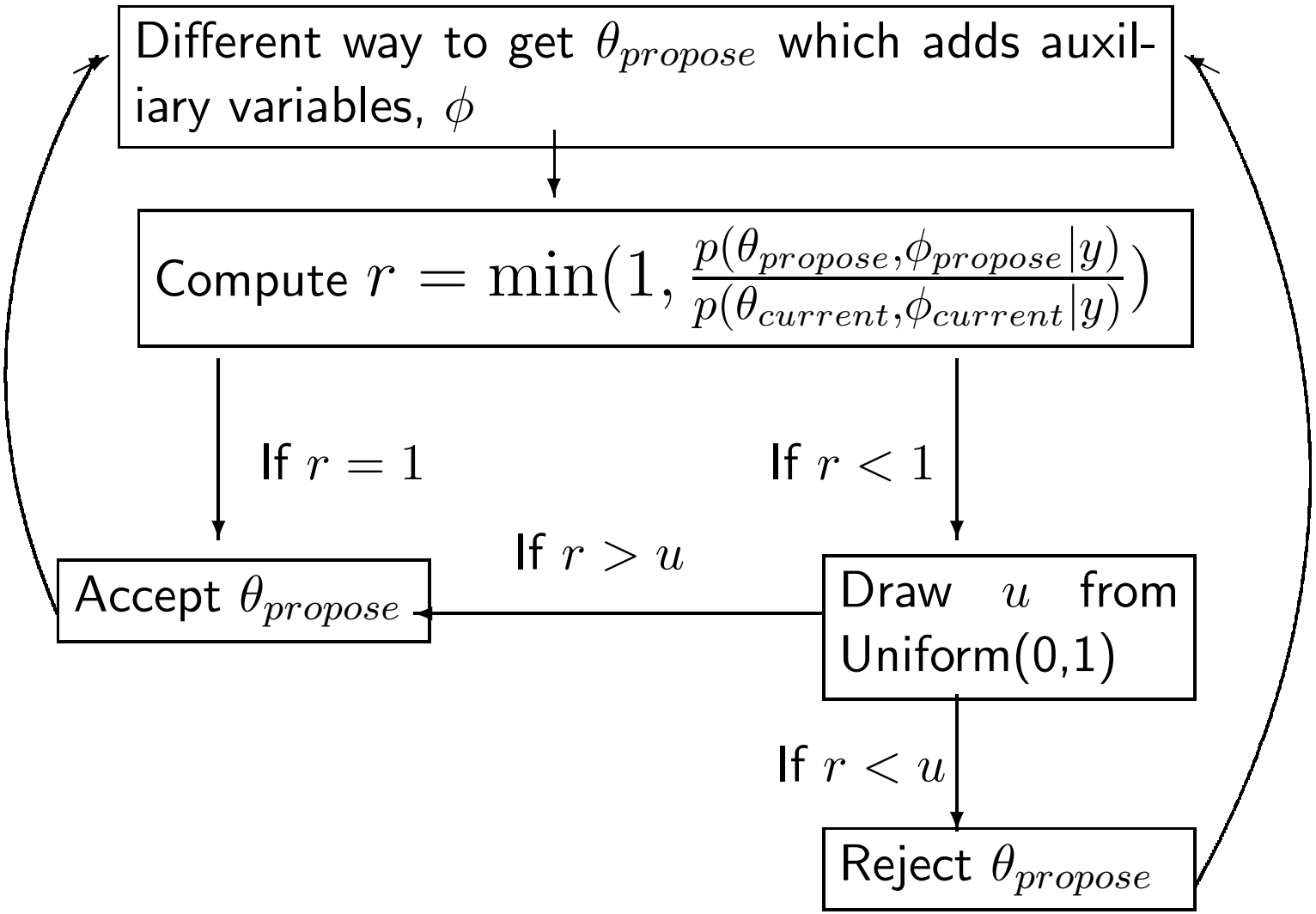
There is a physical analogy of sampling using Hamiltonian dynamics.

Local Zigging and Zapping of Gibbs

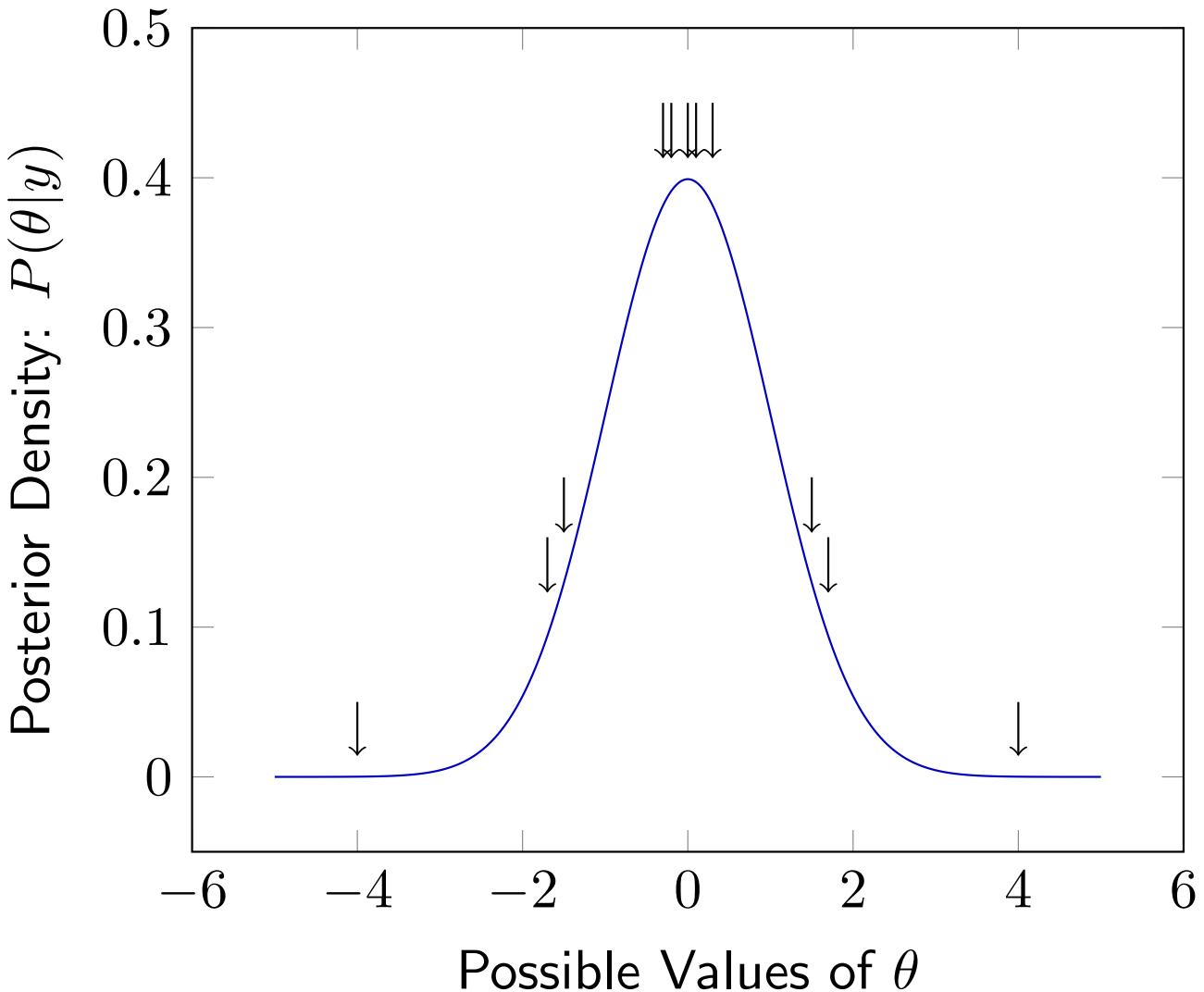




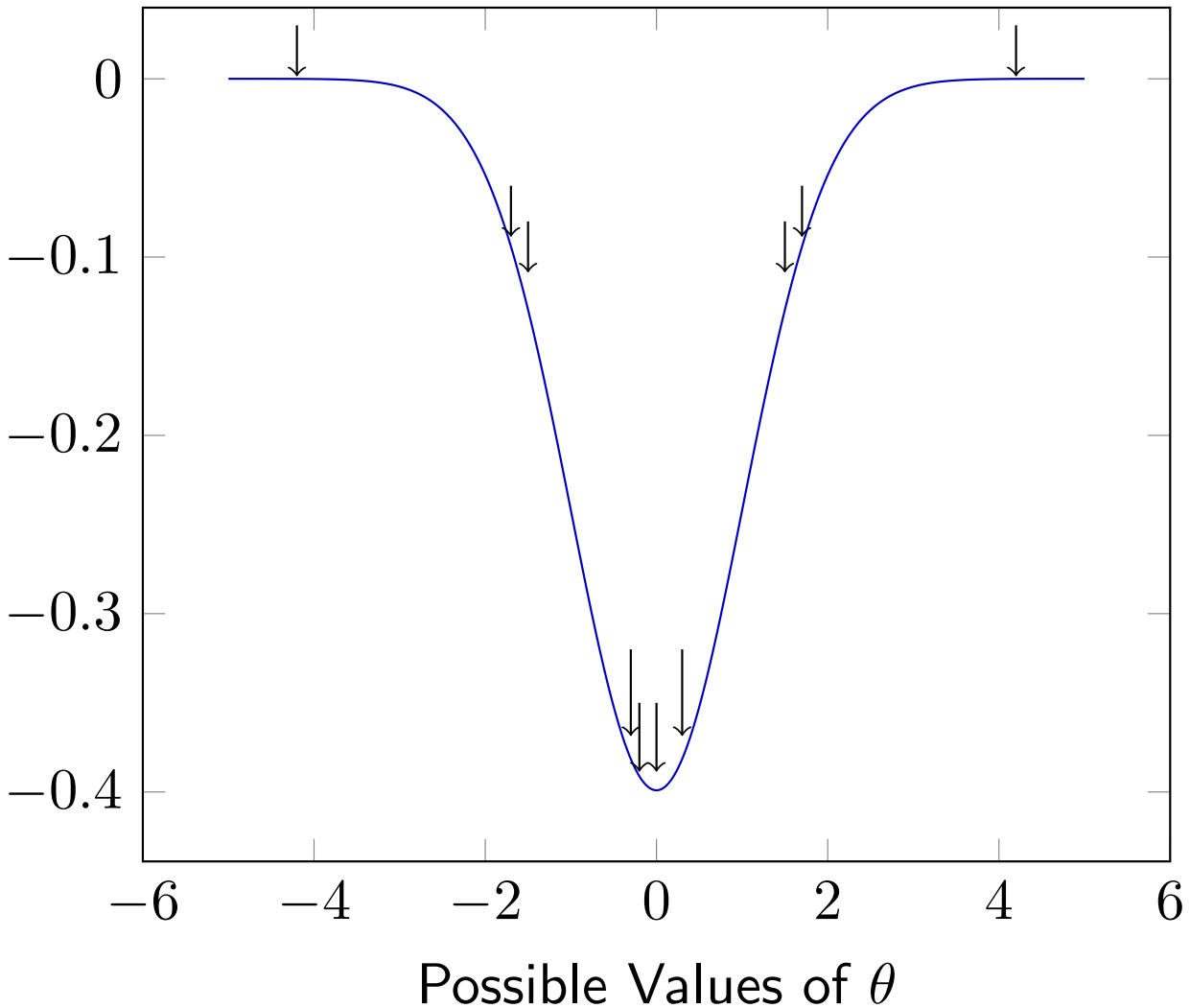
HMC is Version of Metropolis(-Hastings)



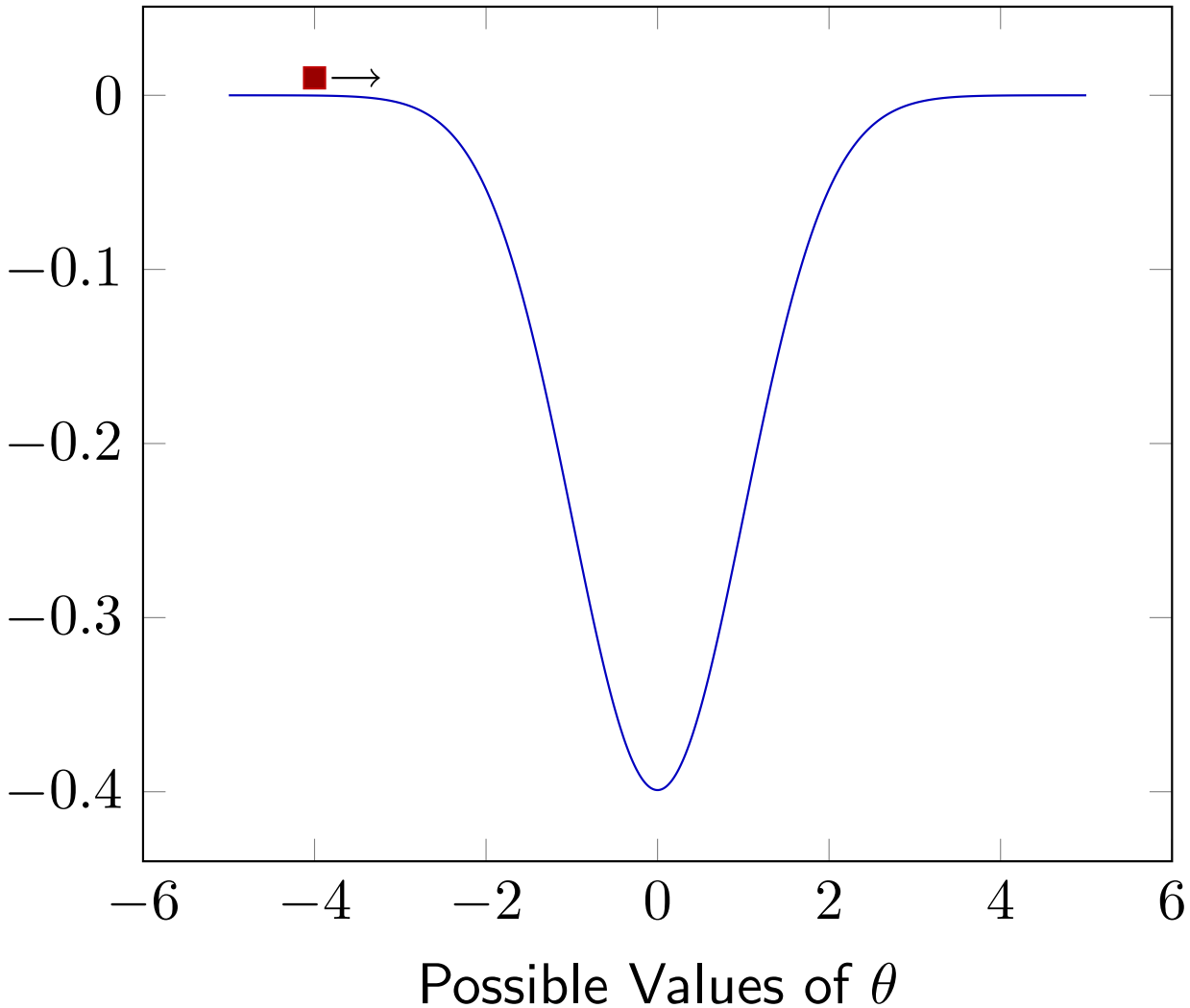
Posterior Density: $P(\theta|y)$



-log(Posterior Density): $-\log(P(\theta|y))$

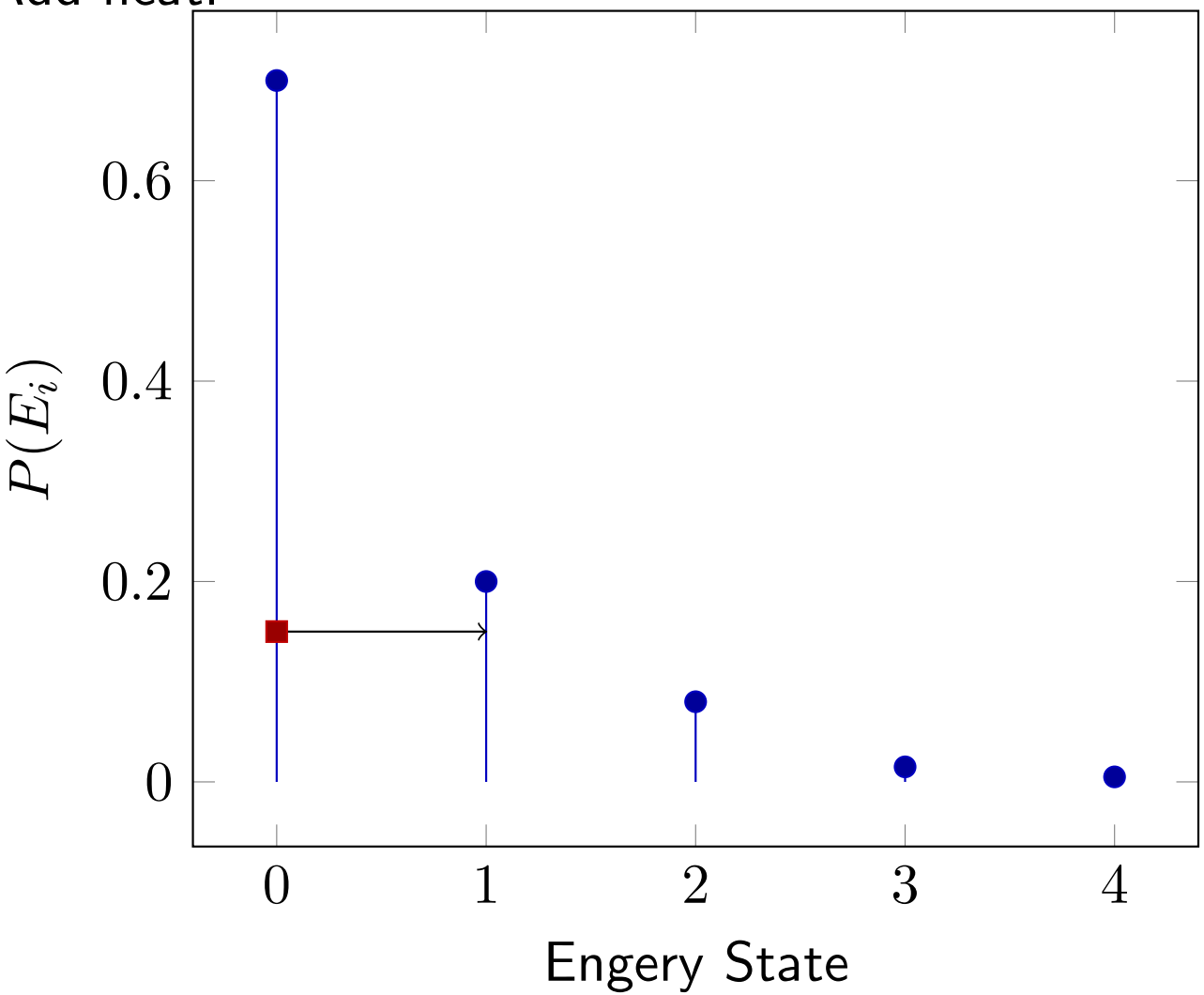


Statistical Mechanics: movement over fixed time period



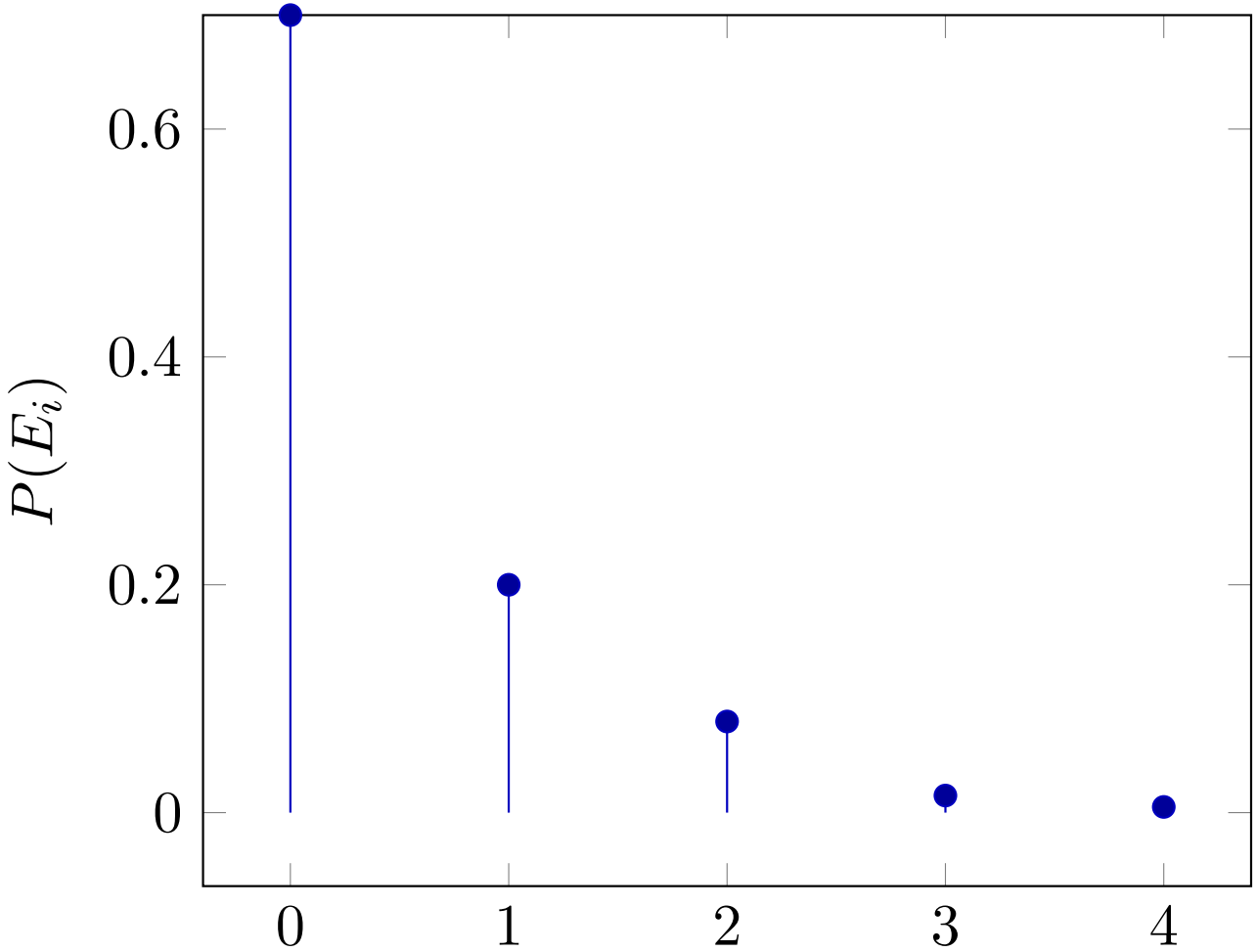
Energy States

Add heat:



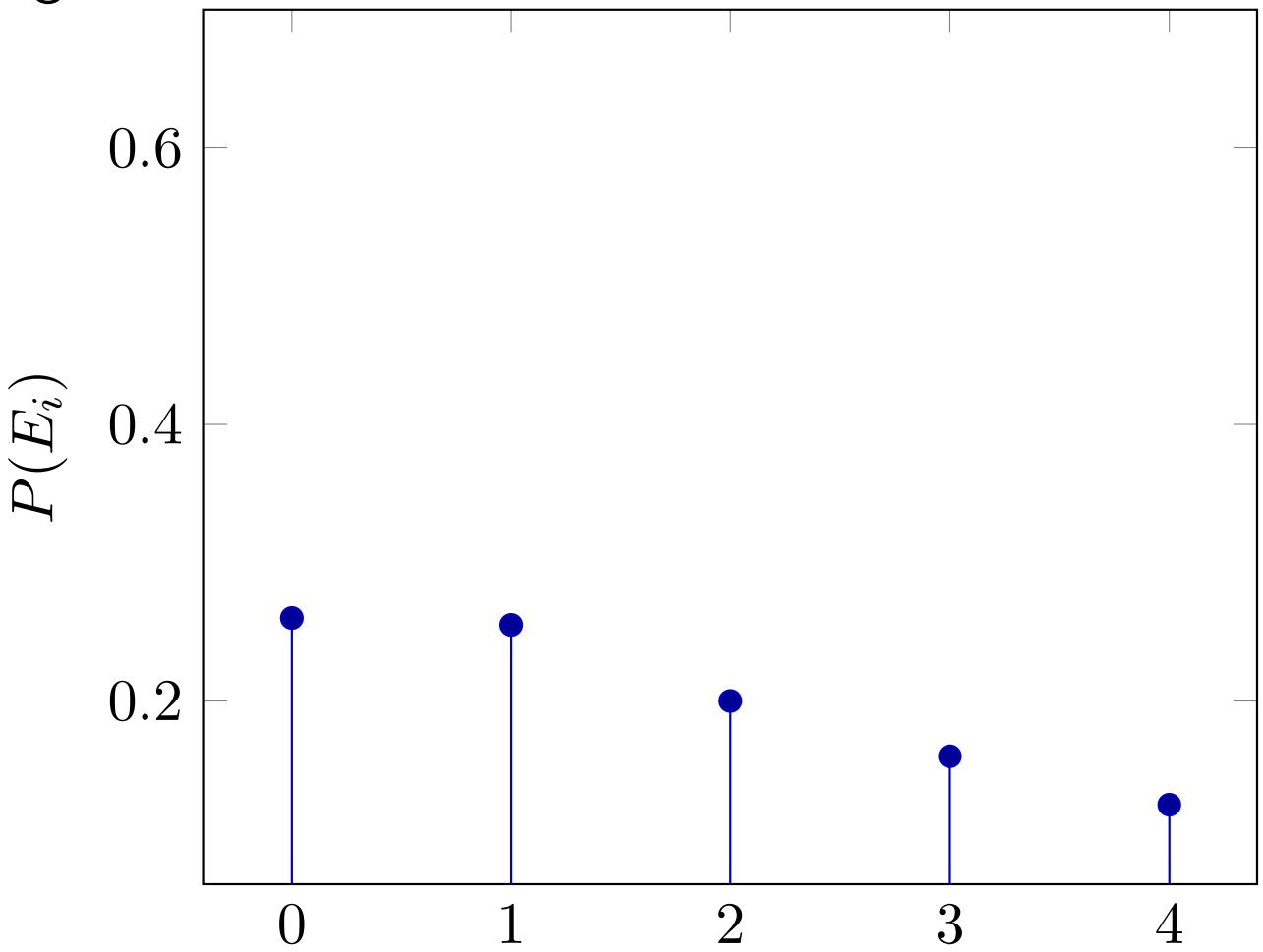
Energy State: $P(E_i) \propto \exp(-E_i/T)$

where E_i is energy state and T is "temperature". In this plot T is low.



Energy States: $P(E_i) \propto \exp(-E_i/T)$

where E_i is energy state and T is "temperature". In this plot T is high.



How this Applies to HMC

- We create one auxiliary variable per parameter that we want to estimate and this acts as "Momentum" or ϕ_i.
- Think of the parameter(s) as having a "location" or "position".
- Use the joint distribution of the parameter θ and ϕ,

$$E(\theta, \phi) = \underbrace{U(\theta)}_{potential} + \underbrace{K(\phi)}_{kinetic} = \text{constant} = \text{total energy}$$

The gravitational potential energy (i.e., $U(\theta)$), is that which is due to the parameter's location or position.

- From previous slides

$$\begin{aligned} Pr(Energy) &\propto \exp(-\text{Energy state}/T) \\ &= \exp(-(U(\theta) + K(\phi))/T) \end{aligned}$$

How this Applies to HMC (continued)

- Kinetic energy is

$$K(\phi) = \sum_{i=1}^{d} \frac{\phi_i^2}{2(\text{mass})}$$

We'll just consider one parameter (i.e., $d = 1$) so

$$K(\phi) = \phi^2/(2\text{mass})$$

- Set Potential Energy to

$$\begin{aligned} U(\theta) &= -\log[P(\theta, y)] \\ &= -\log[P(y|\theta)P(\theta)] \end{aligned}$$

and ...

How this Applies to HMC (continued)

Setting $T = 1$ and mass=1, substitute $K(\phi)$ and $U(\theta)$ into equation for Pr(Energy),

$$
\begin{aligned}
P(\theta, \phi) &\propto \exp[-(-\log(P(y|\theta)P(\theta)) + \frac{\phi^2}{2})] \\
&= P(y|\theta)P(\theta) \times \underbrace{\exp\left(\frac{-\phi^2}{2}\right)}_{N(0,1)} \\
&\equiv P(y|\theta)P(\theta) \times N(\phi, 0, 1)
\end{aligned}
$$

How this Applies to HMC (continued)

- With this setup, the Marginal density of θ, $P(\theta)$, is equal to the posterior distribution of θ:

$$\begin{aligned} P(\theta) &= \int P(\theta, \phi) d\phi \\ &= zP(y|\theta)P(\theta) \underbrace{\int N(\phi, 0, 1) d\phi}_{=1} \\ &= zP(y|\theta)P(\theta) \end{aligned}$$

  where z is just a normalizing constant.
- So, we sample from the joint distribution of θ and ϕ and throw away ϕ.

How do we do this?

Overview of Algorithm:

Iterate through the following steps:

1. Start with draw ϕ_{current} $N(0, 1)$
2. Use "Leap Frog" to get proposals for estimates of θ and ϕ.
3. Compute ratio

$$r = \frac{p(\theta_{\text{proposed}}|y)p(\phi_{\text{prosposed}})}{p(\theta_{\text{current}}|y)p(\phi_{\text{current}})}$$

4. Set value of θ for next iteratation,

$$\theta^t = \begin{cases} \theta_{\text{proposed}} & \text{with probability } \min(1, r) \\ \theta_{\text{current}} & \text{otherwise} \end{cases}$$

Leap Frog Step

Leap frog is a game played by children but here it is used to refer to the splitting of momentum updates into half steps.

- This is the deterministic part of the algorithm.
- This is the mechanism which allows movement much to be more rapid through the parameter space and suppresses the random walking. This does the part of sampling more in areas of high density.
- This mimicks the physical dynamics for continuous time and makes "time" discrete.

We update both θ and ϕ

Leap Frog Steps

Iterate through the following steps L times where $L =$ the number of leap cycles

1. Use the gradient (derivative) of the log-posterior density of θ,

$$\phi \leftarrow \phi + \frac{1}{2}\epsilon \frac{d \log p(\theta|y)}{d\theta}$$

2. Use the ϕ found in step 1 to update θ,

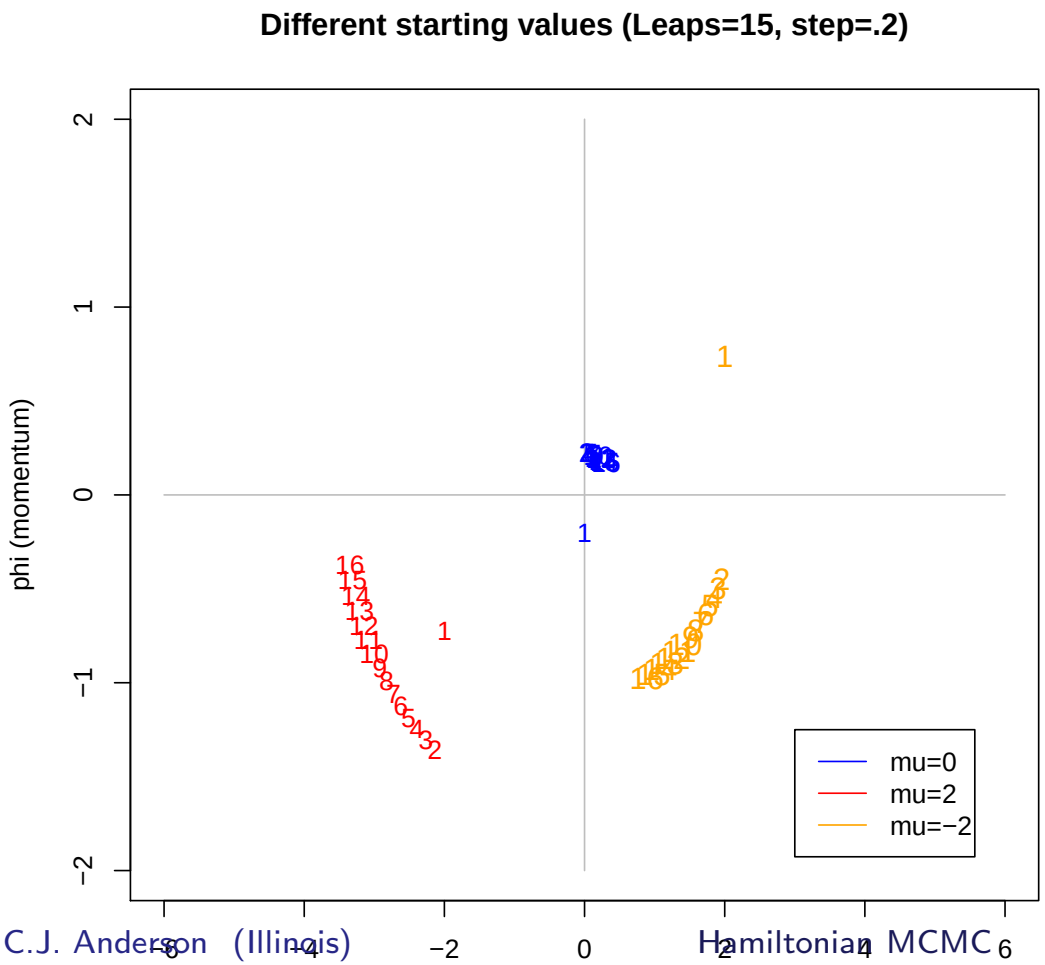
$$\theta \leftarrow \theta + \epsilon M^{-1}\phi$$

3. Do another half step and update ϕ

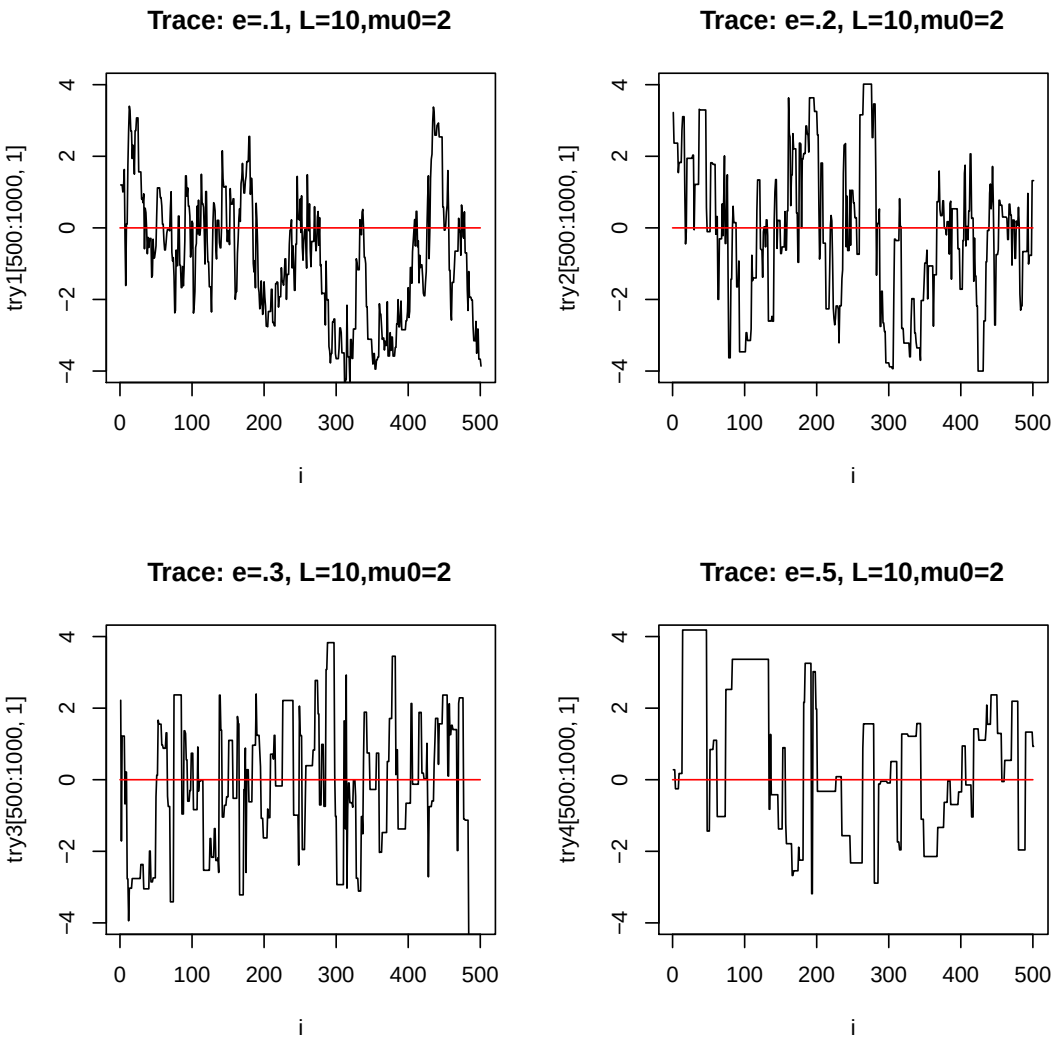
$$\phi \leftarrow \phi + \frac{1}{2}\epsilon \frac{d \log p(\theta|y)}{d\theta}$$

The Beginning of HMC

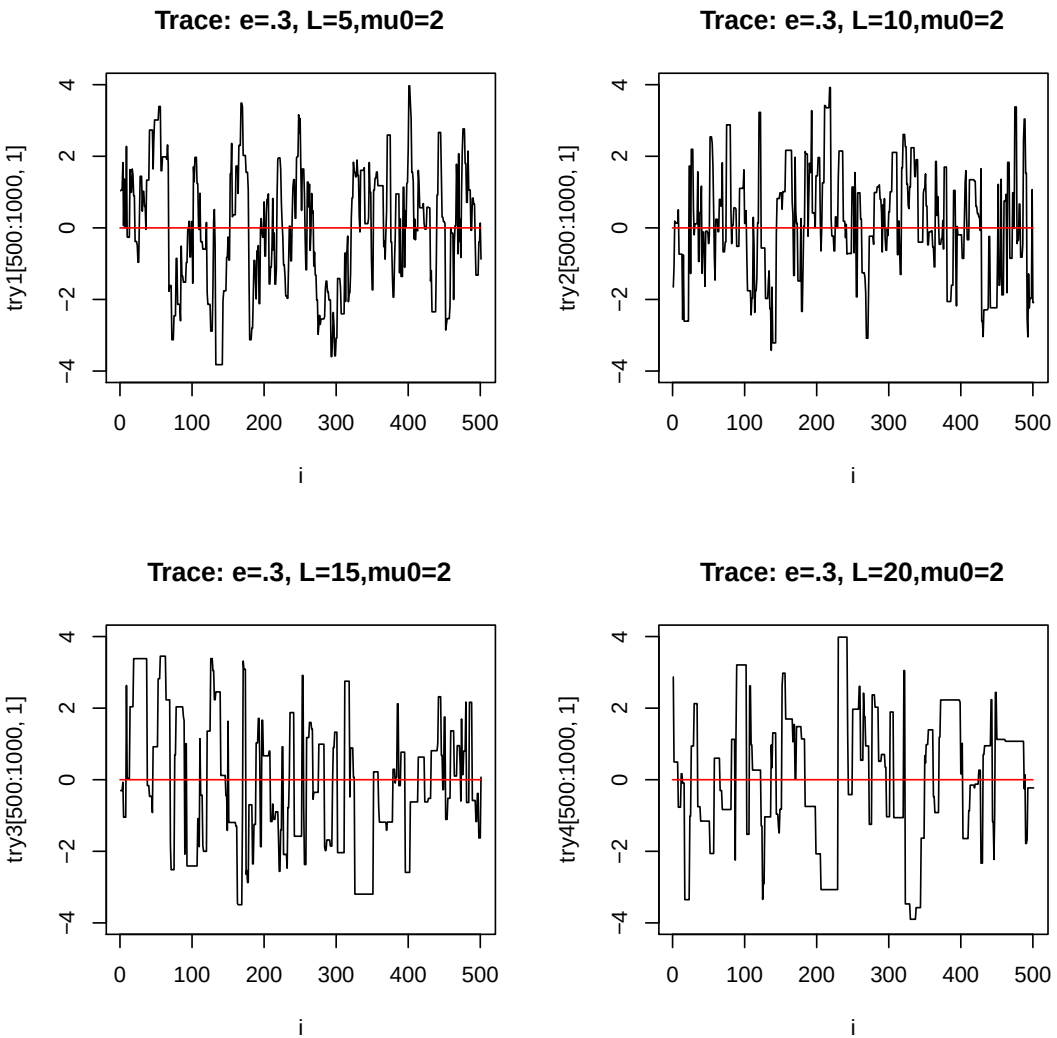
This illustrates what leap steps do (Each has a different starting place).



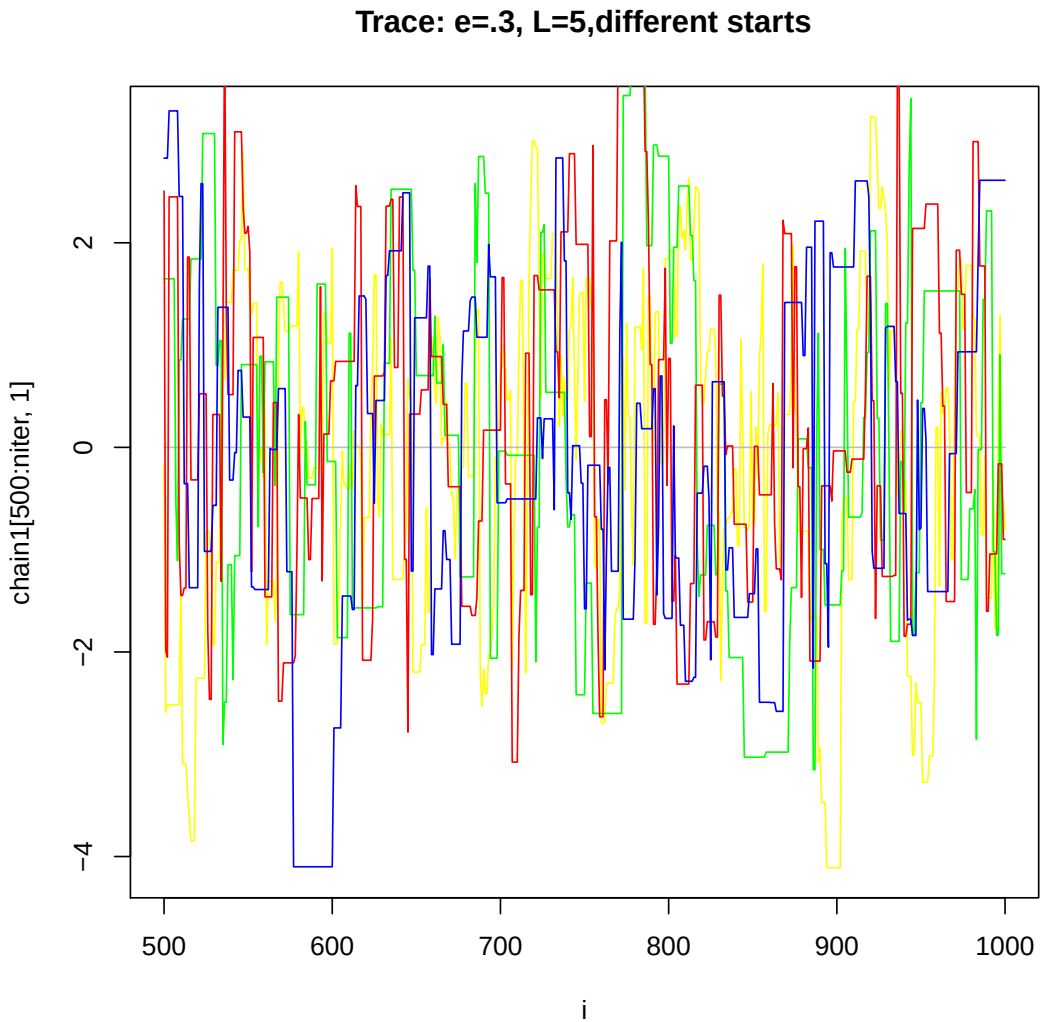
Impact of ϵ



Impact of L


Trace: e=.3, L=5,mu0=2
Trace: e=.3, L=10,mu0=2
Trace: e=.3, L=15,mu0=2
Trace: e=.3, L=20,mu0=2
try1[500:1000, 1]
try2[500:1000, 1]
try3[500:1000, 1]
try4[500:1000, 1]
i
0
100
200
300
400
500
−4
−2
0
2
4

Multiple Chains Running HMC



NUTS

No U-turn Sampler.

- Tuning HMC by trial and error is hard.
- NUTS is implemented in Stan and does the turning— finds L and ϵ.
- Is the default in Stan.
- For more information on this see Hoffman,M.D., & Gelman, A. (2011) The No-U-Turn sampler: adaptively selecting path lengths in Hamiltonian sampling. arXiv.1111.426v1 [stat.CO].
- "NUTS uses a recursive algorithm to build a set of likely candidate points that span a wide swath of the target distribution, stoping automatically when it starts to double back and retrace its steps."
- Also, it adapts the step size ϵ on the fly.

Stan

- It implements HMC (and variational approximation of Bayesian inference, and MLE for penalized maximum likelihood estimation).
- It is a probabilistic programming language to specifying statistical models that can be called through R using the rstan package.
- It's development was spearheaded by Gelman & Carpenter and a 34 member development team.
- The program is named after Stanislaw Ulam (1909–1984) who was a co-inventor of the Monte Carlo method.
- In Gelman et al. text they say that Stan stands for "Sampling Through Adaptive Neighborhoods."
- The Stan program is written in C++ and this shows up in some of the input conventions.

Stan

Input is highly structured and Stan is very picky.

The "Blocks" in this order

- data
- transformed data
- parameters
- transformed parameters
- model (REQUIRED)
- generated quantities

Conventions in Stan

- Comments are indicated by double slashes, "//"
- Stan uses semi-colon (i.e., ;) to mark end of line or command, like a period at the end of a sentence. This is the end-of-line indicator in C++ (SAS also uses this and C++ underlies it was well).
- Order matters!
- In the data and parameter (and others) blocks, you need to specify what type of variables you will be using. Type include real, int (integer), categorical, vector, and matrix.
- If there are lower or upper limits of data or parameters this should be indicated.
- In the parameter block, you need to specify what are the parameters.
- In the model block are priors followed by likelihood.
- In distributions, Stan uses standard deviation rather than

Conventions in Stan

Since gradients need to be computed, to achieve good computational efficiency, Stan has some standard distributions programed, including normal, student_t, cauchy, uniform, and others.

For full list see Table A-1 in Gelman et al. text, pages 576–569.

Simple Example using Stan

For this (and others), I'll use the 23 schools from the NELS data.

We'll start with finding the mean and standard deviation of the data.

See the document on course web-site:
Rmarkdown_stan_nels_set.docx

We will work through the steps in working with `Stan` using this document, starting simple and working to more complex models; that is,

- Estimating mean and standard deviation.
- Simple linear regression.
- Multiple regression.
- HLM with random intercept...takes too long so switch to brms
- HLM with random intercept and slope.

Stan Interfaces

There are a number of different packages that work with Stan. I don't want to rely on those written by authors of specific books (i.e., teaching tools), but rather the following two:

rstanarm: Bayesian Applied Regression Modeling.

- This is designed to work much like other package and to "make Bayesian estimation routine for the most common regression model that applied researchers use."
- Functions available include stan_lm, stan_glm, stan_lmer, stan_glimer, and stan_gamm4.
- Plenty of information and examples the internet.
- Good starting point is https://mc-stan.org/rstanarm/

Stan Interfaces

brms package: Bayesian Regression Models

- This is an interface to fit Bayesian generalized (non)linear multivariate multilevel models using `Stan`.
- Single and multi-level models can be fit.
- Wide range of possible models.
- A good starting point is Bürker, P.C. (2017). An `R` package for Bayesian multilevel models using Stan. *Journal of Statistical Software, 80*, doi: 10.18637/jss/v080.i01
- And the manual!

brms package

We'll go over same examples for nels for which we estimated using `Stan`.

Recall:

- waic: widely applicable information criteria
- loo: leave one out cross-validation