
BETTER CALL Stan
HAMILTONIAN MONTE CARLO
SAUL GOODMAN'S BAYESIAN ADVOCACY:
Maximize your winnings, minimize your risk.
BETTER CALL STAN INFERENCE, LLC
LIKELIHOOD & LITIGATION
POSTERIOR PRIORS
CINNABON
SAUL'S STAN WORKFLOW
DEFINE MODEL
MODEL COMPILER
COMPILED CODE
COMPILED CODE
1. DEFINE MODEL
2. INFERENCE ENGINE
THE HMC ROLLER COASTER
High Probability
MOMENTUM KICK (Dive particle a posk)
SAMPLER PARTICLE
PROBABILITY LANDSCAPE (fillly terrain)
LEAPFROG TRAJECTORY (energy conservation path)
automatically stops steps... automatically steps steps...
Low Probability
CINNABON
NO-U-TURN (NUTS)
NO U-TURN
FINAL INFERENCE
EXPERRES SAMPLES DISTRIBUTIONS EFFICIENTLY!
POSTERIOR samples
Density
Yields clear, useful, probabilistic graph
FINAL INFERENCE

Auxiliary variables in Monte Carlo

- In this lecture we will look at two methods to introduce auxiliary variables to help improve posterior sampling: slice sampling and Hamiltonian Monte Carlo (HMC).
- Both methods focus on sampling from a posterior distribution in the general form without assuming particular form of the distribution. Thus it makes these methods appealing in complex problems.
- We will focus on the general problem of sampling from an unnormalized density

  $$p(x) \propto f(x)$$

  in this lecture and discuss conceptual framework of these two sampling techniques.
- We will also take a brief look at Stan as a highly optimized HMC implementation.

Slice sampling

- Slice sampling is a way of constructing MCMC that sometimes mixes better than M-H algorithm.
- The basic idea is simple: think about a one-dimensional density function, sampling from this density function is equivalent to sampling points uniformly from the area under the density curve, and then gather the x-coordinates of these points.
- More formally, suppose we want to sample from $X \sim p(x) \ \propto \ f(x)$.
- the region under $f(x)$:

$$A = \{(x, u) : 0 < u < f(x)\}$$

- Then if $(X, U) \sim Unif(A)$, $p(x, u) = 1/|A|$ for $(x, u) \in A$. Taking the marginal distribution of X, $\int p(x, u)du = f(x)/A \ \propto \ p(x)$.

Slice sampling

- Operationally, sampling uniformly under $f(x)$ alternates between the two steps:
  1. Sample $u^{(t+1)} \sim U(0, f(x^{(t)}))$.
  2. Sample $x^{(t+1)} \sim U(B)$, where $B = \{x : f(x) \geq u^{(t+1)}\}$.
- Exact sampling from the second step is often difficult. However, we only need to make a move that has $U(B)$ as its stationary distribution.
- For instance, we can use a M-H move with proposal $U(B_r(x^{(t)}))$, where $B_r(x) = \{x' : ||x' - x||, r\}$.
- The radius r can be chosen dynamically, e.g., if it is too large, it is shrunk automatically until an acceptable point is found. See *Slice Sampling* (2003) by Neal for details.

Slice sampling

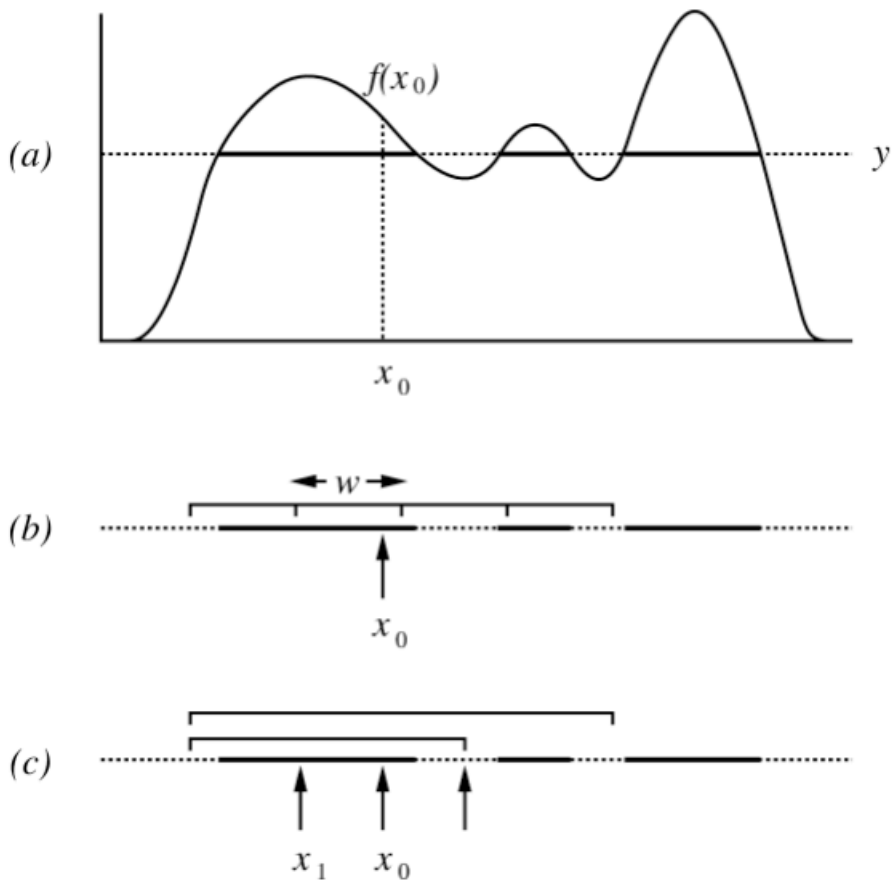


FIG. 1. *A single-variable slice sampling update using the stepping-out and shrinkage procedures. A new point, x_1, is selected to follow the current point, x_0, in three steps.* (a) *A vertical level, y, is drawn uniformly from $(0, f(x_0))$, and used to define a horizontal "slice," indicated in bold.* (b) *An interval of width w is randomly positioned around x_0, and then expanded in steps of size w until both ends are outside the slice.* (c) *A new point, x_1, is found by picking uniformly from the interval until a point inside the slice is found. Points picked that are outside the slice are used to shrink the interval.*

Elliptical slice sampling

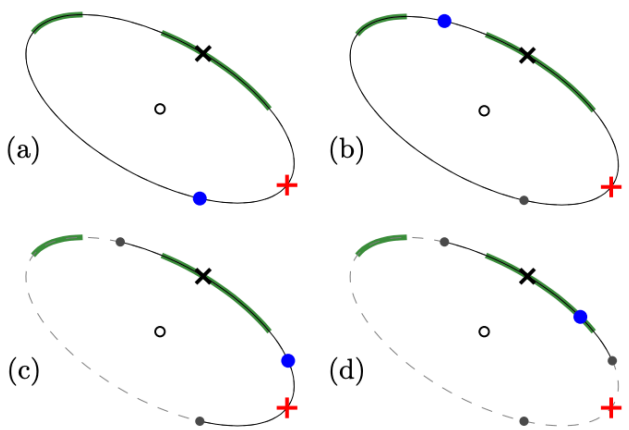
- Slice sampling is easy to implement in one dimension. Things get trickier in high dimensions.
- There are methods that are built on the idea and make use of specific parametric form of the density function.
- A particular case is when $\boldsymbol{x} \in \mathbb{R}^d$ and $p(x) \propto L(x)N(\boldsymbol{x}; 0, \boldsymbol{\Sigma})$. For example, hierarchical model with Gaussian priors.
- For a M-H update, we can draw proposal $\boldsymbol{x}' \sim N(0, \boldsymbol{\Sigma})$ and compute acceptance probability to be $min\{1, L(\boldsymbol{x}')/L(\boldsymbol{x})\}$, i.e., no need to evaluate the Gaussian component.
- An insight about the Gaussian distribution is that if $\boldsymbol{x} \sim N(0, \boldsymbol{\Sigma})$ and $\boldsymbol{v} \sim N(0, \boldsymbol{\Sigma})$ then $\boldsymbol{x}' = \sqrt{1-\epsilon^2}\boldsymbol{x} + \epsilon\boldsymbol{v}$ or equivalently $\boldsymbol{x}' = \boldsymbol{x}cos\theta + \boldsymbol{v}sin\theta \sim N(0, \boldsymbol{\Sigma})$.
- How do we choose θ such that the proposal $\boldsymbol{x}'$ has a high probability of being accepted? We can treat $\theta \sim Unif[0, 2\pi]$ as an auxiliary variable, and it turns out we can do slice sampling on $\theta \in [0, 2\pi]$.
- This is called Elliptical slice sampling (Murray, Adams, and MacKay, 2010). It works the best when $p(\boldsymbol{x})$ is close to the Gaussian density.

Elliptical slice sampling

Input: current state $\mathbf{f}$, a routine that samples from $\mathcal{N}(0, \Sigma)$, log-likelihood function $\log L$.

Output: a new state $\mathbf{f}'$. When $\mathbf{f}$ is drawn from $p^\star(\mathbf{f}) \propto \mathcal{N}(\mathbf{f}; 0, \Sigma)\, L(\mathbf{f})$, the marginal distribution of $\mathbf{f}'$ is also $p^\star$.

1. Choose ellipse: $\boldsymbol{\nu} \sim \mathcal{N}(0, \Sigma)$
2. Log-likelihood threshold:
$$u \sim \text{Uniform}[0, 1]$$
$$\log y \leftarrow \log L(\mathbf{f}) + \log u$$
3. Draw an initial proposal, also defining a bracket:
$$\theta \sim \text{Uniform}[0, 2\pi]$$
$$[\theta_{\min}, \theta_{\max}] \leftarrow [\theta - 2\pi,\ \theta]$$
4. $\mathbf{f}' \leftarrow \mathbf{f} \cos\theta + \boldsymbol{\nu} \sin\theta$
5. **if** $\log L(\mathbf{f}') > \log y$ **then:**
6. Accept: **return** $\mathbf{f}'$
7. **else:**
Shrink the bracket and try a new point:
8. **if** $\theta < 0$ **then:** $\theta_{\min} \leftarrow \theta$ **else:** $\theta_{\max} \leftarrow \theta$
9. $\theta \sim \text{Uniform}[\theta_{\min}, \theta_{\max}]$
10. **GoTo** 4.



Hamiltonian Monte Carlo

- Similar to the difficulty with slice sampling in high dimensions, random walk is highly inefficient in high dimensional spaces.
- Transformations and careful tuning of the step size can mitigate the issue, but they are hard to construct for complex distributions.
- A more promising approach is to make the proposal ‘smarter’ than random walk.
- Hamiltonian Monte Carlo (HMC) is one such method to sample from a target distribution.

HMC: basic idea to sample from $p(x)$ where $x \in \mathbb{R}^d$

- Sample an auxiliary variable $z \in N(0, I_d)$.
- Find a new proposal that leaves $p(x, z)$ roughly constant.
  - This step is done using Hamiltonian dynamics.
- Use M-H step to accept or reject the proposal.

First, let us see how the samples look like:

http://chi-feng.github.io/mcmc-demo/

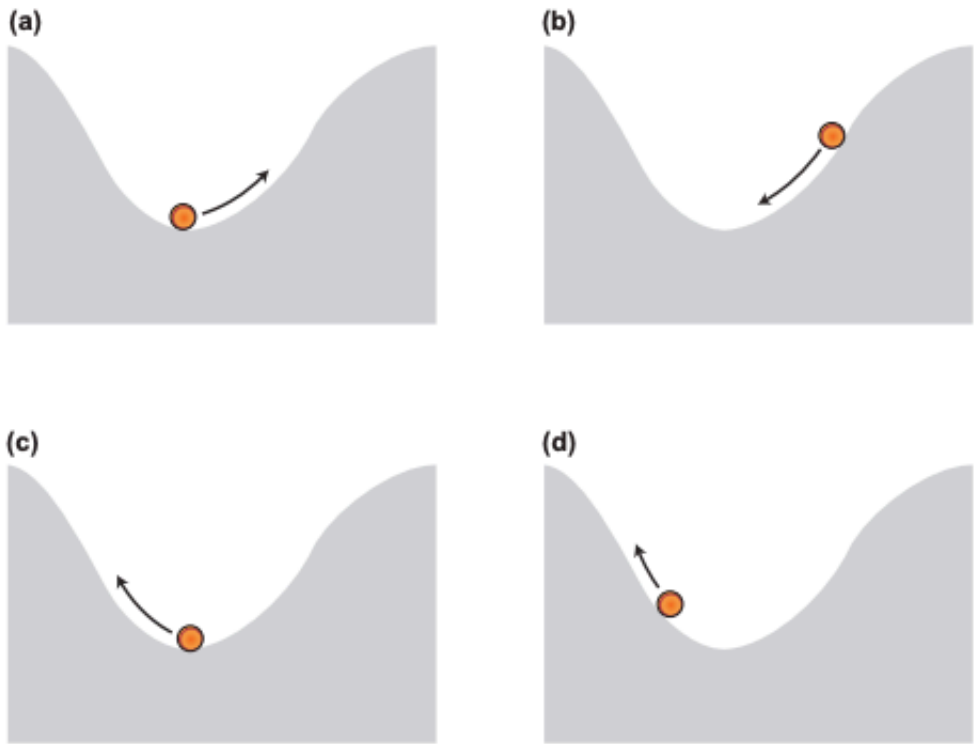
HMC: slightly more details

- Basically, we want the sampler to travel along the contour of (x, z) where

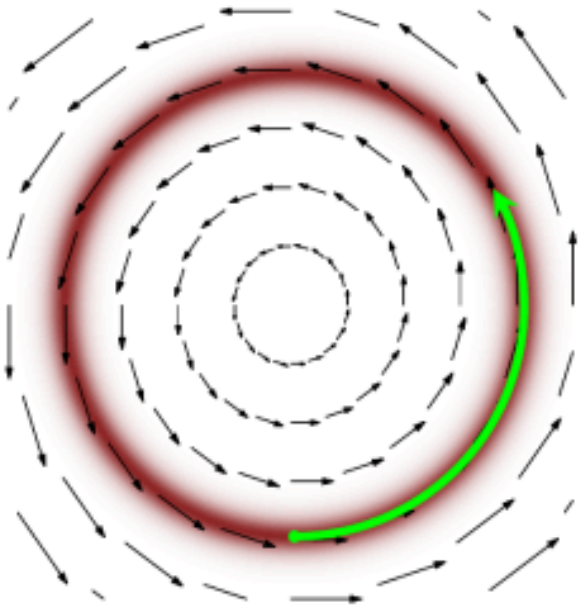
$$H(x, z) = -\log p(x) + \frac{1}{2} z^T z$$

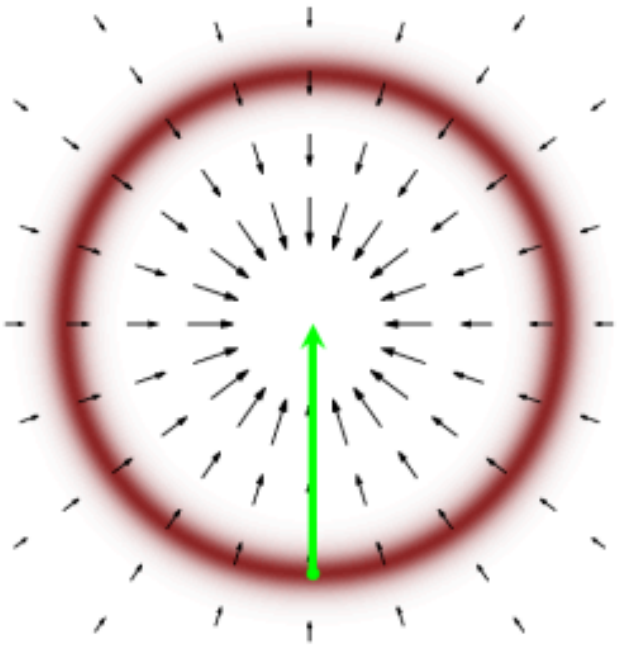
  is constant.
- This is has useful physical interpretation:
  - $x_1, ..., x_d$: position coordinates
  - $z_1, ..., z_d$: momentum coordinates
  - $-\log p(x)$: potential energy
  - $\frac{1}{2} z^T z$: kinetic energy
  - $H(x, z)$: total energy
- Think about a ball rolling on a friction-less surface...

One-dimensional illustration of HMC



Two-dimensional illustration of HMC





Hamiltonian dynamics

- Now we add the 'time' dimension to help us understand how the HMC sampler 'travels' across the parameter space.
- At any given value of (x^*, z^*), consider two processes $x(t)$ and $z(t)$ that starts from the initial state (x^*, z^*) following the dynamic system

$$\frac{\partial x_i}{\partial t} = \frac{\partial H}{\partial z_i} = z_i$$
$$\frac{\partial z_i}{\partial t} = \frac{\partial H}{\partial x_i} = \frac{\partial}{\partial x_i} \log p(x_i)$$

- $p(x, z) \propto \exp(-H(x, z))$ remains constant as (x, z) evolves according to the dynamic system.
- In practice, we cannot solve the dynamic system analytically, and it is necessary to approximate the system via discretization.

HMC: algorithm

HMC uses a numerical integration method called “leapfrog” algorithm that proceeds with the following steps

- Sample $z \sim N(0, diag(m_1, ..., m_d))$.
- Set $x_0 \leftarrow x$, $z_0 \leftarrow z$.
- For $l = 1, ..., L$
  1. $z \leftarrow z + \frac{1}{2}\epsilon \nabla \log p(x)$
  2. $x \leftarrow x + \epsilon M^{-1} z$
  3. $z \leftarrow z + \frac{1}{2}\epsilon \nabla \log p(x)$
- Metroplis accept/reject step: accept new state with probability

$$\alpha = \min\{1, \frac{p(x)N(z; 0, M)}{p(x_0)N(z_0; 0, M)}\}$$

Key parameters to specify are the momentum variance M, the number of steps to take L and step size ϵ. As usual with MCMC, these tuning parameters can be chosen to get a desired acceptance rate.

No U-turn sampler (NUTS)

- One intuition to choose suitable step size is that we may want to avoid the Hamiltonian dynamic trajectory to go all the way back to the starting point.
- But stopping the exploration as soon as it turns back violates detailed balance.
- In practice, NUTS (Hoffman and Gelman, 2014) fix M and ϵ (after adapting them during burn-in) and then adaptively choose L at each iteration.
- Another computational step is to compute the gradient. This is achieved via automatic differentiation.
- NUTS is the default sampler of Stan. Stan contains several other carefully optimized routines to improve the automatic sampler behavior too...so bottom line is, if you want to use HMC, check out Stan first!

Stan

- Stan is a 'black-box' MCMC sampler. It has various integration with R, Python, Matlab, Julia, etc., and can be run as command-line executable.
- We will focus on the R interface here via the package rstan.

Stan example: eight schools

```
library(rstan)

stan_code <- "
data{
  int<lower = 0> J;
  real y[J];
  real<lower = 0> sigma[J];
}
parameters{
  real mu;
  real<lower = 0> tau;
  vector[J] eta;
}
transformed parameters{
  vector[J] theta = mu + tau * eta;
}
model{
  eta ~ normal(0, 1);
  y ~ normal(theta, sigma);
}
"
```

Stan example: eight schools

You can also specify the likelihood calculation directly (which is useful if the likelihood involves analytical marginalization of latent variables)

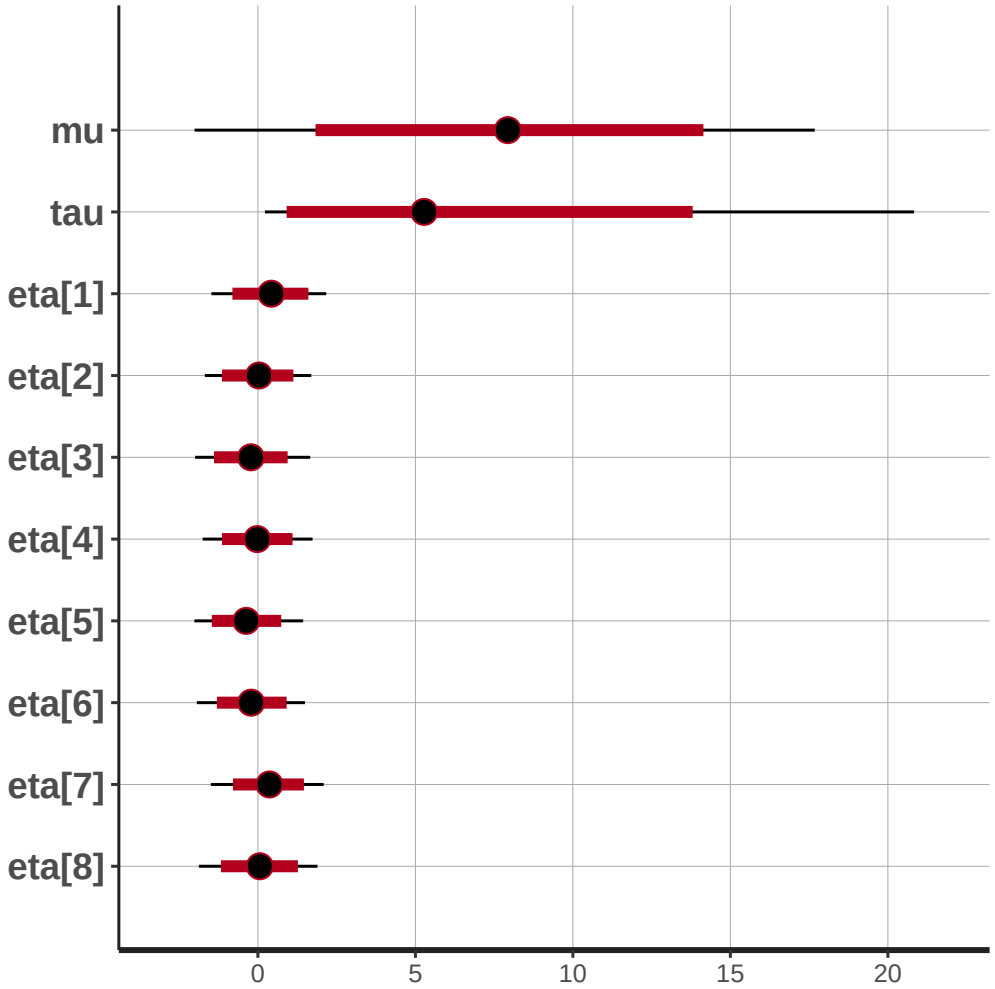
```
stan_code <- "
data{
  int<lower = 0> J;
  real y[J];
  real<lower = 0> sigma[J];
}
parameters{
  real mu;
  real<lower = 0> tau;
  vector[J] eta;
}
transformed parameters{
  vector[J] theta = mu + tau * eta;
}
model{
  target += normal_lpdf(eta | 0, 1);
  target += normal_lpdf(y | theta, sigma);
}
"
```

Stan example: eight schools

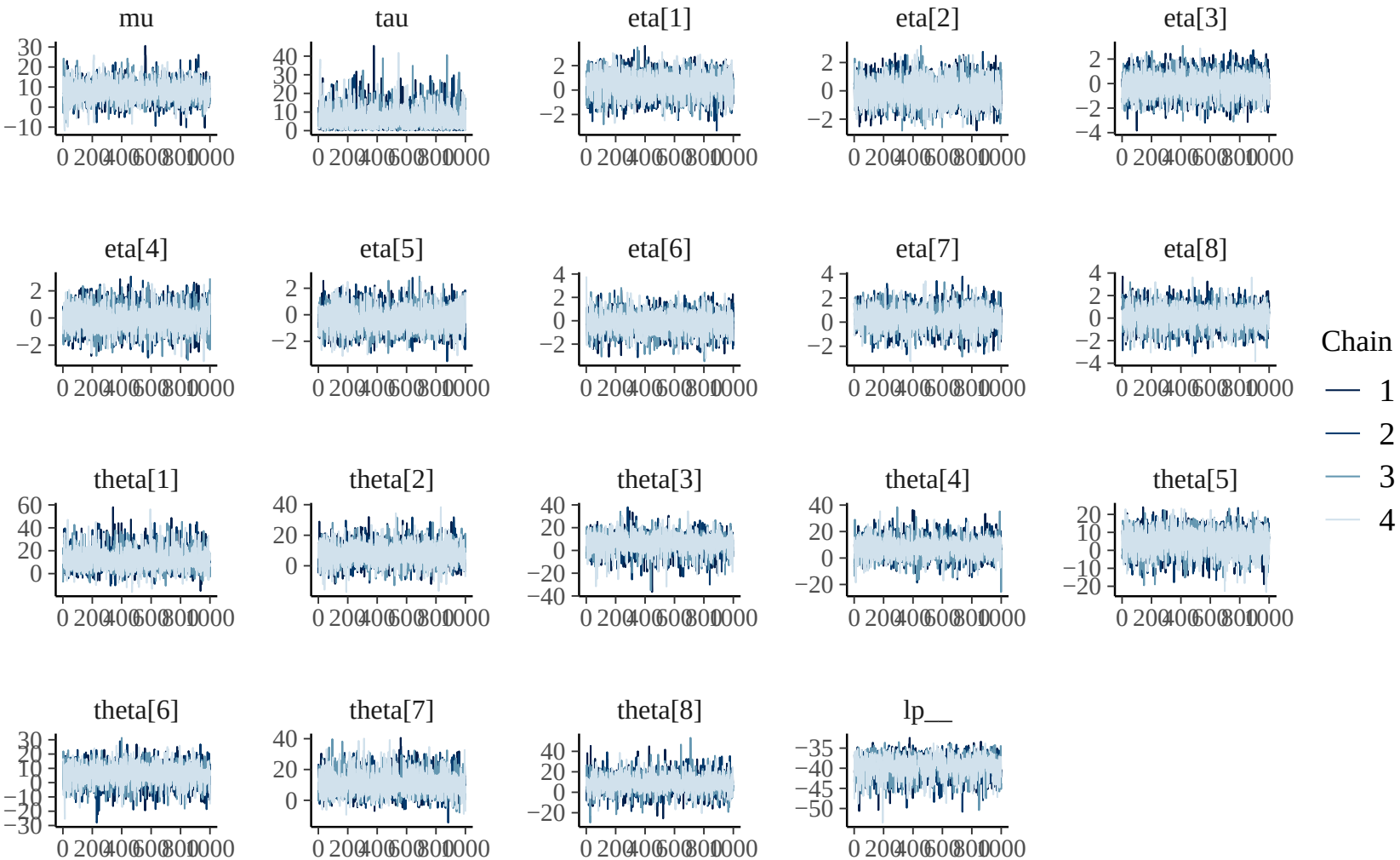
```
data <- list(J = 8,
     y = c(28,8,-3,7,-1,1,18,12),
     sigma = c(15,10,16,11,9,11,10,18))
fit <- stan(model_code = stan_code, data = data,
      chains = 4, warmup = 1000, iter = 2000)
plot(fit)

library(bayesplot)
mcmc_dens_overlay(fit)
mcmc_trace(fit)
```

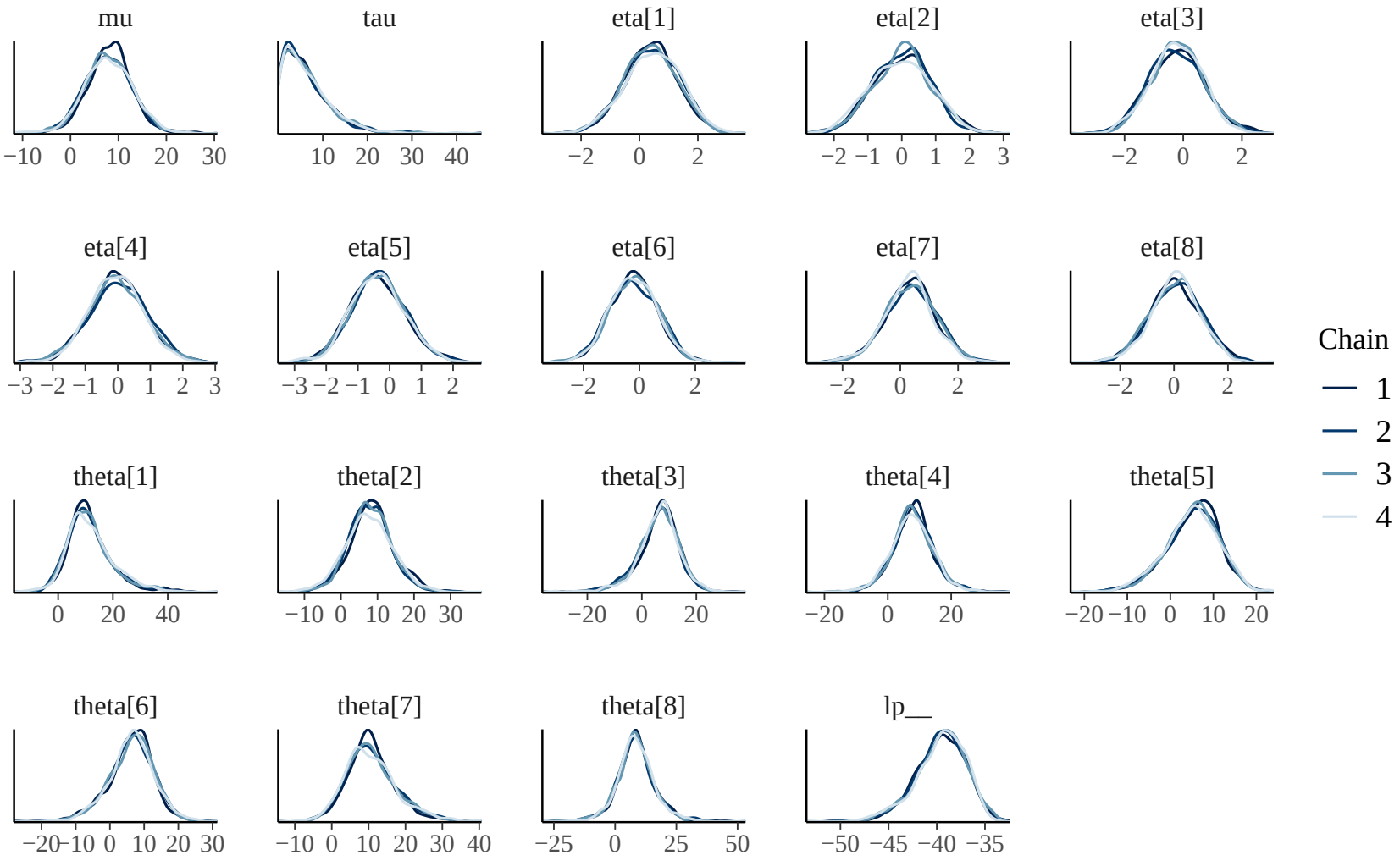
Posterior marginals



Trace plot


mu
tau
eta[1]
eta[2]
eta[3]
eta[4]
eta[5]
eta[6]
eta[7]
eta[8]
theta[1]
theta[2]
theta[3]
theta[4]
theta[5]
theta[6]
theta[7]
theta[8]
lp__
0 200 400 600 800 1000
Chain
1
2
3
4

Posterior density



HMC and Stan

- Some good references to learn more details about HMC:
  - Betancourt, Michael. *A conceptual introduction to Hamiltonian Monte Carlo.* arXiv:1701.02434 (2017).
  - Thomas, Samuel, and Wanzhu Tu. *Learning Hamiltonian Monte Carlo in R.* The American Statistician (2021).
- Stan is a very useful tool for prototyping and building simple models quickly to gain insights.
- It has many many built-in priors/likelihoods and new extensions (variational Bayes, ODE, etc.). See the Stan manual for more details.
- It has some useful model diagnostics tools for HMC.
- There is some learning curve to know what is possible and the syntax, but in general not too difficult.
- But there are fundamental limitations stemming from HMC, e.g., discrete random variables are always tricky if not impossible to work with.