Bayesian Inference II

Hamiltonian Monte Carlo (HMC): a sketch

Leonardo Egidi, legidi@units.it

2025/2026

Guest lecture, February 11

Table of contents i

When MCMC fails?

The momentum distribution

The three steps of an HMC iteration

HMC and Stan

When MCMC fails?

MCMC inefficiency i

- The Metropolis-Hastings algorithm and the Gibbs sampling generate *correlated* variables from a stochastic process called Markov chain. Markov chains carry different convergence properties that can be exploited to provide easier proposals in cases where generic importance sampling does not readily apply.
- An inherent inefficiency in the Gibbs sampler and Metropolis algorithm is their random walk behavior: the simulations can take a long time zigging and zagging while moving through the target distribution.

MCMC inefficiency ii

- Reparameterization and efficient jumping rules can improve the situation, but for complicated models this local random walk behavior remains, especially for high-dimensional target distributions.
- *Hamiltonian Monte Carlo* (HMC) borrows an idea from physics to suppress the local random walk behavior in the Metropolis algorithm, thus allowing it to move much more rapidly through the target distribution.

Moving to Hamiltonian Monte Carlo

- Once we have built a model, Bayesian computation reduces to evaluating expectations, or integrals.

$$\mathrm{E}_\pi(\theta|y) = \int \theta\pi(\theta|y)d\theta. \tag{1}$$

- How do we compute posterior expectations in practice?
  - Construct a Markov chain that explores the parameter space.
  - Anything you would want to do if you could write it analytically, you can do to any accuracy with the draws (history) of the chain

$$\lim_{S\to\infty} \frac{1}{S}\sum_{s=1}^{S} \theta^{(s)} \to \mathrm{E}_\pi(\theta|y)$$

Moving to Hamiltonian Monte Carlo

- To be efficient we need to focus computation on the relevant neighborhoods of parameter space. Relevant neighborhoods, however, are defined not by probability density but rather by probability mass.
- But exactly which neighborhoods end up contributing most to arbitrary expectations?
- The neighborhoods around the maxima of probability distributions feature a lot of probability density, but, especially in a large number of dimensions, or in long tailed distributions, they do not feature much volume. In other words, the *sliver* size $d\theta$ tends to be small there.

The Geometry of High-Dimensional Spaces

- Expectation values are given by accumulating the integrand over a **volume** of parameter space and, while the density is largest around the mode, there is not much volume there.
- To identify the regions of parameter space that dominate expectations we need to consider the behavior of *both the density and the volume*. In high-dimensional spaces the volume behaves very differently from the density, resulting in a tension that concentrates the significant regions of parameter space away from either extreme.

The Geometry of High-Dimensional Spaces

- Consider a rectangular partitioning centered around a distinguished point, such as the mode (example from Betancourt, 2017):

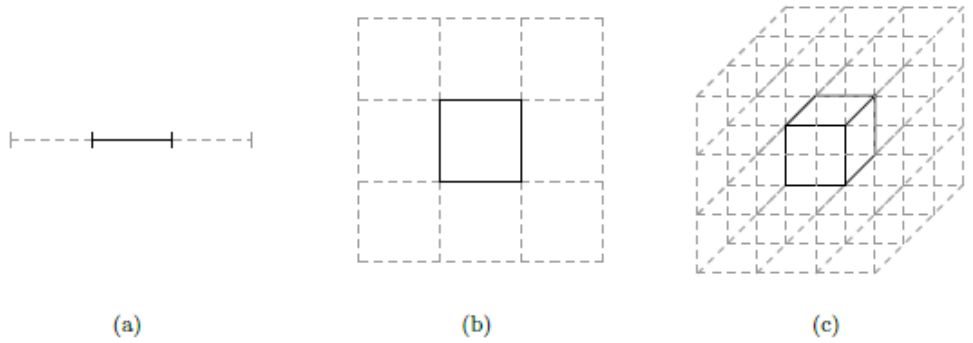


FIG 1. *To understand how the distribution of volume behaves with increasing dimension we can consider a rectangular partitioning centered around a distinguished point, such as the mode. (a) In one dimension the relative weight of the center partition is 1/3, (b) in two dimensions it is 1/9, (c) and in three dimensions it is only 1/27. Very quickly the volume in the center partition becomes negligible compared to the neighboring volume.*

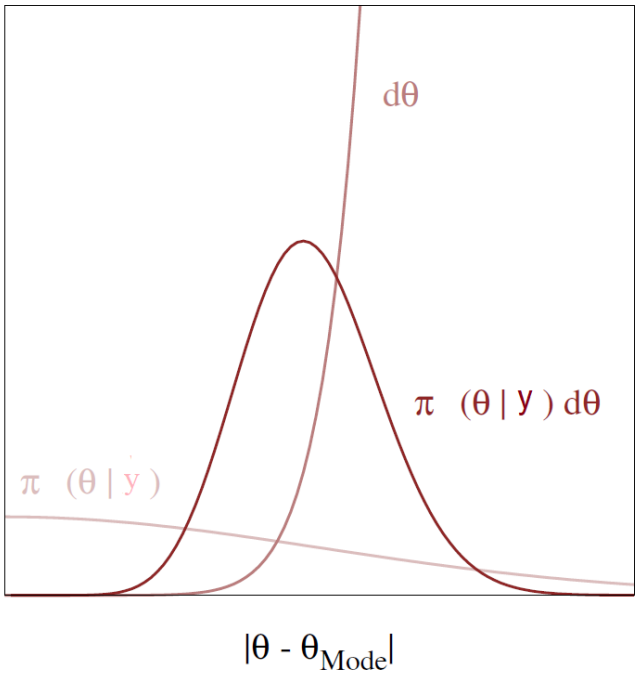
- One of the characteristic properties of high-dimensional spaces is that there is much more volume outside any given neighborhood than inside of it!

The Geometry of High-Dimensional Spaces

- Generically, then, volume is *largest* out in the tails of the target distribution away from the mode, and this disparity grows exponentially with the dimension of parameter space.
- The neighborhood immediately around the mode features large densities, but in more than a few dimensions the small volume of that neighborhood prevents it from having much contribution to any expectation. On the other hand, the complimentary neighborhood far away from the mode features a much larger volume, but the vanishing densities lead to similarly negligible contributions expectations
- The only significant contributions come from the neighborhood between these two extremes known as the **typical set**.

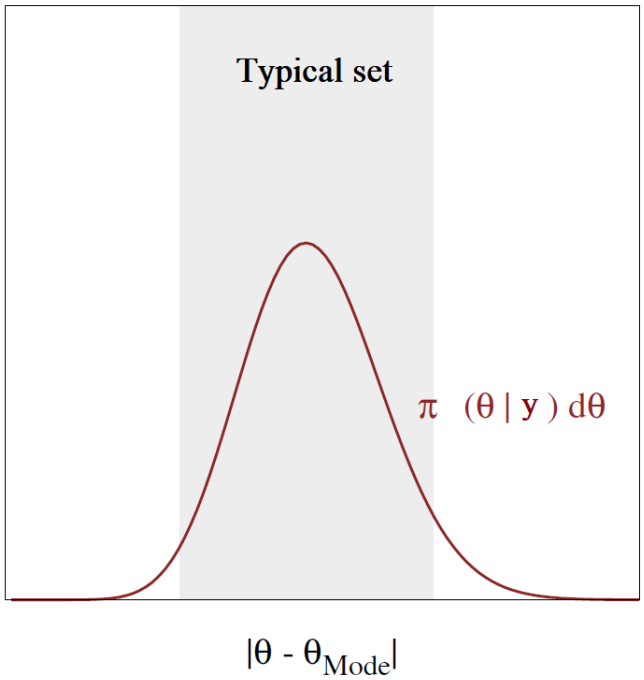
Typical set

Thus, relevant neighborhoods are defined not by probability density but rather by probability mass.



Typical set

Probability mass concentrates on a hypersurface called the typical set that surrounds the mode.



Moving to Hamiltonian Monte Carlo i

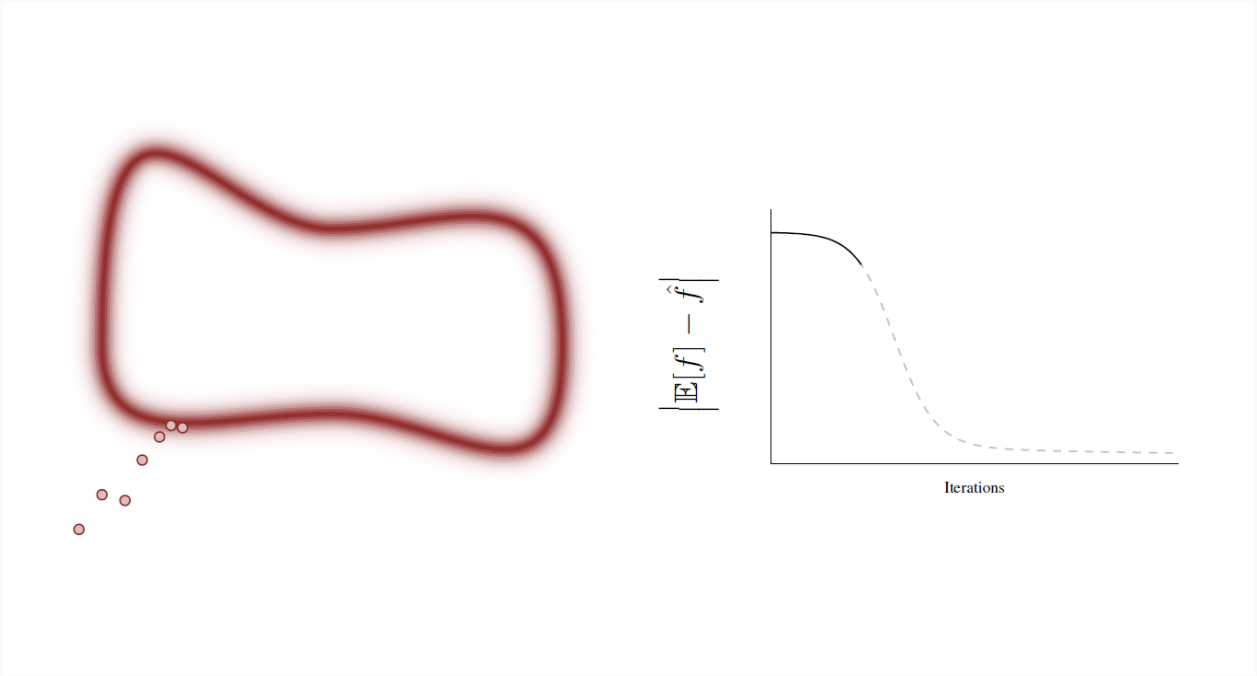
- We can accurately estimate expectations by averaging over the typical set instead of the entirety of parameter space. We need a method able to do it!
- MCMC uses a Markov chain to stochastically explore the typical set. However, some inefficiencies arise:
  - *random walk behaviour* (Gibbs sampling and random walk MH): the simulations can take a long time *zigging* and *zagging* while moving through the target distribution;
  - *finite time* exploration is not guaranteed...
  - *stuck* in *high curvature regions*, which are are hardly explored by Markov Chains.

Moving to Hamiltonian Monte Carlo ii

- HMC borrows strengths from physics to suppress the random walk behaviour in the Metropolis algorithm, thus allowing it to move much more rapidly through the target distribution.
- Ideal behaviour of the chains is then achieved by the so-called **geometric ergodicity**.
- To inquiry the differential structure of the target distribution, HMC uses the gradient of the log-posterior distribution, $\frac{d \log(\pi(\theta|y))}{d\theta}$: this will imply an adjustment of the algorithm towards the typical set area.
- Hamiltonian Monte Carlo is the unique procedure for automatically generating this coherent exploration for sufficiently well-behaved target distributions.

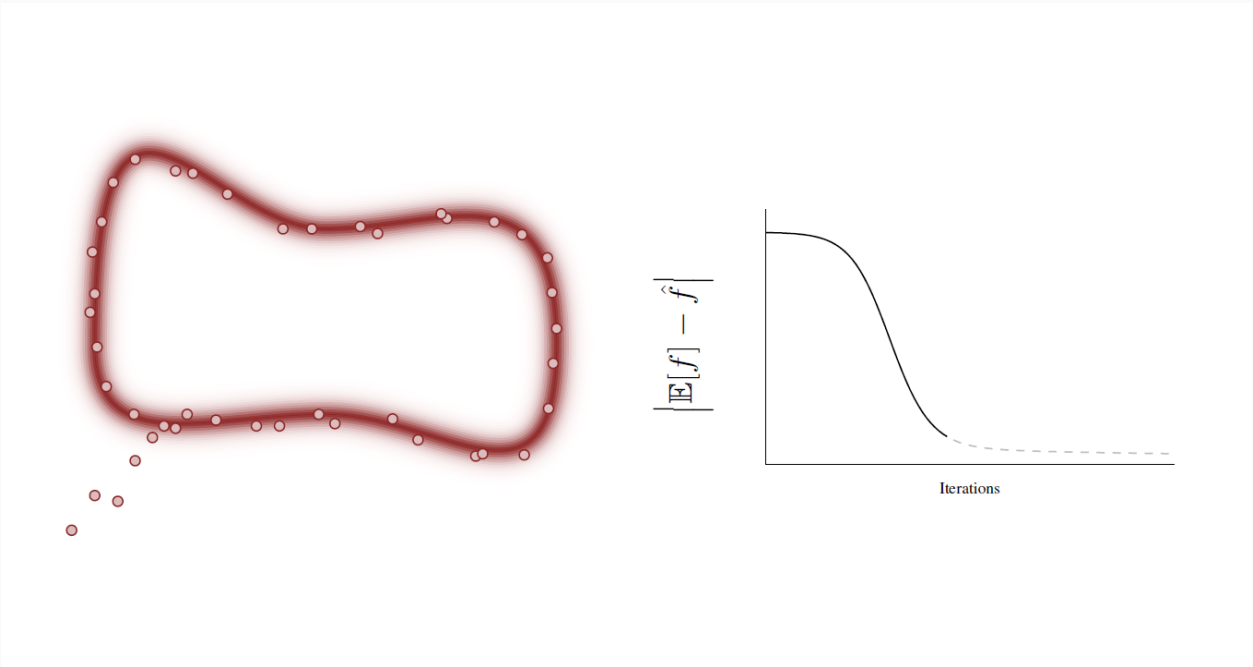
When something goes wrong

Under ideal conditions, MCMC estimators converge to the true expectations in a very practical progression.



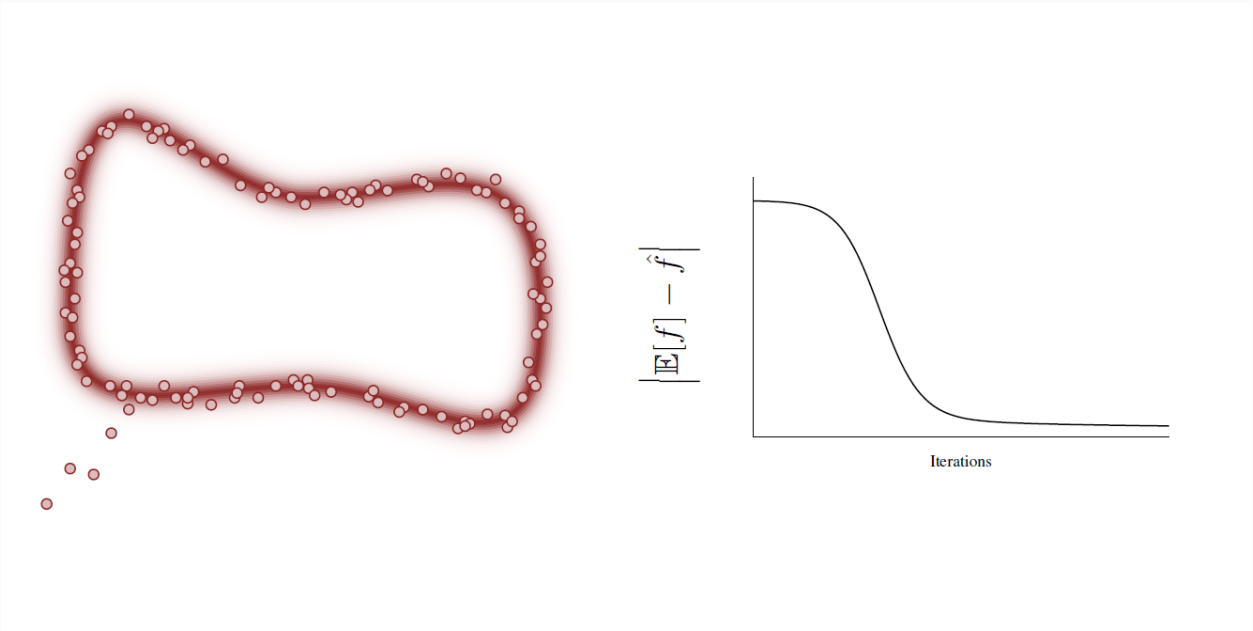
When something goes wrong

Under ideal conditions, MCMC estimators converge to the true expectations in a very practical progression.



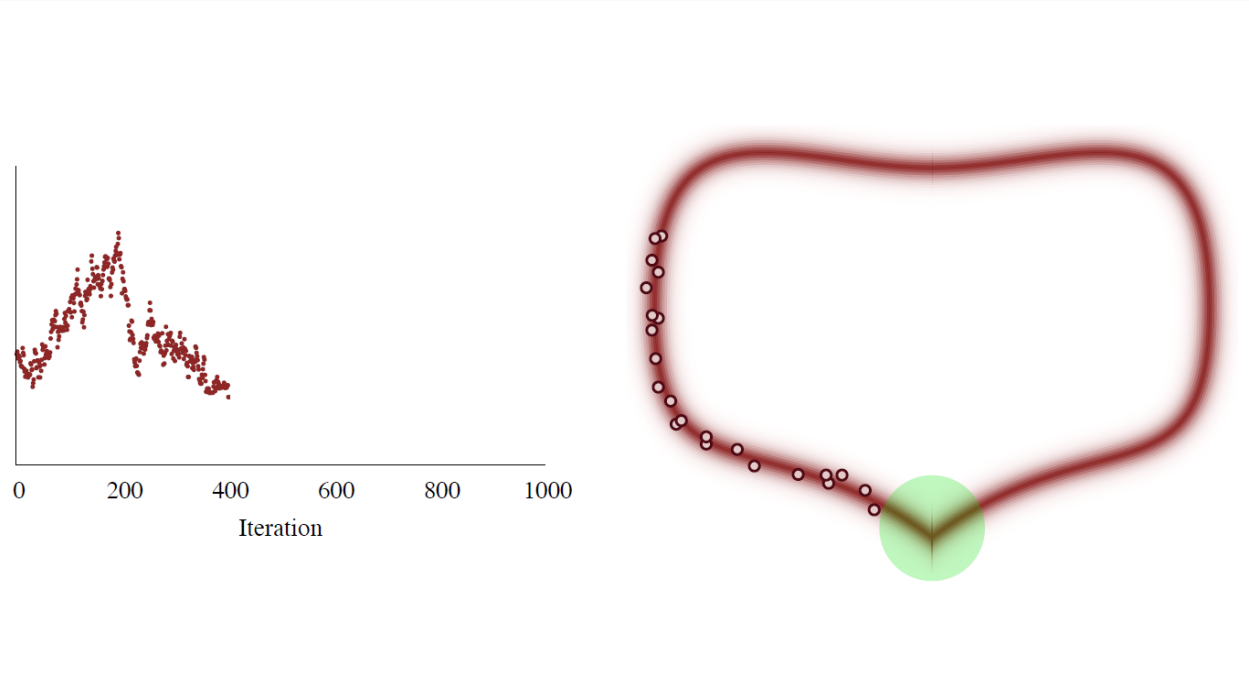
When something goes wrong

Under ideal conditions, MCMC estimators converge to the true expectations in a very practical progression.



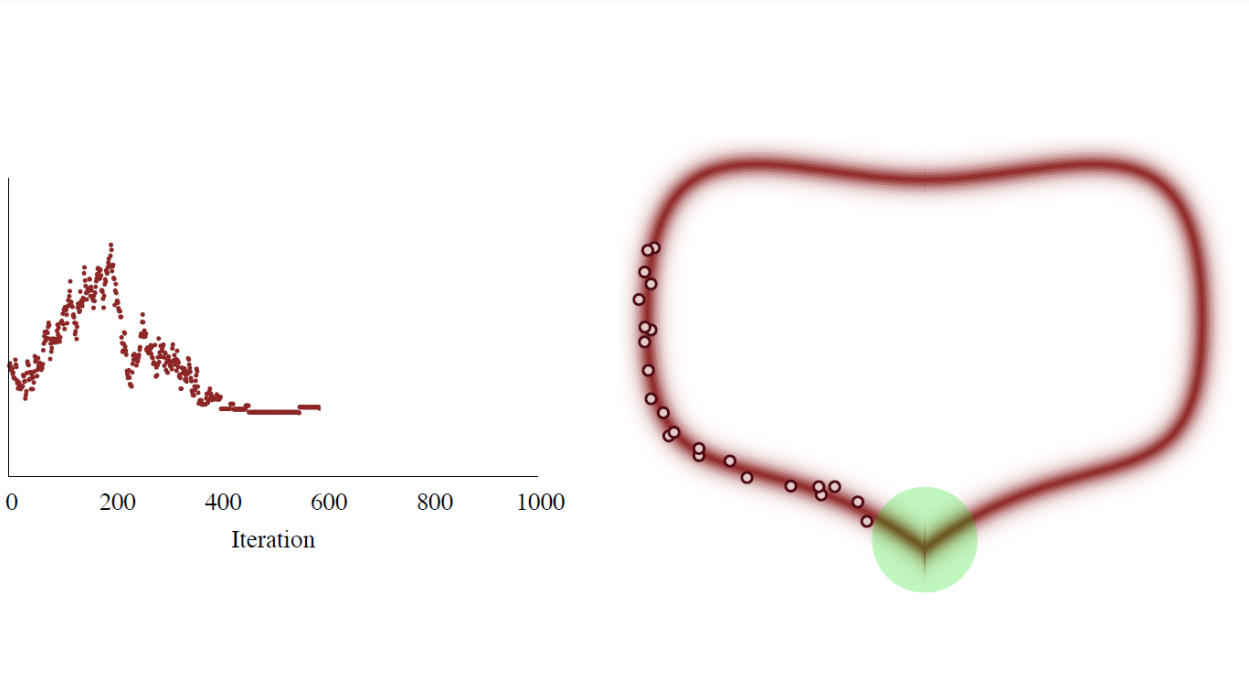
When something goes wrong

There are many pathological posterior geometries, however, that spoil these ideal conditions.



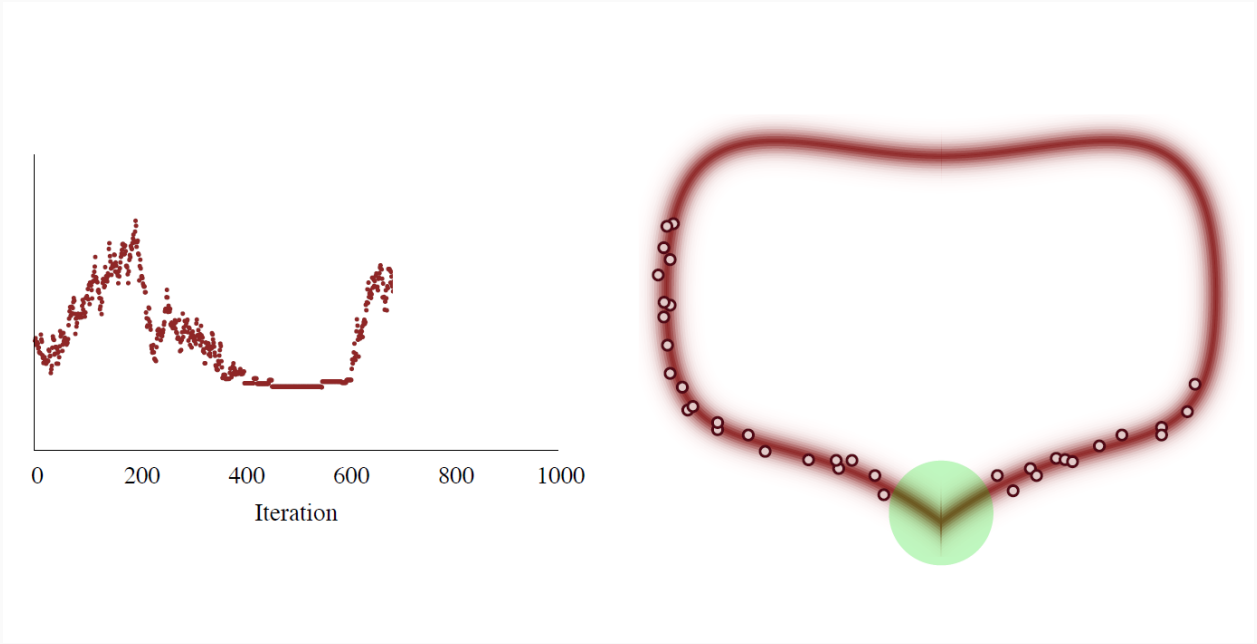
When something goes wrong

There are many pathological posterior geometries, however, that spoil these ideal conditions.



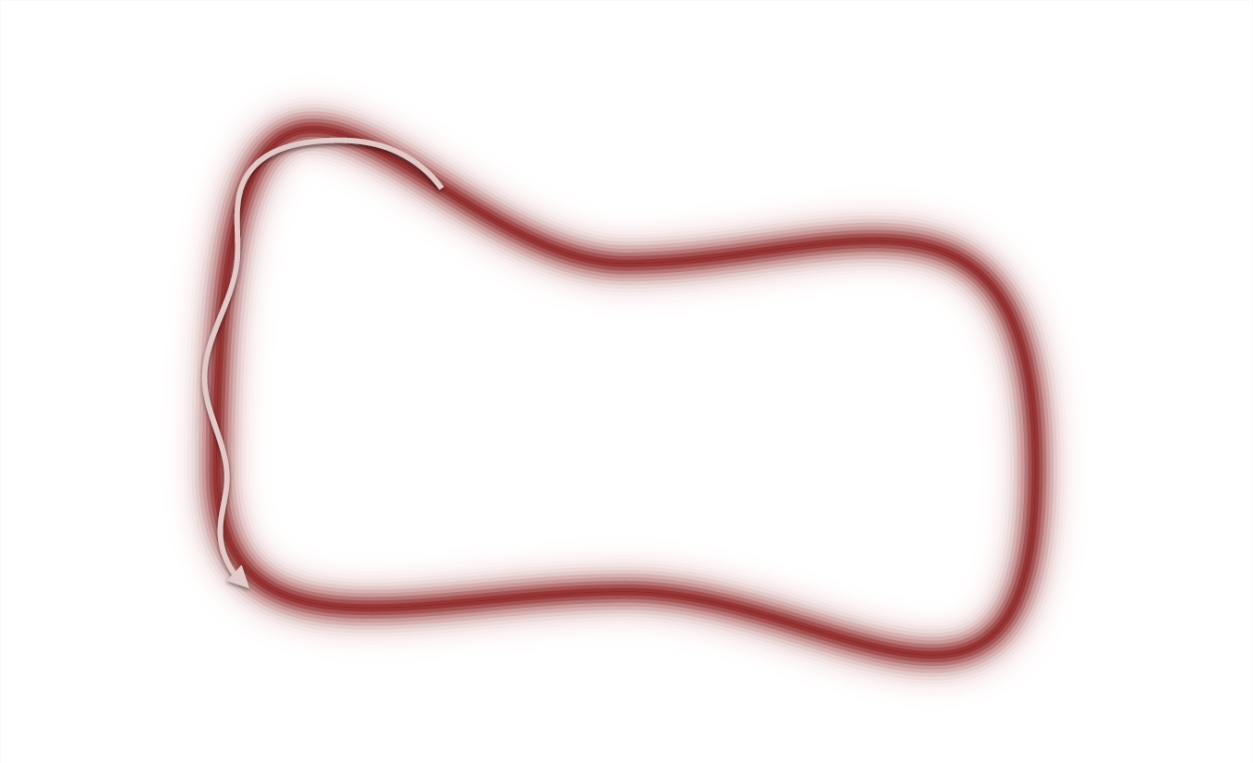
When something goes wrong

There are many pathological posterior geometries, however, that spoil these ideal conditions.



Hamiltonian Monte Carlo

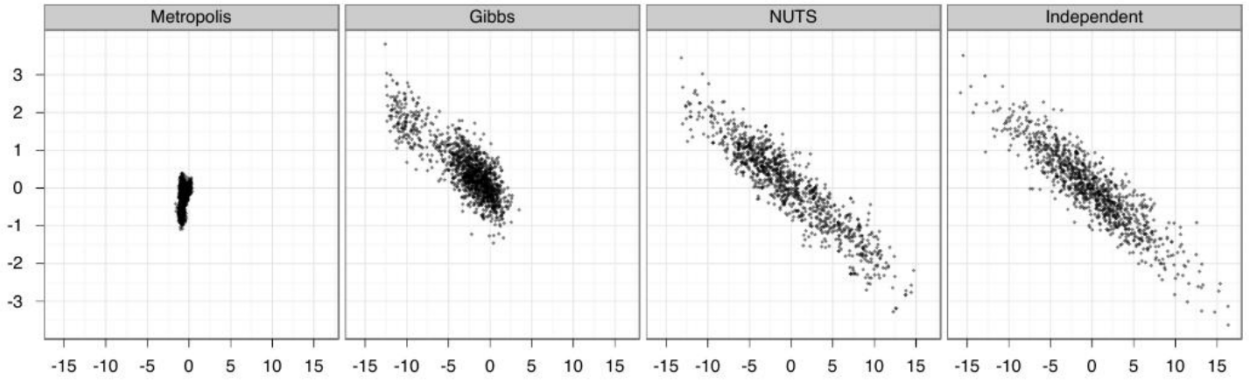
- Hamiltonian Monte Carlo yields fast, and *robust*, exploration of the distributions common in practice, by exploring the geometry of the typical set through the gradient of the target distribution.



Hamiltonian Monte Carlo: bivariate Gaussian

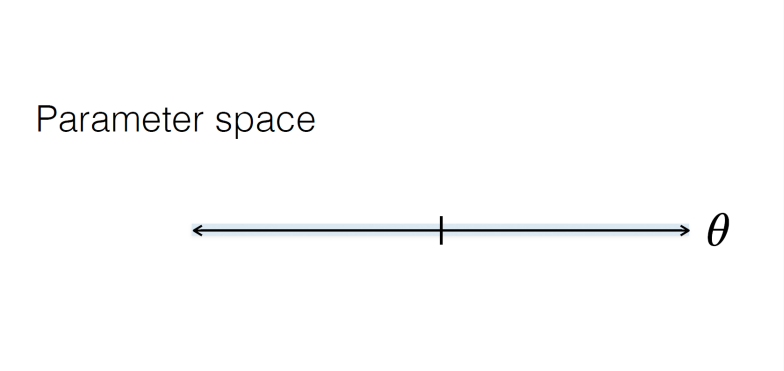
Comparison of algorithms on **highly correlated** 250-dimensional Gaussian distribution

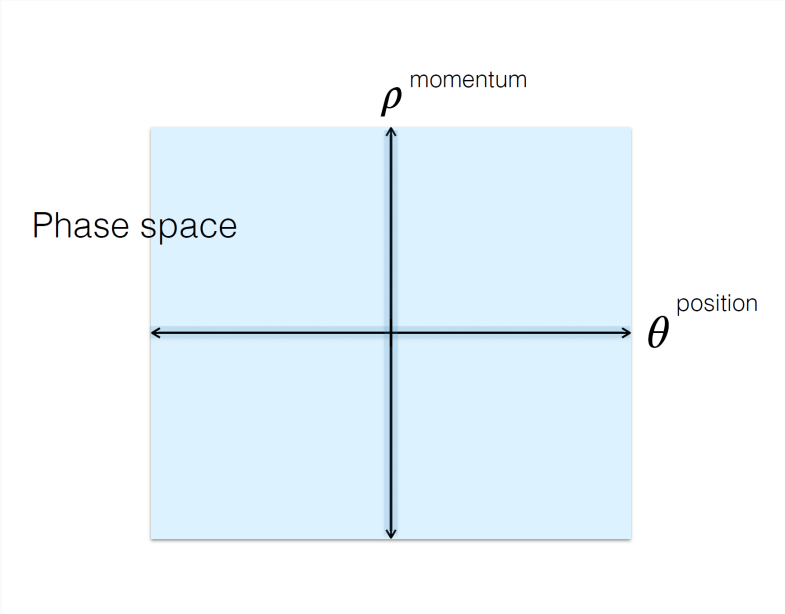
- Do **1,000,000** draws with both Random Walk Metropolis and Gibbs, thinning by 1000
- Do **1,000** draws using Stan's NUTS algorithm (no thinning)
- Do 1,000 independent draws (we can do this for multivariate normal)



Add the phase space

For each component θ_j in the target space, Hamiltonian Monte Carlo adds a *momentum* variable ρ_j . Both θ and ρ are then updated together in a new Metropolis algorithm, in which the jumping distribution for θ is determined largely by ρ.





Add the phase space

- Each iteration of HMC proceeds via several steps, during which the position and momentum evolve based on rules imitating the behavior of position the steps can move rapidly where possible through the space of θ and even can turn corners in parameter space to preserve the total *energy* of the trajectory.
- Hamiltonian Monte Carlo is also called *hybrid Monte Carlo* because it combines MCMC and deterministic simulation methods.
- In HMC, the posterior density $\pi(\theta|y)$ (which, as usual, needs only be computed up to a multiplicative constant) is augmented by an independent distribution $\pi(\rho)$ on the **momenta**, thus defining a joint distribution,

$$\pi(\theta, \rho|y) = \pi(\rho)\pi(\theta|y) \tag{2}$$

.

Add the phase space

- We simulate from the joint distribution but we are only interested in the simulations of θ; the vector ρ is thus an auxiliary variable, introduced only to enable the algorithm to move faster through the parameter space.
- HMC also requires the *gradient* of the log-posterior density. In practice the gradient must be computed analytically. If θ has d dimensions, this gradient is:

$$\frac{d \log \pi(\theta|y)}{d\theta} = \left(\frac{d \log \pi(\theta|y)}{d\theta_1}, \ldots, \frac{d \log \pi(\theta|y)}{d\theta_d} \right).$$

- For most of the models we consider in this course, this vector is easy to determine analytically and then program.

The momentum distribution

The momentum distribution

- It is usual to give ρ a multivariate normal distribution (recall that ρ has the same dimension as θ) with mean 0 and covariance set to a prespecified *mass matrix* M (so called by analogy to the physical model of Hamiltonian dynamics).
- To keep it simple, we commonly use a diagonal mass matrix, M. If so, the components of ρ are independent, with

$$\rho_j \sim \mathcal{N}(0, M_{jj}),\ j = 1, \ldots, d.$$

- It can be useful for M to roughly scale with the inverse covariance matrix of the posterior distribution, but the algorithm works in any case; better scaling of M will merely make HMC more efficient.

The three steps of an HMC iteration

The three steps of an HMC iteration i

HMC proceeds by a series of iterations (as in any Metropolis algorithm), with each iteration having three parts (a-b-c below):

1. **Update** ρ with a random draw from its posterior distribution—which, as specified, is the same as its prior distribution, $\rho \sim \mathcal{N}(0, M)$.
2. **Update** simultaneously θ and ρ via a discrete mimicking of physical *Hamiltonian dynamics* **equations**:

$$\frac{d\theta}{dt} = +\frac{\partial H}{\partial \rho} = \frac{\partial K}{\partial \rho}$$
$$\frac{d\rho}{dt} = -\frac{\partial H}{\partial \theta} = -\frac{\partial K}{\partial \theta} - \frac{\partial V}{\partial \theta},$$

where $K(\theta, \rho)$ is called the *kinetic energy*, and $V(\theta)$ is the *potential energy*.

The three steps of an HMC iteration ii

- To do this discretization, we can establish the following relationship between the posterior/target distribution and the Hamiltonian dynamics, by defining the **Hamiltonian Function** $H(\theta, \rho)$:

$$\begin{aligned} \pi(\theta, \rho|y) &= \exp\{-H(\theta, \rho)\} \\ H(\theta, \rho) &= -\log \pi(\theta, \rho|y) \\ &= -\log \pi(\rho|\theta, y) - \log \pi(\theta|y) \\ &= K(\theta, \rho) + V(\theta) \\ &= \text{kinetic} + \text{potential} \end{aligned} \tag{3}$$

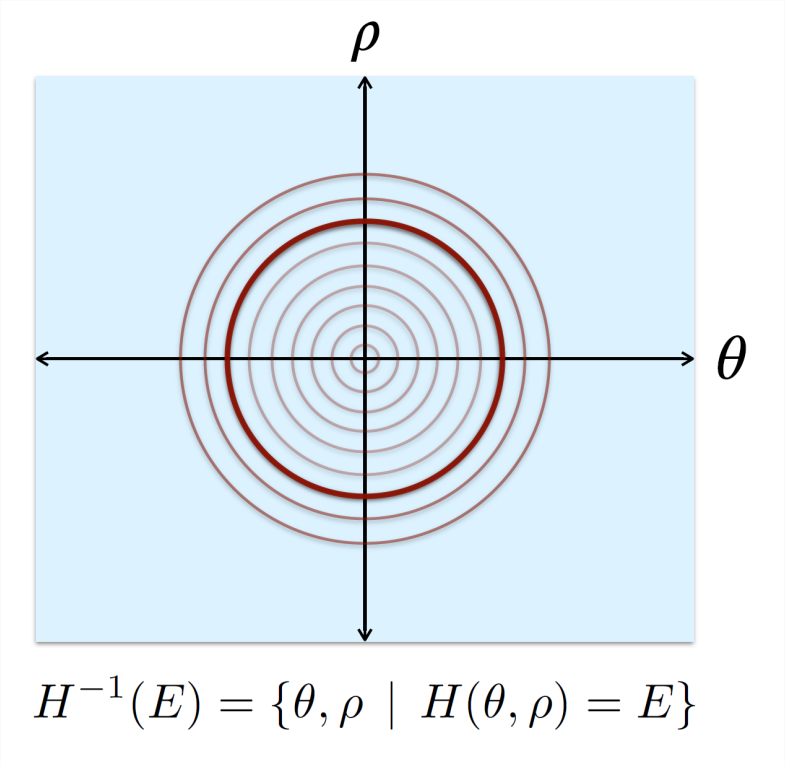
- Thus, the quantity $\partial V/\partial\theta$ is the gradient of the (log) target distribution.

The three steps of an HMC iteration iii

- The *canonical* density function $\pi(\theta, \rho|y)$ (2) ensures that if we marginalize out the momentum we immediately recover our target distribution. It does not depend on a particular choice of parameterization, and we can write it in terms of an invariant Hamiltonian function.
- For such characteristic, it captures the invariant probabilistic structure of the phase space distribution and, most importantly, the geometry of its typical set.
- Because the Hamiltonian captures the geometry of the typical set, we should be able to use it to generate a *vector field* oriented with the typical set of the canonical distribution and hence the trajectories that we are after. Indeed, the desired vector field can be generated from a given Hamiltonian with Hamilton's equations.

The three steps of an HMC iteration iv

- Phase space decomposes into concentric **energy** level sets of the Hamiltonian, $H^{-1}(E)$. Instead of specifying a point in phase space with its position and momentum, we can specify it with an energy, E, and its position on the corresponding level set, $\theta_E \in H^{-1}(E)$.



The three steps of an HMC iteration v

- To approximate the trajectory of the Hamiltonian dynamics and solve the differential equations (3), we need some *integrators*. HMC can be implemented in practice by using a *leapfrog* integrator assuming a time discretization—or **step size**—ϵ.
- This step involves L *leapfrog steps* (to be defined in a moment), each scaled by ϵ. In a leapfrog step, both θ and ρ are changed, each in relation to the other. The L leapfrog steps proceed as follows:

  Repeat the following steps L times:

  (a) Use the gradient (the vector derivative) of the log-posterior density of θ to make a half-step of ρ:

$$\rho \leftarrow \rho + \frac{1}{2}\epsilon \frac{d \log \pi(\theta|y)}{d\theta}.$$

The three steps of an HMC iteration vi

(b) Use the *momentum* vector ρ to update the *position* vector θ:

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

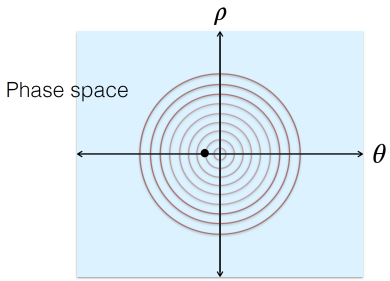
(c) Again use the gradient of θ to half-update ρ:

$$\rho \leftarrow \rho + \frac{1}{2} \epsilon \frac{d \log \pi(\theta|y)}{d\theta}.$$

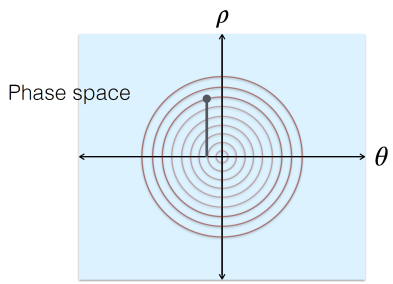
- Except at the first and last step, updates (c) and (a) above can be performed together.

The three steps of an HMC iteration vii

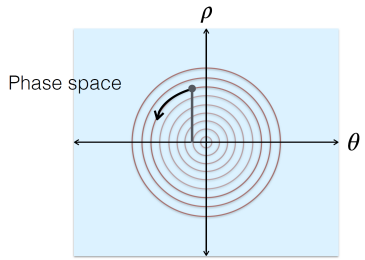
Pick an initialization point



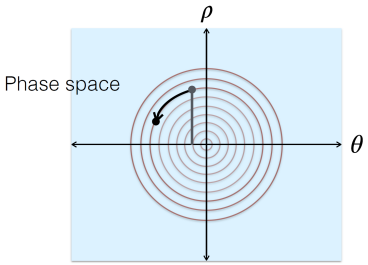
Sample momenta to lift into phase space



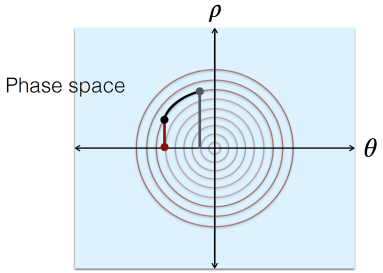
Deterministic exploration within energy levels sets



Deterministic exploration within energy levels sets



Project back down to parameter space



The auxiliary momenta are discarded and we are left with a point in the typical set of the target distribution

Parameter space

The three steps of an HMC iteration viii

- Some remarks sofar:
  - This algorithm (a)–(c) is called a 'leapfrog' because of the splitting of the momentum updates into half steps: is a *discrete approximation* to physical Hamiltonian dynamics in which both position and momentum evolve in continuous time.
  - In the limit of ϵ near zero, the leapfrog algorithm preserves the joint density $\pi(\theta, \rho|y)$.
  - For finite ϵ, the joint density $\pi(\theta, \rho|y)$ does not remain entirely constant during the leapfrog steps but it will vary only slowly if ϵ is small

The three steps of an HMC iteration ix

3. **Accept-reject step** Label $\theta^{(t-1)}, \rho^{(t-1)}$ as the value of the parameter and momentum vectors at the start of the leapfrog process and θ^*, ρ^* as the value after the L steps. In the accept-reject step, we compute the *acceptance ratio*:

$$R = \frac{\pi(\theta^*|y)\pi(\rho^*)}{\pi(\theta^{(t-1)}|y)\pi(\rho^{(t-1)})}.$$

4. **Final assignment** Set:

$$\theta^t = \begin{cases} \theta^* & \text{with probability } \min(R, 1) \\ \theta^{(t-1)} & \text{otherwise.} \end{cases}$$

5. **Repeat** these iterations until approximate convergence, as assessed by $\hat{R}$ being near 1 and the effective sample size being large enough for all quantities of interest.

The three steps of an HMC iteration x

Some remarks:

- After the point (4), the auxiliary momenta are discarded and we are left with a point in the typical set of the target distribution
- HMC can be tuned in three places: (i) the probability distribution for the momentum variables ρ (which, in our implementation requires specifying the diagonal elements of a covariance matrix, that is, a scale parameter for each of the d dimensions of the parameter vector), (ii) the scaling factor ϵ of the leapfrog steps, and (iii) the number of leapfrog steps L per iteration.
- Theory suggests that HMC is optimally efficient when its acceptance rate is approximately 65% (see diagnostics part).

HMC and Stan

Efficient HMC in Stan

- Hamiltonian Monte Carlo takes a bit of effort to program and tune. In more complicated settings, though, we have found HMC to be faster and more reliable than basic Markov chain simulation algorithms.
- To mitigate the challenges of programming and tuning, we use a computer program, Stan (Sampling through adaptive neighborhoods) to automatically apply HMC given a Bayesian model.
- The key steps of the algorithm are: (a) data and model input, (b) computation of the log posterior density (up to an arbitrary constant that cannot depend on the parameters in the model) and its gradients, (c) a warm-up phase in which the tuning parameters are set, (d) an implementation of the **no-U-turn sampler** to move through the parameter space, and (e) convergence monitoring and inferential summaries at the end.

Diagnostics i

In an ideal world, our simulation algorithm would return *iid* samples from the target (posterior) distribution.

However, MCMC/HMC simulation has two short-comings

- The distribution of the samples, $\tilde{\pi}(\theta^{(s)})$ only converges to the target distribution as $s \to \infty$.
- The samples are dependent.

Now we shall consider how we deal with these issues. In typical practice, one monitors the performance of an MCMC algorithm by:

- inspecting the value of the acceptance rate (in M-H only)
- constructing graphs
- computing diagnostic statistics on the stream of simulated values

Diagnostics ii

In both MCMC and HMC methods, some diagnostics about the convergence of the algorithm must be always checked. Quickly:

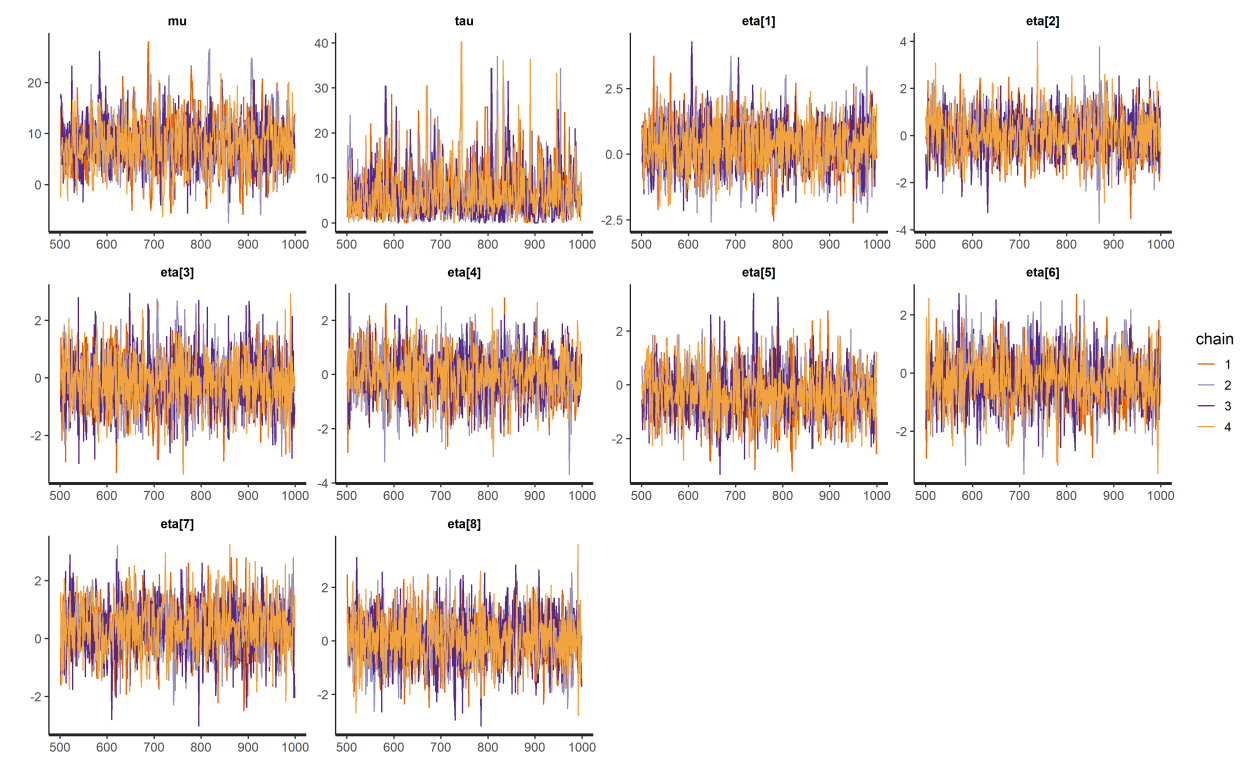
- Running multiple chains from different starting values that are over-dispersed relative to the posterior distribution (and checking the chains mixing).
- Checking the autocorrelation in the draws: a strong correlation between successive iterates may prevent the algorithm from exploring the entire region of the parameters' space.
- Running diagnostic tests: Gelman-Rubin statistics $\hat{R}$, Geweke diagnostics,...

Diagnostics iii

- **Gelman-Rubin statistic** $\hat{R}$:estimate the mixing chains and their convergence to the stationary distribution. As a golden rule, convergence and mixing of the chains is reached when $\hat{R} \leq 1.1$.
- **Effective sample size**: ESS estimates the reduction in the true number of samples, compared to iid samples, due to the autocorrelation in the chain. As a rule, the higher is the ESS, and the better is the performance.

In the next slide, parameters' traceplots: 4 chains (usual choice for Stan), perfect mixing.

Diagnostics iv



Divergences i

- In order to approximate the exact solution of the Hamiltonian dynamics we need to choose a step size ϵ of the leapfrog integrator governing how far we move each time we evolve the system forward. That is, the step size controls the resolution of the sampler.
- Unfortunately, for particularly hard problems there are features of the target distribution that are too small for this resolution. Consequently the sampler misses those features and returns biased estimates. Fortunately, this mismatch of scales manifests as divergences which provide a practical diagnostic.

Divergences ii

- A divergence arises when the simulated Hamiltonian trajectory departs from the true trajectory as measured by departure of the Hamiltonian value from its initial value. When this divergence is too high, the simulation has gone off the rails and cannot be trusted.
- The Stan interfaces report divergences as warnings and provide ways to access which iterations encountered divergences
- If the posterior is highly curved, very small step sizes are required for this gradient-based simulation of the Hamiltonian to be accurate. When the step size is too large (relative to the curvature), the simulation diverges from the true Hamiltonian.

Divergences iii

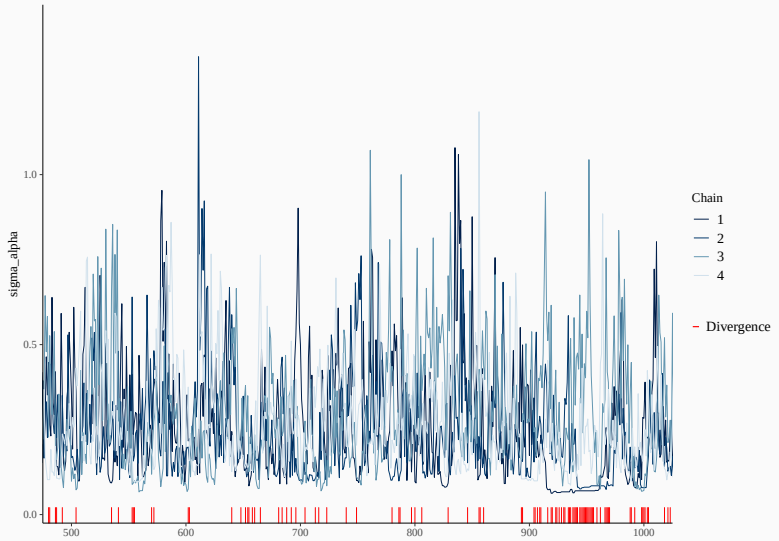
- In Stan, we can find some divergences:
  - *Divergent transitions after warmup* Recommendations: (1) Increase the target acceptance rate (2) Reparameterize your model.
  - *Maximum treedepth exceeded* Warnings about hitting the maximum treedepth are not as serious as warnings about divergent transitions. While divergent transitions are a validity concern, hitting the maximum treedepth is an efficiency concern. Recommendations: Increase the maximum allowed treedepth.
  - *BFMI low* You may see a warning that says some number of chains had an estimated Bayesian Fraction of Missing Information (BFMI) that was too low. This implies that the adaptation phase of the Markov Chains did not turn out well and those chains likely did not explore the posterior distribution efficiently.

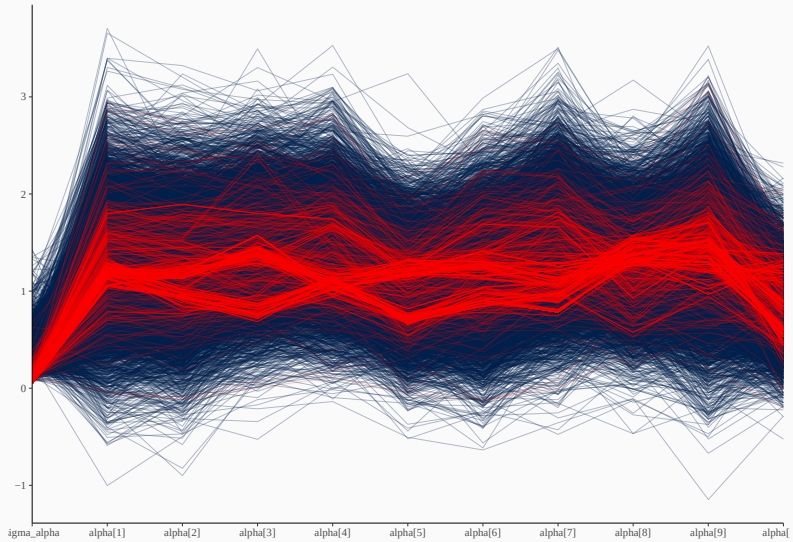
Divergences iv

- For a complete list, check: https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup.
- In Stan, the `adapt_delta` argument is the target average proposal acceptance probability during Stan's adaptation period, and increasing it will force Stan to take smaller steps. The downside is that sampling will tend to be slower because a smaller step size means that more steps are required. Since the validity of the estimates is not guaranteed if there are post-warmup divergences, the slower sampling is a minor cost.
- If the divergent transitions cannot be eliminated by increasing the `adapt_delta` parameter, we have to find a different way to write the model that is logically equivalent but simplifies the geometry of the posterior distribution. This problem occurs frequently with hierarchical models

Divergences v

- Luckily, Stan allows graphical inspection for the divergences:





Further reading

- *A conceptual introduction to Hamiltonian Monte Carlo*, by M.Betancourt.
- Chapters 11-12 from *Bayesian Data Analysis*, by A. Gelman et al.