

Lecture 2: a romp through MCMC

Ben Lambert¹

`ben.c.lambert@gmail.com`

¹Imperial College London

Monday 22nd July, 2019

Lecture outcomes

- 1 Appreciate how sampling can be used to gain insight into a distribution.
- 2 Grasp how **dependent sampling** via **MCMC** allows sampling from the posterior.
- 3 Understand the mechanics of Random Walk Metropolis and how it works intuitively.
- 4 Know that judging convergence of chains to the posterior is *hard*.
- 5 Learn how adaptive covariance MCMC, Gibbs sampling and Hamiltonian Monte Carlo can speed up sampling in most cases.

- 1 Understanding a distribution by sampling from it
- 2 Introducing dependent sampling
- 3 Random Walk Metropolis
- 4 Judging convergence of chains to posterior
- 5 Adaptive covariance MCMC
- 6 Gibbs sampling
- 7 Introduction to Hamiltonian Monte Carlo

What is (independent) sampling and how can it give insight to distributions?

- Suppose we have a large (infinite) urn filled with coloured balls.
- The number of colours and the frequencies of each are **unknown**.
- **Question:** how can we determine the underlying probability distribution of ball colour?

What is (independent) sampling and how can it give insight to distributions?

Answer: we draw lots of balls from the urn and count the **sampled** frequencies!

What is (independent) sampling and how can it give insight to distributions?

- Drawing one ball from the urn is the act of taking a single **sample**.
- If the balls in the urn are swishing about then the colour of the next ball does not depend on the current ball's colour.
- Here the samples are (conditionally-) **independent**.
- Independent sampling gives us a very **efficient** way of gaining insight into a distribution.

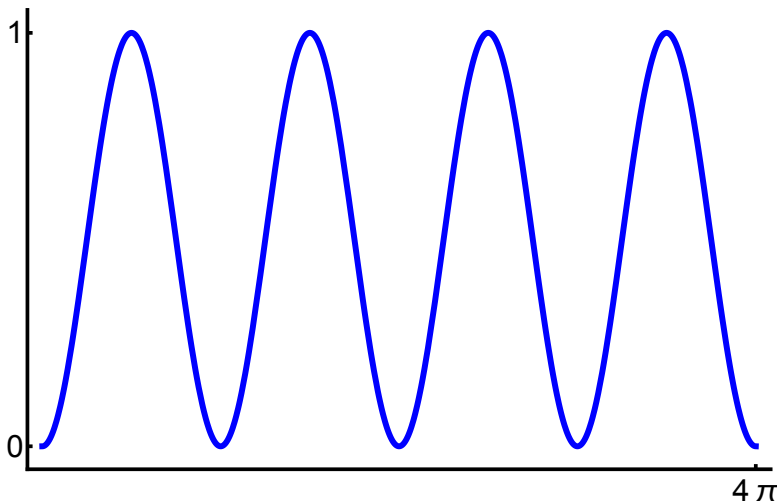
Sampling from a continuous distribution

- Suppose we have a large (infinite) urn filled with balls of differing sizes.
- The distribution of sizes is **unknown**.
- **Question:** can we use same method to determine the underlying probability distribution of ball size? **Answer:** yes!

Sampling from a continuous distribution

Generating independent samples: sine curve

Question: how can we generate independent samples from the following (un-normalised) PDF?



Generating independent samples: sine curve

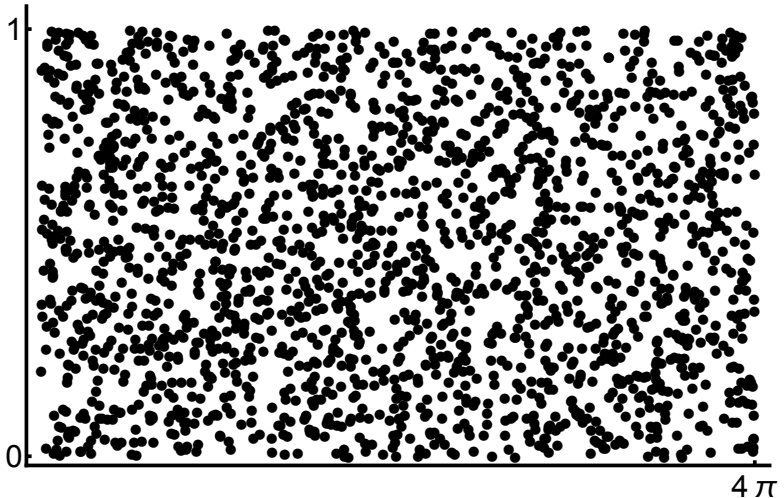
Answer: do the following a large number of times:

- 1 Generate **x** coordinates: uniformly-distributed points from $(0, 4\pi)$; where 4π is the domain of the function.
- 2 Generate **y** coordinates: uniformly-distributed points from $(0, 1)$; where 1 is the maximum value of the function.
- 3 If $y < p(x)$, then **accept** **x** coordinate as a sample.
- 4 If $y \geq p(x)$, then **reject** **x** coordinate as a sample.

Known as **Rejection** sampling.

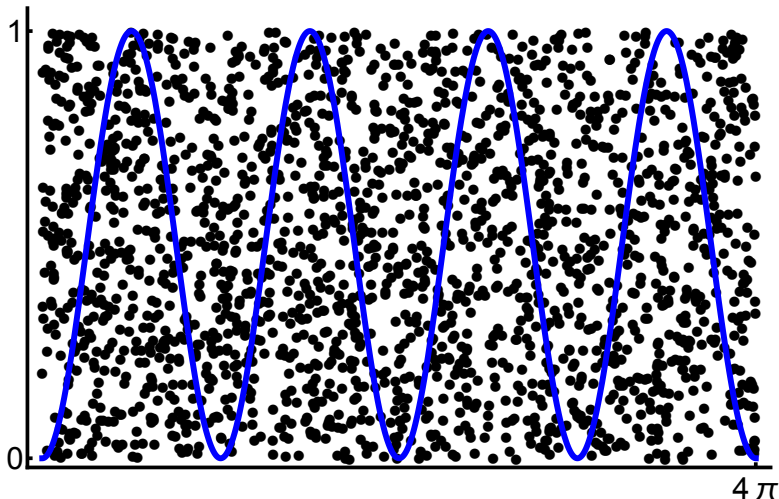
Generating independent samples: sine curve

Generate x and y coordinates.



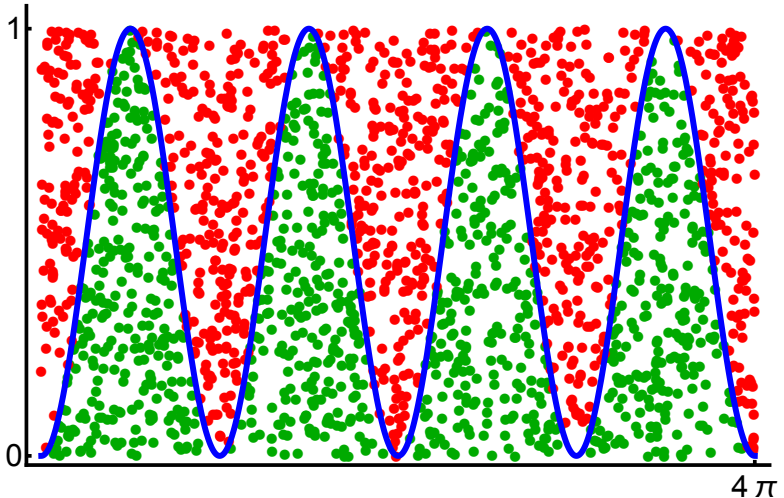
Generating independent samples: sine curve

Overlay pdf.



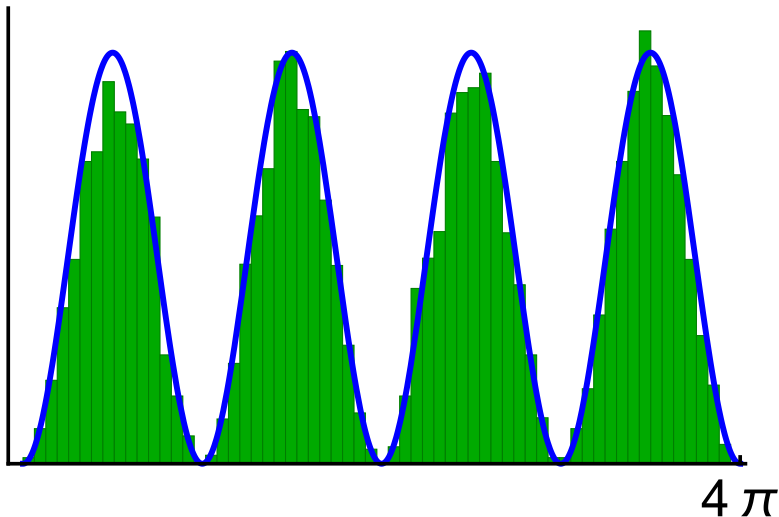
Generating independent samples: sine curve

Accept x coordinates as samples if $y < p(x)$.



Generating independent samples: sine curve

The resultant samples.



Why do sampling in the first place?

Typically we want to calculate the posterior mean of some parameter, θ_1 :

$$\begin{aligned} E(\theta_1|X) &= \int_{\Theta_1} \int_{\Theta_{-1}} \theta_1 \times p(\theta_1, \theta_{-1}|X) d\theta_{-1} d\theta_1 \\ &= \int_{\Theta_1} \theta_1 \times p(\theta_1|X) d\theta_1 \end{aligned}$$

where θ_{-1} corresponds to the $d - 1$ other parameters of the model.

This integral (the top line) is just too difficult to calculate exactly for all but the simplest models \implies we instead use sampling to approximate it!

Why is generating independent samples difficult?

- **Rejection sampling** requires generation of a large number of random points to produce relatively few samples.
- This inefficiency increases (exponentially) with the dimensionality of the distribution; i.e. for posteriors with more parameters.
- Other methods exist (inverse-transform sampling and importance sampling, for example) but they suffer from complexity and/or inefficiency issues.
- We cannot calculate the denominator so are unable to use some of these methods.
- Even if we had the denominator the complexity of most models means that independent sampling isn't possible.

Is sampling finished?



- 1 Understanding a distribution by sampling from it
- 2 Introducing dependent sampling
- 3 Random Walk Metropolis
- 4 Judging convergence of chains to posterior
- 5 Adaptive covariance MCMC
- 6 Gibbs sampling
- 7 Introduction to Hamiltonian Monte Carlo

What is dependent sampling?

Definition:

“A sampling algorithm where the next sample **depends** on the current value.”

And the list of all (accepted) positions of the sampler form the sample.

Dependent samplers as Markov Chains (Monte Carlo)

- Where to step next is determined via a distribution *conditional* on the current parameter value.
- This stepping is probabilistic \implies *Monte Carlo*.
- The conditional distribution only depends on the current value of the sampler meaning it is memoryless about past path.
- This memoryless means that the path of the sampler is a *1st order Markov Chain*.

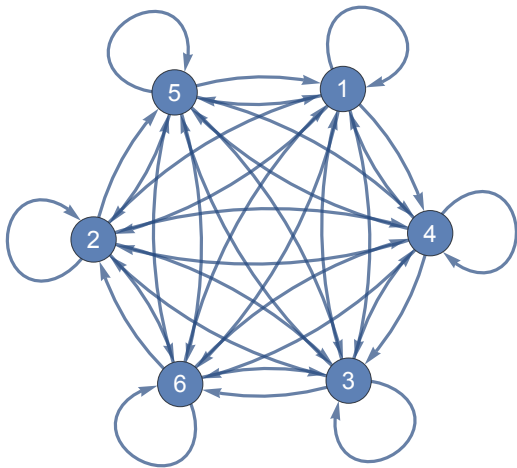


Example dependent sampler: live and let die

A standard die has six faces, all of which are equally likely to be obtained on a given throw.



Standard die



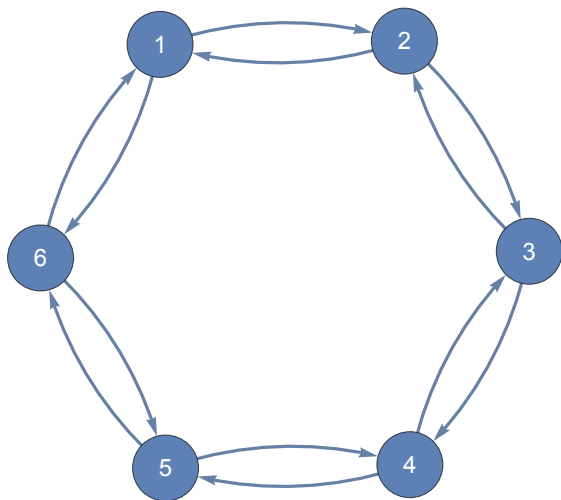
Introducing a Markovian die

Now suppose we have a die where from each number only consecutive numbers can be obtained,

- $1 \rightarrow 2$ or $1 \rightarrow 6$; each with probability $1/2$.
- $2 \rightarrow 3$ or $2 \rightarrow 1$; each with probability $1/2$.
- ...
- $6 \rightarrow 1$ or $6 \rightarrow 5$; each with probability $1/2$.

This die has **dependence** – the next value we obtain depends on the current value! **However** has same unconditional distribution (i.e. across all throws) as independent die \rightarrow same mean.

A Markovian die



A Markovian die

Question: Which of these two dies – the independent and Markovian one – is better able to estimate **mean** of the die?

Answer: shake it off (again)!



Markovian die

Independent die

Effective sample size

Question: How do we quantify performance of two sampling algorithms?

Answer: Estimate the number of effective samples per X iterations,

“The **effective sample size** for X iterations is the equivalent number of samples from an independent sampler.”

Effective sample size: depends on dependence

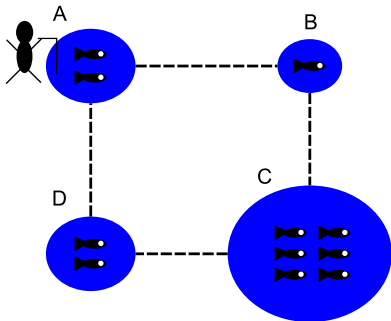
- The Markovian die performs worse than the independent sampler.
- This is due to the **dependence** of throwing the Markovian die \implies takes longer for sampler to explore parameter space!
- As dependence \uparrow the gap between the independent sampler and the Markovian one increases.
- Therefore conclude that as dependence \uparrow the effective sample size \downarrow .

Effective sample size: summary

- The worth of a sample is **not** dictated by its number of samples per second.
- More important is the net information gained per second.
- As dependence of sampler increases there is less incremental information gained per sample.
- Quantify this using concept of effective sample size – the equivalent number of samples from an independent sampler.
- Stan calculates this **automatically** for all parameters! No need to know formula.

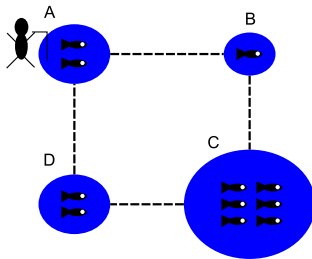
- 1 Understanding a distribution by sampling from it
- 2 Introducing dependent sampling
- 3 Random Walk Metropolis**
- 4 Judging convergence of chains to posterior
- 5 Adaptive covariance MCMC
- 6 Gibbs sampling
- 7 Introduction to Hamiltonian Monte Carlo

David Robinson's fishing



- David Robinson (a more fortunate cousin of Robinson Crusoe) is marooned on an island.
- Access to four freshwater lakes of different sizes; each with a supply of fish.

David Robinson's fishing



- Robinson does not know the amount of fish in each lake.
- He also does not know the number of lakes!
- However, the amount of fish in each lake is proportionate to its area.
- From a particular lake he can see the two adjoining lakes, and can estimate their area.

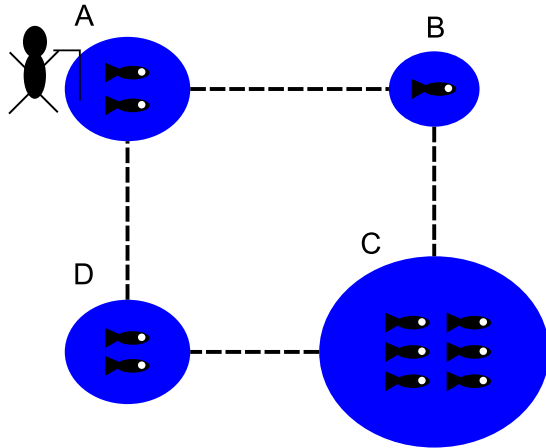
David Robinson's fishing

- He has a terrible memory (too much coconut toddy), and each day forgets any estimates of lake size he made previously.
- He wants to fish (at maximum) one new lake per day.
- He possesses a coin and a solar-powered calculator that can generate (pseudo-)random numbers uniformly distributed between 0 and 1.
- He is initially “washed up” next to lake A.



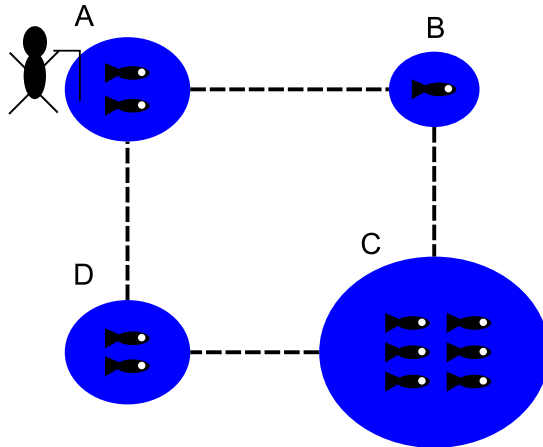
David Robinson's fishing

- **Question:** What strategy should he use to fish as sustainably as possible?



David Robinson's fishing

- **Remember:** Robinson doesn't know the # of lakes, nor the amount of fish in each!



David Robinson's fishing

Answer: visit each lake in proportion to the fish it contains, by doing the following:

- 1 Each night he flips the coin.
- 2 If it's heads (tails) he proposes a move to the neighbouring lake in the clockwise (anticlockwise) direction.
- 3 Calculates the ratio of the size of the proposed lake to the current one.
- 4 Compares the ratio with a (pseudo-)random number from the calculator.
- 5 If the ratio exceeds the generated number, he moves. If not, he stays put and fishes the same lake tomorrow.

David Robinson's fishing: does it work?

David Robinson's fishing: summary

- Robinson lacked knowledge of *numbers* of fish in each lake and the number of lakes.
- Knows that the number of fish in each lake is proportionate to its size.
- His memory stops him remembering the exact sizes.
- Each night he flips a coin; heads (tails) \implies consider clockwise (anticlockwise) neighbouring lake.
- Estimates ratio of size of selected lake to current one.
- If ratio exceeds a uniform random number he moves. If not he stays where he is.
- After about 100 days his “random” strategy is quite similar from an “omniscient” one.

Defining Random Walk Metropolis

Robinson's strategy is an example of the "Random Walk Metropolis" algorithm. This has the following form:

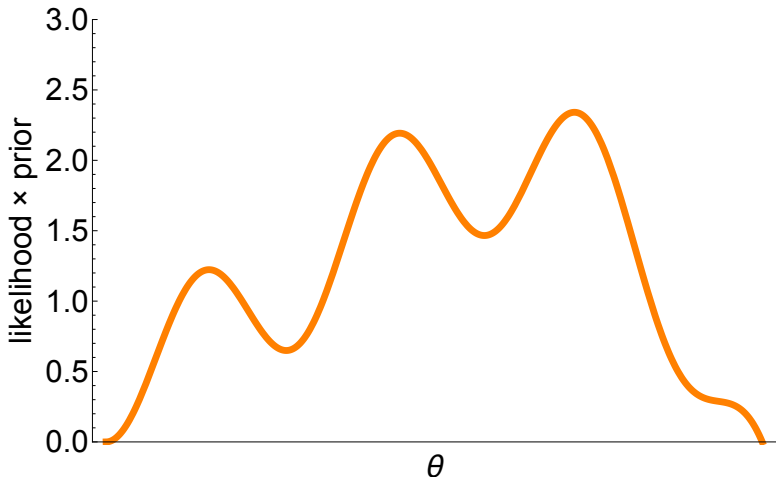
- Generate a random starting location θ_0 .
- Iterate the following for $t = 1, \dots, T$:
 - Propose a new location from a jumping distribution:
 $\theta_{t+1} \sim J(\theta_{t+1}|\theta_t)$.
 - Calculate the ratio:

$$r = \frac{\text{likelihood}(\theta_{t+1}) \times \text{prior}(\theta_{t+1})}{\text{likelihood}(\theta_t) \times \text{prior}(\theta_t)} \quad (1)$$

- Compare r with a uniformly-distributed number u between 0 and 1.
- If $r \geq u \implies$ we move.
- Otherwise, we remain at our current position.

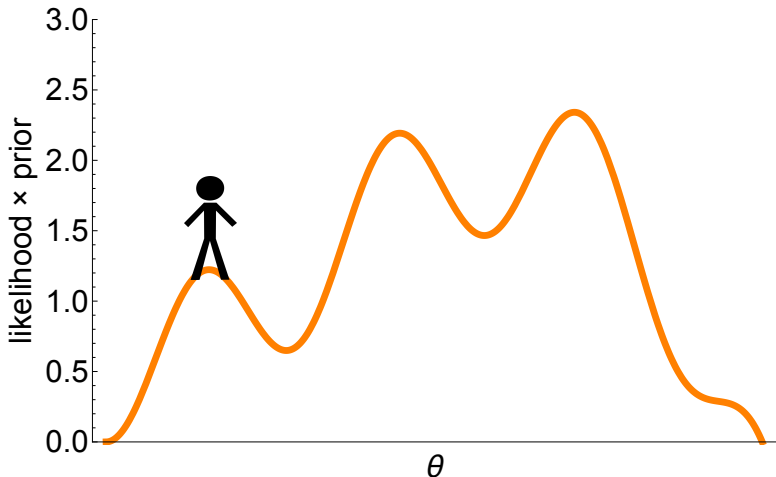
Defining Random Walk Metropolis

Start with the un-normalised density.



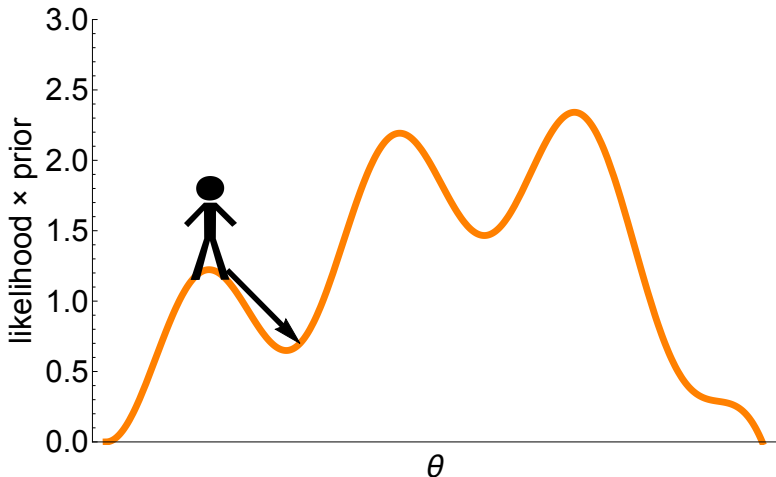
Defining Random Walk Metropolis

Select a random starting location.



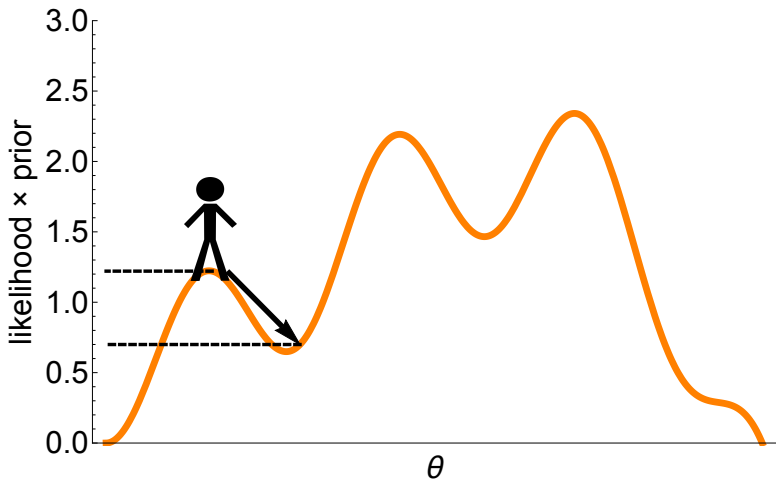
Defining Random Walk Metropolis

Propose a new location using jumping distribution.



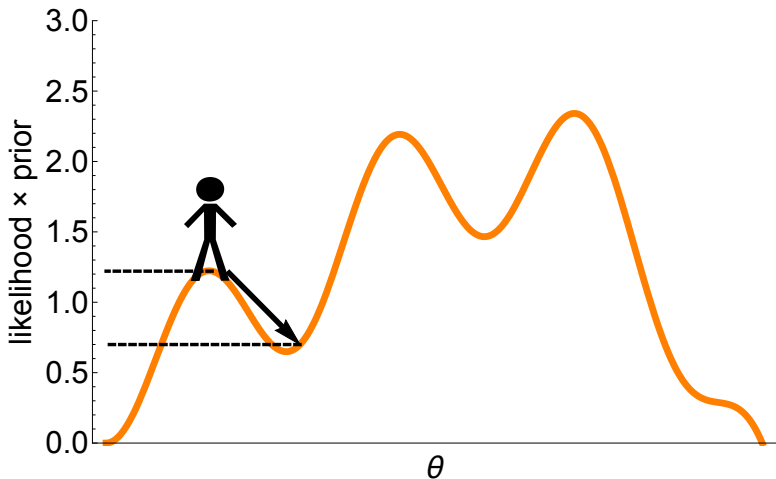
Defining Random Walk Metropolis

Calculate ratio of likelihood \times prior at proposed to current location, and find $r \approx 0.58$.



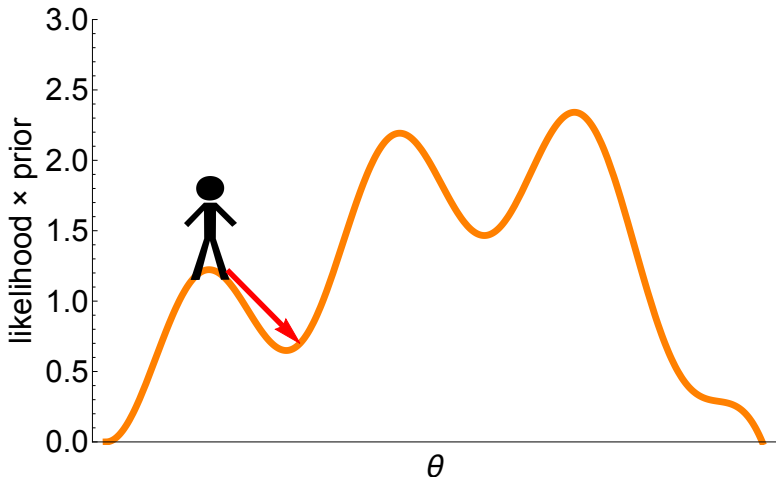
Defining Random Walk Metropolis

Compare $r \approx 0.58$ with random real between 0 and 1. For example suppose we obtain $u = 0.823$.



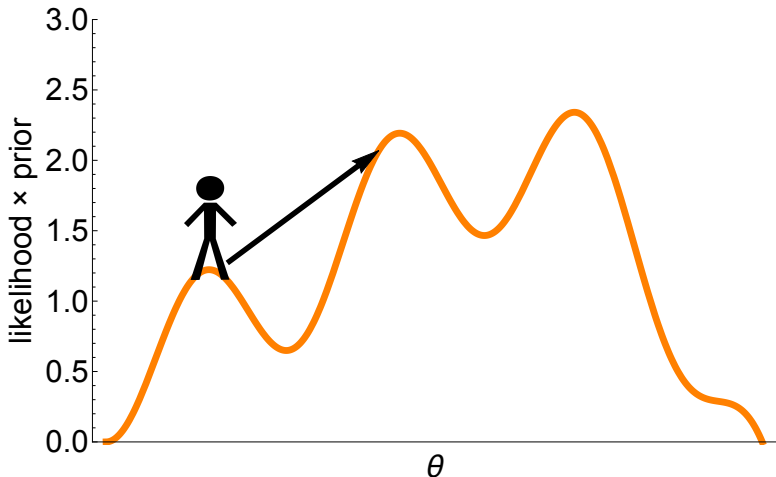
Defining Random Walk Metropolis

Since $r < u$ we remain at our original location.



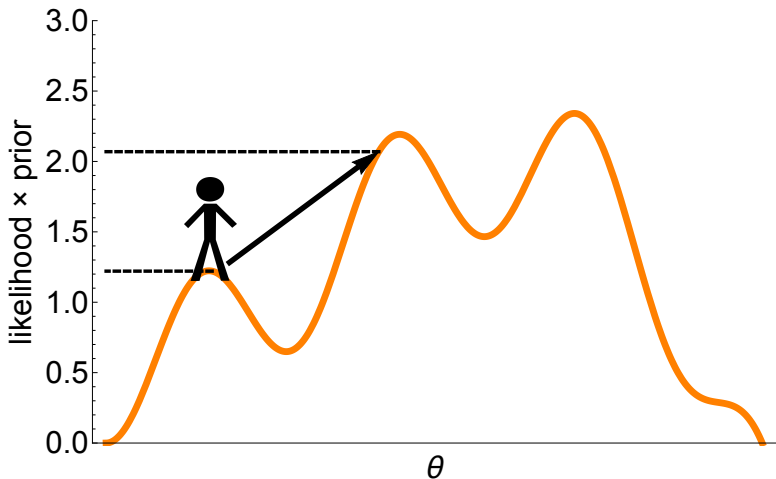
Defining Random Walk Metropolis

Generate a new proposed step using jumping distribution.



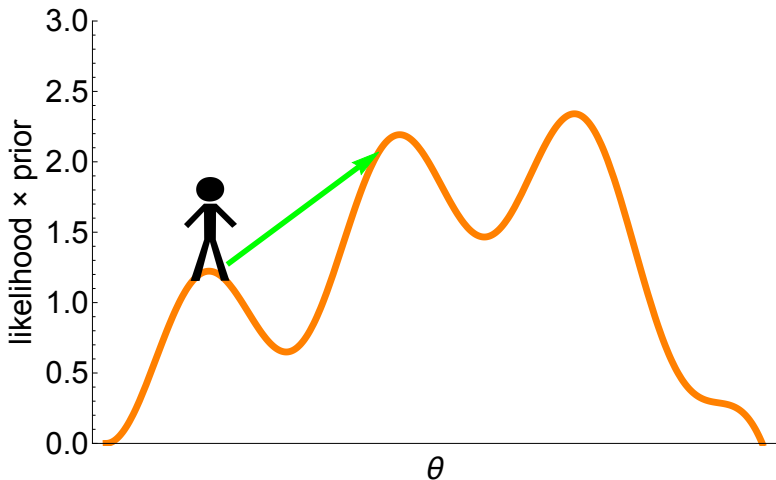
Defining Random Walk Metropolis

Calculate ratio of likelihood \times prior at proposed to current location, and find $r \approx 1.75$.



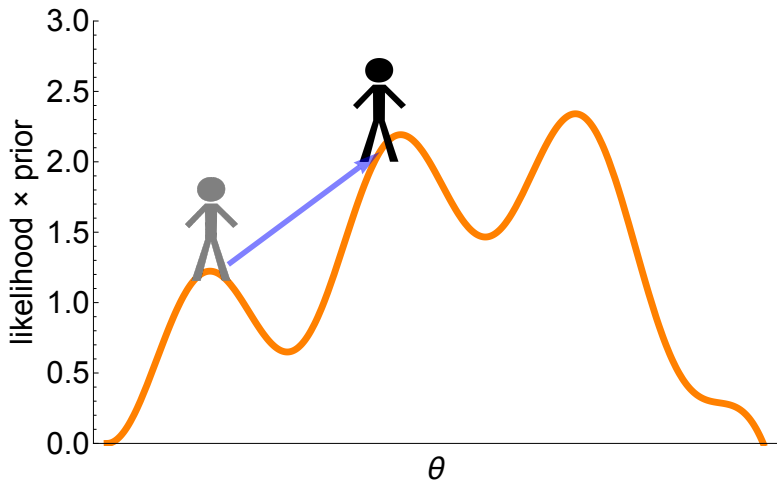
Defining Random Walk Metropolis

Since $r > 1$ (maximum possible u) \implies we move to new location.



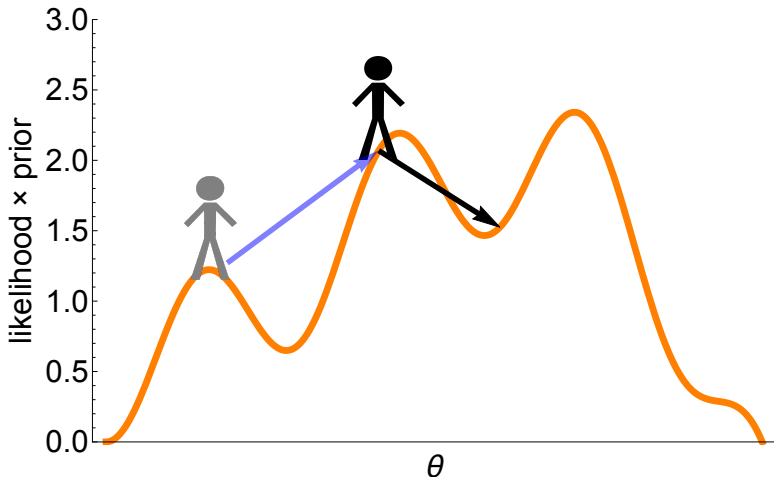
Defining Random Walk Metropolis

Since $r > 1$ (maximum possible u) \implies we move to new location.



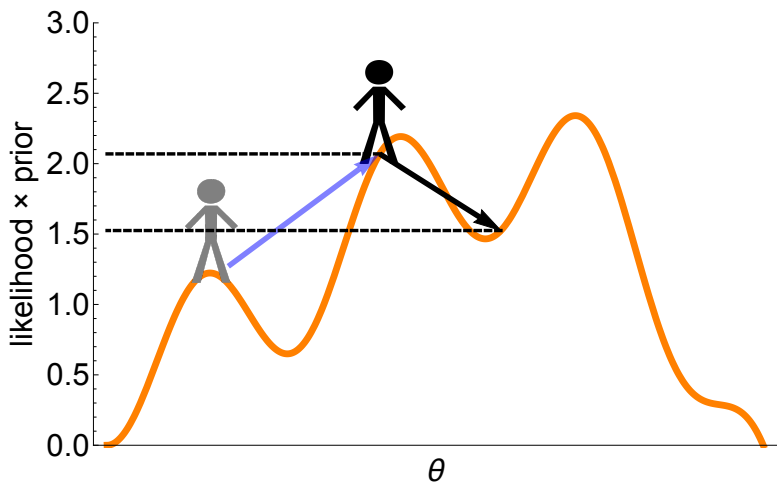
Defining Random Walk Metropolis

Propose a new step using jumping distribution.



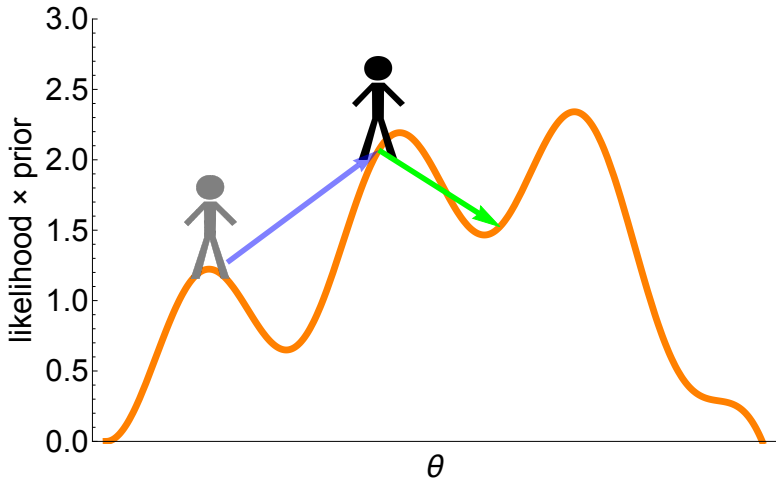
Defining Random Walk Metropolis

Calculate $r \approx 0.75$.



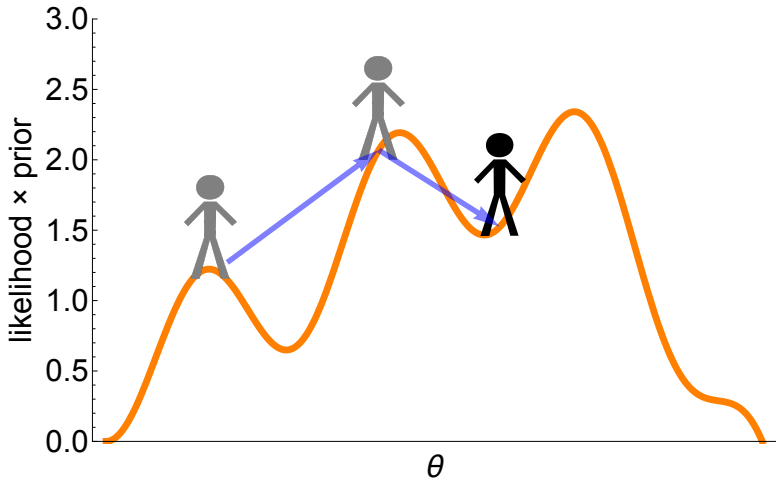
Defining Random Walk Metropolis

Generate $u = 0.278 < r \implies$ we move!



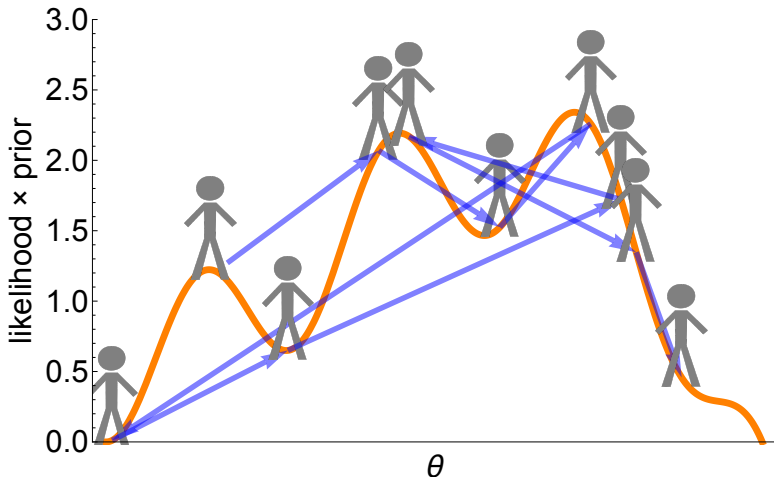
Defining Random Walk Metropolis

Generate $u = 0.278 < r \implies$ we move!



Defining Random Walk Metropolis

Repeat a large number of times.



Random Walk Metropolis: benefits

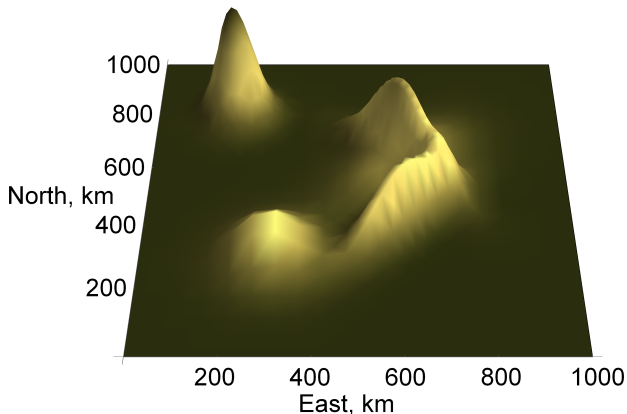
- Under quite general conditions the Random Walk Metropolis sampler converges **asymptotically** to the posterior.
- However for a sufficiently large sample size the sampling distribution may be practically indistinguishable from the true posterior.
- The algorithm requires us to be able to calculate the ratio:

$$r = \frac{\text{likelihood}(\theta_{t+1}) \times \text{prior}(\theta_{t+1})}{\text{likelihood}(\theta_t) \times \text{prior}(\theta_t)} \quad (2)$$

- The ratio uses **only** the numerator of Bayes' rule \implies we side-step calculating the denominator!

Random Walk Metropolis in action

Can we use Random Walk Metropolis to sample from the continuous distribution below?



Random Walk Metropolis in action

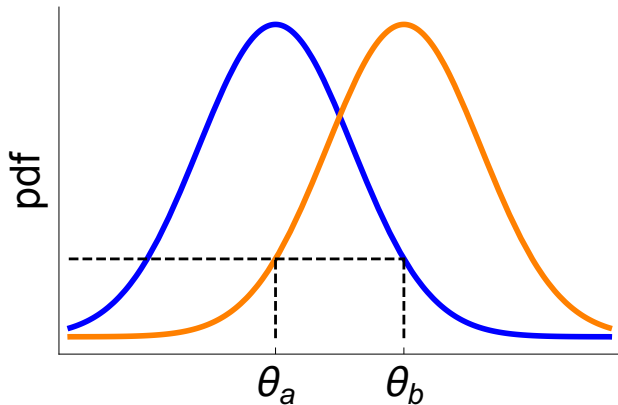
Random Walk Metropolis in action

Random Walk Metropolis: short summary

- Algorithm works by starting in a randomly-determined position in parameter space.
- In each iteration we generate a proposed (local) step from our current position.
- We then move based the ratio of the proposed **un-normalised** posterior to our current location \implies no need to calculate troublesome denominator.
- The path of our positions over time forms our **sample**.
- If we repeat the above for a (large) number of steps \implies sampling distribution \approx posterior.

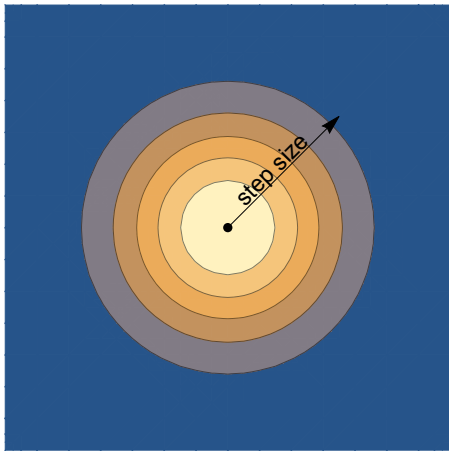
How do we choose the jumping distribution?

- Sometimes called the “proposal distribution”.
- In Random Walk Metropolis we use a symmetric distribution (relaxed in Metropolis-Hastings):
 $\implies J(\theta_a|\theta_b) = J(\theta_b|\theta_a)$



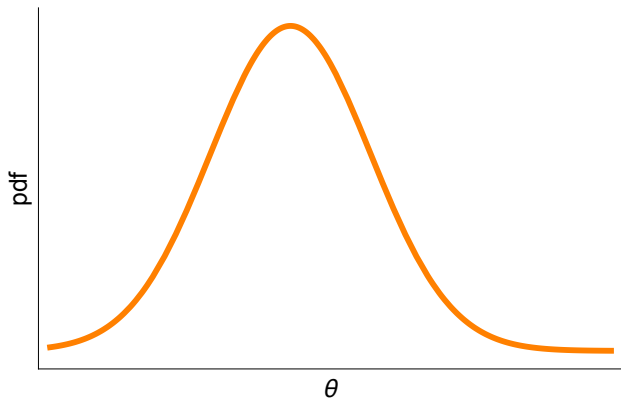
The importance of step size

Question: how should we decide on the jumping kernel's step size?



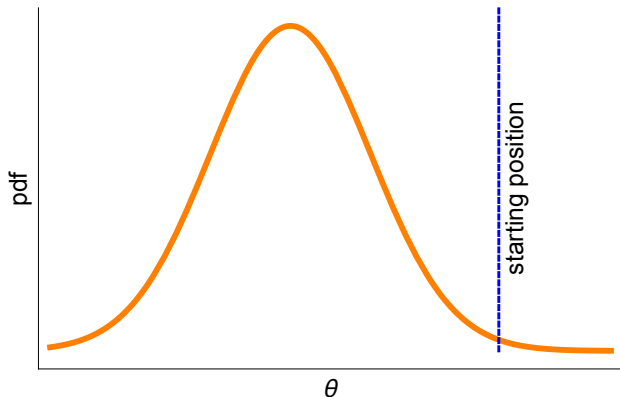
Another example posterior distribution

Assume a unimodal distribution from which we want to sample.



Another example posterior distribution

Start three algorithms with different step sizes at same point.



The importance of step size: too small

The importance of step size: too large

The importance of step size: just right

Step size: summary

- Whilst step size does not affect asymptotic convergence, it does affect finite sample performance.
- If step size is too small we do not find the typical set (area of high probability mass).
- If step size is too large we find the typical set, but do not explore it efficiently.
- Therefore do an initial run of sampler to find optimal step size before starting proper.

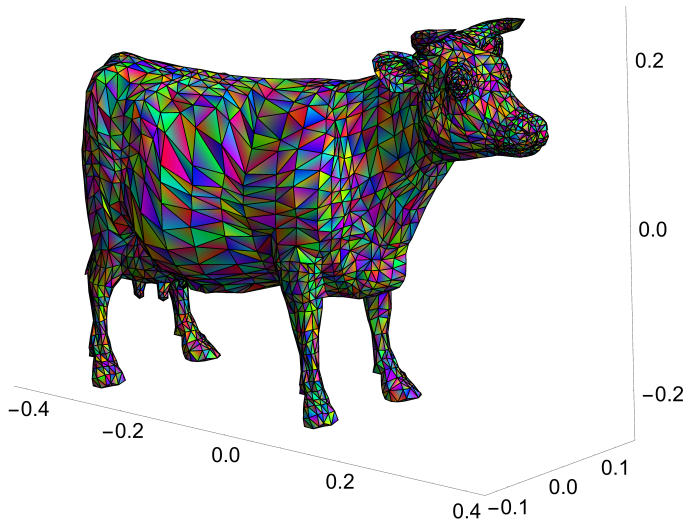
MCMC with unnormalised posterior: cow distribution

- **Question:** Can we use MCMC to generate samples from the following unnormalised distribution?

$$p(r) \propto \exp(-100r) \quad (3)$$

where r is the shortest euclidean distance from an (x,y,z) point to cow surface.

Cow distribution



Example Random Walk Metropolis: cow revisited

- 1 Understanding a distribution by sampling from it
- 2 Introducing dependent sampling
- 3 Random Walk Metropolis
- 4 Judging convergence of chains to posterior**
- 5 Adaptive covariance MCMC
- 6 Gibbs sampling
- 7 Introduction to Hamiltonian Monte Carlo

Why do we need to monitor convergence?

Recap the steps of Metropolis:

- ① Propose an initial position θ_0 using a initial proposal distribution $\pi(\theta) \neq p(\theta|X)$.
- ② For $t = 1, \dots, T$ do:
 - Propose a new location: $\theta_{t+1} \sim J(\theta_{t+1}|\theta_t)$.
 - Accept/reject move based on

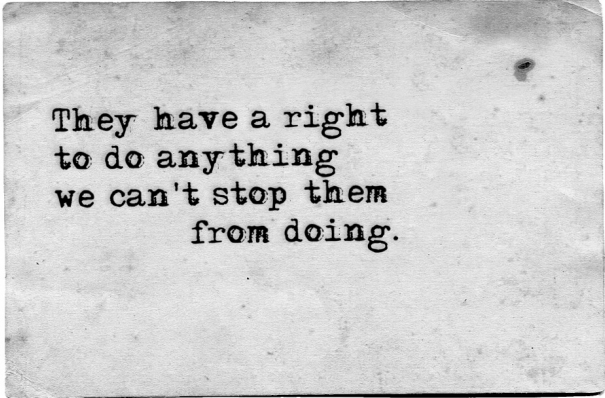
$$r = \frac{p(X|\theta_{t+1})p(\theta_{t+1})}{p(X|\theta_t)p(\theta_t)} > u \sim \text{Unif}(0, 1) \quad (4)$$

Why do we need to monitor convergence?

- Start with an initial proposal distribution $\pi(\theta) \neq p(\theta|X)$.
- Repeatedly take steps and use the Metropolis accept/reject rule $\implies \pi(\theta_t)$; the sampling distribution at time t .
- Under a set of quite general assumptions we are guaranteed that asymptotically: $\pi(\theta_t) \rightarrow p(\theta|X)$.
- However, when practically can we assume:
 $\pi(\theta_t) \approx p(\theta|X)$?

How to measure convergence?

- To monitor convergence to the posterior \implies need the posterior.
- But we don't have the posterior \Leftarrow the reason we are doing the sampling in the first place!



They have a right
to do anything
we can't stop them
from doing.

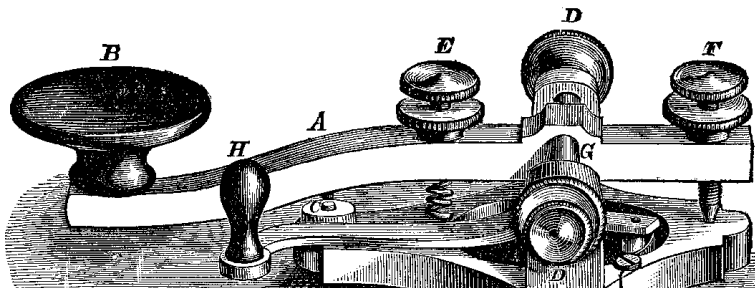
Two strategies for monitoring convergence

Strategy 1: measure distributional separation.

- For example Kullback-Leibler:

$$KL = \int p(\theta|X) \log \left(\frac{p(\theta|X)}{\pi(\theta_t)} \right) d\theta \quad (5)$$

- Motivated by information theory.
- Can use un-normalised posterior to do this.
- Again integral is too difficult to do.



Two strategies for monitoring convergence

Strategy 2: monitor the approach to a stationary distribution.

- We know asymptotically this will happen.
- By design of Metropolis stepping and accept/reject rules, we know the stationary distribution is the posterior.



Monitoring convergence of a single chain

Initial idea:

- Compare summaries (mean, variance, etc.) of sampling distribution for a chain at time t with itself at time $t + T$.
- If their rate of change is below a threshold \implies convergence.

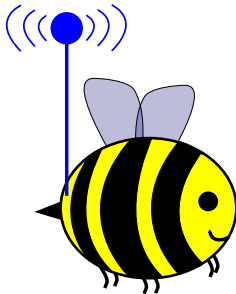
Monitoring convergence of a single chain

Question: What is the problem with this idea?

Convergence monitoring: Bob's bees

Thought experiment:

- Imagine a house of unknown shape.
- We have an unlimited supply of bees, each equipped with a GPS tracker allowing us to accurately monitor their position.
- **Question:** How can we use these to estimate the shape of the house?



Convergence monitoring: Bob's bees

Answer:

- Release one (at a random location in the house) and monitor its path over time.
- Stop/collect bee after summary measures of its path stop changing.

Convergence monitoring: single bee

Convergence monitoring: single bee, a bit later

Convergence monitoring: single bee, a bit bit later

Convergence monitoring: single bee

Question: what's the actual shape of the house?

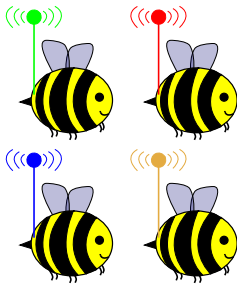
Convergence monitoring: single bee

Single chain problems: summary

- One way to monitor convergence is to look for convergence in a single chain's summary statistics.
- This method is very susceptible to the curse of hindsight problem ("Now we've definitely converged on the posterior. We hadn't a minute ago.")
- Particularly because chains often get stuck in subregions of θ space.

The solutions: lots of bees

- Release lots of bees starting at dispersed locations in parameter space.
- Stop recording when an individual bee's path is indistinguishable from all others'.



Convergence monitoring: multiple bees

Convergence monitoring: multiple bees (a lot later)

Multiple chain convergence monitoring: summary

- Start a number of chains in random dispersed locations in θ space.
- Chains do *not* interact with one another (in Metropolis).
- Run each sampler until it is hard to distinguish one chain's path from all others'.
- Less susceptible to “curse of hindsight”, since we can see if chains aren't mixing.
- Not foolproof! There still may be an area of high probability mass that we miss. However, less likely to fail compared to a single chain.
- The more chains, the better!

Judging convergence

Single bee in a house.

Judging convergence

Multiple bees in a house released in a single room.

Judging convergence

Question: have we converged?

Judging convergence

Multiple bees in new house released in highly dispersed rooms.

Judging convergence

Multiple bees in new house released in highly dispersed rooms...much later.

Multiple chain convergence monitoring: open questions

- ① How to determine “random dispersed locations” at which to start the chains?
 - Ideally use an initial proposal distribution similar to posterior shape.
 - Otherwise a good rule of thumb is “Any point you don’t mind having in a sample is a good starting point”, Charles Geyer.
- ② Which summary statistics to monitor to determine convergence?
- ③ At what threshold are “between chain” statistics sufficiently similar?

Gelman and Rubin's \hat{R}

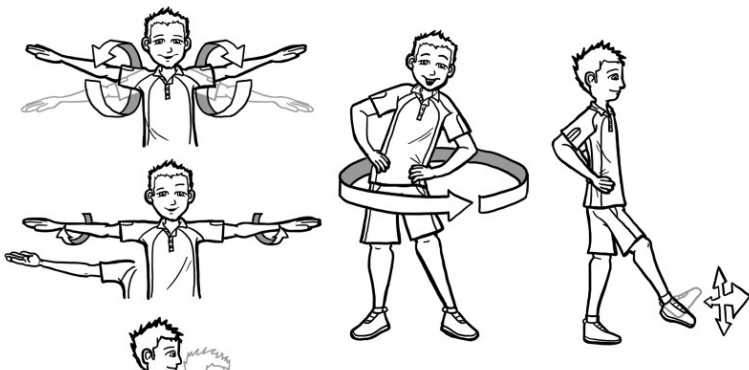
- Gelman and Rubin (1992) had the idea of comparing within-chain to between-chain variability.
- They quantified this comparison using:

$$\hat{R} = \sqrt{\frac{W + \frac{1}{n}(B - W)}{W}} \quad (6)$$

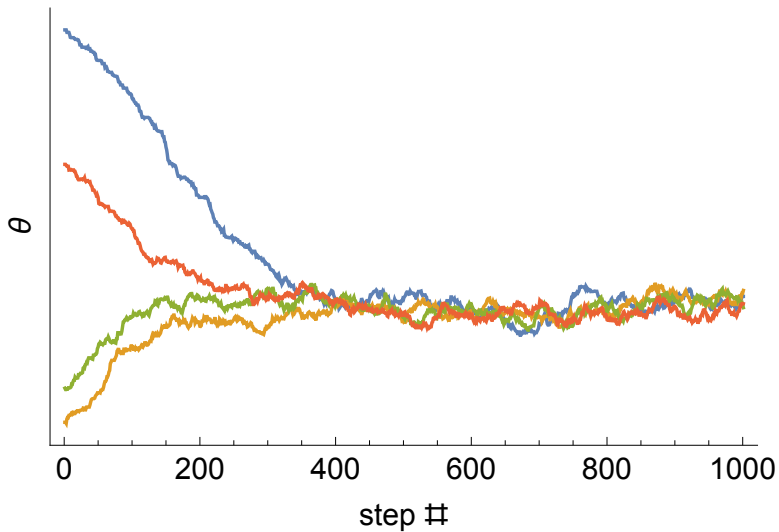
- Where “within-chain” variability, $W = \frac{1}{m} \sum_{j=1}^m s_j^2$, for m chains.
- And “between-chain” variability, $B = \frac{n}{m-1} \sum_{j=1}^m (\bar{\theta}_j - \bar{\theta})^2$.
- When we start $B \gg W$ since we start in an overdispersed position.
- In convergence $B \rightarrow W \implies \hat{R} \rightarrow 1$ (in practice $\hat{R} < 1.1$ usually suffices).

Warm up period

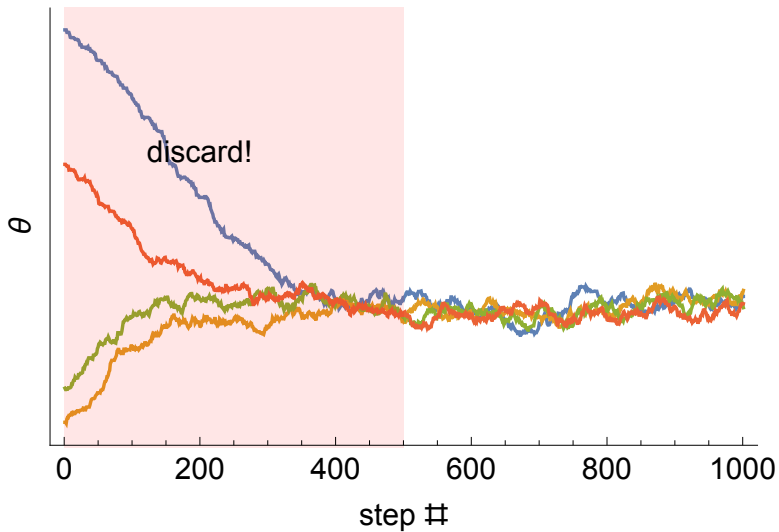
- The initial proposal distribution is *not* the posterior.
- We therefore discard the beginning part of the chain called the “warm up” to lessen the effect of the starting position.
- Typically discard first half of converged chains (can also cut chains in two to monitor intra-chain convergence).



Warm up period



Warm up period



- 1 Understanding a distribution by sampling from it
- 2 Introducing dependent sampling
- 3 Random Walk Metropolis
- 4 Judging convergence of chains to posterior
- 5 Adaptive covariance MCMC**
- 6 Gibbs sampling
- 7 Introduction to Hamiltonian Monte Carlo

Inefficient exploration of the typical set by Random Walk Metropolis

Adaptive covariance MCMC: adjusting the proposal kernel to posterior geometry

- The problem with RWM is that the proposals - being in random directions - are unlikely by chance to align with areas of high density.
- Adaptive covariance MCMC adjusts the proposal kernel dynamically to obtain a higher acceptance probability.

Sketch of adaptive covariance algorithm(s)

Begin by generating n samples using Random Walk Metropolis. Start with $\Sigma_0 = \text{identity matrix}$. Then,

- 1 Estimate sample mean: $\mu_t = \frac{1}{n} \sum_{i=1}^t \theta_i$.
- 2 Estimate sample covariance matrix:
$$\Omega_t = (\theta^{\{t\}} - \mu_t)(\theta^{\{t\}} - \mu_t)'$$
- 3 Proposal kernel: $\Sigma_t = (1 - a^t)\Sigma_{t-1} + a^t\Omega_t$.

$\lim_{t \rightarrow \infty} a^t = 0$ is key to ensuring convergence to posterior distribution.

Adaptive covariance MCMC: adjusting the proposal kernel to posterior geometry

Adaptive covariance MCMC: summary

- RWM is inefficient due to random directionality of proposals.
- Adaptive covariance MCMC dynamically changes proposal kernel to match (global) posterior geometry leading to significant speed ups.
- ACMCMC can be used whenever RWM can be \implies very general algorithm; particularly useful for ODE and PDE models, where gradients of solution (necessary for HMC) are expensive.
- ACMCMC and loads of other algorithms are available in PINTS: <https://github.com/pints-team/pints>.
- Problem: adapting to global geometry often leads to very poor local exploration.

- 1 Understanding a distribution by sampling from it
- 2 Introducing dependent sampling
- 3 Random Walk Metropolis
- 4 Judging convergence of chains to posterior
- 5 Adaptive covariance MCMC
- 6 Gibbs sampling**
- 7 Introduction to Hamiltonian Monte Carlo

Defining the Gibbs sampler

For a parameter vector: $\theta = (\theta_1, \theta_2, \theta_3)$:

- Select a random starting location: $(\theta_1^0, \theta_2^0, \theta_3^0)$, along the same lines as for Random Walk Metropolis.
- For each iteration $t = 1, \dots, T$ do:
 - ① Select a random parameter update ordering, for example $(\theta_3, \theta_2, \theta_1)$.
 - ② Independently sample from the conditional posterior for each parameter in order using the most up-to-date parameters.

Defining the Gibbs sampler

First we sample:

$$\theta_3^1 \sim p(\theta_3 | \theta_2^0, \theta_1^0) \quad (7)$$

Then conditional on freshly-sampled θ_3^1 :

$$\theta_2^1 \sim p(\theta_2 | \theta_1^0, \theta_3^1) \quad (8)$$

Then conditional on freshly-sampled θ_3^1 and θ_2^1 :

$$\theta_1^1 \sim p(\theta_1 | \theta_2^1, \theta_3^1) \quad (9)$$

Defining the Gibbs sampler

Important: in Gibbs sampling there is no rejection of steps
⇒ unlike Random Walk Metropolis!

One of the reasons Gibbs can be more efficient.

Example application of Gibbs sampling: speed of motion of neighbouring birds in a flock

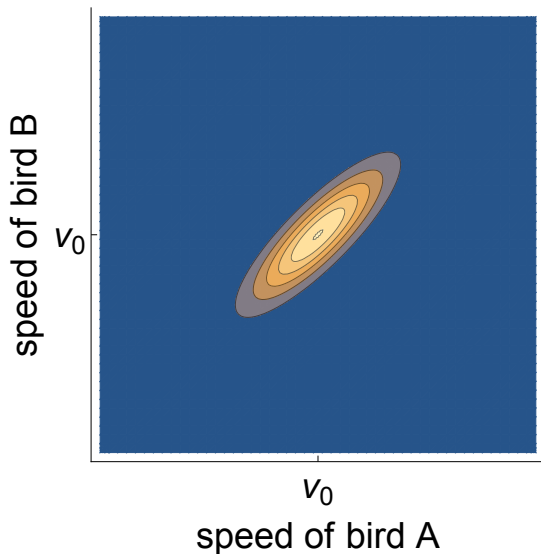
Suppose we record the speed of bird A (v_A) and bird B (v_B) in a flock along a particular axis.

Based on observations we find that the joint posterior distribution over speeds is a multivariate normal distribution:

$$\begin{pmatrix} v_A \\ v_B \end{pmatrix} \sim N \left[\begin{pmatrix} v_0 \\ v_0 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right]$$

Of course here we have an analytic expression for the posterior distribution, but this example illustrates how the method works for more general problems.

Example application of Gibbs sampling: speed of motion of neighbouring birds in a flock



Finding the conditional distributions

In most circumstances we cannot find the conditional distributions however here it is possible.

If we knew v_B :

$$v_A \sim N(v_0 + \rho(v_B - v_0), 1 - \rho^2) \quad (10)$$

Alternatively, if we knew v_A :

$$v_B \sim N(v_0 + \rho(v_A - v_0), 1 - \rho^2) \quad (11)$$

Use Gibbs sampling to conditionally sample: $v_A|v_B$ then $v_B|v_A$.

Remember: in Gibbs sampling we accept **all** steps unlike Random Walk Metropolis.

Gibbs sampling the posterior distribution over birds' speeds

Comparing Random Walk Metropolis with Gibbs

Highly correlated parameters: problems with Random Walk Metropolis and Gibbs

- Gibbs performs well on this simple problem.
- However if we increase the posterior correlation between parameters, how does each sampler fare?

Highly correlated parameters: both poor at finding the typical set

Highly correlated parameters: also both poor at exploring the typical set

Other problems with Gibbs

- Requires that the conditional distributions can be derived and sampled from.
- Relies on us “knowing” a reasonable amount of the maths behind each problem.
- Maths is hard and we would like to avoid it if possible!

Often we can only sample from the conditional distributions for a few parameters \implies use Random Walk Metropolis for others (essentially the method used by BUGS and JAGS.)

Gibbs sampling: summary

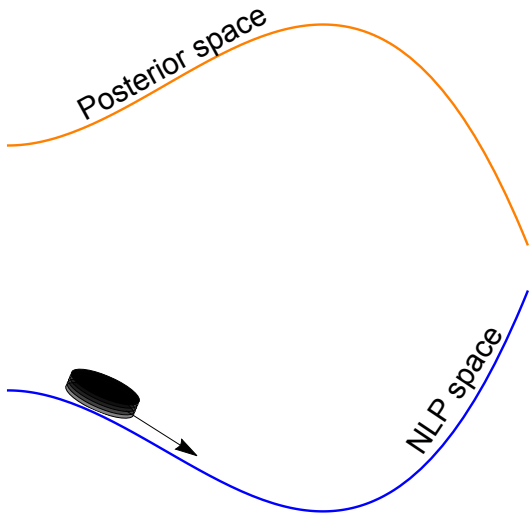
- Gibbs sampling works by cycling through each parameter dimension, and sampling from the distribution conditional on all other parameters.
- (If “joint-conditional” distributions of the form $p(\theta_1, \theta_2 | \theta_3)$ can be sampled, then this is a more efficient form of Gibbs.)
- Depends on us knowing the conditional distribution for each parameter \implies in majority of circumstances not possible.
- Can be more efficient than Random Walk Metropolis; especially useful for discrete parameter models.
- Not a panacea. (What is a panacea?)

- 1 Understanding a distribution by sampling from it
- 2 Introducing dependent sampling
- 3 Random Walk Metropolis
- 4 Judging convergence of chains to posterior
- 5 Adaptive covariance MCMC
- 6 Gibbs sampling
- 7 Introduction to Hamiltonian Monte Carlo**

Introduction to Hamiltonian Monte Carlo

- Assume a space related to posterior space can be thought of as a landscape.
- Imagine an ice puck moving over the frictionless surface of this terrain.
- At defined time points we measure the location of the puck, and instantaneously give the puck a shove in a random direction.
- The locations traced out by the puck represent proposed steps from our sampler.
- Based on the height of the posterior and momentum of the puck we accept/reject steps.

Why does this physical analogy help us?



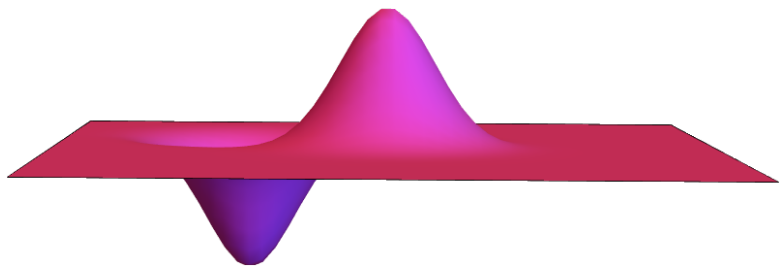
Why does this physical analogy help us?

- Allow the potential energy of the puck to be determined partly by the posterior density.
- \implies puck will move in the “natural” directions dictated by the posterior geometry.
- And will visit areas of low NLP \implies high posterior density.
- **NLP** stands for the **negative log (un-normalised) posterior**,

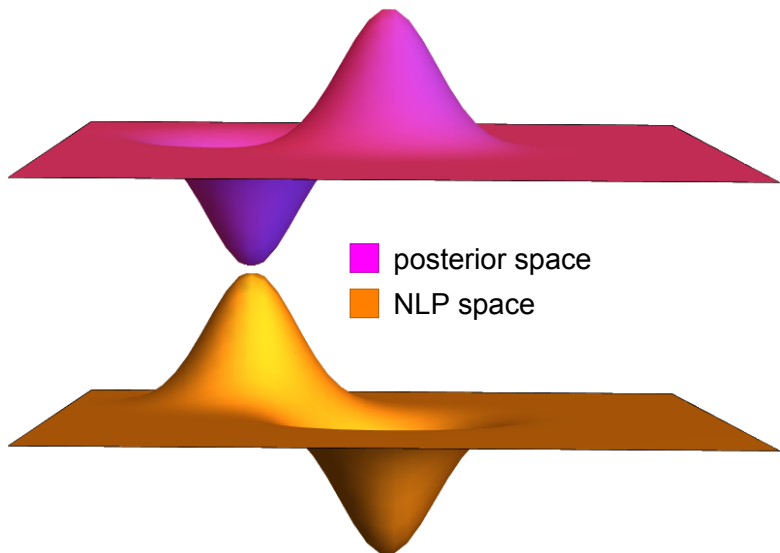
$$NLP = -\log [p(X|\theta) \times p(\theta)] \quad (12)$$

Important to remember that HMC uses the **log** of the posterior. (When coding up model in Stan “sampling” statements amount to incrementing the log probability.)

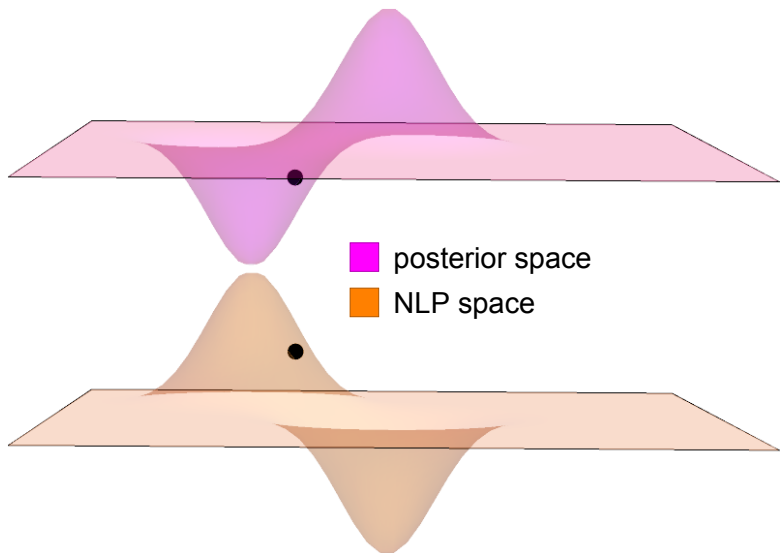
Simulating the puck's motion in NLP space: start with a posterior



Simulating the puck's motion in NLP space: find NLP space



Simulating the puck's motion in NLP space:
consider a point in posterior space



The path traced out for 100 different shoves from same distribution

The HMC algorithm

The HMC algorithm works as follows,

- Start at random location θ_0 .
- For $i = 1, \dots, N$ do:
 - ① Give puck random initial momentum, $k \sim N(0, \Sigma)$.
 - ② Simulate puck's movement across NLP surface for a fixed time T (number of discrete steps of numerical integration algo).
 - ③ Compute a ratio:

$$r = \frac{p(\theta_t|X)}{p(\theta_{t-1}|X)} \times \frac{p(k')}{p(k)} \quad (13)$$

where k is the final momentum.

- ④ If $r > u \implies$ move to θ_t , otherwise return to θ_{t-1} .

The HMC algorithm

The HMC algorithm

The HMC algorithm

Simulating the puck's motion in NLP space:
randomly shoving the puck at intervals of 50 steps

RWM and Gibbs performance

Hamiltonian Monte Carlo performance

Hamiltonian Monte Carlo: summary

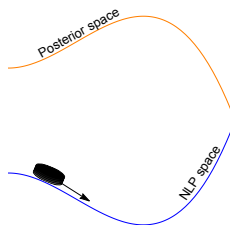
- Imagine a puck sliding across a frictionless surface of the negative log posterior (NLP).
- Give puck random shoves at predefined time intervals, and simulate path of puck for predefined time interval.
- Puck will tend to visit areas of low NLP which correspond to areas of high posterior density!
- Algorithm often performs considerably better than Random Walk Metropolis or Gibbs.
- Stan uses a fancy variant of HMC known as NUTS (No U-Turn Sampler) that determines the optimal time intervals to simulate puck for.

Summary

- ① Sampling can be used to gain insight into a distribution.
- ② Independent sampling from posterior not generally possible
 \implies shift to dependent sampling.
- ③ Random Walk Metropolis is a MCMC algorithm that allows *dependent* sampling from the posterior.
- ④ The efficiency of Metropolis depends on choosing the right step size.
- ⑤ Monitoring of sampler's convergence to the posterior is non-trivial.
- ⑥ The use of multiple chains makes it harder to make a mistake although not impossible.
- ⑦ Adaptive covariance MCMC, Gibbs sampling and HMC can speed up sampling dramatically.

Not sure I understand?

Hamiltonian Monte Carlo.



Hamilton in Monte Carlo.

