Monte Carlo methods

Nordic Probabilistic AI School

Instructor:

Charles Margossian (Flatiron Institute, New York, USA)

Teaching Assistants:

Austin Garrett (?)

Bob Pepin (University of Copenhagen, Denmark)

Outline:

- Aim and scope of Monte Carlo methods
- Markov chain Monte Carlo
- Application: Bayesian linear regression
- Application: Disease transmission model
- Importance sampling and model comparison
- Discussion

Remark: This is a course on effectively applying Monte Carlo methods, rather than developing them from scratch.

However, principled use of MC methods requires a rigorous understanding of how they are implemented. So we need to talk about theory and low-level implementation—all whilst remaining pragmatic.

Code for exercises:

- We'll use `R` scripts and the probabilistic programming language `Stan` .
- This is <u>not</u> a course on coding in `R`: the methods apply generally and `Stan` can be interfaced with other languages, including `Python` and `Julia`.
- Code for the exercises at `https://github.com/charlesm93/stanTutorial/tree/main/Nordic_Prob_AI`

I
Aim and scope of Monte Carlo methods

Goal: compute an intractable expectation value with respect to $\pi(\theta)$.

Goal: compute an intractable expectation value with respect to $\pi(\theta)$.

- Statistical physics
$$\pi(\theta) \propto \exp\left(-E(\theta)\right)$$
- Bayesian inference
$$\pi(\theta) = p(\theta \mid y) \propto p(\theta)p(y \mid \theta)$$
- Variational inference
$$\pi(\theta) = q(\theta)$$

Goal: compute an intractable expectation value with respect to $\pi(\theta)$.

- Statistical physics
$$\pi(\theta) \propto \exp\left(-E(\theta)\right)$$
- Bayesian inference
$$\pi(\theta) = p(\theta \mid y) \propto p(\theta)p(y \mid \theta)$$
- Variational inference
$$\pi(\theta) = q(\theta)$$

What might we want to learn about these distributions?

Goal: compute an intractable expectation value with respect to $\pi(\theta)$.

- Statistical physics
$$\pi(\theta) \propto \exp\left(-E(\theta)\right)$$
- Bayesian inference
$$\pi(\theta) = p(\theta \mid y) \propto p(\theta) p(y \mid \theta)$$
- Variational inference
$$\pi(\theta) = q(\theta)$$

What might we want to learn about these distributions?

- Expectation, variance, and quantiles of $f(\theta)$ with respect to π.

Goal: compute an intractable expectation value with respect to $\pi(\theta)$.

- Statistical physics
$$\pi(\theta) \propto \exp\left(-E(\theta)\right)$$
- Bayesian inference
$$\pi(\theta) = p(\theta \mid y) \propto p(\theta)p(y \mid \theta)$$
- Variational inference
$$\pi(\theta) = q(\theta)$$

What might we want to learn about these distributions?

- Expectation, variance, and quantiles of $f(\theta)$ with respect to π.
- **Monte Carlo:** draw samples and construct sample estimators,
$$\theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(N)} \sim \pi(\theta).$$
- When no exact simulation is possible, use Markov chain Monte Carlo, or importance sampling.

II

Markov chain Monte Carlo

Quantities of interest can often be expressed as integrals with respect to a probability measure

$$\mathbb{E}[f(\theta)] = \int f(\theta)\, p(\theta \mid y)\, \mathrm{d}\theta$$

Quantities of interest can often be expressed as integrals with respect to a probability measure

$$\mathbb{E}[f(\theta)] = \int f(\theta)\ p(\theta \mid y)\ \mathrm{d}\theta$$

Monte Carlo estimator:

$$\theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(N)} \overset{\text{iid}}{\sim} p(\theta \mid y)$$

$$\widehat{\mathbb{E}}[f(\theta)] = \frac{1}{N} \sum_{n=1}^{N} f\left(\theta^{(n)}\right)$$

Quantities of interest can often be expressed as integrals with respect to a probability measure

$$\mathbb{E}[f(\theta)] = \int f(\theta)\, p(\theta \mid y)\, \mathrm{d}\theta$$

Monte Carlo estimator:

$$\theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(N)} \overset{\text{iid}}{\sim} p(\theta \mid y)$$

$$\widehat{\mathbb{E}}[f(\theta)] = \frac{1}{N} \sum_{n=1}^{N} f\left(\theta^{(n)}\right)$$

Can get a sample estimator for mean, variance and quantiles.

How good is our Monte Carlo estimator $\widehat{\mathbb{E}}[f(\theta)]$?

How good is our Monte Carlo estimator $\widehat{\mathbb{E}}[f(\theta)]$?

Ultimately want to control the expected squared error,

$$\mathbb{E}\left[\left(\widehat{\mathbb{E}}[f(\theta)] - \mathbb{E}[f(\theta)]\right)^2\right] = \text{Bias}^2 + \text{Var}\left[\widehat{\mathbb{E}}[f(\theta)]\right]$$

How good is our Monte Carlo estimator $\widehat{\mathbb{E}}[f(\theta)]$?

Ultimately want to control the expected squared error,

$$\mathbb{E}\left[\left(\widehat{\mathbb{E}}[f(\theta)] - \mathbb{E}[f(\theta)]\right)^2\right] = \text{Bias}^2 + \text{Var}\left[\widehat{\mathbb{E}}[f(\theta)]\right]$$

If $\theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(N)}$ are i.i.d,

$$\text{Bias} = 0, \qquad \text{Var}\left[\widehat{\mathbb{E}}[f(\theta)]\right] = \frac{1}{N}\text{Var}[\theta]$$

How good is our Monte Carlo estimator $\widehat{\mathbb{E}}[f(\theta)]$?

Ultimately want to control the expected squared error,

$$\mathbb{E}\left[\left(\widehat{\mathbb{E}}[f(\theta)] - \mathbb{E}[f(\theta)]\right)^2\right] = \text{Bias}^2 + \text{Var}\left[\widehat{\mathbb{E}}[f(\theta)]\right]$$

If $\theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(N)}$ are i.i.d,

$$\text{Bias} = 0, \qquad \text{Var}\left[\widehat{\mathbb{E}}[f(\theta)]\right] = \frac{1}{N}\text{Var}[\theta]$$

We also have a *central limit theorem*, i.e. for large N

$$\widehat{\mathbb{E}}[f(\theta)] \overset{\text{approx}}{\sim} \text{normal}\left(\mathbb{E}f(\theta), \sqrt{\frac{\text{Var}[f(\theta)]}{N}}\right).$$

In practice, we cannot generate iid samples from $p(\theta \mid y)$.

In practice, we cannot generate iid samples from $p(\theta \mid y)$.

Markov chain Monte Carlo:

- Start with an initial draw $\theta^{(0)} \sim p_0(\theta)$.
- Apply a transition kernel, $\theta^{(i+1)} \sim \Gamma(\theta^{(i+1)} \mid \theta^{(i)})$.

In practice, we cannot generate iid samples from $p(\theta \mid y)$.

Markov chain Monte Carlo:

- Start with an initial draw $\theta^{(0)} \sim p_0(\theta)$.
- Apply a transition kernel, $\theta^{(i+1)} \sim \Gamma(\theta^{(i+1)} \mid \theta^{(i)})$.

Under certain conditions,

$$\theta^{(n)} \xrightarrow[n \to \infty]{d} p(\theta \mid y),$$

and $p(\theta \mid y)$ is the stationary distribution.

In practice, we cannot generate iid samples from $p(\theta \mid y)$.

Markov chain Monte Carlo:

- Start with an initial draw $\theta^{(0)} \sim p_0(\theta)$.
- Apply a transition kernel, $\theta^{(i+1)} \sim \Gamma(\theta^{(i+1)} \mid \theta^{(i)})$.

Under certain conditions,

$$\theta^{(n)} \xrightarrow[n\to\infty]{d} p(\theta \mid y),$$

and $p(\theta \mid y)$ is the stationary distribution.

In practice, for “large enough” n,

$$\theta^{(n)} \overset{\text{approx.}}{\sim} p(\theta \mid y).$$

In practice, we cannot generate iid samples from $p(\theta \mid y)$.

Markov chain Monte Carlo:

- Start with an initial draw $\theta^{(0)} \sim p_0(\theta)$.
- Apply a transition kernel, $\theta^{(i+1)} \sim \Gamma(\theta^{(i+1)} \mid \theta^{(i)})$.

Under certain conditions,

$$\theta^{(n)} \xrightarrow[n \to \infty]{d} p(\theta \mid y),$$

and $p(\theta \mid y)$ is the stationary distribution.

In practice, for "large enough" n,

$$\theta^{(n)} \overset{\text{approx.}}{\sim} p(\theta \mid y).$$

- The first samples suffer from a large bias.

In practice, we cannot generate iid samples from $p(\theta \mid y)$.

Markov chain Monte Carlo:

- Start with an initial draw $\theta^{(0)} \sim p_0(\theta)$.
- Apply a transition kernel, $\theta^{(i+1)} \sim \Gamma(\theta^{(i+1)} \mid \theta^{(i)})$.

Under certain conditions,

$$\theta^{(n)} \xrightarrow[n \to \infty]{d} p(\theta \mid y),$$

and $p(\theta \mid y)$ is the stationary distribution.

In practice, for "large enough" n,

$$\theta^{(n)} \overset{\text{approx.}}{\sim} p(\theta \mid y).$$

- The first samples suffer from a large bias.
- Discard these samples during a burn-in or *warmup* phase.

Example: Metropolis algorithm [Metropolis et al., 1953]

Example: Metropolis algorithm [Metropolis et al., 1953]

1. Start at an initial point, $\theta^{(0)} \sim p_0$.

Example: Metropolis algorithm [Metropolis et al., 1953]

1. Start at an initial point, $\theta^{(0)} \sim p_0$.
2. Apply the transition kernel N times:
    1. Propose a new sample
    $$\theta^{(i+1)} \sim \text{normal}\left(\theta^{(i)}, \sigma^2 I\right)$$
    2. Accept the proposal with probability
    $$\Pr = \min\left(\frac{p(\theta^{(i+1)} \mid z)}{p(\theta^{(i)} \mid z)}, 1\right).$$

Example: Metropolis algorithm [Metropolis et al., 1953]

1. Start at an initial point, $\theta^{(0)} \sim p_0$.
2. Apply the transition kernel N times:
    1. Propose a new sample
    $$\theta^{(i+1)} \sim \text{normal}\left(\theta^{(i)}, \sigma^2 I\right)$$
    2. Accept the proposal with probability
    $$\Pr = \min\left(\frac{p(\theta^{(i+1)} \mid z)}{p(\theta^{(i)} \mid z)}, 1\right).$$
3. Return the chain $(\theta^{(1)}, \theta^{(2)}, ..., \theta^{(N)})$.

Example: Metropolis algorithm [Metropolis et al., 1953]

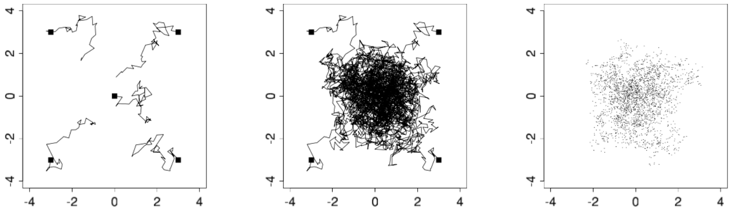


Figure from [Gelman et al., 2013].

Example: Metropolis algorithm [Metropolis et al., 1953]

Benefits:

- Only requires evaluating $p(\theta, y) = p(\theta)p(y \mid \theta)$.
- Asymptotically, the algorithm samples from $p(\theta \mid y)$.

Drawbacks:

- In the finite regime, the samples are **biased**.
- The samples are <u>not</u> independent; there are correlated, which **increases the variance** of our Monte Carlo estimators.

Example: Continuous diffusion process

In the limit where we take infinitesimally small steps, many MCMC algorithms can be approximated by a random diffusion process [Gelman et al., 1997, Roberts and Rosenthal, 1998].

- Initial distribution: $p_0 = \text{normal}(\mu_0, \sigma_0^2)$.
- Target distribution: $p = \text{normal}(\mu, \sigma^2)$.

Example: Continuous diffusion process

In the limit where we take infinitesimally small steps, many MCMC algorithms can be approximated by a random diffusion process [Gelman et al., 1997, Roberts and Rosenthal, 1998].

- Initial distribution: $p_0 = \text{normal}(\mu_0, \sigma_0^2)$.
- Target distribution: $p = \text{normal}(\mu, \sigma^2)$.

Then after time T,

$$\theta^{(T)} \sim \text{normal}\left[(\mu_0 - \mu)e^{-T} + \mu, \quad \left(\sigma_0^2 - \sigma^2\right) e^{-2T} + \sigma^2\right].$$

Example: Continuous diffusion process

In the limit where we take infinitesimally small steps, many MCMC algorithms can be approximated by a random diffusion process
[Gelman et al., 1997, Roberts and Rosenthal, 1998].

- Initial distribution: $p_0 = \text{normal}(\mu_0, \sigma_0^2)$.
- Target distribution: $p = \text{normal}(\mu, \sigma^2)$.

Then after time T,

$$\theta^{(T)} \sim \text{normal}\left[(\mu_0 - \mu)e^{-T} + \mu, \quad \left(\sigma_0^2 - \sigma^2\right) e^{-2T} + \sigma^2\right].$$

For T large enough, the bias becomes negligible.

Variance of Monte Carlo estimator

Suppose the chain is *stationary*; i.e. we started at $p_0 = p(\theta \mid y)$ or we already ran the chain for an infinitely long time.

Variance of Monte Carlo estimator

Suppose the chain is *stationary*; i.e. we started at $p_0 = p(\theta \mid y)$ or we already ran the chain for an infinitely long time.

- Under certain conditions, Monte Carlo estimators observe a central limit theorem, meaning that for large N,

$$\frac{1}{N}\sum_i f(\theta^{(n)}) \overset{\text{approx}}{\sim} \text{Normal}\left(\mathbb{E}[f(\theta)], \frac{\text{Var} f(\theta)}{N_{\text{eff}}}\right)$$

  where N_{eff} is the **effective sample size (ESS)**.

Variance of Monte Carlo estimator

Suppose the chain is *stationary*; i.e. we started at $p_0 = p(\theta \mid y)$ or we already ran the chain for an infinitely long time.

- Under certain conditions, Monte Carlo estimators observe a central limit theorem, meaning that for large N,

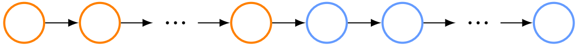
$$\frac{1}{N} \sum_i f(\theta^{(n)}) \overset{\text{approx}}{\sim} \text{Normal}\left(\mathbb{E}[f(\theta)], \frac{\text{Var} f(\theta)}{N_{\text{eff}}} \right)$$

  where N_{eff} is the **effective sample size (ESS)**.

- $$N_{\text{eff}} = \frac{N}{1 + 2\sum_{t=1}^{\infty} \rho_t}.$$

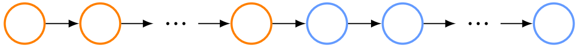
  ρ_t is the chain's autocorrelation between $\theta^{(i)}$ and $\theta^{(i+t)}$.

Handling the error of MCMC



In practice, MCMC proceeds in two phases:

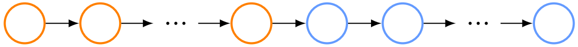
Handling the error of MCMC



In practice, MCMC proceeds in two phases:

Warmup phase: We run the process for several steps for the bias to become negligible but don't use any of those samples in our Monte Carlo estimator.

Handling the error of MCMC



In practice, MCMC proceeds in two phases:

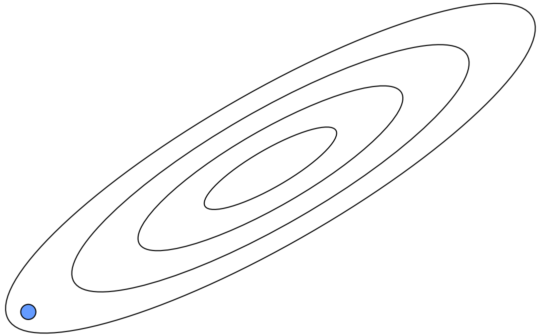
Warmup phase: We run the process for several steps for the bias to become negligible but don't use any of those samples in our Monte Carlo estimator.

Sampling phase: Collect enough samples to have a large ESS and reduce the variance of the Monte Carlo estimator.

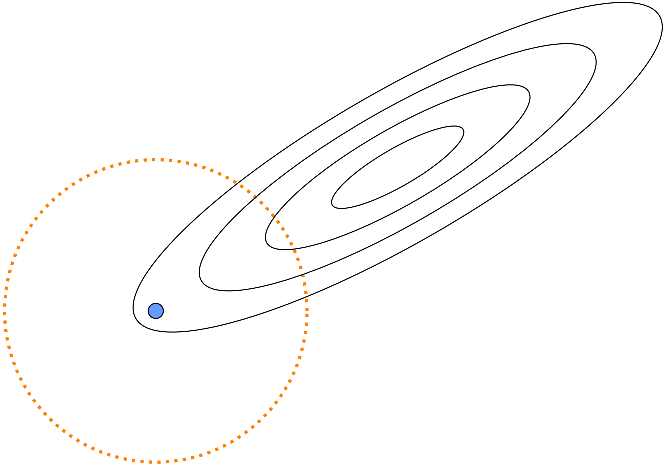
Question: Which transition kernel should we choose? Many choices!

Metropolis, Gibbs, Metropolis-adjusted Langevin approximation, Hamiltonian Monte Carlo,...

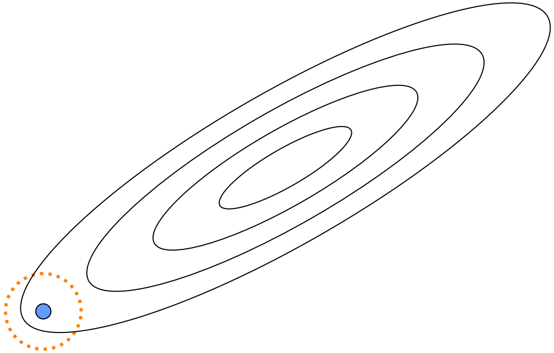
Example: ill-conditioned Gaussian



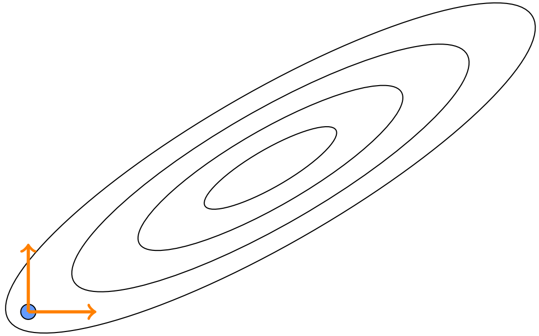
Kernel: Metropolis-Hastings proposal (low acceptance probability)



Kernel: Metropolis-Hastings proposal (small step)



Kernel: Gibbs sampler (moves one coordinate at a time)



Hamiltonian Monte Carlo

- Treat the Markov chain as a physical particle, which evolves over $\mathbb{R}^D$, subject to a *potential*:
$$U(\theta) = -\log p(\theta \mid y).$$

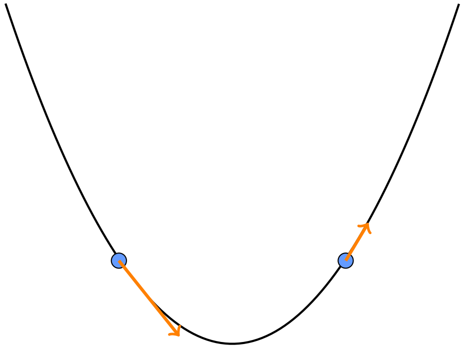
Hamiltonian Monte Carlo

- Treat the Markov chain as a physical particle, which evolves over $\mathbb{R}^D$, subject to a *potential*:
$$U(\theta) = -\log p(\theta \mid y).$$
- Give the particle a random shove, by giving it a *momentum* $\xi_0 \in \mathbb{R}^D$,
$$\xi_0 \sim \text{normal}(0, M)$$

Hamiltonian Monte Carlo

- Treat the Markov chain as a physical particle, which evolves over $\mathbb{R}^D$, subject to a *potential*:
$$U(\theta) = -\log p(\theta \mid y).$$
- Give the particle a random shove, by giving it a *momentum* $\xi_0 \in \mathbb{R}^D$,
$$\xi_0 \sim \text{normal}(0, M)$$
- Simulate a the laws of classical mechanics for a time T,
$$(\theta_0, \xi_0) \to (\theta_T, \xi_T).$$

$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = M^{-1}\xi, \quad \frac{\mathrm{d}\xi}{\mathrm{d}t} = \nabla_\theta \log p(\theta \mid y).$$

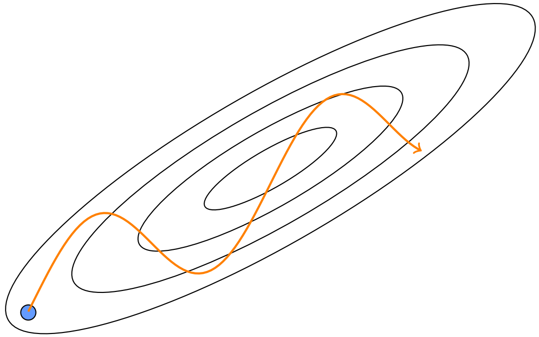


A particle <u>accelerates</u> when $U(\theta) = -\log p(\theta \mid y)$ decreases.

It <u>decelerates</u> when $U(\theta) = -\log p(\theta \mid y)$ increases.

This is based on $\nabla_\theta \log p(\theta \mid y)$.

Kernel: Hamiltonian Monte Carlo



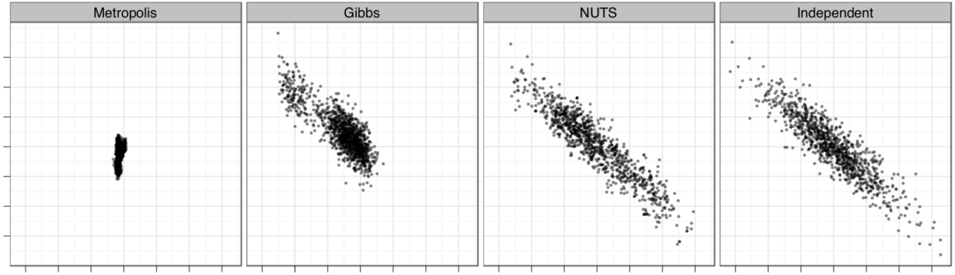
Some challenges in implementing HMC:

- Need to compute $\nabla_\theta \log p(\theta \mid y) \rightarrow$ automatic differentiation
- Tuning parameters:
  - The length of the Hamiltonian trajectory
  - The precision with which we solve Hamilton's equations of motion (step size)
  - The mass matrix of the fictitious particle

Some challenges in implementing HMC:

- Need to compute $\nabla_\theta \log p(\theta \mid y) \rightarrow$ automatic differentiation
- Tuning parameters:
    - The length of the Hamiltonian trajectory
    - The precision with which we solve Hamilton's equations of motion (step size)
    - The mass matrix of the fictitious particle

- `Stan` implements autodiff and the No-U-Turn Sampler (NUTS) [Hoffman and Gelman, 2014].
    - Adaptive setting of trajectory length to avoid U-Turns
    - Adaptive tuning of step size and mass matrix during warmup



- Figure from [Hoffman and Gelman, 2014].
- Geometric structures are common in complex models and arise naturally in high dimensions.
- `Stan` implements an improved NUTS sampler [Betancourt, 2018]. I highly recommends Betancourt's paper on HMC for those who want to dive deeper!

III
Application: Bayesian Linear Regression
(warmup example)

How `Stan` works

- The `Stan` file specifies the joint distribution

$$p(\theta, y) = p(y|\theta)p(\theta) \propto p(\theta \mid y)$$

- The input includes:
  - the data, y
  - tuning parameters for the algorithm

- The output can include:
  - an approximate sample from the posterior distribution
  - summaries of the run which can help us diagnose problems.

Inference algorithms in `Stan`

- Hamiltonian Monte Carlo (HMC)
- No-U Turn Sampler (NUTS)
- Automatic differentiation variational inference (ADVI)
- Pathfinder Variational Inference
- ...

We can manage the `Stan` file, the input, and the output using a scripting language, such as:

- `R`
- `Python`
- `Julia`
- The command line
- . . .

The `Stan` documentation is your friend:
`https://mc-stan.org/users/documentation/`!!

And so is its community: `https://discourse.mc-stan.org/`.

Example: Bayesian linear regression

The data generating process is:

$$\begin{aligned} p(\beta) &= \text{Normal}(2, 1) \\ p(\sigma) &= \text{Normal}^{+}(1, 1) \\ p(y \mid \beta, \sigma) &= \text{Normal}\left(\beta x, \sigma\right) \end{aligned}$$

Our goal is to estimate $\theta = (\beta, \sigma)$, based on the observation $z = (x, y)$ and prior knowledge we have of β and σ.

Writing the Stan file

Stan retains certain C++ features:

- Variables need to be declared.
- Each statement must end with a semi-colon.

For example:

```
real x;
```

Writing the Stan file

```
data {
 Declare the data that will be given as an input.
}

parameters {
 Declare the parameters we want to sample.
}

model {
 Compute the log joint distribution.
}
```

code demo

Convergence diagnostic

Are the chains still biased by their initializations?

Proposition: Start multiple chains at a different locations and check that they all converge to the same distribution.

Examine:

1. the trace plots
2. the density plots
3. the $\widehat{R}$ statistic

$$\widehat{R} := \frac{\text{Standard deviation across all chains}}{\text{Standard deviation within chain}}$$

- If the chains sample from the same target, expect $\widehat{R} \approx 1$.
- If the chains are disagreement, $\widehat{R} \gg 1$.

code demo

A more in-depth look at $\widehat{R}$

Let $\theta^{(nm)}$ be the n^{th} sample from the m^{th} chain.

Let $f : \Theta \to \mathbb{R}$ be some function of interest.

A more in-depth look at $\widehat{R}$

Let $\theta^{(nm)}$ be the n^{th} sample from the m^{th} chain.

Let $f : \Theta \to \mathbb{R}$ be some function of interest.

Can write $\widehat{R}$ as

$$\widehat{R} = \sqrt{\frac{N-1}{N} + \frac{\widehat{B}}{\widehat{W}}},$$

where

- $\widehat{B}$ is the sample variance of $\bar{f}\left(\theta^{(\cdot m)}\right)$.
- $\widehat{W}$ is the (average) within-chain variance.

A more in-depth look at $\widehat{R}$

Let $\theta^{(nm)}$ be the nth sample from the mth chain.

Let $f : \Theta \to \mathbb{R}$ be some function of interest.

Can write $\widehat{R}$ as

$$\widehat{R} = \sqrt{\frac{N-1}{N} + \frac{\widehat{B}}{\widehat{W}}},$$

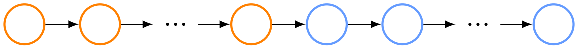
where

- $\widehat{B}$ is the sample variance of $\bar{f}\left(\theta^{(\cdot m)}\right)$.
- $\widehat{W}$ is the (average) within-chain variance.

$\widehat{R} \leq 1 + \epsilon \iff \widehat{B} \lessapprox 2\epsilon\widehat{W} + \mathcal{O}(\epsilon^2)$.

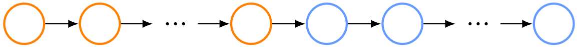
Want to make sure $\mathrm{Var}\left[f\left(\bar{\theta}^{(\cdot m)}\right)\right]$ is small.

A more in-depth look at $\widehat{R}$



Question. What can $\mathrm{Var}\left[f\left(\bar{\theta}^{(\cdot m)}\right)\right]$ teach us about convergence and bias decay?

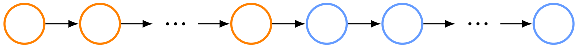
A more in-depth look at $\widehat{R}$



Question. What can $\text{Var}\left[f\left(\bar{\theta}^{(\cdot m)}\right)\right]$ teach us about convergence and bias decay?

$$\text{Var}\left(\bar{f}^{(\cdot m)}\right) = \text{Var}\left(\mathbb{E}\left(\bar{f}^{(\cdot m)} \mid f^{(0)}\right)\right) + \mathbb{E}\left(\text{Var}\left(\bar{f}^{(\cdot m)} \mid f^{(0)}\right)\right).$$

A more in-depth look at $\widehat{R}$

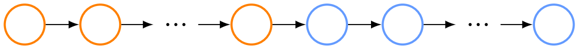


Question. What can $\text{Var}\left[f\left(\bar{\theta}^{(\cdot m)}\right)\right]$ teach us about convergence and bias decay?

$$\text{Var}\left(\bar{f}^{(\cdot m)}\right) = \text{Var}\left(\mathbb{E}\left(\bar{f}^{(\cdot m)} \mid f^{(0)}\right)\right) + \mathbb{E}\left(\text{Var}\left(\bar{f}^{(\cdot m)} \mid f^{(0)}\right)\right).$$

The nonstationary variance measures how well the chains forget their starting points.

A more in-depth look at $\widehat{R}$



Question. What can $\text{Var}\left[f\left(\bar{\theta}^{(\cdot m)}\right)\right]$ teach us about convergence and bias decay?

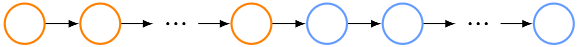
$$\text{Var}\left(\bar{f}^{(\cdot m)}\right) = \text{Var}\left(\mathbb{E}\left(\bar{f}^{(\cdot m)} \mid f^{(0)}\right)\right) + \mathbb{E}\left(\text{Var}\left(\bar{f}^{(\cdot m)} \mid f^{(0)}\right)\right).$$

The nonstationary variance measures how well the chains forget their starting points.

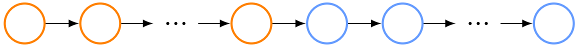
As we warmup the chains, both the nonstationary variance and squared bias decay to 0, and so $\widehat{R}$ acts as a “proxy clock” for bias.

A more in-depth look at $\widehat{R}$

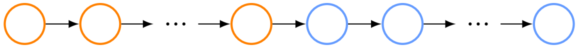
- What quantity does $\widehat{R}$ measure and how close to 1 should it be?
  - [Vehtari et al., 2021] propose checking that $\widehat{R} \leq 1.01$.
  - [Margossian et al., 2024] examine $\widehat{R}$ for nonstationary chains and propose a more direct measure of the nonstationary variance.



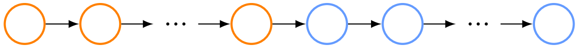
- Ideally, $\widehat{R}$ tells us if the warmup phase is long enough.
  In practice, it tells us if both the warmup and sampling phase are long enough.



- Ideally, $\widehat{R}$ tells us if the warmup phase is long enough.
  In practice, it tells us if both the warmup and sampling phase are long enough.
- ESS and Monte Carlo standard error (MCSE) tell us if the sampling phase is long enough.
  Rule of thumb: aim for ESS ≥ 100. (we'll think harder about that)



- Ideally, $\widehat{R}$ tells us if the warmup phase is long enough.
  In practice, it tells us if both the warmup and sampling phase are long enough.
- ESS and Monte Carlo standard error (MCSE) tell us if the sampling phase is long enough.
  Rule of thumb: aim for ESS ≥ 100. (we'll think harder about that)
- ESS_{tail} quantifies information for tail estimates [Vehtari et al., 2021].



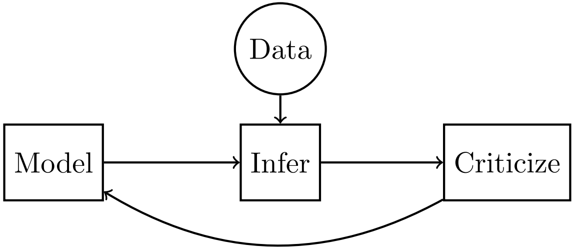
- Ideally, $\widehat{R}$ tells us if the warmup phase is long enough.
  In practice, it tells us if both the warmup and sampling phase are long enough.
- ESS and Monte Carlo standard error (MCSE) tell us if the sampling phase is long enough.
  Rule of thumb: aim for ESS ≥ 100. (we'll think harder about that)
- ESS_{tail} quantifies information for tail estimates [Vehtari et al., 2021].
- Median, $M(\theta)$ and Median Absolute Deviation (MAD),

$$M(|\theta^{(i)} - M(\theta)|)$$

  can be helpful when the first moments are not finite.

Posterior predictive checks

- Recall Box's loop (from *Bayesian Workflow*)!
- Does our model accurately describe the data?



Posterior predictive checks (“check trained model”)

Proposition:

Each time we draw a sample, $\theta^{(i)} = \left(\beta^{(i)}, \sigma^{(i)}\right)$, we will also simulate data, according to:

$$y_{\text{pred}}^{(i)} \sim \text{Normal}\left(x\beta^{(i)}, \sigma^{(i)}\right)$$

Posterior predictive checks ("check trained model")

Proposition:
Each time we draw a sample, $\theta^{(i)} = \left(\beta^{(i)}, \sigma^{(i)}\right)$, we will also simulate data, according to:

$$y_{\text{pred}}^{(i)} \sim \text{Normal}\left(x\beta^{(i)}, \sigma^{(i)}\right)$$

Want to study the posterior predictive distribution,

$$p(y_{\text{pred}} \mid y) = \int_{\Theta} p(y_{\text{pred}} \mid \theta) p(\theta \mid y) \mathrm{d}\theta.$$

code demo

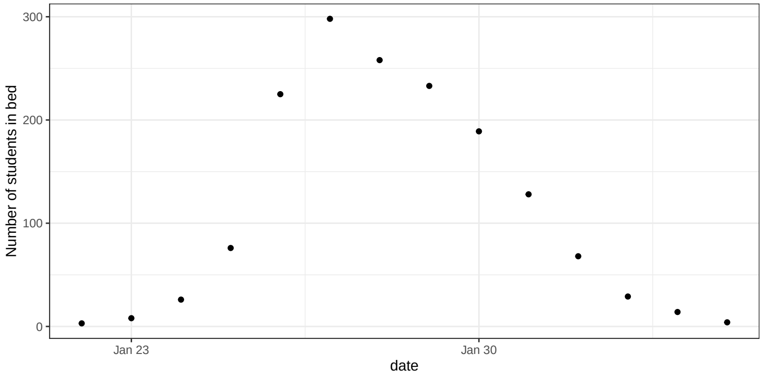
Improving the model

- The ppc suggest our model can improve with an intercept parameter.
- *Exercise: repeat the above procedure, but this time add an intercept parameter β_0. Check that the inference is reliable and perform new posterior predictive checks.*

III
Disease transmission model

1978 influenza outbreak in a British boarding school.

Data: daily number of students in bed.



Susceptible-Infected-Recovered (SIR) model

$$\begin{tikzcd}
\mathit{Susceptible} \arrow[r, "\beta S \frac{I}{N}"] & \mathit{Infected} \arrow[r, "\gamma I"] & \mathit{Recovered}
\end{tikzcd}$$

N: total number of individuals,
$N = S + I + R$.

$$\begin{aligned}
\dot{S} &= -\beta SI/N \\
\dot{I} &= \beta SI/N - \gamma I \\
\dot{R} &= \gamma I
\end{aligned}$$

Susceptible-Infected-Recovered (SIR) model

$$\begin{tikzcd} Susceptible \arrow[r, "\beta S \frac{I}{N}"] & Infected \arrow[r, "\gamma I"] & Recovered \end{tikzcd}$$

$$\begin{aligned} \dot{S} &= -\beta SI/N \\ \dot{I} &= \beta SI/N - \gamma I \\ \dot{R} &= \gamma I \end{aligned}$$

N: total number of individuals, $N = S + I + R$.

β: transmission rate.

γ: rate of recovery of infected individuals.

Susceptible-Infected-Recovered (SIR) model

$$\begin{tikzcd} \boxed{Susceptible} \arrow[r, "\beta S \frac{I}{N}"] & \boxed{Infected} \arrow[r, "\gamma I"] & \boxed{Recovered} \end{tikzcd}$$

$$\begin{aligned} \dot{S} &= -\beta SI/N \\ \dot{I} &= \beta SI/N - \gamma I \\ \dot{R} &= \gamma I \end{aligned}$$

N: total number of individuals, $N = S + I + R$.

β: transmission rate.

γ: rate of recovery of infected individuals.

- I/N: the proportion of infectious individuals.
- $\beta(I/N)$: the probability that a single susceptible individual becomes infected in one day.

Which measurement model should we use?

Which measurement model should we use?

1. *Poisson* likelihood parameterized by $\lambda(t) = I(t)$.
   - Then $\mathbb{E}(y(t)) = I(t)$ and $\operatorname{Var}(y(t)) = I(t)$.

Which measurement model should we use?

1. *Poisson* likelihood parameterized by $\lambda(t) = I(t)$.
   - Then $\mathbb{E}(y(t)) = I(t)$ and $\text{Var}\,(y(t)) = I(t)$.

2. *Negative-Binomial* parameterized by $\mu = I(t)$ and ϕ.
   - Then $\mathbb{E}(y(t)) = I(t)$ and $\text{Var}\,(y(t)) = I(t) + \frac{I(t)^2}{\phi}$.

Which measurement model should we use?

1. *Poisson* likelihood parameterized by $\lambda(t) = I(t)$.
   - Then $\mathbb{E}(y(t)) = I(t)$ and $\operatorname{Var}(y(t)) = I(t)$.

2. *Negative-Binomial* parameterized by $\mu = I(t)$ and ϕ.
   - Then $\mathbb{E}(y(t)) = I(t)$ and $\operatorname{Var}(y(t)) = I(t) + \frac{I(t)^2}{\phi}$.

   - In `Stan` use `neg_binomial_2`.
   - Define in `parameters` block ϕ^{-1}.

Which prior should we use?

- $p(\beta) = \text{normal}^+(2, 1)$: restricts β to be positive and $p(\beta < 4) = 0.975$.
- $p(\gamma) = \text{normal}^+(0.4, 0.5)$: restricts γ to be positive and $p(\gamma < 1) = 0.9$, i.e. 90% of the time, we expect the average time spent in bed to be less than 1 day).
- $p(\phi^{-1}) = \text{exponential}(5)$, see [Grinsztajn et al., 2021].

code demo

Additional controls for the MCMC sampler.

- Number of (parallel) chains
- Number of warmup iterations
- Number of sampling iterations

Exercise: Write and fit an SIR model for the 1978 influenza outbreak.

- *Check the standard diagnostics ($\widehat{R}$ and ESS) and examine the density and trace plots. Is the inference reliable?*
- *Do posterior predictive checks: does the model accurately describe the data?*
- *Report β, γ, $R_0 = \beta/\gamma$ and the recovery time $T = 1/\gamma$.*
- *How precise are your estimates of T? Should you run longer Markov chains? Shorter ones?*
- *Compare the two proposed measurement models: Poisson and negative binomial.*

For more discussion about this model (e.g. choice of priors, sensitivity tests), see [Grinsztajn et al., 2021].

For more discussion about the length of Markov chains, see [**?**].

IV
Model Comparison

Question: For the SIR model, do we get better predictions with the Poisson or the negative binomial likelihood?

Question: For the SIR model, do we get better predictions with the Poisson or the negative binomial likelihood?

- **Proposition:** Test *model predictions* on a validation set.

Question: For the SIR model, do we get better predictions with the Poisson or the negative binomial likelihood?

- **Proposition:** Test *model predictions* on a validation set.
    - Split the data into a **training** and a **validation** set.

Question: For the SIR model, do we get better predictions with the Poisson or the negative binomial likelihood?

- **Proposition:** Test *model predictions* on a validation set.
  - Split the data into a **training** and a **validation** set.
  - **Training set:** The data y_{train} used to learn the parameters, and on which we condition the posterior,
$$p(\theta \mid y_{\text{train}}).$$

Question: For the SIR model, do we get better predictions with the Poisson or the negative binomial likelihood?

- **Proposition:** Test *model predictions* on a validation set.
  - Split the data into a **training** and a **validation** set.
  - **Training set:** The data y_{train} used to learn the parameters, and on which we condition the posterior,
    $$p(\theta \mid y_{\text{train}}).$$
  - **Validation set:** The data y_{val} we use to "test" the model's predictions.

    Example: At $t = 12$, the model predicts $\tilde{y}(t = 12)$. Compute the *prediction error*,
    $$\text{Err} = \left(\tilde{y}(t = 12) - y_{\text{val}}(t = 12)\right)^2 .$$

Testing *uncertainty calibration* in (point) predictions

Testing *uncertainty calibration* in (point) predictions

Suppose we have a normal likelihood, with point estimates for the learned parameters,

$$\mathrm{Normal}\left(\hat{\mu}(t), \hat{\sigma}\right).$$

Our “best” prediction is $\tilde{y}(t) = \hat{\mu}(t)$.

Testing *uncertainty calibration* in (point) predictions

Suppose we have a normal likelihood, with point estimates for the learned parameters,

$$\text{Normal}\left(\hat{\mu}(t), \hat{\sigma}\right).$$

Our "best" prediction is $\tilde{y}(t) = \hat{\mu}(t)$.

Then the prediction error is

$$\text{Err} = \left(\hat{\mu}(t) - y_{\text{val}}(t)\right)^2,$$

and $\hat{\sigma}$ is unaccounted for!

Testing *uncertainty calibration* in (point) predictions

Suppose we have a normal likelihood, with point estimates for the learned parameters,

$$\mathrm{Normal}\left(\hat{\mu}(t), \hat{\sigma}\right).$$

Our "best" prediction is $\tilde{y}(t) = \hat{\mu}(t)$.

Then the prediction error is

$$\mathrm{Err} = \left(\hat{\mu}(t) - y_{\mathrm{val}}(t)\right)^2,$$

and $\hat{\sigma}$ is unaccounted for!

Instead, let's evaluate the *point-estimate log predictive density*,

$$\begin{aligned} \text{p-lpd} &= \log p(y_{\mathrm{val}}(t) \mid \hat{\mu}, \hat{\sigma}) \\ &= \text{const.} - \log \hat{\sigma} - \frac{1}{2\hat{\sigma}^2}(y_{\mathrm{val}}(t) - \hat{\mu}(t))^2. \end{aligned}$$

Testing *uncertainty calibration* in (point) predictions

Suppose we have a Bernoulli likelihood, with point estimates for the learned parameters,

$$\text{Bernoulli}(\hat{\pi}(t)).$$

Our "best" prediction is $\tilde{y}(t) = \mathbb{I}[\hat{\pi}(t) > 0.5]$.

Then the prediction error is

$$\text{Err} = \mathbb{I}[\tilde{y}(t) = y_{\text{val}}(t)].$$

Instead, let's evaluate the *point-estimate log predictive density*,

$$\begin{aligned}
\text{p-lpd} &= \log p(y_{\text{val}}(t) \mid \hat{\pi}(t)) \\
&= y_{\text{val}}(t) \log \hat{\pi}(t) + (1 - y_{\text{val}}(t)) \log(1 - \hat{\pi}(t)).
\end{aligned}$$

Testing *uncertainty calibration* in Bayesian predictions

We have a general strategy which accounts for uncertainty in the likelihood for a fixed θ,

$$\text{p-lpd} = \log p(y_{\text{val}}(t) \mid \theta).$$

Testing *uncertainty calibration* in Bayesian predictions

We have a general strategy which accounts for uncertainty in the likelihood for a fixed θ,

$$\text{p-lpd} = \log p(y_{\text{val}}(t) \mid \theta).$$

In a Bayesian framework, we integrate with respect to the posterior and obtain the *expected log predictive density*,

$$\begin{aligned} \text{elpd} &= \log p(y_{\text{val}}(t) \mid y_{\text{train}}) \\ &= \log \int_\Theta p(y_{\text{val}}(t) \mid \theta) p(\theta \mid y_{\text{train}}) \mathrm{d}\theta. \end{aligned}$$

How do we split the data (t, y) into a training and a test set?

Proposition: Do *leave-one-out cross validation* and compute

$$\mathrm{elpd}_{\mathrm{loo}} = \sum_{i=1}^{N} \log p(y_i \mid y_{-i}),$$

where

$$p(y_i \mid y_{-i}) = \int_{\Theta} p(y_i \mid \theta) p(\theta \mid y_{-i}) \mathrm{d}\theta.$$

Recap.

Prediction error based on "best" prediction, $(y_{\text{val}} - \tilde{y})^2$

$\rightarrow$ point-wise log predictive score, p-lpd $= \log p(y_{\text{val}} \mid \hat{\theta})$

$\rightarrow$ expected log predictive score, elpd $= \log p(y_{\text{val}} \mid y_{\text{train}})$

$\rightarrow$ loo CV, $\text{elpd}_{\text{loo}} = \sum_{i=1}^{N} \log p(y_i \mid y_{-i})$

How do we estimate elpd_{loo} efficiently?

Importance Sampling

- **Idea:** Suppose we need to estimate an expectation with respect to $\ell(\theta)$,

$$\int_{\Theta} f(\theta)\ell(\theta)\mathrm{d}\theta,$$

  but using samples from $q(\theta)$.

Importance Sampling

- **Idea:** Suppose we need to estimate an expectation with respect to $\ell(\theta)$,

$$\int_\Theta f(\theta)\ell(\theta)\mathrm{d}\theta,$$

  but using samples from $q(\theta)$.
- For example, $\ell(\theta) = p(\theta \mid y_{-i})$ and $q(\theta) = p(\theta \mid y)$.

Importance Sampling

- **Idea:** Suppose we need to estimate an expectation with respect to $\ell(\theta)$,
$$\int_{\Theta} f(\theta)\ell(\theta)\mathrm{d}\theta,$$
but using samples from $q(\theta)$.
- For example, $\ell(\theta) = p(\theta \mid y_{-i})$ and $q(\theta) = p(\theta \mid y)$.
- Then, note that
$$\int_{\Theta} f(\theta)\ell(\theta)\mathrm{d}\theta = \int_{\Theta} f(\theta)\frac{\ell(\theta)}{q(\theta)}q(\theta)\mathrm{d}\theta.$$

Importance Sampling

- **Idea:** Suppose we need to estimate an expectation with respect to $\ell(\theta)$,

$$\int_\Theta f(\theta)\ell(\theta)\mathrm{d}\theta,$$

  but using samples from $q(\theta)$.
- For example, $\ell(\theta) = p(\theta \mid y_{-i})$ and $q(\theta) = p(\theta \mid y)$.
- Then, note that

$$\int_\Theta f(\theta)\ell(\theta)\mathrm{d}\theta = \int_\Theta f(\theta)\frac{\ell(\theta)}{q(\theta)}q(\theta)\mathrm{d}\theta.$$

- The IS Monte Carlo estimator is

$$\widehat{\mathbb{E}}f(\theta) = \frac{1}{S}\sum_{s=1}^{S} f(\theta^{(s)})\frac{\ell(\theta^{(s)})}{q(\theta^{(s)})}.$$

- The IS Monte Carlo estimator is

$$\widehat{\mathbb{E}} f(\theta) = \sum_{s=1}^{S} f(\theta^{(s)}) \frac{\ell(\theta^{(s)})}{q(\theta^{(s)})}.$$

- The IS Monte Carlo estimator is

$$\widehat{\mathbb{E}} f(\theta) = \sum_{s=1}^{S} f(\theta^{(s)}) \frac{\ell(\theta^{(s)})}{q(\theta^{(s)})}.$$

  Practical concern: it is not uncommon for the IS estimator to have a non-finite variance and so we need $q(\theta) \approx \ell(\theta)$!

- The IS Monte Carlo estimator is

$$\widehat{\mathbb{E}}f(\theta) = \sum_{s=1}^{S} f(\theta^{(s)}) \frac{\ell(\theta^{(s)})}{q(\theta^{(s)})}.$$

Practical concern: it is not uncommon for the IS estimator to have a non-finite variance and so we need $q(\theta) \approx \ell(\theta)$!

Proposition

When the y_j's are independent conditioned on θ, the importance sampling Monte Carlo estimator is

$$\widehat{p}(y_i \mid y_{-i}) = \frac{1}{\sum_{s=1}^{S} \frac{1}{p(y_i|\theta^{(s)})}},$$

where $\theta^{(s)} \sim p(\theta \mid y)$.

Practical concerns:

- Several steps need to be taken to stabilize IS estimators.

Practical concerns:

- Several steps need to be taken to stabilize IS estimators.
- We will use *Pareto-smoothed importance sampling* (PSIS) [Vehtari et al., 2017].

Practical concerns:

- Several steps need to be taken to stabilize IS estimators.
- We will use *Pareto-smoothed importance sampling* (PSIS) [Vehtari et al., 2017].
- PSIS comes equipped with a $\hat{k}$ diagnostic:
  - if $\hat{k} < 0.5$, PSIS estimators is reliable.
  - if $\hat{k} \geq 0.7$, importance weights have non-finite variance.

Practical concerns:

- Several steps need to be taken to stabilize IS estimators.
- We will use *Pareto-smoothed importance sampling* (PSIS) [Vehtari et al., 2017].
- PSIS comes equipped with a $\hat{k}$ diagnostic:
    - if $\hat{k} < 0.5$, PSIS estimators is reliable.
    - if $\hat{k} \geq 0.7$, importance weights have non-finite variance.

- The R package `loo` computes PSIS.
- In `Stan` 's **`generated quantities`**, need to compute `log_lik`, where

```
log_lik[i] = log p(cases[i] | theta);
```

Exercise: Compare the predictive scores of the SIR models.

- *Evaluate in* `generated quantities` *the log probability mass functions using* `poisson_lpmf` *and* `neg_binomial_2_lpmf`.
- *In R, use the* `loo` *package to compute the PSIS estimates of the* elpd_{loo}.
- *Check* $\hat{k}$ *to see if the IS estimators are reliable.*
- *Which likelihood achieves the best predictive score?*

V
Concluding Remarks

Where does `Stan` fit in the Bayesian modeler's toolkit?

Historical contribution:

- `Stan` was born around 2012.
- First intended as a well programmed version of `BUGS` and `JAGS`.

Where does `Stan` fit in the Bayesian modeler's toolkit?

Historical contribution:

- `Stan` was born around 2012.
- First intended as a well programmed version of `BUGS` and `JAGS`.

Several algorithms were developed as part of `Stan` 's development:

- Adaptive Hamiltonian Monte Carlo [Hoffman and Gelman, 2014, Betancourt, 2018]
- ADVI: a black box variational inference [Kucukelbir et al., 2017]
- PathFinder: an improved variational inference [Zhang et al., 2022].

- Delayed rejection HMC [Modi et al., 2023]
- Adjoint-differentiated Laplace approximation [Margossian et al., 2020]
- GIST sampler [Bou-Rabee et al., 2024]

`Stan` by the people, for the people

- `Stan` is open source: `https://github.com/stan-dev`
- Contributing new functions to `Stan` : `https://github.com/stan-dev/stan/wiki/Contributing-New-Functions-to-Stan`

- Straightforward to interface `Stan` with your own gradient-based algorithm (written in your language of choice), via `BridgeStan`: `https://roualdes.github.io/bridgestan/latest/`.

- A lot of other probabilistic programming languages out there: `PyMC`, `Turing`, `TensorFlow Probability`, `PyTorch`, etc.

Where do Monte Carlo methods fit in the probabilistic modeling toolkit?
(a high-level comparison)

Markov chain Monte Carlo

Variational inference

Where do Monte Carlo methods fit in the probabilistic modeling toolkit?
(a high-level comparison)

Markov chain Monte Carlo

- use general transition kernel

Variational inference

- requires an approximating model

Where do Monte Carlo methods fit in the probabilistic modeling toolkit?
(a high-level comparison)

Markov chain Monte Carlo

- use general transition kernel
- initially error goes down slowly

Variational inference

- requires an approximating model
- initially error goes down quickly

Where do Monte Carlo methods fit in the probabilistic modeling toolkit?
(a high-level comparison)

Markov chain Monte Carlo

- use general transition kernel
- initially error goes down slowly
- asymptotically unbiased

Variational inference

- requires an approximating model
- initially error goes down quickly
- asymptotically biased

Where do Monte Carlo methods fit in the probabilistic modeling toolkit?
(a high-level comparison)

Markov chain Monte Carlo

- use general transition kernel
- initially error goes down slowly
- asymptotically unbiased
- several diagnostics available

Variational inference

- requires an approximating model
- initially error goes down quickly
- asymptotically biased
- not obvious how to check inference (though good candidates exist)

Where do Monte Carlo methods fit in the probabilistic modeling toolkit?
(a high-level comparison)

Markov chain Monte Carlo

- use general transition kernel
- initially error goes down slowly
- asymptotically unbiased
- several diagnostics available

- can check trained model

Variational inference

- requires an approximating model
- initially error goes down quickly
- asymptotically biased
- not obvious how to check inference (though good candidates exist)

- can check trained model

Where does MCMC fit in the probabilistic modeling toolkit?

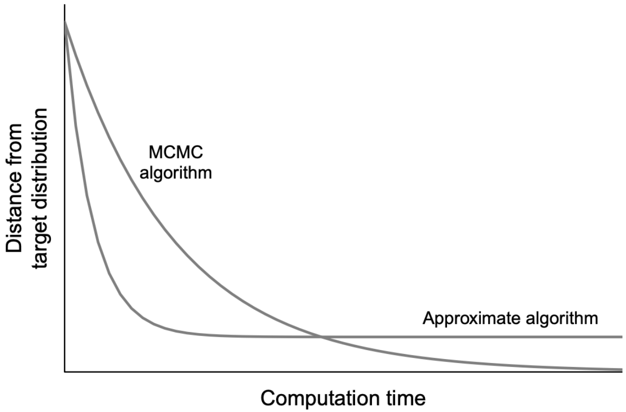


Figure from [Gelman et al., 2020].

References I

[Betancourt, 2018] Betancourt, M. (2018).
A conceptual introduction to Hamiltonian Monte Carlo.
arXiv:1701.02434v1.

[Bou-Rabee et al., 2024] Bou-Rabee, N., Carpenter, B., and Marsden, M. (2024).
Gist: Gibbs self-tuning for locally adaptive hamiltonian monte carlo.
arXiv 2404.15253.

[Gelman et al., 2013] Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013).
Bayesian Data Analysis.
Chapman & Hall.

[Gelman et al., 1997] Gelman, A., Gilks, W. R., and Roberts, G. O. (1997).
Weak convergence and optimal scaling of random walk Metropolis algorithms.
Annals of Applied Probability, 7(1):110–120.

[Gelman et al., 2020] Gelman, A., Vehtari, A., Simpson, D., Margossian, C. C., Carpenter, B., Yao, Y., Kennedy, L., Gabry, J., Bürkner, P.-C., and Modrák, M. (2020).
Bayesian workflow.
arXiv:2011.01808.

References II

[Grinsztajn et al., 2021] Grinsztajn, L., Semenova, E., Margossian, C. C., and Riou, J. (2021).
Bayesian workflow for disease transmission modeling in Stan.
Statistics in Medicine.

[Hoffman and Gelman, 2014] Hoffman, M. D. and Gelman, A. (2014).
The No-U-Turn Sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo.
Journal of Machine Learning Research, pages 1593–1623.

[Kucukelbir et al., 2017] Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A., and Blei, D. (2017).
Automatic differentiation variational inference.
Journal of machine learning research, 18:1 – 45.

[Margossian et al., 2024] Margossian, C. C., Hoffman, M. D., Sountsov, P., Riou-Durand, L., Vehtari, A., and Gelman, A. (2024).
Nested $\widehat{R}$: Assessing the convergence of Markov chain Monte Carlo when running many short chains.
Bayesian Analysis.

[Margossian et al., 2020] Margossian, C. C., Vehtari, A., Simpson, D., and Agrawal, R. (2020).
Hamiltonian Monte Carlo using an adjoint-differentiated Laplace approximation: Bayesian inference for latent Gaussian models and beyond.
Neural Information Processing Systems.

References III

[Metropolis et al., 1953] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953).
Equations of state calculations by fast computing machines.
Journal of Chemical Physics, 26.

[Modi et al., 2023] Modi, C., Barnett, A., and Carpenter, B. (2023).
Delayed rejection Hamiltonian Monte Carlo for sampling multiscale distributions.
Bayesian Analysis.

[Roberts and Rosenthal, 1998] Roberts, G. O. and Rosenthal, J. S. (1998).
Optimal scaling of discrete approximations to Langevin diffusions.
Journal of the Royal Statistical Society, Series B, 60:255–268.

[Vehtari et al., 2017] Vehtari, A., Gelman, A., and Gabry, J. (2017).
Practical bayesian model evaluation using leave-one-out cross-validation and waic.
Statistics and Computing, 27:1413–1432.

[Vehtari et al., 2021] Vehtari, A., Gelman, A., Simpson, D., Carpenter, B., and Bürkner, P.-C. (2021).
Rank-normalization, folding, and localization: An improved $\widehat{R}$ for assessing convergence of MCMC (with discussion).
Bayesian Analysis, 16:667–718.

References IV

[Zhang et al., 2022] Zhang, L., Carpenter, B., Gelman, A., and Vehtari, A. (2022).
Pathfinder: Parallel quasi-Newton variational inference.
Journal of Machine Learning Research, 23(306):1–49.