Bayesian Workflow for Hierarchical and ODE-based Models using Stan

September 2023

Summer School on Advanced Bayesian Methods

mc-stan.org

Instructor:

Charles Margossian
Flatiron Institute

Teaching Assistant:

Maxime Fajgenblat
KU Leuven

Outline:

- Review of Bayesian analysis
- Markov chain Monte Carlo
- Basics of Stan
- ODE-based models
- Leave-one-out cross validation

- Hamiltonian Monte Carlo
- Tuning ODEs in a Bayesian context
- Hierarchical models

- Torsten: an extension of Stan for pharmacometrics
- Population models

An `R` notebook to do the exercises can be found at:

https://github.com/charlesm93/stanTutorial

You can run the `R` code on your local machine or on the `Colab` cloud server.

I

Review of Bayesian Analysis

What is a Bayesian model?

What is a Bayesian model?

Defined as a joint distribution

$$p(\theta, y)$$

over observed variables y and unknowns θ.

What is a Bayesian model?

Defined as a joint distribution

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

over observed variables y and unknowns θ.

What is a Bayesian model?

Defined as a joint distribution

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

over observed variables y and unknowns θ.

$p(y \mid \theta)$ is the *likelihood*:

For a fixed θ, defines a data generating process.

What is a Bayesian model?

Defined as a joint distribution

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

over observed variables y and unknowns θ.

$p(y \mid \theta)$ is the *likelihood*:

For a fixed θ, defines a data generating process.

$p(\theta)$ is the *prior*:

quantitative assumptions and understanding about θ
information from previous analysis
regularization tool

RESEARCH ARTICLE

Estimation of SARS-CoV-2 mortality during the early stages of an epidemic: A modeling study in Hubei, China, and six regions in Europe

Anthony Hauser[1], Michel J. Counotte[1], Charles C. Margossian[2], Garyfallos Konstantinoudis[3], Nicola Low[1], Christian L. Althaus[1], Julien Riou[1,4]*

RESEARCH ARTICLE

Estimation of SARS-CoV-2 mortality during the early stages of an epidemic: A modeling study in Hubei, China, and six regions in Europe

Anthony Hauser[1], Michel J. Counotte[1], Charles C. Margossian[2], Garyfallos Konstantinoudis[3], Nicola Low[1], Christian L. Althaus[1], Julien Riou[1,4]*

Likelihood:

Epidemiological model of the disease dynamic

Measurement model: test results, hospital deaths.

RESEARCH ARTICLE

Estimation of SARS-CoV-2 mortality during the early stages of an epidemic: A modeling study in Hubei, China, and six regions in Europe

Anthony Hauser[1], Michel J. Counotte[1], Charles C. Margossian[2], Garyfallos Konstantinoudis[3], Nicola Low[1], Christian L. Althaus[1], Julien Riou[1,4]*

Likelihood:

Epidemiological model of the disease dynamic
Measurement model: test results, hospital deaths.

Prior:

Constraints on interpretable parameters
Meta-analysis for asymptomatic rate

Bayesian inference

Given observations y, want to learn about θ

Bayesian inference

Given observations y, want to learn about θ

Proposition: Learn a posterior distribution

$$p(\theta \mid y) = \frac{p(\theta)p(y \mid \theta)}{p(y)}$$

Bayesian inference

Given observations y, want to learn about θ

Proposition: Learn a posterior distribution

$$p(\theta \mid y) = \frac{p(\theta)p(y \mid \theta)}{p(y)}$$

For some transformation f of the parameters θ, can learn

$$p(f(\theta) \mid y)$$

Bayesian inference

Given observations y, want to learn about θ

Proposition: Learn a posterior distribution

$$p(\theta \mid y) = \frac{p(\theta)p(y \mid \theta)}{p(y)}$$

For some transformation f of the parameters θ, can learn

$$p(f(\theta) \mid y)$$

f may be predictions about the future, a useful summary (e.g. R_0 infection rate), etc.

Bayesian inference

Given observations y, want to learn about θ

Proposition: Learn a posterior distribution

$$p(\theta \mid y) = \frac{p(\theta)p(y \mid \theta)}{p(y)}$$

For some transformation f of the parameters θ, can learn

$$p(f(\theta) \mid y)$$

f may be predictions about the future, a useful summary (e.g. R_0 infection rate), etc.

The variance in $p(f(\theta) \mid y)$ accounts for both the modeled noise and the uncertainty in our estimation of θ.

Example: normal-normal model

$$\begin{aligned} p(\theta) &= \text{normal}(\mu, \tau) \\ p(y_n \mid \theta) &= \text{normal}(\theta, \sigma) \end{aligned}$$

Suppose we have N i.i.d observations $y_1, y_2, \cdots, y_N$.

Example: normal-normal model

$$\begin{aligned} p(\theta) &= \text{normal}(\mu, \tau) \\ p(y_n \mid \theta) &= \text{normal}(\theta, \sigma) \end{aligned}$$

Suppose we have N i.i.d observations $y_1, y_2, \cdots, y_N$.

Then

$$p(\theta \mid \mathbf{y}) = \text{normal}\left(\frac{\mu/\tau^2 + N\bar{y}/\sigma^2}{1/\tau^2 + N/\sigma^2}, \frac{1}{1/\tau^2 + N/\sigma^2}\right)$$

Example: normal-normal model

$$\begin{aligned} p(\theta) &= \text{normal}(\mu, \tau) \\ p(y_n \mid \theta) &= \text{normal}(\theta, \sigma) \end{aligned}$$

Suppose we have N i.i.d observations $y_1, y_2, \cdots, y_N$.

Then

$$p(\theta \mid \mathbf{y}) = \text{normal}\left(\frac{\mu/\tau^2 + N\bar{y}/\sigma^2}{1/\tau^2 + N/\sigma^2}, \frac{1}{1/\tau^2 + N/\sigma^2}\right)$$

In practice, the posterior is <u>not</u> tractable.

Need to estimate summary quantities: expectation values, variance, quantiles, $\cdots$

Bayesian learning

Suppose we obtain data over two observations, $\mathbf{y}_1$ and $\mathbf{y}_2$.

Bayesian learning

Suppose we obtain data over two observations, $\mathbf{y}_1$ and $\mathbf{y}_2$. Then the following two procedure are equivalent,

(1)

- Start with a prior $p(\theta)$.
- Compute the posterior $p(\theta \mid \mathbf{y}_1, \mathbf{y}_2)$.

or

(2)

- Start with a prior $p(\theta)$.
- Compute the posterior $p(\theta \mid \mathbf{y}_1)$
- Use $p(\theta \mid \mathbf{y}_1)$ as a new prior.
- Compute the posterior $\tilde{p}(\theta \mid \mathbf{y}_2) \propto p(\mathbf{y}_2 \mid \theta) p(\theta \mid \mathbf{y}_1)$.

Bayesian learning

Suppose we obtain data over two observations, $\mathbf{y}_1$ and $\mathbf{y}_2$. Then the following two procedure are equivalent,

(1)

- Start with a prior $p(\theta)$.
- Compute the posterior $p(\theta \mid \mathbf{y}_1, \mathbf{y}_2)$.

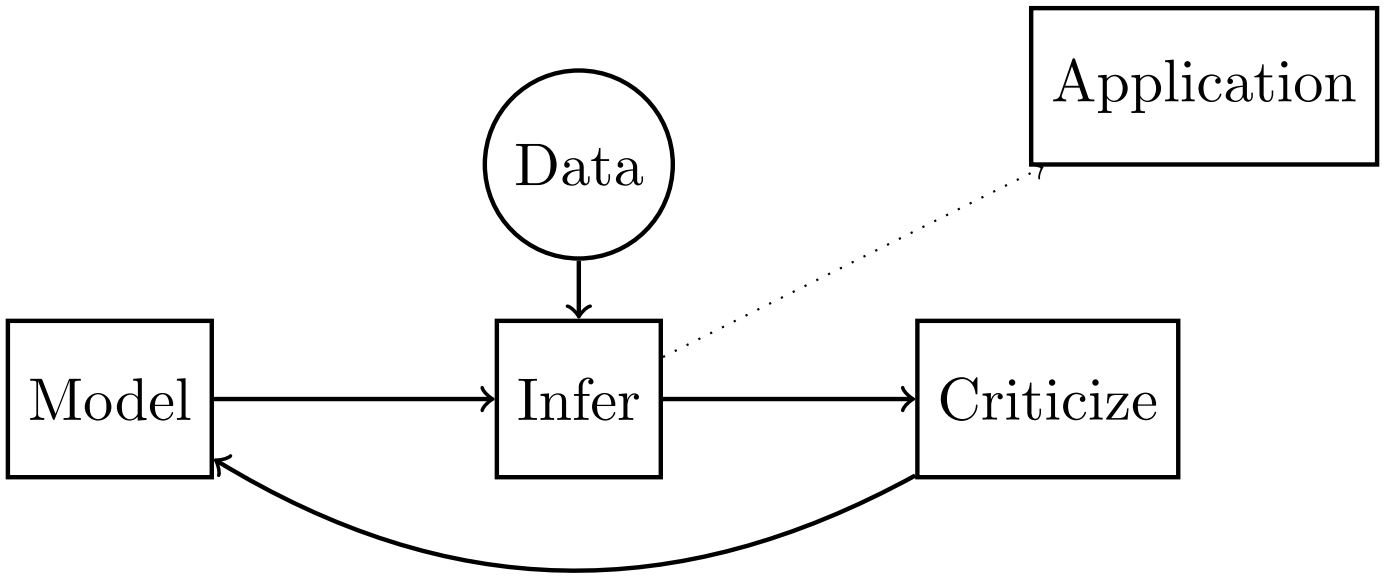
or

(2)

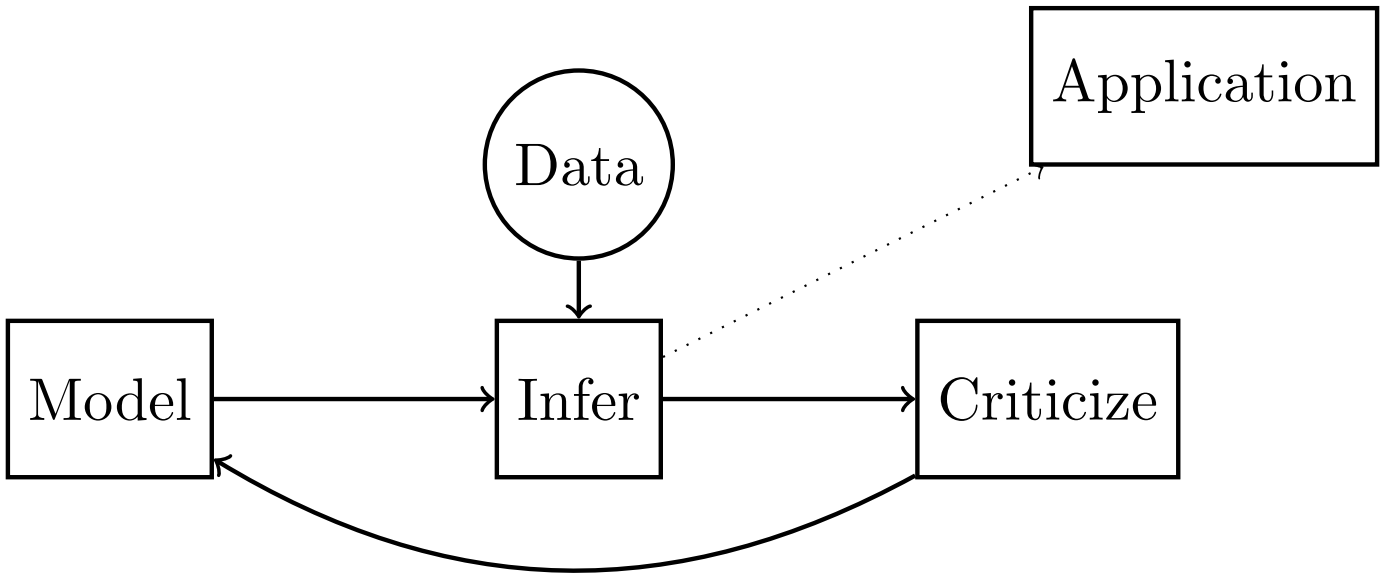
- Start with a prior $p(\theta)$.
- Compute the posterior $p(\theta \mid \mathbf{y}_1)$
- Use $p(\theta \mid \mathbf{y}_1)$ as a new prior.
- Compute the posterior $\tilde{p}(\theta \mid \mathbf{y}_2) \propto p(\mathbf{y}_2 \mid \theta) p(\theta \mid \mathbf{y}_1)$.

$\tilde{p}(\theta \mid \mathbf{y}_2) = p(\theta \mid \mathbf{y}_1, \mathbf{y}_2)$.

Bayesian workflow



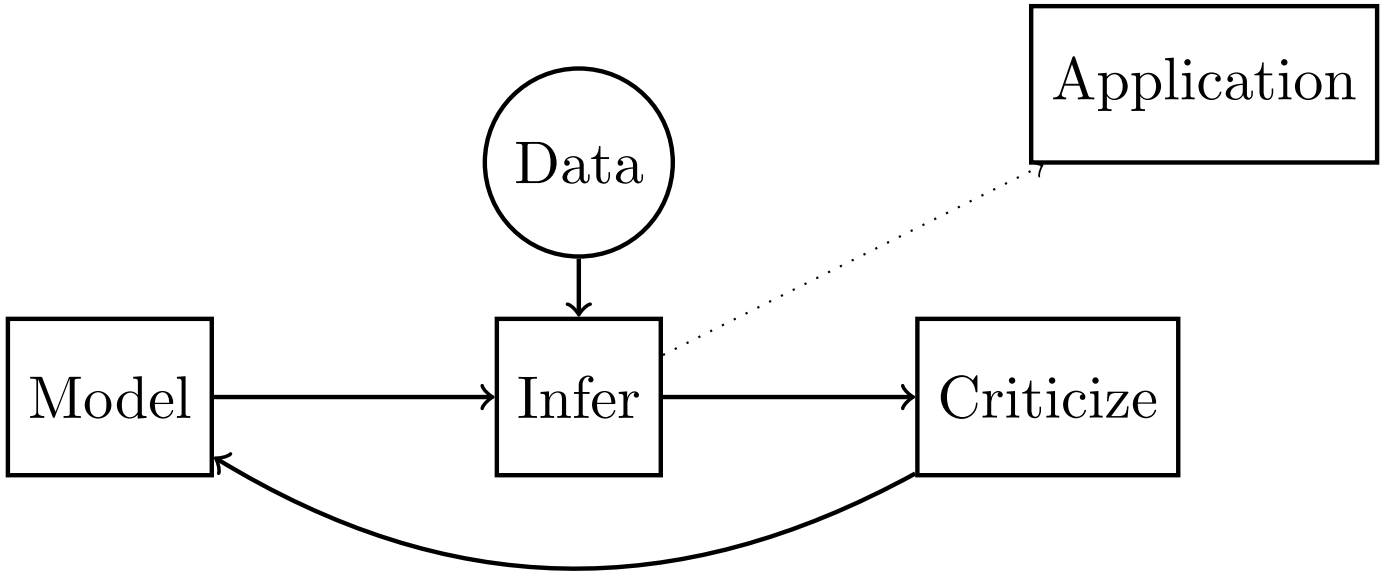
Bayesian workflow



Model

$p(y, \theta)$

Bayesian workflow



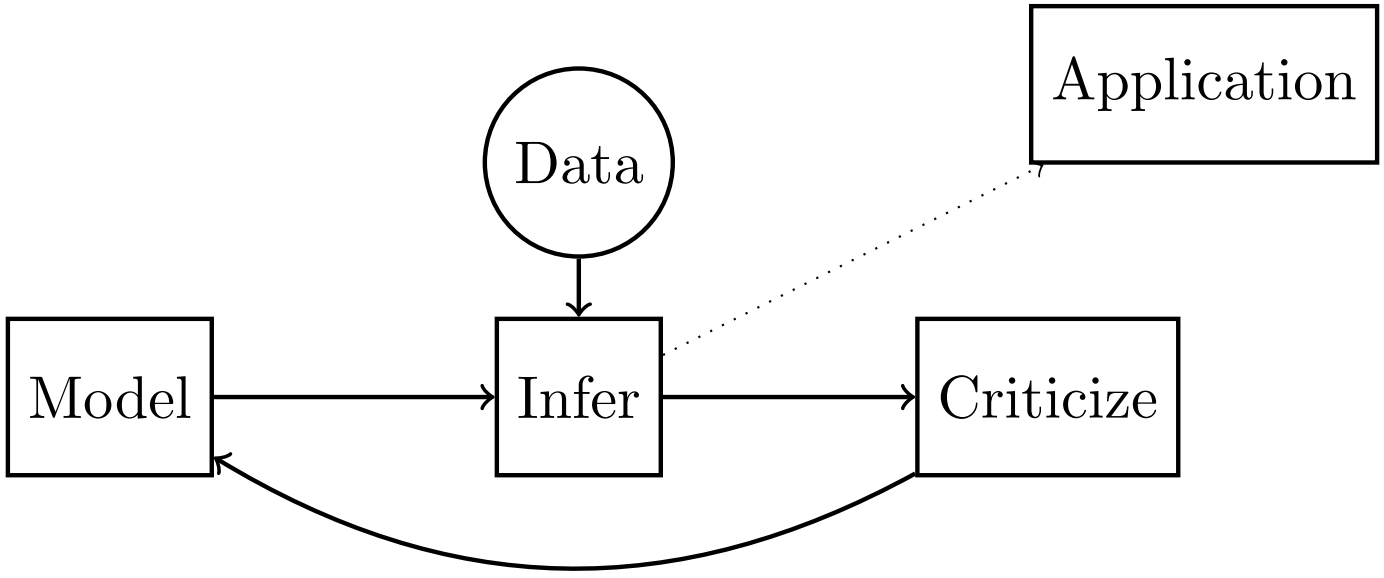
Model

$p(y, \theta)$

Infer

$p(f(\theta) \mid y)$

Bayesian workflow



Model

$p(y, \theta)$

Infer

$p(f(\theta) \mid y)$

Criticize

Simulation,
prediction,
cross-validation,
etc.

RESEARCH ARTICLE

Estimation of SARS-CoV-2 mortality during the early stages of an epidemic: A modeling study in Hubei, China, and six regions in Europe

Anthony Hauser[1], Michel J. Counotte[1], Charles C. Margossian[2], Garyfallos Konstantinoudis[3], Nicola Low[1], Christian L. Althaus[1], Julien Riou[1,4]*

The published model is the $\sim$15th iteration.

RESEARCH ARTICLE

Estimation of SARS-CoV-2 mortality during the early stages of an epidemic: A modeling study in Hubei, China, and six regions in Europe

Anthony Hauser[1], Michel J. Counotte[1], Charles C. Margossian[2], Garyfallos Konstantinoudis[3], Nicola Low[1], Christian L. Althaus[1], Julien Riou[1,4]*

The published model is the ∼15th iteration.

Grinsztajn et al. Bayesian workflow for disease transmission model in Stan, *Statistics in Medicine* (2021)

Gelman et al. Bayesian workflow, *arXiv:2011.01808* (2020)

II

Markov chain Monte Carlo

Characterizing the posterior distribution

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

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

Characterizing the posterior distribution

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:

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

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

Characterizing the posterior distribution

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:

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

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

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,

$$\lim_{n \to \infty} \theta^{(n)} \sim p(\theta \mid y),$$

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,

$$\lim_{n \to \infty} \theta^{(n)} \sim p(\theta \mid y),$$

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

In practice, for large n,

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

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,

$$\lim_{n \to \infty} \theta^{(n)} \sim p(\theta \mid y),$$

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

In practice, for large n,

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

- 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,

$$\lim_{n \to \infty} \theta^{(n)} \sim p(\theta \mid y),$$

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

In practice, for large n,

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

- 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 in the *parameter space*, $\theta^{(0)} \sim p_0$.

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

1. Start at an initial point in the *parameter space*, $\theta^{(0)} \sim p_0$.
2. Apply the transition kernel N times:
   1. Take a random step in the parameter space, from $\theta^{(i)}$ to $\theta^{(i+1)}$ to propose a new sample.
   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 in the *parameter space*, $\theta^{(0)} \sim p_0$.
2. Apply the transition kernel N times:
   1. Take a random step in the parameter space, from $\theta^{(i)}$ to $\theta^{(i+1)}$ to propose a new sample.
   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

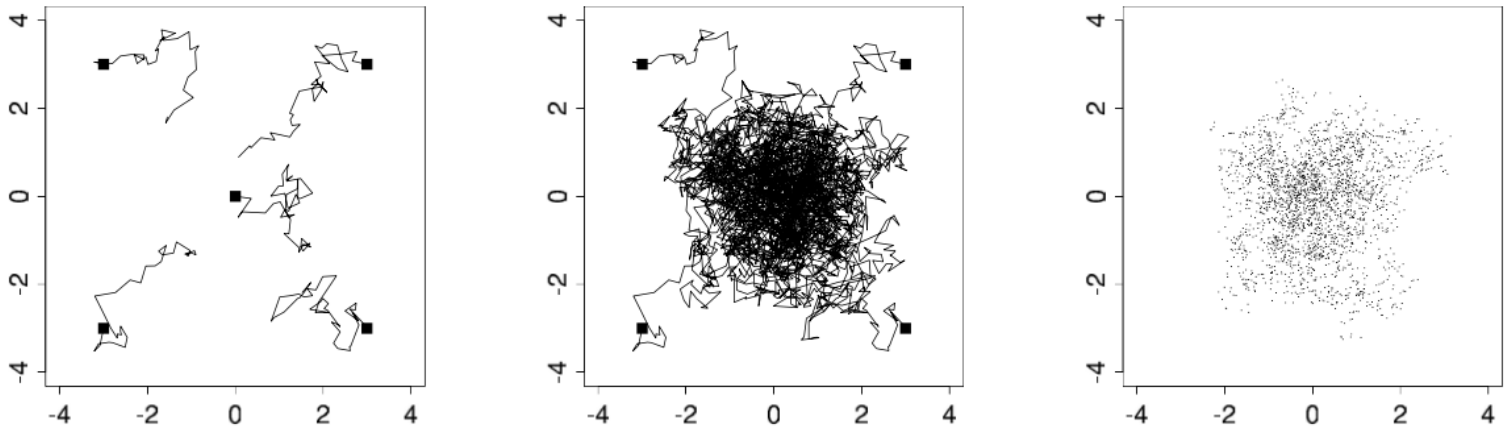


Figure from [Gelman et al., 2013].

Example: Metropolis algorithm

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 not 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 $p_0 = p(\theta \mid z)$ 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 $p_0 = p(\theta \mid z)$ 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 $p_0 = p(\theta \mid z)$ 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,

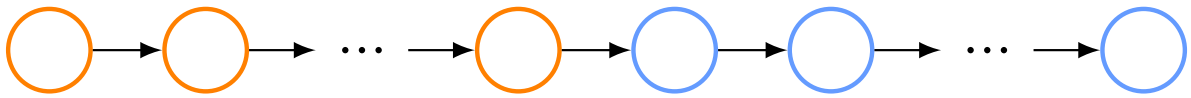
$$\frac{1}{N}\sum_i f(\theta^{(n)}) \overset{\text{approx}}{\sim} \text{Normal}\left(\mathbb{E}[f(\theta)], \frac{\text{Var} f(\theta)}{N_{\text{eff}}}\right)$$

  where N_{eff} is the **effective sample size (ESS)**.
- The ESS by a process with autocorrelation ρ_t is

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

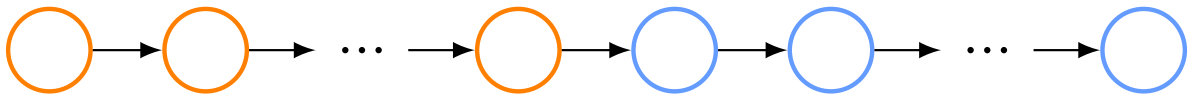
  Here ρ_t is the chain's autocorrelation for two variables separated by t iterations.

Handling the error of MCMC



In practice, MCMC proceeds in two phases:

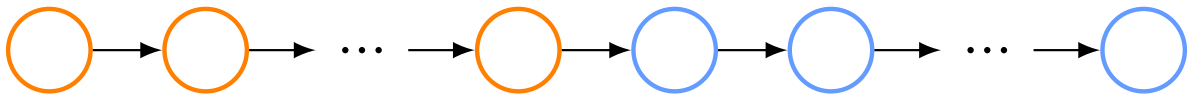
Handling the error of MCMC



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, Γ, should we choose?
Many choices!

Metropolis, Metropolis-Hastings, Gibbs, **Hamiltonian Monte Carlo**, Metropolis-adjusted Langevin, ...

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

Metropolis, Metropolis-Hastings, Gibbs, **Hamiltonian Monte Carlo**, Metropolis-adjusted Langevin, ...

Hamiltonian Monte Carlo:

- Scales in high dimensions and can approximate ill-conditioned posteriors.

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

Metropolis, Metropolis-Hastings, Gibbs, **Hamiltonian Monte Carlo**, Metropolis-adjusted Langevin, ...

Hamiltonian Monte Carlo:

- Scales in high dimensions and can approximate ill-conditioned posteriors.
- Gradient based, requires evaluating $\nabla_\theta \log p(\theta \mid y)$.
- Difficult to tune!

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

Metropolis, Metropolis-Hastings, Gibbs, **Hamiltonian Monte Carlo**, Metropolis-adjusted Langevin, ...

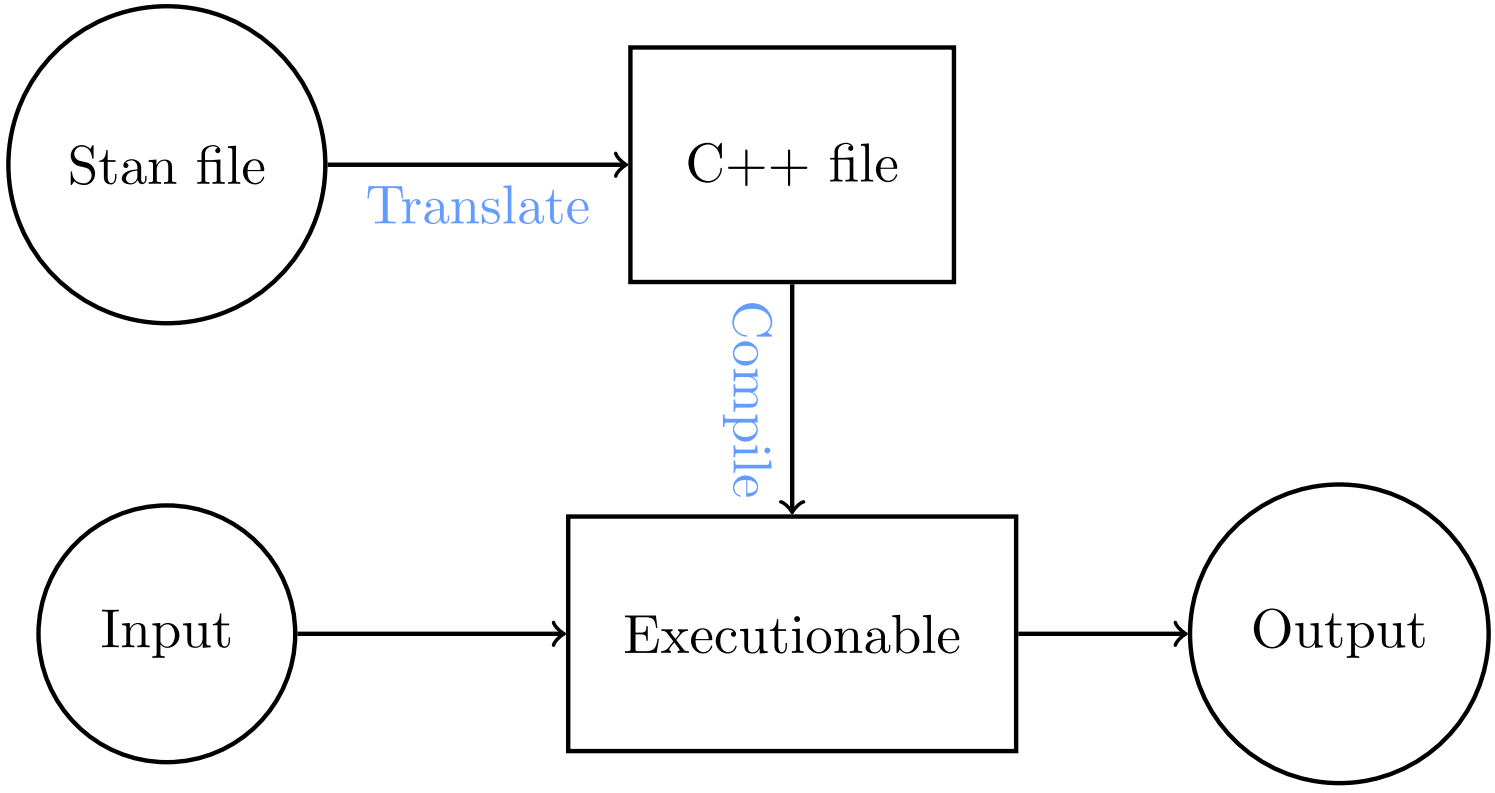
Hamiltonian Monte Carlo:

- Scales in high dimensions and can approximate ill-conditioned posteriors.
- Gradient based, requires evaluating $\nabla_\theta \log p(\theta \mid y)$.
- Difficult to tune!
- `Stan` provides automated calculations of gradients and a self-tuning HMC algorithm.

III
Basics of Stan

- `Stan` is an expressive language for joint distributions.
- It "automatically" computes derivatives.
- It "automatically" performs inference algorithms.

How Stan works



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
- . . .

Example: Bayesian linear regression

The data generating process is:

$$p(y \mid \theta) = \text{Normal}\,(\beta x, \sigma)$$

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

Example: Bayesian linear regression

As a prior, we use:

- $\beta \sim \text{Normal}(2.0, 1.0)$
- $\sigma \sim \text{Gamma}(1.0, 1.0)$

which encode information from previously observed data.

Writing the `Stan` file

We need a statement that specifies the log joint distribution. Recall:

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

Then:

$$\log p(\theta, y) = \log p(y \mid \theta) + \log p(\theta)$$

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

A `Stan` program is divided into coding blocks:

- data
- parameter
- model

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.
}
```

Writing the Stan file

```
model {
 target += normal_lpdf(y | beta * x, sigma);

 // or equivalently

 y ~ normal(beta * x, sigma);
}
```

Writing the `Stan` file

Live demo.

Convergence diagnostic

Are the chains still biased by their initializations?

Proposition: Start each chain at a different location and check that they all converge to the same distribution. Look at:

- the trace plots and the density plots to compare estimates from each chain.
- the $\widehat{R}$ statistic.

The $\widehat{R}$ statistic,

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

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, $\widehat{R} \gg 1$.

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

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

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

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

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 $\bar{\theta}^{(\cdot m)}$.
- $\widehat{W}$ is the (average) within-chain variance.

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

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

Can write $\widehat{R}$ as

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

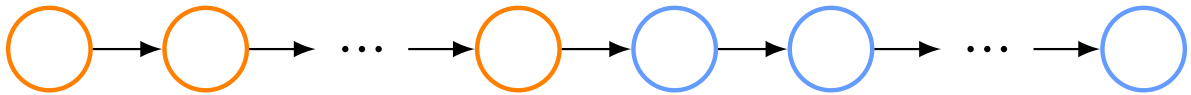
where

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

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

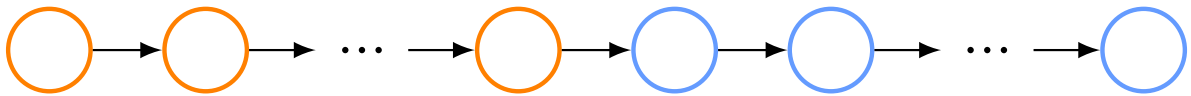
Want to make sure Var $\left(\bar{\theta}^{(\cdot m)}\right)$ is small.

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



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

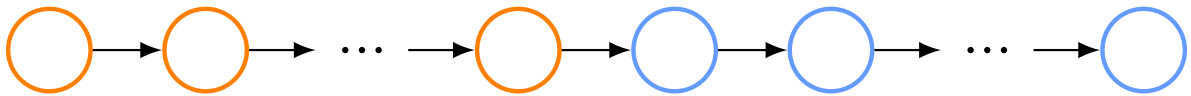
A more in-depth look at $\widehat{R}$



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

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

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

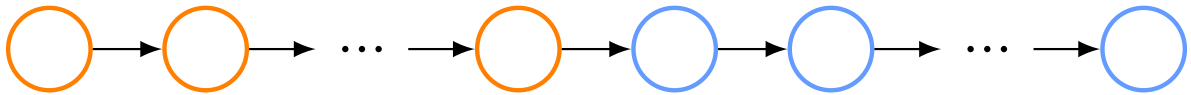


Question. What can $\mathrm{Var}\left(\bar{\theta}^{(\cdot m)}\right)$ teach us about convergence and bias decay?

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

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

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



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

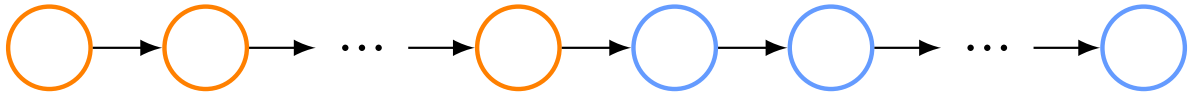
$$\mathrm{Var}\left(\bar{\theta}^{(\cdot m)}\right) = \mathrm{Var}\left(\mathbb{E}\left(\bar{\theta}^{(\cdot m)} \mid \theta^{(0)}\right)\right) + \mathbb{E}\left(\mathrm{Var}\left(\bar{\theta}^{(\cdot m)} \mid \theta^{(0)}\right)\right).$$

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

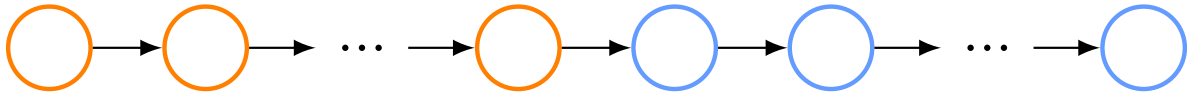
As we warmup the chains, both the nonstationary variance and squared bias decay to 0, and so $\widehat{R}$ acts as a "proxy clock" for bias.

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

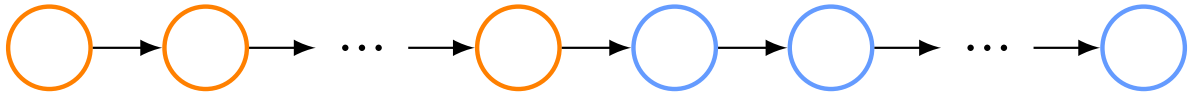
- What quantity does $\widehat{R}$ measure and how close to 1 should it be?
  - [Vehtari et al., 2021] propose checking that $\widehat{R} \leq 1.01$.
  - [Moins et al., 2022] examine the property of $\widehat{R}$ for stationary chains.
  - [Margossian et al., 2023] examine $\widehat{R}$ for nonstationary chains and propose a more direct measure of the nonstationary variance.



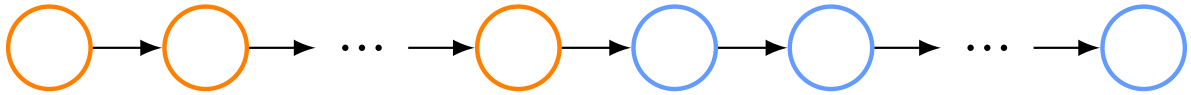
- $\widehat{R}$ (ideally) tells us if the warmup phase is long enough.



- $\widehat{R}$ (ideally) tells us if the warmup phase is long enough.
- ESS and Monte Carlo standard error (MCSE) tell us if the sampling phase is long enough.



- $\widehat{R}$ (ideally) tells us if the warmup phase is long enough.
- ESS and Monte Carlo standard error (MCSE) tell us if the sampling phase is long enough.
- ESS_{tail} quantifies information for tail estimates [Vehtari et al., 2021].



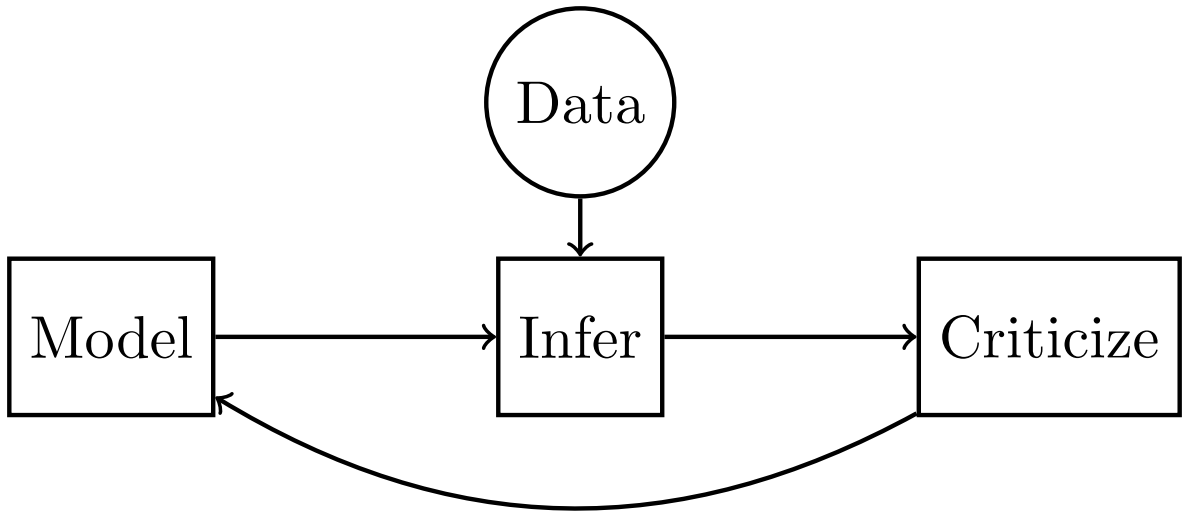
- $\widehat{R}$ (ideally) tells us if the warmup phase is long enough.
- ESS and Monte Carlo standard error (MCSE) tell us if the sampling phase is long enough.
- ESS_{tail} quantifies information for tail estimates [Vehtari et al., 2021].
- Median, $M(\theta)$ and Median Absolute Deviation (MAD),

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

can be helpful when the first moments are not finite.

Posterior predictive checks

- Recall Box's loop.
- Does our model accurately describe the data?



Posterior predictive checks

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

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

Posterior predictive checks

Proposition:
Each time we draw a sample, $\theta^{(i)} = (\beta^{(i)}, \sigma^{(i)})$*, 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,

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

Posterior predictive checks

To do this, we will use the `generated quantities` block.

Live 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.

General resources to use Stan

- Stan User's Guide (https://mc-stan.org/docs/stan-users-guide/index.html)
- Stan Language Reference Manual (https://mc-stan.org/docs/reference-manual/index.html)
- Stan Language Functions Reference (https://mc-stan.org/docs/functions-reference/index.html)

- Stan Forum (http://discourse.mc-Stan.org/)

Parallel chains

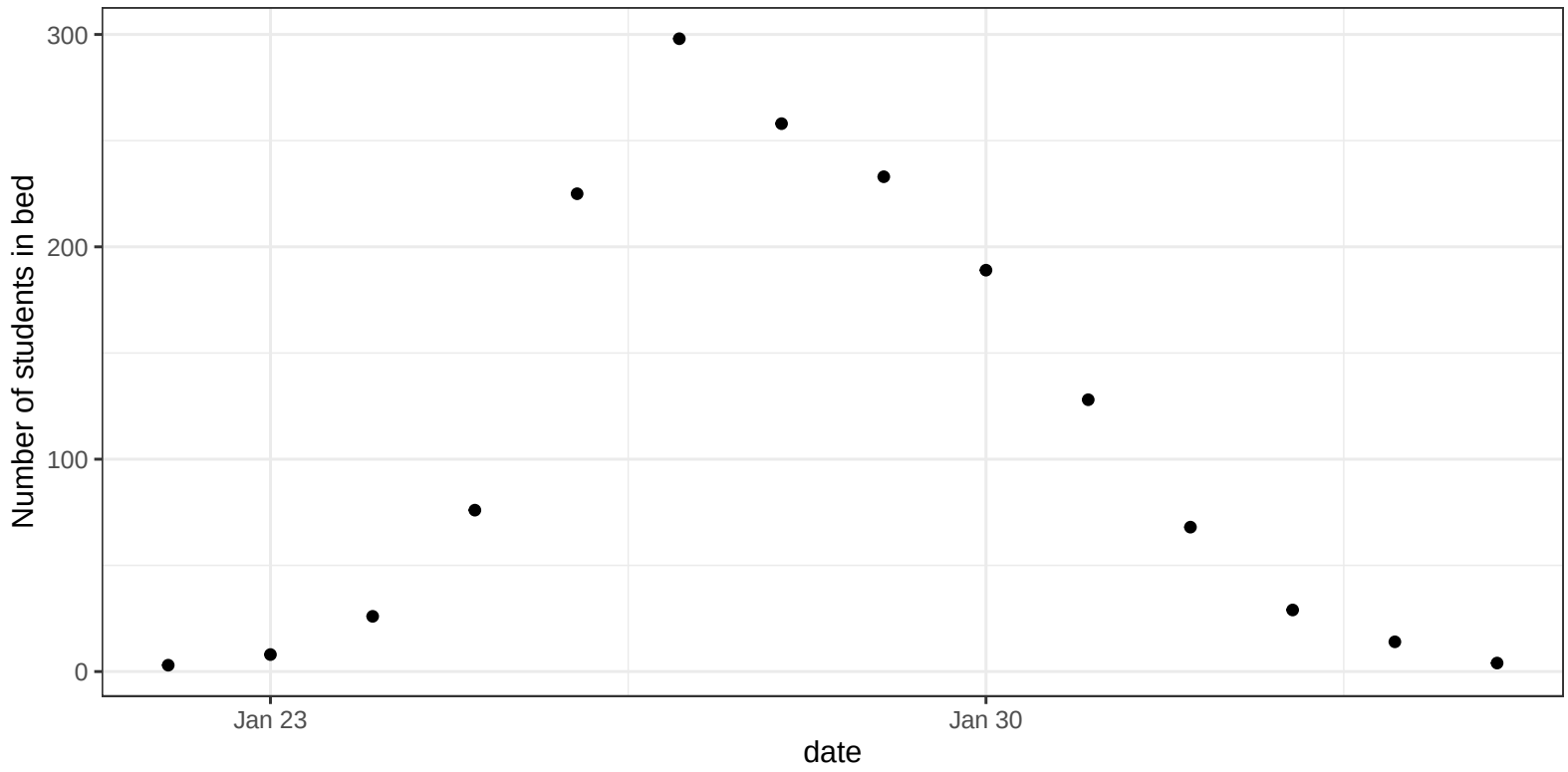
- Each chain is completely independent and can be run on a different core.

```
fit <- mod$sample(file = "model/linear.stan",
                  data = data, chains = 4,
                  init = init,
                  parallel_chains = 4)
```

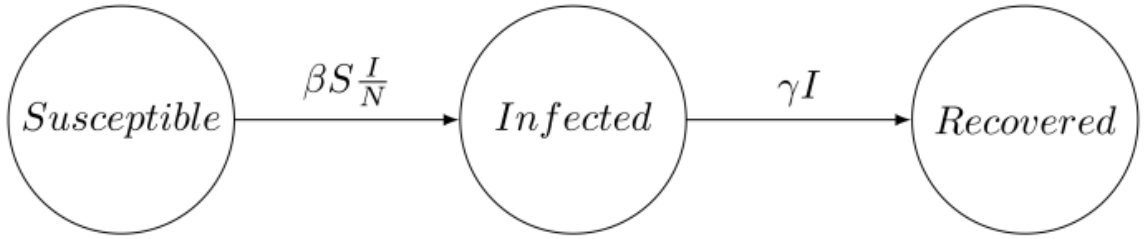
IV
ODE-based models

1978 influenza outbreak in a British boarding school.

Data: daily number of students in bed.

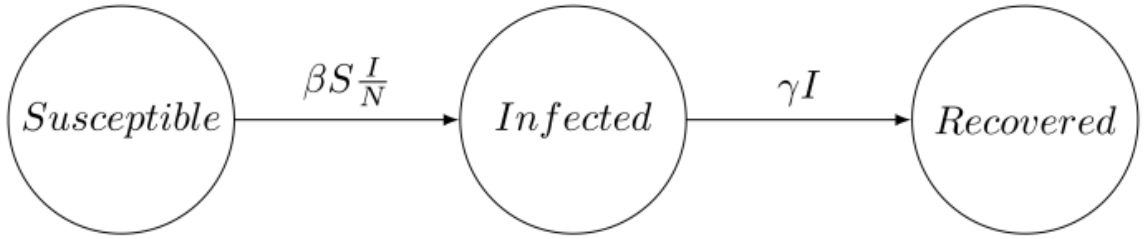


Susceptible-Infected-Recovered (SIR) model



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

Susceptible-Infected-Recovered (SIR) model

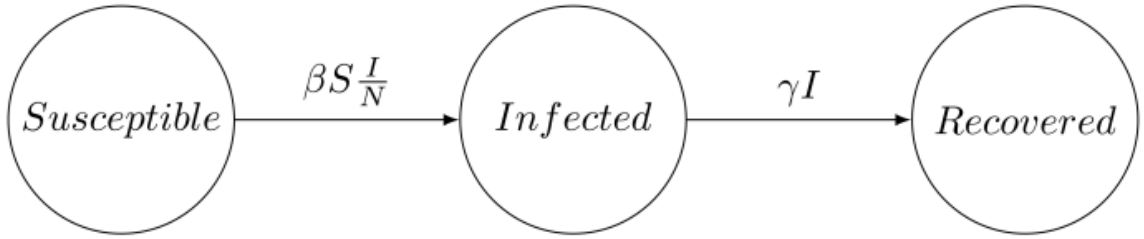


$$\begin{aligned} \dot{S} &= -\beta SI/N \\ \dot{I} &= \beta SI/N - \gamma I \\ \dot{R} &= \gamma I \end{aligned}$$

β: transmission rate.

γ: rate of recovery of infected individuals.

Susceptible-Infected-Recovered (SIR) model



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

β: transmission rate.

γ: rate of recovery of infected individuals.

Interpretation:

- I/N is the proportion of infectious individuals.
- $\beta(I/N)$ is then 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 $\text{Var}\,(y(t)) = I(t)$.

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 $\text{Var}\,(y(t)) = I(t)$.

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

   - 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].

Need additional blocks to fit this model:

functions: Here we'll construct a function that returns $\{\dot{S}, \dot{I}, \dot{R}\}$, which we can then pass to an ODE solver.

- `vector sir (real t, vector y, real beta, real gamma, int N) { ··· return dy_dt };`
- `t`: time
- `y`: the solution to the ODE, $y(t) = [S(t), I(t), R(t)]$

Need additional blocks to fit this model:

functions: Here we'll construct a function that returns $\{\dot{S}, \dot{I}, \dot{R}\}$, which we can then pass to an ODE solver.

- `vector sir (real t, vector y, real beta, real gamma, int N) { ··· return dy_dt };`
- `t`: time
- `y`: the solution to the ODE, $y(t) = [S(t), I(t), R(t)]$

transformed parameters: Allows us to do manipulations on the parameters

- compute $I(t)$ by solving the ODE:
  ```
  array[n_days] vector[3] y
    = ode_rk45(sir, y0, t0, ts, beta, gamma, N);
  ```
- `y0`: initial condition for $t = t_0$.
- `ts`: times at which we require a solution.

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 the posterior predictive checks: does the model accurately describe the data?*
- *Report β, γ and*
$$R_0 = \beta/\gamma.$$
- *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].

V
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,

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

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,

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

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!

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

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

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,

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

To be Bayesian, 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

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

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 with 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 with 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

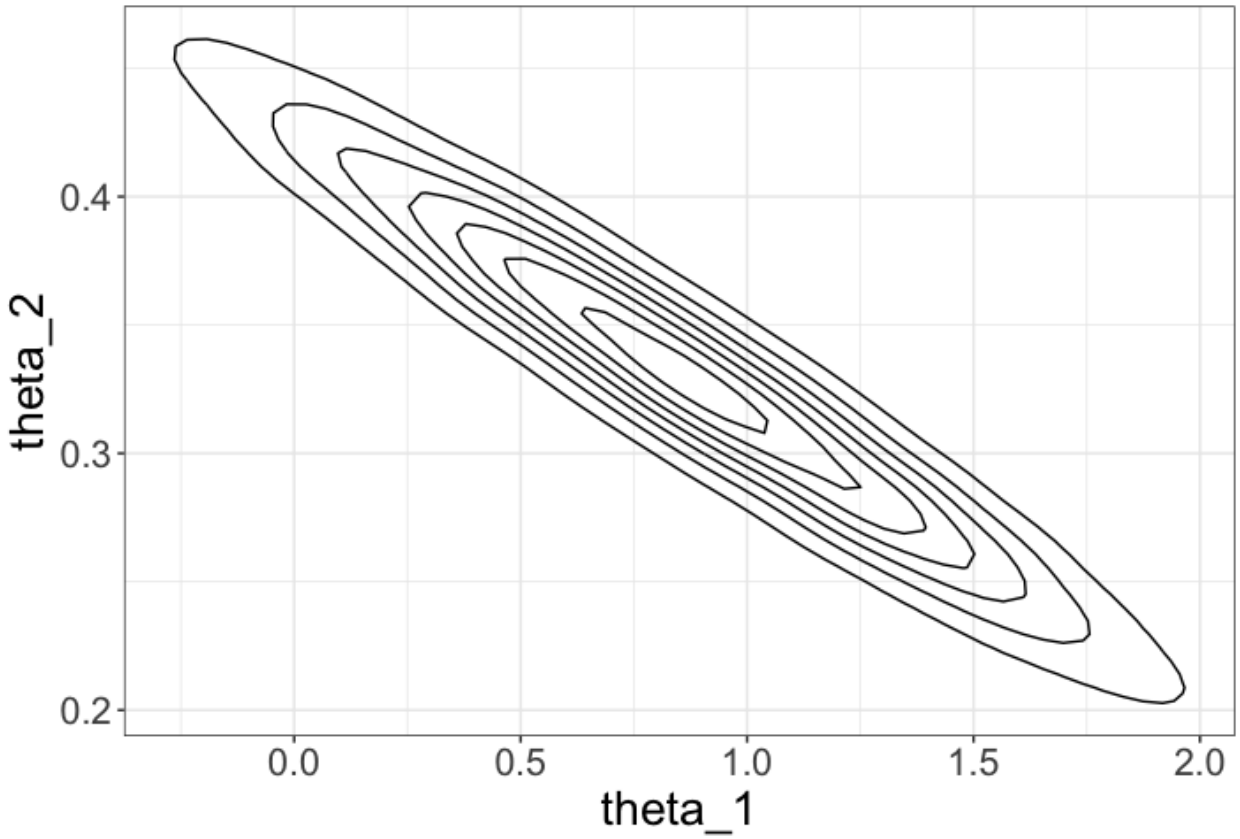
```
log_lik[i] = log p(cases[i] | theta);
```

Exercise: *Compare the predictive scores of the SIR models.*

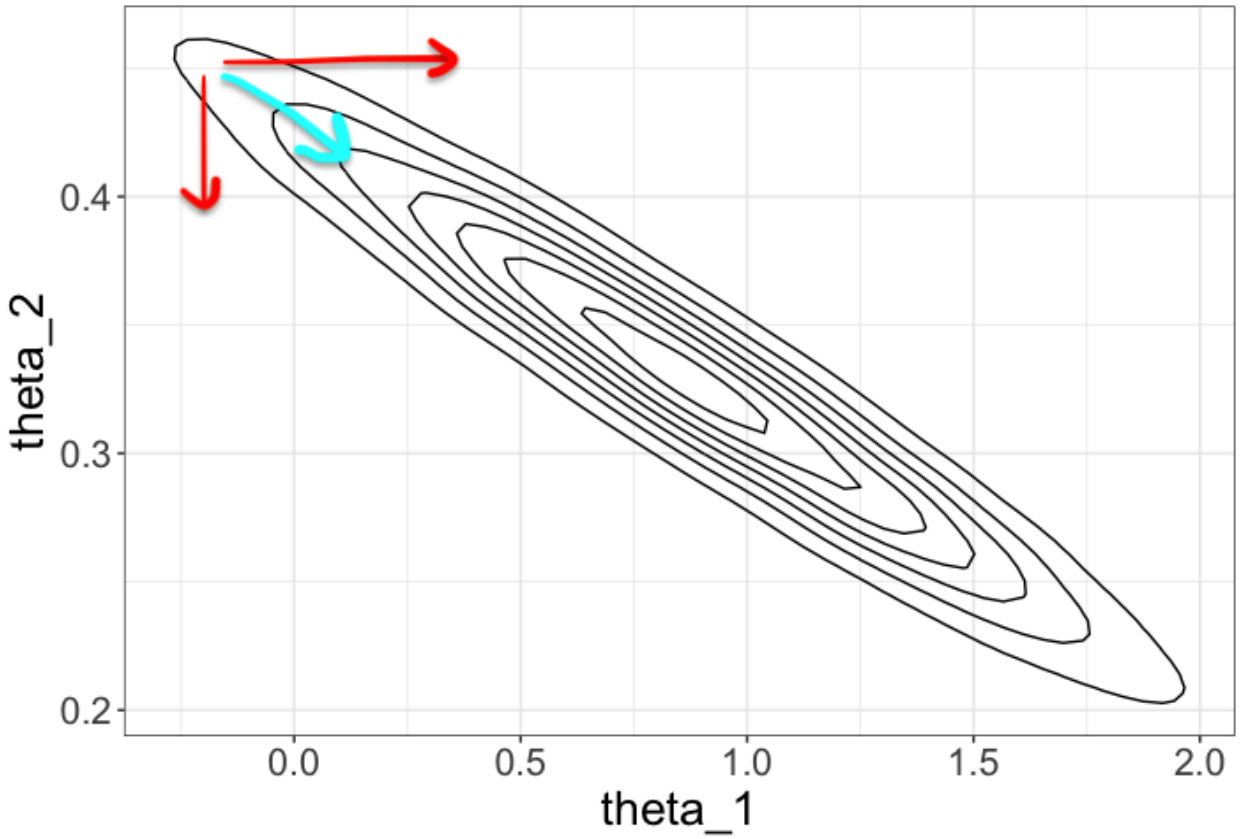
- *Evaluate in* `generated quantities` *the log probability mass functions using* `poisson_lpmf` *and* `neg_binomial_2_lpmf`*.*
- *In R, use the* `loo` *package to compute the PSIS estimates of the* $elpd_{loo}$*.*
- *Check* $\hat{k}$ *to see if the IS estimators are reliable.*
- *Which likelihood achieves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

VI
Hamiltonian Monte Carlo

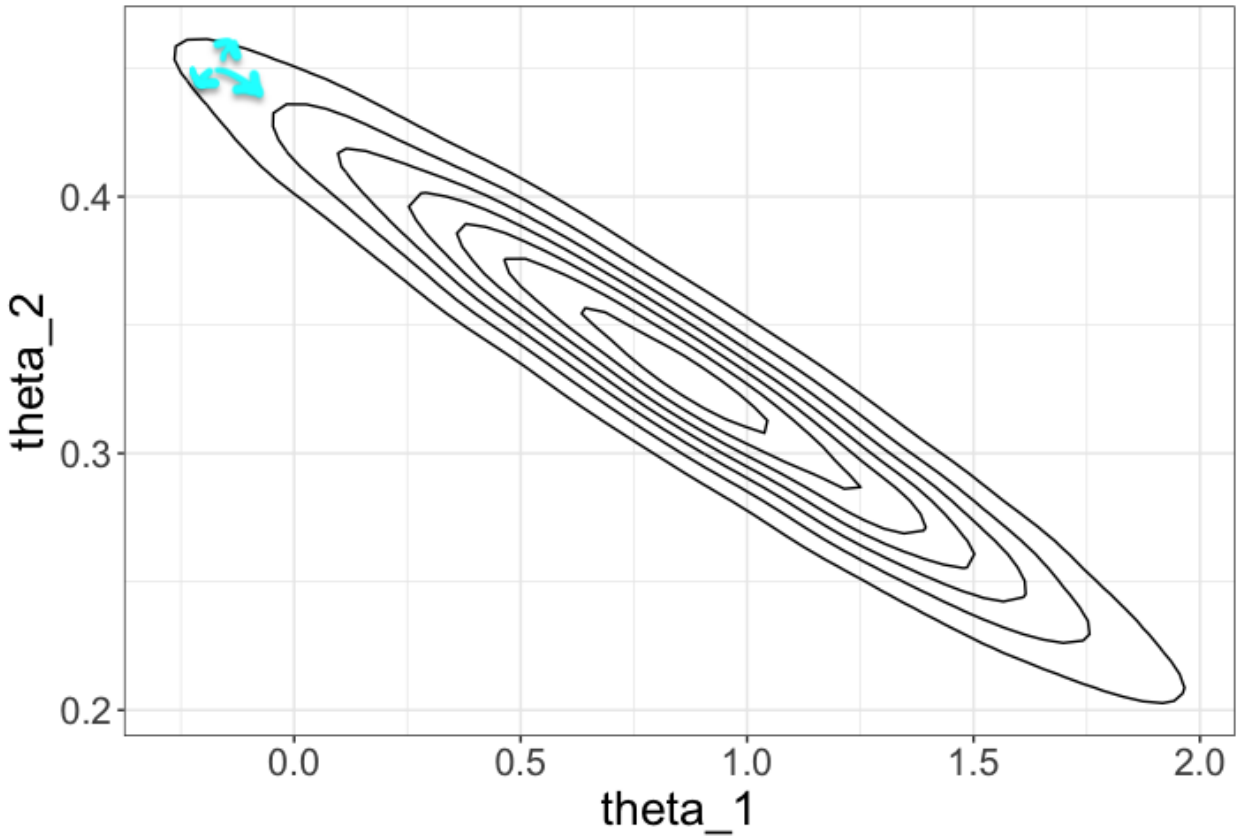
Geometric structure in the distribution



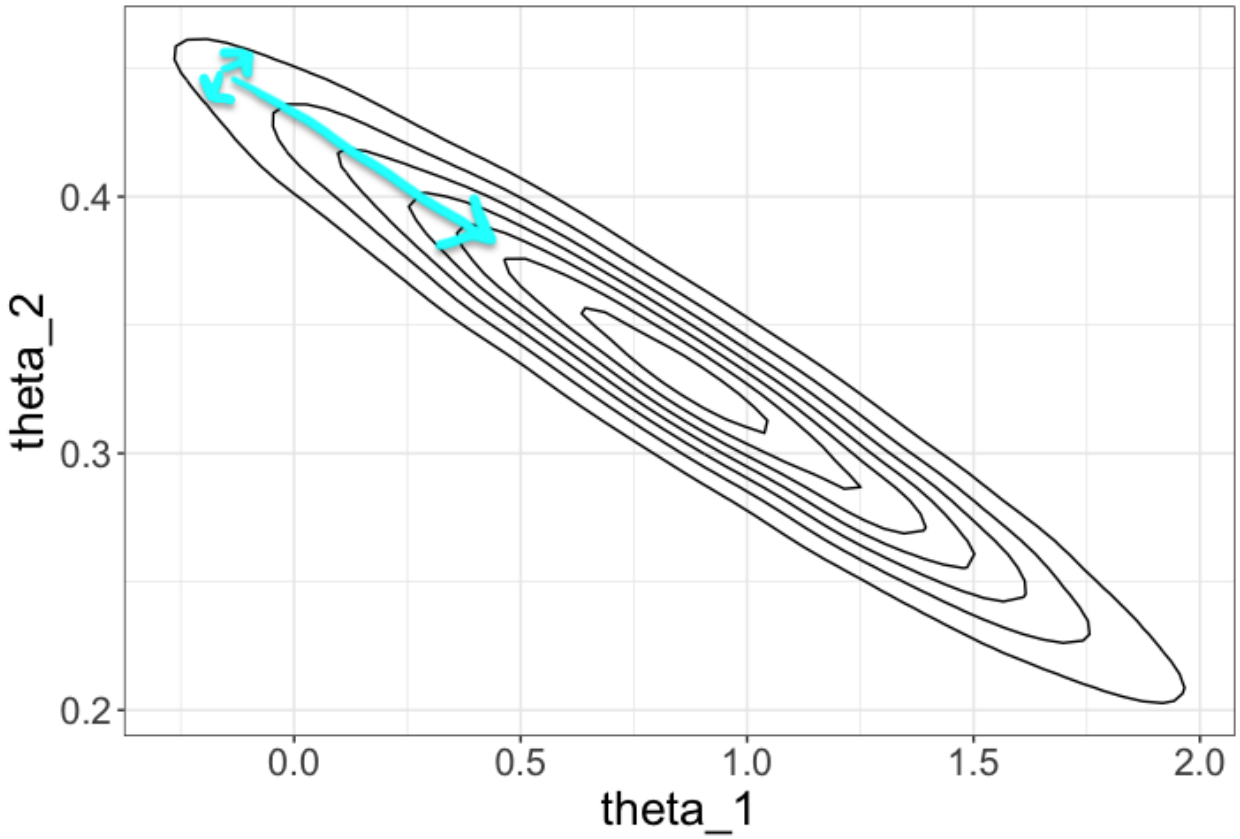
Geometric structure in the distribution



Geometric structure in the distribution



Geometric structure in the distribution



Hamiltonian Monte Carlo

- Treat the Markov chain as a physical particle, which evolves over $\mathbb{R}^D$.

Hamiltonian Monte Carlo

- Treat the Markov chain as a physical particle, which evolves over $\mathbb{R}^D$.
- Give the particle a random shove, by it with 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 $\mathbb{R}^D$.
- Give the particle a random shove, by it with a *momentum* $\xi_0 \in \mathbb{R}^D$,

$$\xi_0 \sim \text{normal}(0, M)$$

- Treat the negative log density as a physical *potential*,

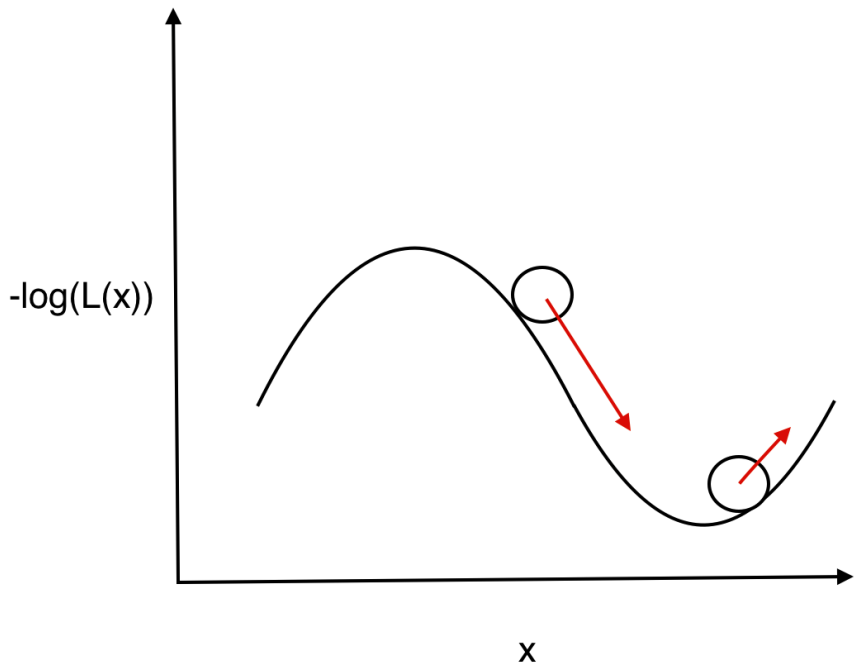
$$U(\theta) = -\log p(\theta \mid y).$$

- Simulate a the laws of classical mechanics for a time T,

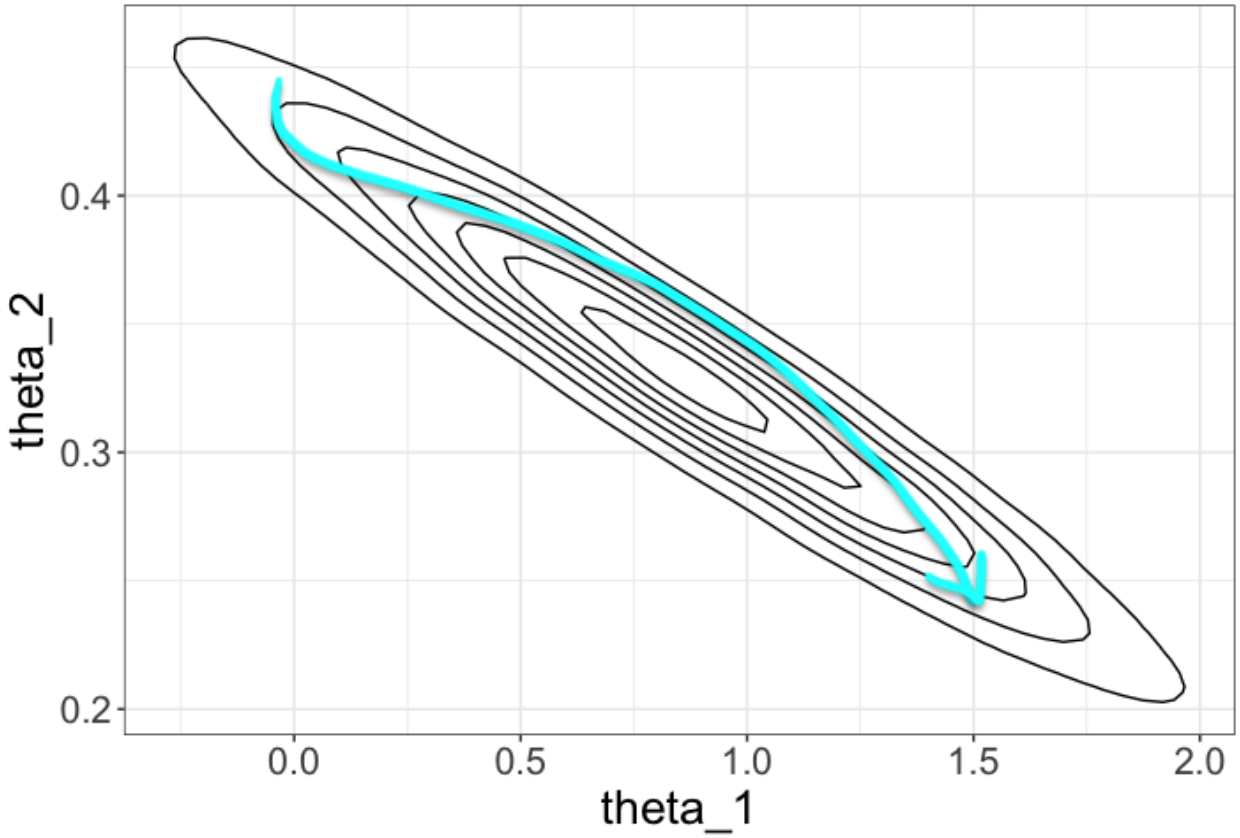
$$(\theta_0, \xi_0) \to (\theta_T, \xi_T).$$

Hamiltonian Dynamics

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



Geometric structure in the distribution



Canonical distribution (for stationary Markov chains),

$$
\begin{aligned}
p(\xi, \theta) &= p(\xi) p(\theta \mid y) \\
&\propto \exp\left\{ - \left(\frac{1}{2} \xi^T M^{-1} \xi - \log p(\theta \mid y) \right) \right\}
\end{aligned}
$$

Canonical distribution (for stationary Markov chains),

$$\begin{aligned} p(\xi, \theta) &= p(\xi) p(\theta \mid y) \\ &\propto \exp\left\{ - \left(\frac{1}{2} \xi^T M^{-1} \xi - \log p(\theta \mid y) \right) \right\} \end{aligned}$$

$H(\theta, \xi)$ is the energy (or *Hamiltonian* of the system) of the system,

$$H(\theta, \xi) = \underbrace{T(\xi)}_{\text{kinetic}} + \underbrace{U(\theta)}_{\text{potential}} .$$

Canonical distribution (for stationary Markov chains),

$$\begin{aligned} p(\xi, \theta) &= p(\xi) p(\theta \mid y) \\ &\propto \exp\left\{ - \left(\frac{1}{2} \xi^T M^{-1} \xi - \log p(\theta \mid y) \right) \right\} \end{aligned}$$

$H(\theta, \xi)$ is the energy (or *Hamiltonian* of the system) of the system,

$$H(\theta, \xi) = \underbrace{T(\xi)}_{\text{kinetic}} + \underbrace{U(\theta)}_{\text{potential}} \quad .$$

During a Hamiltonian trajectory, $H(\theta, \xi)$ stays constant.

Canonical distribution (for stationary Markov chains),

$$\begin{aligned} p(\xi, \theta) &= p(\xi) p(\theta \mid y) \\ &\propto \exp\left\{ - \left(\frac{1}{2} \xi^T M^{-1} \xi - \log p(\theta \mid y) \right) \right\} \end{aligned}$$

$H(\theta, \xi)$ is the energy (or *Hamiltonian* of the system) of the system,

$$H(\theta, \xi) = \underbrace{T(\xi)}_{\text{kinetic}} + \underbrace{U(\theta)}_{\text{potential}} \quad .$$

During a Hamiltonian trajectory, $H(\theta, \xi)$ stays constant.

HMC can be seen as a Gibbs sampler, alternating between a *random* step $p(H \mid \theta)$ and a *deterministic* step $\delta(\theta \mid H)$.

Algorithm 1: Leapfrog integrator for simulating Hamiltonian trajectory.

input: trajectory length L, step size ϵ, mass matrix M, $\theta(0)$

$\xi(0) \sim \text{normal}(0, M)$

$t \leftarrow 0$

for $i \in \{0, 1, \cdots, L-1\}$ **do**

$\xi(t + \epsilon/2) \leftarrow \xi(t) + \frac{\epsilon}{2} \nabla_\theta \log p(\theta(t) \mid y)$

$\theta(t + \epsilon) \leftarrow \theta(t) + \epsilon M^{-1} \xi(t + \epsilon/2)$

$\xi(t + \epsilon) \leftarrow \xi(t + \epsilon/2) - \frac{\epsilon}{2} \nabla_\theta p(\theta(t + \epsilon) \mid y)$

$t \leftarrow t + \epsilon$

end

return: $\theta(T = L\epsilon)$

Algorithm 2: Leapfrog integrator for simulating Hamiltonian trajectory.

input: trajectory length L, step size ϵ, mass matrix M, $\theta(0)$

$\xi(0) \sim \text{normal}(0, M)$

$t \leftarrow 0$

for $i \in \{0, 1, \cdots, L-1\}$ **do**

$\xi(t + \epsilon/2) \leftarrow \xi(t) + \frac{\epsilon}{2} \nabla_\theta \log p(\theta(t) \mid y)$

$\theta(t + \epsilon) \leftarrow \theta(t) + \epsilon M^{-1} \xi(t + \epsilon/2)$

$\xi(t + \epsilon) \leftarrow \xi(t + \epsilon/2) - \frac{\epsilon}{2} \nabla_\theta \log p(\theta(t + \epsilon) \mid y)$

$t \leftarrow t + \epsilon$

end

return: $\theta(T = L\epsilon)$

Tuning Hamiltonian Monte Carlo [Hoffman and Gelman, 2014]

Step size ϵ:

- Leapfrog computes inexact Hamiltonian trajectories.

Tuning Hamiltonian Monte Carlo [Hoffman and Gelman, 2014]

Step size ϵ:

- Leapfrog computes inexact Hamiltonian trajectories.
- This leads to *violation of energies*

$$H(\theta(0), \xi(0)) \neq H(\theta(T), \xi(T))$$

Tuning Hamiltonian Monte Carlo [Hoffman and Gelman, 2014]

Step size ϵ:

- Leapfrog computes inexact Hamiltonian trajectories.
- This leads to *violation of energies*

$$H(\theta(0), \xi(0)) \neq H(\theta(T), \xi(T))$$

- This motivates a Metropolis accept/reject step,*

$$\Pr(\text{accept}) = \min\left(1, \exp\left(H(\theta(0), \xi(0)) - H(\theta(T), \xi(T))\right)\right).$$

*`Stan` uses something a bit more sophisticated than a Metropolis step, see [Betancourt, 2017].

Tuning Hamiltonian Monte Carlo [Hoffman and Gelman, 2014]

Step size ϵ:

- Leapfrog computes inexact Hamiltonian trajectories.
- This leads to *violation of energies*

$$H(\theta(0), \xi(0)) \neq H(\theta(T), \xi(T))$$

- This motivates a Metropolis accept/reject step,*

$$\Pr(\text{accept}) = \min\left(1, \exp\left(H(\theta(0), \xi(0)) - H(\theta(T), \xi(T))\right)\right).$$

- `Stan` uses a stochastic optimization strategy to adjust ϵ and hit a target acceptance rate, `adapt_delta`,

$$\delta_{\text{adapt}} = 0.8.$$

*`Stan` uses something a bit more sophisticated than a Metropolis step, see [Betancourt, 2017].

Tuning Hamiltonian Monte Carlo [Hoffman and Gelman, 2014]

Trajectory length L:

- If L is too short, the Markov chain has a large autocorrelation.
- If L is too long, we use too much compute for little gain.

Tuning Hamiltonian Monte Carlo [Hoffman and Gelman, 2014]

Trajectory length L:

- If L is too short, the Markov chain has a large autocorrelation.
- If L is too long, we use too much compute for little gain.

- **Idea:** Maximize the squared jump distance

$$(\theta(T) - \theta(0))^T(\theta(T) - \theta(0)).$$

Tuning Hamiltonian Monte Carlo [Hoffman and Gelman, 2014]

Trajectory length L:

- If L is too short, the Markov chain has a large autocorrelation.
- If L is too long, we use too much compute for little gain.

- **Idea:** Maximize the squared jump distance

$$(\theta(T) - \theta(0))^T (\theta(T) - \theta(0)).$$

- *No U-Turn* criterion,

$$\frac{\mathrm{d}}{\mathrm{d}t} \frac{(\theta(T) - \theta(0))^T (\theta(T) - \theta(0))}{2} = (\theta(T) - \theta(0))^T \xi(T).$$

- Run simulation until $(\theta(T) - \theta(0))^T \xi(T) = 0$.

Tuning Hamiltonian Monte Carlo [Hoffman and Gelman, 2014]

Mass matrix M:

- The mass matrix determines the *resistance to acceleration*.

Tuning Hamiltonian Monte Carlo [Hoffman and Gelman, 2014]

Mass matrix M:

- The mass matrix determines the *resistance to acceleration*.
- **Idea:** set

$$M = \widehat{\Sigma}^{-1}(\theta),$$

where $\widehat{\Sigma}(\theta)$ is the sample covariance of the posterior.

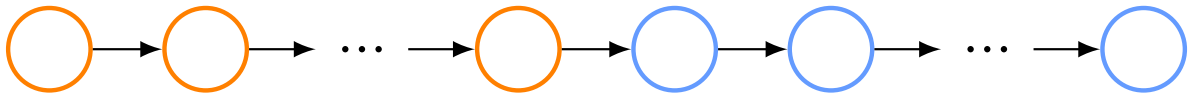
Tuning Hamiltonian Monte Carlo [Hoffman and Gelman, 2014]

Mass matrix M:

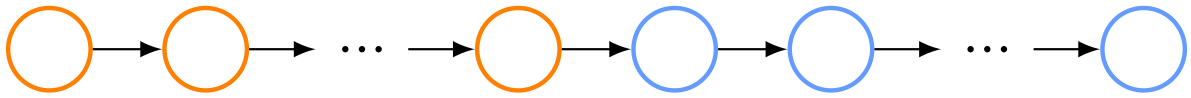
- The mass matrix determines the *resistance to acceleration*.
- **Idea:** set

$$M = \widehat{\Sigma}^{-1}(\theta),$$

  where $\widehat{\Sigma}(\theta)$ is the sample covariance of the posterior.
- To make the leapfrog cheaper, `Stan` uses a diagonal M.



Stan 's HMC learns ϵ and M during the warmup phase and then "freezes" these tuning parameters.



`Stan` 's HMC learns ϵ and M during the warmup phase and then "freezes" these tuning parameters.

The resulting algorithm is called the No U Turn Sampler (NUTS) or *dynamic Hamiltonian Monte Carlo*.

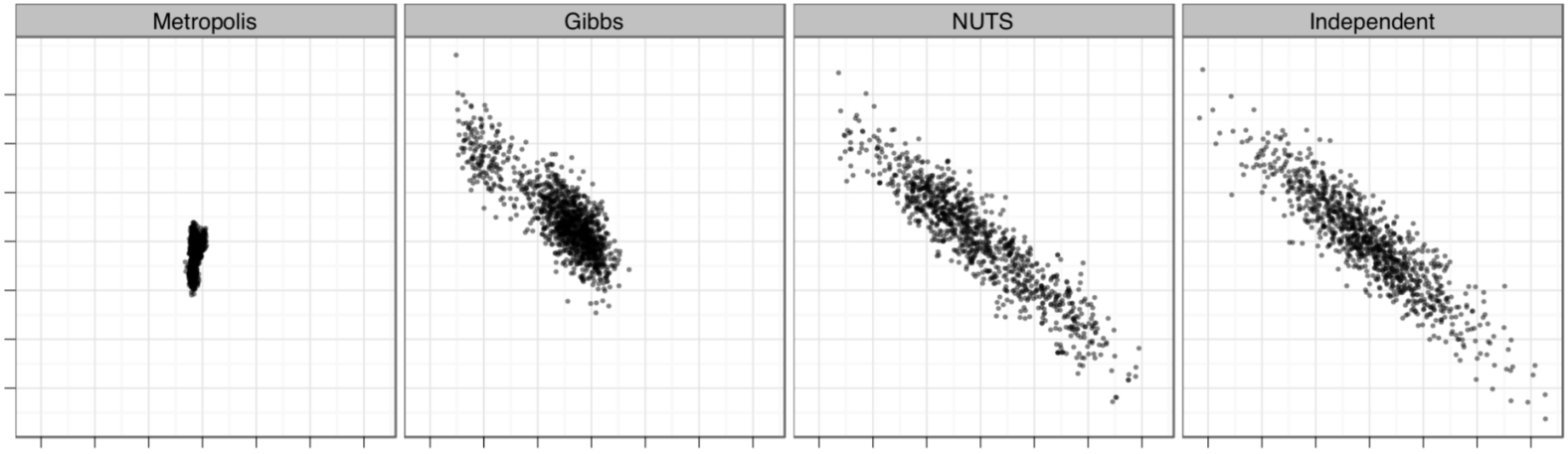


Figure from [Hoffman and Gelman, 2014].

To implement HMC, need

$$\begin{aligned}
\nabla_\theta \log p(\theta \mid y) &= \nabla_\theta \left(\log p(\theta) + \log p(y \mid \theta) - \log p(y) \right) \\
&= \nabla_\theta \left(\log p(\theta) + \log p(y \mid \theta) \right) .
\end{aligned}$$

To implement HMC, need

$$\begin{aligned}\nabla_\theta \log p(\theta \mid y) &= \nabla_\theta \left(\log p(\theta) + \log p(y \mid \theta) - \log p(y)\right) \\ &= \nabla_\theta \left(\log p(\theta) + \log p(y \mid \theta)\right).\end{aligned}$$

Stan implements *automatic differentiation*.

To implement HMC, need

$$\begin{aligned}\nabla_\theta \log p(\theta \mid y) &= \nabla_\theta \left(\log p(\theta) + \log p(y \mid \theta) - \log p(y)\right) \\ &= \nabla_\theta \left(\log p(\theta) + \log p(y \mid \theta)\right).\end{aligned}$$

`Stan` implements *automatic differentiation*.

Autodiff uses the chain rule to propagate derivatives through compositions of "analytical" functions; for some introductions, see [Baydin et al., 2018, Margossian, 2019].

Autodiff for implicit functions

Suppose our likelihood depends on an implicit function g,

$$\log p(y \mid \theta) = f \circ g(\theta).$$

Autodiff for implicit functions

Suppose our likelihood depends on an implicit function g,

$$\log p(y \mid \theta) = f \circ g(\theta).$$

E.g. g solves a differential equation.

$$\dot{g} = h(g, t, \theta).$$

Autodiff for implicit functions

Suppose our likelihood depends on an implicit function g,

$$\log p(y \mid \theta) = f \circ g(\theta).$$

E.g. g solves a differential equation.

$$\dot{g} = h(g, t, \theta).$$

Then

$$\nabla_\theta \log p(y \mid \theta) = \frac{\mathrm{d}f}{\mathrm{d}g} \cdot \frac{\mathrm{d}g}{\mathrm{d}\theta}.$$

...

$$\nabla_\theta \log p(y \mid \theta) = \frac{\mathrm{d}f}{\mathrm{d}g} \cdot \frac{\mathrm{d}g}{\mathrm{d}\theta}.$$

$\mathrm{d}g/\mathrm{d}\theta$ saves the augmented differential equation,

$$\frac{\mathrm{d}\dot{g}}{\mathrm{d}\theta} = \frac{\mathrm{d}}{\mathrm{d}\theta} h(g, \theta, t),$$

...

$$\nabla_\theta \log p(y \mid \theta) = \frac{\mathrm{d}f}{\mathrm{d}g} \cdot \frac{\mathrm{d}g}{\mathrm{d}\theta}.$$

$\mathrm{d}g/\mathrm{d}\theta$ saves the augmented differential equation,

$$\frac{\mathrm{d}\dot{g}}{\mathrm{d}\theta} = \frac{\mathrm{d}}{\mathrm{d}\theta} h(g, \theta, t),$$

...

$$\nabla_\theta \log p(y \mid \theta) = \frac{\mathrm{d}f}{\mathrm{d}g} \cdot \frac{\mathrm{d}g}{\mathrm{d}\theta}.$$

$\mathrm{d}g/\mathrm{d}\theta$ saves the augmented differential equation,

$$\frac{\mathrm{d}\dot{g}}{\mathrm{d}\theta} = \frac{\mathrm{d}}{\mathrm{d}\theta} h(g, \theta, t),$$

If $g \in \mathbb{R}^N$ and $\theta \in \mathbb{R}^K$, need to solve an ODE with $N + NK$ states.

...

$$\nabla_\theta \log p(y \mid \theta) = \frac{\mathrm{d}f}{\mathrm{d}g} \cdot \frac{\mathrm{d}g}{\mathrm{d}\theta}.$$

$\mathrm{d}g/\mathrm{d}\theta$ saves the augmented differential equation,

$$\frac{\mathrm{d}\dot{g}}{\mathrm{d}\theta} = \frac{\mathrm{d}}{\mathrm{d}\theta} h(g, \theta, t),$$

If $g \in \mathbb{R}^N$ and $\theta \in \mathbb{R}^K$, need to solve an ODE with $N + NK$ states.

Stan also supports an *adjoint method*, which is an augmented ODE with $2N + K$ states solved by $\mathrm{d}f/\mathrm{d}g \cdot \mathrm{d}g/\mathrm{d}\theta$ (but harder to use!).

...

$$\nabla_\theta \log p(y \mid \theta) = \frac{\mathrm{d}f}{\mathrm{d}g} \cdot \frac{\mathrm{d}g}{\mathrm{d}\theta}.$$

$\mathrm{d}g/\mathrm{d}\theta$ saves the augmented differential equation,

$$\frac{\mathrm{d}\dot{g}}{\mathrm{d}\theta} = \frac{\mathrm{d}}{\mathrm{d}\theta} h(g, \theta, t),$$

If $g \in \mathbb{R}^N$ and $\theta \in \mathbb{R}^K$, need to solve an ODE with $N + NK$ states.

Stan also supports an *adjoint method*, which is an augmented ODE with $2N + K$ states solved by $\mathrm{d}f/\mathrm{d}g \cdot \mathrm{d}g/\mathrm{d}\theta$ (but harder to use!).

For a discussion on autodiff for implicit functions, see [Margossian and Betancourt, 2022].

Computational cost of each `Stan` block

`data`: variable declaration.

`parameter`: variable declaration.

Computational cost of each `Stan` block

`data`: variable declaration.

`transformed data`: evaluated once.

`parameter`: variable declaration.

Computational cost of each `Stan` block

`data`: variable declaration.

`transformed data`: evaluated once.

`parameter`: variable declaration.

`transformed parameter`: evaluated and differentiated once per leapfrog step, i.e. multiple times per iteration.

`model`: evaluated and differentiated once per leapfrog step, i.e. multiple times per iteration.

Computational cost of each Stan block

`data`: variable declaration.

`transformed data`: evaluated once.

`parameter`: variable declaration.

`transformed parameter`: evaluated and differentiated once per leapfrog step, i.e. multiple times per iteration.

`model`: evaluated and differentiated once per leapfrog step, i.e. multiple times per iteration.

`generated quantities`: evaluated once per iteration during the sampling phase.

VII
Tuning ODEs in a Bayesian Context

It is possible in Stan to specify the tuning parameters of the ODE integrator.

```
array[n_days] vector[3] y
 = ode_rk45_tol(sir, y0, t0, ts,
                rel_tol, abs_tol, max_num_steps, ···)
```

It is possible in Stan to specify the tuning parameters of the ODE integrator.

```
array[n_days] vector[3] y
 = ode_rk45_tol(sir, y0, t0, ts,
                rel_tol, abs_tol, max_num_steps, ···)
```

- `rel_tol` or δ_{rel}: the relative tolerance to error.
- `abs_tol` or δ_{abs}: the absolute tolerance to error.
- `max_num_steps` : the maximum number of steps before the integrator "gives" up.

$\hat{u}$: numerical solution

$\hat{\epsilon}$: estimated error.

$\hat{u}$: numerical solution

$\hat{\epsilon}$: estimated error.

Solve adaptively reduces the step size until

$$\sqrt{\sum_{i}^{N} \frac{1}{N} \frac{\hat{\epsilon}_i^2}{(\delta_{\text{abs}} + \delta_{\text{rel}} \hat{u}_i)^2}} < 1.$$

$\hat{u}$: numerical solution

$\hat{\epsilon}$: estimated error.

Solve adaptively reduces the step size until

$$\sqrt{\sum_{i}^{N} \frac{1}{N} \frac{\hat{\epsilon}_i^2}{(\delta_{\text{abs}} + \delta_{\text{rel}} \hat{u}_i)^2}} < 1.$$

When $N = 1$, this is equivalent to

$$\hat{\epsilon}_1^2 < (\delta_{\text{abs}} + \delta_{\text{rel}} \hat{u}_1)^2.$$

$\hat{u}$: numerical solution

$\hat{\epsilon}$: estimated error.

Solve adaptively reduces the step size until

$$\sqrt{\sum_{i}^{N} \frac{1}{N} \frac{\hat{\epsilon}_i^2}{(\delta_{\text{abs}} + \delta_{\text{rel}} \hat{u}_i)^2}} < 1.$$

When $N = 1$, this is equivalent to

$$\hat{\epsilon}_1^2 < (\delta_{\text{abs}} + \delta_{\text{rel}} \hat{u}_1)^2.$$

Remark: In Stan we're also setting a tolerance for the augmented ODE system which propagates derivatives.

Exercise: *Fit an SIR model and specify tuning parameters for the ODE solvers.*

- *The default we used was* $\delta_{tol} = \delta_{rel} = 10^{-6}$. *You may experiment with either stricter or more lenient tolerances.*
- *Compare the runtime between the default tuning parameter and your choice using* `fit$time()`.
- *Compare the returned posterior for* β, γ *and* R_0.

Can we check if the tolerance is strict enough?

An importance sampling approach for reliable and efficient inference in Bayesian ordinary differential equation models

Juho Timonen[1], Nikolas Siccha[1], Ben Bales[2], Harri Lähdesmäki[1], and Aki Vehtari[1]

[1]Department of Computer Science, Aalto University, Finland
[2]Earth Institute, University of Columbia, New York, USA

In practice, MCMC does not target $p(\theta \mid y)$ but a numerical approximation

$$p_\delta(\theta \mid y) \propto p_\delta(y \mid \theta) p(\theta)$$

In practice, MCMC does not target $p(\theta \mid y)$ but a numerical approximation

$$p_\delta(\theta \mid y) \propto p_\delta(y \mid \theta) p(\theta)$$

To correct for the error, can use IS estimator

$$\widehat{\mathbb{E}}_{\mathrm{IS}} f(\theta) = \frac{\sum_{s=1}^{S} f(\theta^{(s)}) r(\theta^{(s)})}{\sum_{s=1}^{S} r(\theta^{(s)})},$$

where

$$r(\theta^{(s)}) = \frac{p(y \mid \theta^{(s)})}{p_\delta(y \mid \theta^{(s)})}.$$

In practice, MCMC does not target $p(\theta \mid y)$ but a numerical approximation

$$p_\delta(\theta \mid y) \propto p_\delta(y \mid \theta) p(\theta)$$

To correct for the error, can use IS estimator

$$\widehat{\mathbb{E}}_{\text{IS}} f(\theta) = \frac{\sum_{s=1}^{S} f(\theta^{(s)}) r(\theta^{(s)})}{\sum_{s=1}^{S} r(\theta^{(s)})},$$

where

$$r(\theta^{(s)}) = \frac{p(y \mid \theta^{(s)})}{p_\delta(y \mid \theta^{(s)})}.$$

Only feasible if

$$p_\delta(y \mid \theta^{(s)}) \approx p(y \mid \theta^{(s)}),$$

which we can check with PSIS and $\widehat{k}$.

$$\widehat{\mathbb{E}}_{\mathrm{IS}} f(\theta) = \frac{\sum_{s=1}^{S} f(\theta^{(s)}) r(\theta^{(s)})}{\sum_{s=1}^{S} r(\theta^{(s)})},$$

where

$$r(\theta^{(s)}) = \frac{p(y \mid \theta^{(s)})}{p_\delta(y \mid \theta^{(s)})}.$$

$$\widehat{\mathbb{E}}_{\mathrm{IS}} f(\theta) = \frac{\sum_{s=1}^{S} f(\theta^{(s)}) r(\theta^{(s)})}{\sum_{s=1}^{S} r(\theta^{(s)})},$$

where

$$r(\theta^{(s)}) = \frac{p(y \mid \theta^{(s)})}{p_\delta(y \mid \theta^{(s)})}.$$

Problem: we cannot compute $p(y \mid \theta^{(s)})$.

$$\widehat{\mathbb{E}}_{\mathrm{IS}} f(\theta) = \frac{\sum_{s=1}^{S} f(\theta^{(s)}) r(\theta^{(s)})}{\sum_{s=1}^{S} r(\theta^{(s)})},$$

where

$$r(\theta^{(s)}) = \frac{p(y \mid \theta^{(s)})}{p_\delta(y \mid \theta^{(s)})}.$$

Problem: we cannot compute $p(y \mid \theta^{(s)})$.

Instead we'll use a golden benchmark with $\delta^* \ll \delta$ and

$$r_{\delta,\delta^*}(\theta^{(s)}) = \frac{p_{\delta^*}(y \mid \theta^{(s)})}{p_\delta(y \mid \theta^{(s)})}.$$

$$\widehat{\mathbb{E}}_{\mathrm{IS}} f(\theta) = \frac{\sum_{s=1}^{S} f(\theta^{(s)}) r(\theta^{(s)})}{\sum_{s=1}^{S} r(\theta^{(s)})},$$

where

$$r(\theta^{(s)}) = \frac{p(y \mid \theta^{(s)})}{p_\delta(y \mid \theta^{(s)})}.$$

Problem: we cannot compute $p(y \mid \theta^{(s)})$.

Instead we'll use a golden benchmark with $\delta^* \ll \delta$ and

$$r_{\delta,\delta^*}(\theta^{(s)}) = \frac{p_{\delta^*}(y \mid \theta^{(s)})}{p_\delta(y \mid \theta^{(s)})}.$$

[Timonen et al., 2023] propose a strategy to find a suitable δ^*. We'll just stick to $\delta^* = 10^{-10}$.

Computational cost of each Stan block

`data`: variable declaration.

`transformed data`: evaluated once.

`parameter`: variable declaration.

`transformed parameter`: evaluated and differentiated once per leapfrog step, i.e. multiple times per iteration.

`model`: evaluated and differentiated once per leapfrog step, i.e. multiple times per iteration.

`generated quantities`: evaluated once per iteration during the sampling phase.

Exercise: *Check the tolerance of the ODE integrator in the SIR model.*

- *In* `generated quantities` *compute* `log_ratios` $= \sum_{i=1}^{N} \log p_{\delta^*}(y_i \mid \theta) - \log p_{\delta}(y_i \mid \theta)$.
- *Do a PSIS fit and check whether* $\widehat{k}$ *has an acceptable value. If applicable compute the IS estimator and compare to MCMC estimator.*
- *What are the least strict tolerances with which we still get accurate posterior estimates?*

Choices of ODE integrators in `Stan`

- `rk45`: Runge-Kutta $4^{\text{th}}/5^{\text{th}}$ order. Good place to start.
- `bdf`: Backward differentiation. Recommended for stiff systems.
- `adams`: Adams-Moulton solver – higher-order than rk45 and useful when a high precision is required for a very smooth solution.
- `ckrk`: a variant on rk45 for non-stiff and semi-stiff systems. Designed for problems where the solution evolves rapidly, where the derivatives becomes large.

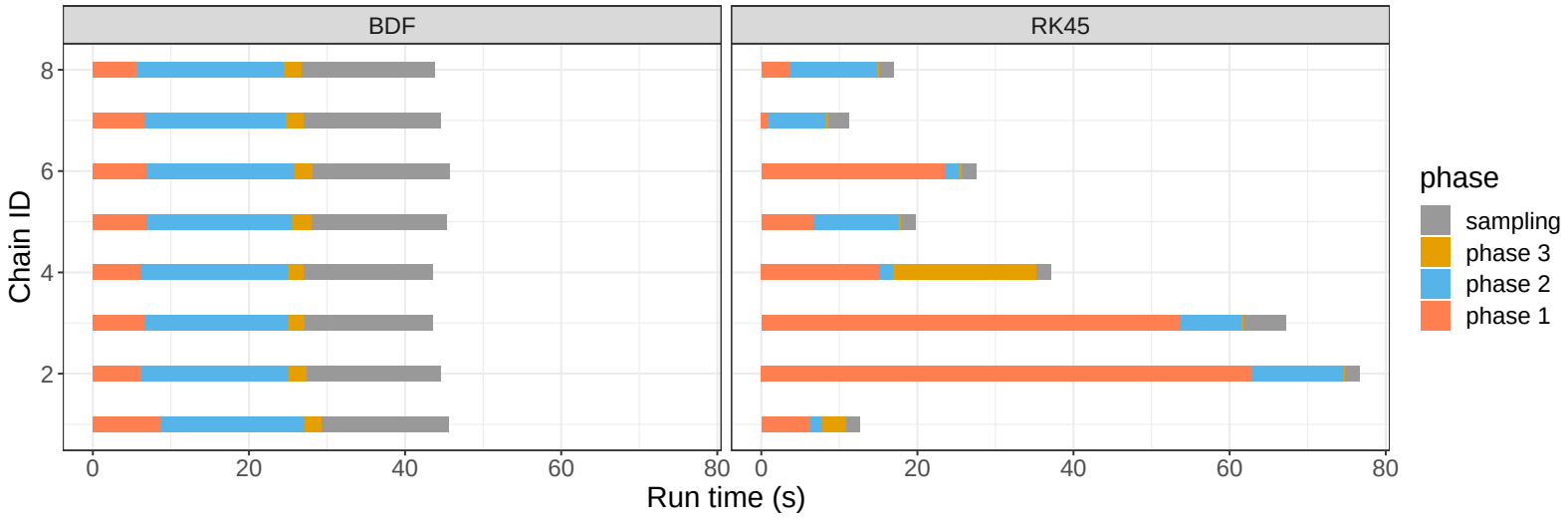
For more, see `https://mc-stan.org/docs/stan-users-guide/ode-solver.html`.

Case study: Michaelis-Menten pharmacokinetic model [Margossian et al., 2021]

- Which numerical integrator should we use in `Stan` ?
  - RK4$^{\text{th}}$/5$^{\text{th}}$ (non-stiff solver)
  - BDF (stiff solver)

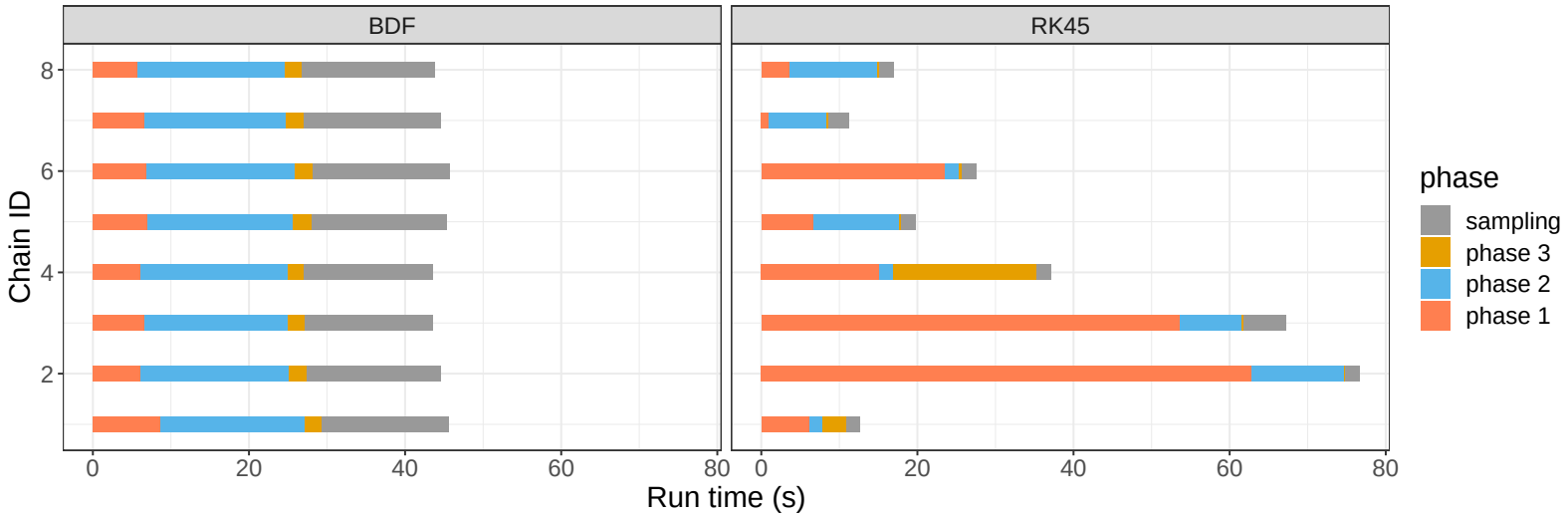
Case study: Michaelis-Menten pharmacokinetic model [Margossian et al., 2021]

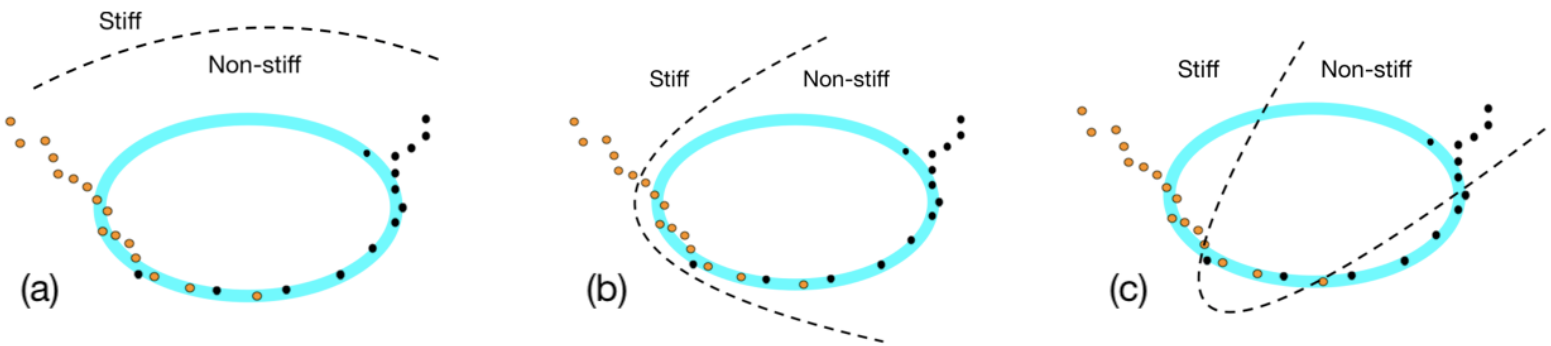
- Which numerical integrator should we use in `Stan` ?
  - RK4$^{\text{th}}$/5$^{\text{th}}$ (non-stiff solver)
  - BDF (stiff solver)



Case study: Michaelis-Menten pharmacokinetic model [Margossian et al., 2021]

- Which numerical integrator should we use in `Stan` ?
  - RK4$^{\text{th}}$/5$^{\text{th}}$ (non-stiff solver)
  - BDF (stiff solver)





Some ideas:

- Switch ODE during MCMC phases.

	Phase I	Phase II	Phase III	Sampling
RK45	RK45	RK45	RK45	RK45
BDF	BDF	BDF	BDF	BDF
Early switch	BDF	RK45	RK45	RK45
Late switch	BDF	BDF	RK45	RK45

Some ideas:

- Switch ODE during MCMC phases.

	Phase I	Phase II	Phase III	Sampling
RK45	RK45	RK45	RK45	RK45
BDF	BDF	BDF	BDF	BDF
Early switch	BDF	RK45	RK45	RK45
Late switch	BDF	BDF	RK45	RK45

- Use careful initializations, e.g. with fast approximation of $p(\theta \mid y)$ to bypass difficult regions.

VIII
Hierarchical Modeling

Suppose our data can be split into groups.

- medical measurements are grouped by patients
- sport measurements are grouped by players
- people's voting intention can be grouped by states, age group, etc.

Suppose our data can be split into groups.

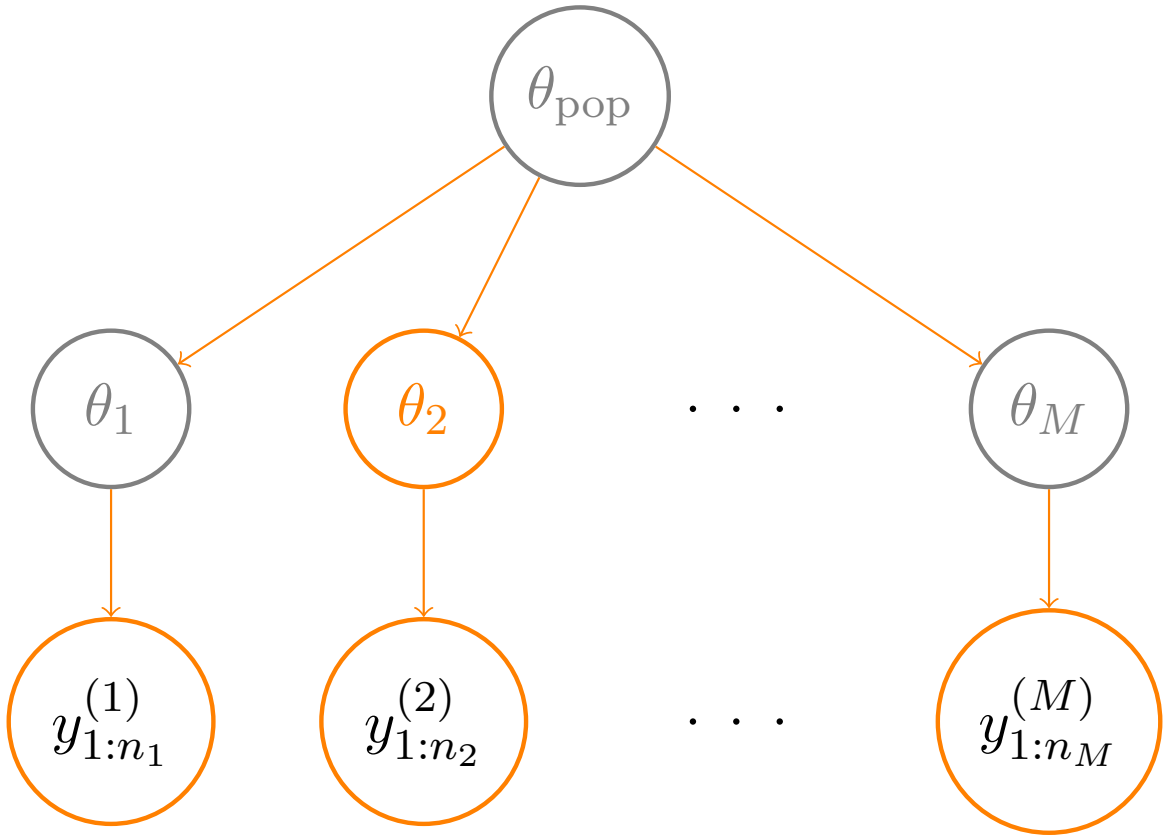
- medical measurements are grouped by patients
- sport measurements are grouped by players
- people's voting intention can be grouped by states, age group, etc.

With a hierarchical model, we can:

- model heterogeneity between groups
- estimate how similar groups are to one another.
- estimate "local" parameters using information from the entire population.

Hierarchical model

$$\begin{aligned} \theta_i &\sim p(\theta_i \mid \theta_{\text{pop}}) \\ y^{(i)}_{1:n_i} &\sim p(y^{(i)}_{1:n_i} \mid \theta_i) \end{aligned}$$



Example: 8 schools experiment [Gelman et al., 2013, Chapter 5]

How effective are prep programs for a standardized exam?

- y_i: estimated coaching effect for school i, based on student scores and covariate adjustments.
- σ_i: sampling standard deviation.
- θ_i: latent coaching effect for school i.
- μ: population level coaching effect
- τ: population standard deviation.

Example: 8 schools experiment [Gelman et al., 2013, Chapter 5]

How effective are prep programs for a standardized exam?

- y_i: estimated coaching effect for school i, based on student scores and covariate adjustments.
- σ_i: sampling standard deviation.
- θ_i: latent coaching effect for school i.
- μ: population level coaching effect
- τ: population standard deviation.

Generative model:

$$\begin{aligned} \mu &\sim \text{normal}(5, 3) \\ \tau &\sim \text{normal}^{+}(0, 10) \\ \theta_i &\sim \text{normal}(\mu, \tau) \\ y_i &\sim \text{normal}(\theta_i, \sigma_i) \end{aligned}$$

Exercise: *Write and fit the 8 schools model.*

- *Check the inference, i.e.* $\widehat{R}$ *and ESS.*
- *Report any warning messages.*
- *Record the estimated posterior and .9 coverage for* μ, τ *and* θ_1.

Divergent transitions

"There were 29 divergent transitions after warmup."

A divergent transition occurs when we fail to accurately compute a Hamiltonian trajectory, i.e. energy conservation is brutally violated.

Demo: Plot divergent transitions amongst MCMC draws.

In a hierarchical model, the joint prior $p(\tau, \theta)$ induces a funnel [Neal, 2003, Betancourt and Girolmi, 2015], which induces a high (sometimes non-finite) curvature.

Potential fixes

1. Increase the target acceptance rate of dynamic HMC.

   Forces the leapfrog integrator to use a smaller step size.
   `Stan` 's default is `adapt_delta` $= 0.8$.

   Exercise: Increase ***`adapt_delta`*** *and report results.*

Potential fixes

2. Use a non-centered parameterization.

   Consider the alternative data generative process,

$$
\begin{aligned}
\mu &\sim \text{normal}(5, 3) \\
\tau &\sim \text{normal}^{+}(0, 10) \\
\eta_i &\sim \text{normal}(0, 1) \\
\theta_i &= \mu + \tau \eta_i \\
y_i &\sim \text{normal}(\theta_i, \sigma_i)
\end{aligned}
$$

Potential fixes

2. Use a non-centered parameterization.

   Consider the alternative data generative process,

$$
\begin{aligned}
\mu &\sim \text{normal}(5, 3) \\
\tau &\sim \text{normal}^{+}(0, 10) \\
\eta_i &\sim \text{normal}(0, 1) \\
\theta_i &= \mu + \tau \eta_i \\
y_i &\sim \text{normal}(\theta_i, \sigma_i)
\end{aligned}
$$

   What structure do we expect from the joint prior $p(\tau, z)$?

Potential fixes

2. Use a non-centered parameterization.

   Consider the alternative data generative process,

$$
\begin{aligned}
\mu &\sim \text{normal}(5, 3) \\
\tau &\sim \text{normal}^{+}(0, 10) \\
\eta_i &\sim \text{normal}(0, 1) \\
\theta_i &= \mu + \tau \eta_i \\
y_i &\sim \text{normal}(\theta_i, \sigma_i)
\end{aligned}
$$

   What structure do we expect from the joint prior $p(\tau, z)$?

   Exercise: Implement a non-centered parameterization of the 8 schools model and report results.

Potential fixes

3. Marginalize out the local variable θ.
   - Use MCMC to sample from the marginal posterior,

$$p(\mu, \tau \mid y) \propto p(\mu)p(\tau)p(y \mid \mu, \tau).$$

   - Then recover θ by sampling from the conditional

$$\theta \sim p(\theta \mid \mu, \tau, y).$$

Potential fixes

3. Marginalize out the local variable θ.
   - Use MCMC to sample from the marginal posterior,

$$p(\mu, \tau \mid y) \propto p(\mu)p(\tau)p(y \mid \mu, \tau).$$

   - Then recover θ by sampling from the conditional

$$\theta \sim p(\theta \mid \mu, \tau, y).$$

   - This strategy works here because the marginal likelihood and the conditional admit analytical expressions.

$$\begin{aligned} p(y_i \mid \mu, \tau) &= \text{normal}\left(\mu, \sqrt{\tau^2 + \sigma_i^2}\right) \\ p(\theta_i \mid \mu, \tau, y) &= \text{normal}\left(\frac{y_i/\sigma_i^2 + \mu/\tau^2}{1/\sigma_i^2 + 1/\tau^2}, \sqrt{\frac{1}{1/\sigma_i^2 + 1/\tau^2}}\right) \end{aligned}$$

Potential fixes

3. Marginalize out the local variable θ.
   - Use MCMC to sample from the marginal posterior,

$$p(\mu, \tau \mid y) \propto p(\mu)p(\tau)p(y \mid \mu, \tau).$$

   - Then recover θ by sampling from the conditional

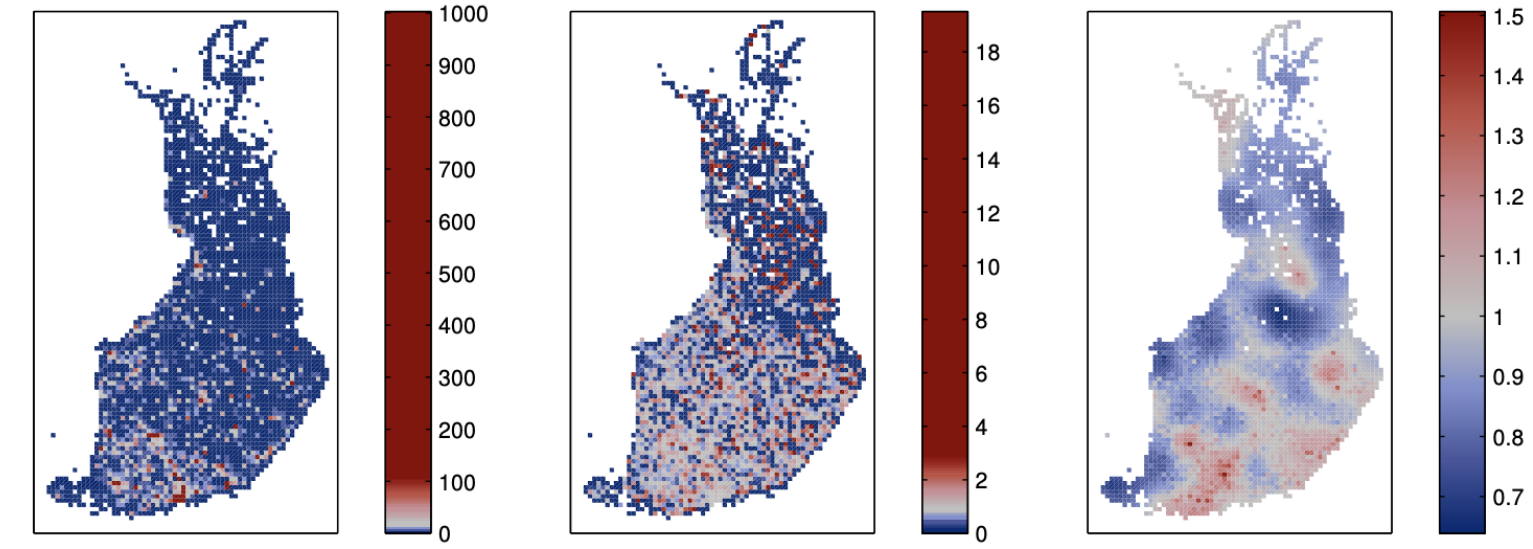
$$\theta \sim p(\theta \mid \mu, \tau, y).$$

   - This strategy works here because the marginal likelihood and the conditional admit analytical expressions.

$$\begin{aligned} p(y_i \mid \mu, \tau) &= \text{normal}\left(\mu, \sqrt{\tau^2 + \sigma_i^2}\right) \\ p(\theta_i \mid \mu, \tau, y) &= \text{normal}\left(\frac{y_i/\sigma_i^2 + \mu/\tau^2}{1/\sigma_i^2 + 1/\tau^2}, \sqrt{\frac{1}{1/\sigma_i^2 + 1/\tau^2}}\right) \end{aligned}$$

Exercise: *Write and fit a* `Stan` *model that samples from the marginal posterior and use* `generated quantities` *to recover draws for* θ*. Compare your results to previous strategies.*

Disease map of Finland [Vanhatalo et al., 2010]



(g) Number of deaths (h) Raw relative risk (i) Smoothed risk

Mortality count due to alcoholism across the country

- The country is split into 911 cells and all cells have the same area.
- In most cells, the population is sparse.

Disease map of Finland [Vanhatalo et al., 2010]

- The death count in cell i is

$$y_i \sim \text{Poisson}\left(y_e^i \exp(\theta_i)\right),$$

  where y_e^i is the standardized expected number of deaths, based on covariates, and $\exp(\theta_i)$ is the (relative) risk.
- Moreover $y_e^i \exp(\theta_i)$ is the expected number of deaths.

Disease map of Finland [Vanhatalo et al., 2010]

- The death count in cell i is

$$y_i \sim \text{Poisson}\left(y_e^i \exp(\theta_i)\right),$$

  where y_e^i is the standardized expected number of deaths, based on covariates, and $\exp(\theta_i)$ is the (relative) risk.
- Moreover $y_e^i \exp(\theta_i)$ is the expected number of deaths.
- Expect similar risks in neighboring counties,

$$\boldsymbol{\theta} \sim \text{Normal}\left(0, K(\alpha, \rho)\right).$$

- The covariance between θ_i and θ_j is

$$K_{ij} = \alpha^2 \exp\left(-\frac{||x_i - x_j||^2}{2\rho^2}\right),$$

  where x_i is the 2D location of cell i.

Disease map of Finland [Vanhatalo et al., 2010]
Full model:

$$
\begin{aligned}
\alpha &\sim \text{invGamma}(10, 10) \\
\rho &\sim \text{invGamma}(2.42, 14.8) \\
\boldsymbol{\theta} &\sim \text{Normal}(0, K(\alpha, \rho)) \\
y_i &\sim \text{Poisson}\left(y_e^i \exp(\theta_i)\right)
\end{aligned}
$$

Disease map of Finland [Vanhatalo et al., 2010]
Full model:

$$
\begin{aligned}
\alpha &\sim \text{invGamma}(10, 10) \\
\rho &\sim \text{invGamma}(2.42, 14.8) \\
\boldsymbol{\theta} &\sim \text{Normal}(0, K(\alpha, \rho)) \\
y_i &\sim \text{Poisson}\left(y_e^i \exp(\theta_i)\right)
\end{aligned}
$$

Tipp:

- This model admits a non-centered parameterization, with

$$\boldsymbol{\eta} \sim \text{normal}(0, I) \; ; \; \boldsymbol{\theta} = L\eta,$$

  where L is the *Cholesky decomposition* of K, that is $K = LL^T$ and L is lower-triangular.
- For numerical stability, can add a "jitter" $\epsilon = 10^{-8}$ along the diagonal of K, to make sure eigenvalues are positive.

Exercise: *Fit the disease map model.*

- *For convenience, we only examine 100 cells.*
- *Make sure there are no divergent transitions.*
- *Examine* $\widehat{R}$*, ESS, and the trace plots for* α*,* ρ *and* θ_1*.*

Tips:

- *The type for* `x` *is* `array[n_obs] vector[n_coordinates]`*.*
- *The following* `Stan` *functions may come in handy:*
  - `gp_exp_quad_cov(x, alpha, rho)`
  - `cholesky_decompose(Sigma)`
  - `inv_gamma()`
  - `poisson_log()` *(but ok to use* `poisson()`*)*

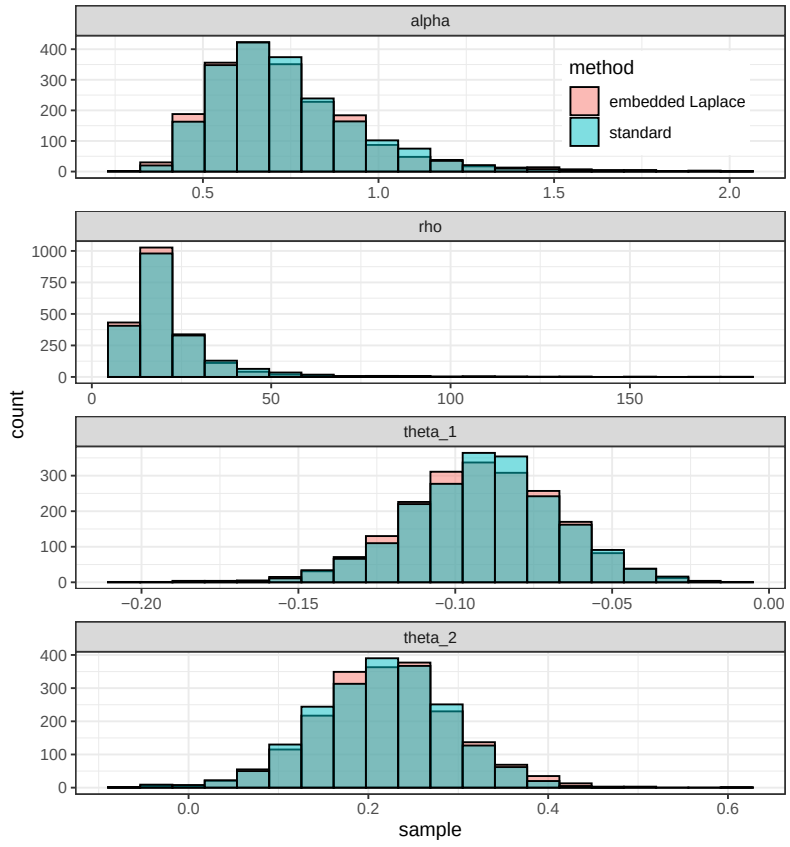
Can we marginalize out θ when $p(\boldsymbol{y} \mid \boldsymbol{\theta})$ is non-Gaussian?

- Can do a *Laplace approximation*,

$$\text{normal}(\mu^*, \Sigma^*) \approx p(\boldsymbol{\theta} \mid \boldsymbol{y}, \phi),$$

  where μ^* matches the mode of $p(\boldsymbol{\theta} \mid \boldsymbol{y}, \phi)$ and Σ^* its curvature.
- This also gives us an approximation for $p(\mathbf{y} \mid \phi)$.
- This is the driving idea behind the *integrated Laplace approximation* [Rue et al., 2009].

Stan supports a prototype integrated Laplace approximation [Margossian et al., 2020, Margossian et al., 2023].



For this application, integrated Laplace approximation is ∼10 times faster and does not require adjusting `adapt_delta`.

- `Stan` 's integrated Laplace approximation is a prototype.
- Works well for standard likelihoods and general linear models.

- Stan 's integrated Laplace approximation is a prototype.
- Works well for standard likelihoods and general linear models.

- The *adjoint differentiated Laplace approximation* [Margossian et al., 2020, Margossian, 2023] allows users to specify their own covariance function K and likelihood, rather than picking from a menu of options.
- The underlying autodiff method to compute $\nabla_\phi \log p_{\mathcal{G}}(\mathbf{y} \mid \phi)$ scales when ϕ is high-dimensional.
- **Ongoing work:** diagnostics to check if approximation is reliable.

- For more, see https://htmlpreview.github.io/?https://github.com/charlesm93/StanCon2020/blob/master/notebook-2022/lgm_stan.html

Strategies to deal with the geometry of hierarchical models:

- Increase the target acceptance probability, i.e. reduce the step size of the leap frog integrator.
- Use a non-centered parameterization.
- Marginalize out the local variables.

Strategies to deal with the geometry of hierarchical models:

- Increase the target acceptance probability, i.e. reduce the step size of the leap frog integrator.
- Use a non-centered parameterization.
- Marginalize out the local variables.

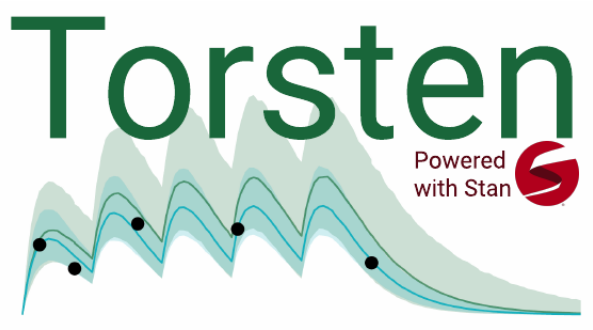
- Riemannian HMC: evaluate a dynamic mass matrix based on the local curvature [Girolami et al., 2011].
- Delayed rejection HMC: reduce step size after rejection [Modi et al., 2023].

IX
Torsten

`Torsten` offers additional built-in functions to write pharmacokinetics/pharmacodynamics (PK/PD) models

Each Torsten function requires users to specify:

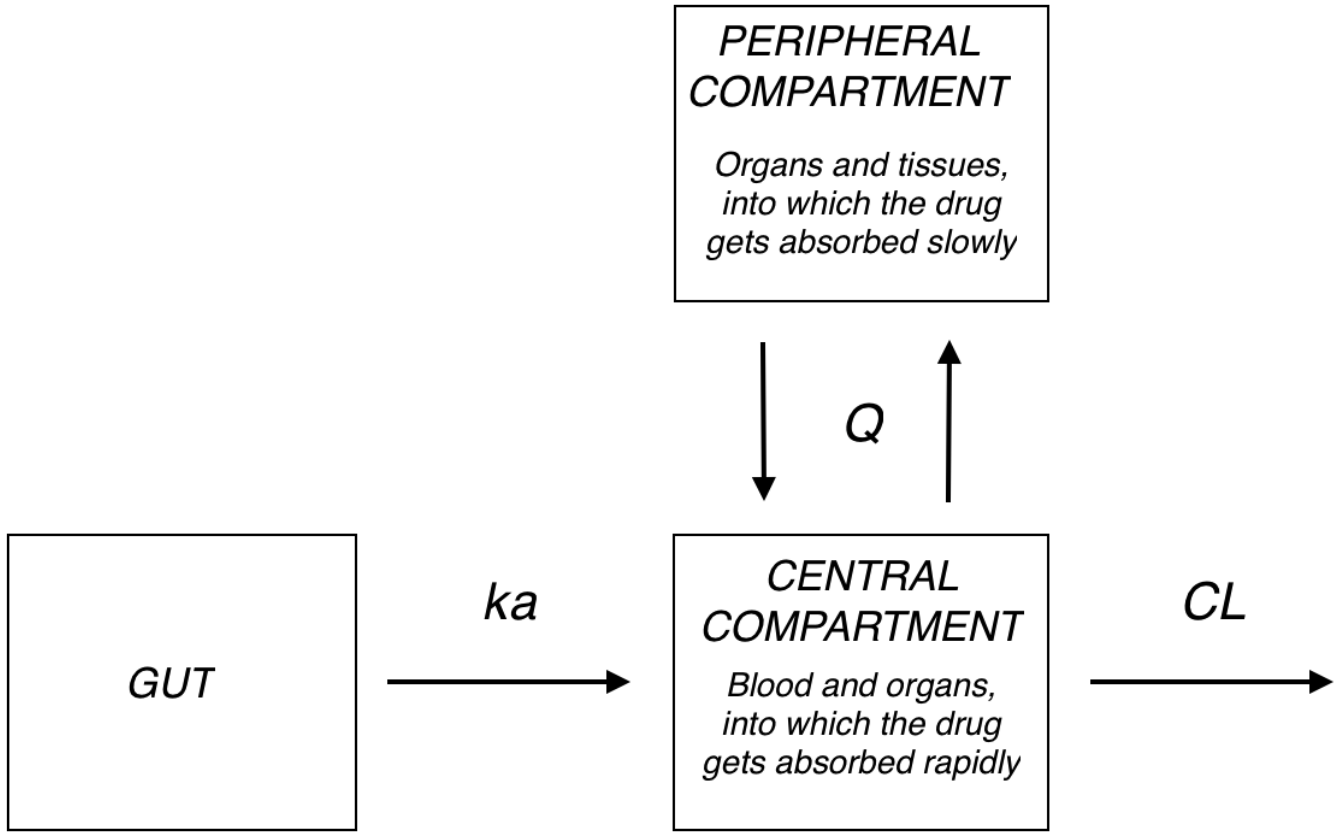
- A system of ODEs and a method to solve it.
- An event schedule, following the `PREDPP` convention from `NONMEM`



Helpful references:

- User manual:
  `https://metrumresearchgroup.github.io/Torsten/`
- Tutorial in *CPT: P&SP* [Margossian et al., 2022]

Two compartment model with absorption from the gut



Two compartment model with absorption from the gut

$$
\begin{aligned}
y'_{\text{gut}} &= -k_a y_{\text{gut}} \\
y'_{\text{cent}} &= k_a y_{\text{gut}} - \left(\frac{CL}{V_{\text{cent}}} + \frac{Q}{V_{\text{cent}}} \right) y_{\text{cent}} + \frac{Q}{V_{\text{peri}}} y_{\text{peri}} \\
y'_{\text{peri}} &= \frac{Q}{V_{\text{cent}}} y_{\text{cent}} - \frac{Q}{V_{\text{peri}}} y_{\text{peri}}
\end{aligned}
$$

Two compartment model with a absorption from the gut

Denote $\theta = \{CL, Q, VC, VP, k_a\}$, the ODE parameters. Then

$$y' = f(y, t, \theta)$$

Given an initial condition $y_0 = y(t_0)$, solving the above ODE gives us the *natural evolution* of the system at any given time point.

The event schedule

An event can be a(n):

- Sate changer: an (exterior) intervention that alters the state of the system; for example a bolus dosing or the beginning of an infusion.
- Observation: measurement of a quantity of interest at a certain time.

Example: single patient model

Event schedule:

- Bolus doses with 1200 mg, administered every 12 hours, for a total of 15 doses.
- Many observations for the first, second, and last doses
- Additional observation every 12 hours

The observation are plasma drug concentration measurement.

See `data/twoCpt.data.json`.

Torsten function

```
matrix pmx_solve_rk45(function system, int nCmt,
                      real[] time, real[] amt,
                      real[] rate, real[] ii,
                      real[] evid, int[] cmt,
                      int[] addl, int[] ss,
                      real[] theta,
                      real rel_tol, real abs_tol,
                      int max_num_steps)
```

- Returns a `matrix[nCmt, nEvent]` with the drug mass in each compartment at each event
- Takes in:
  - an ODE to solve, which takes in `theta`.
  - an event schedule
  - parameters for the ODEs
  - (optional) tuning parameters for ODEs

System function

Declare system in the `functions` block.

```
 vector system(real time,
               vector y,
               real[] theta,
               real[] x_r,
               int[] x_i) {
  array[3] real dydt;
  real CL = theta[1];
  real Q = theta[2];
  ⋮
  return dydt;
}
```

Remark: Torsten uses an older API for the ODE integrator, meaning f must follow a stricter signature (although not a less flexible one).

Prior:

$$\begin{aligned} CL &\sim \text{logNormal}(\log 10, 0.25) \\ Q &\sim \text{logNormal}(\log 15, 0.5) \\ VC &\sim \text{logNormal}(\log 35, 0.25) \\ VP &\sim \text{logNormal}(\log 105, 0.5) \\ ka &\sim \text{logNormal}(\log 2.5, 1) \\ \sigma &\sim \text{Normal}^{+}(0, 1) \end{aligned}$$

Likelihood:

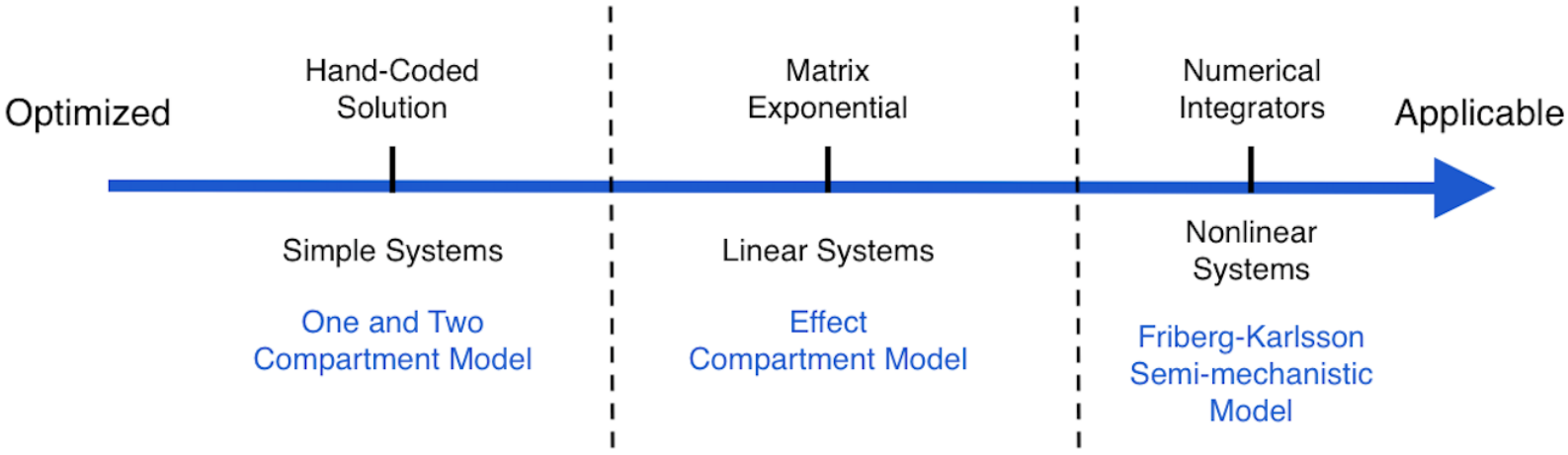
$$cObs \sim \text{logNormal}\left(\log\left(\frac{y_2}{VC}\right), \sigma\right)$$

Exercice: write, fit, and criticize the two compartment model for a single patient.

- *There's a bit of bookkeeping involved, so we'll write the* `data` *block together.*
- *As always, check the inference:* $\widehat{R}$*, ESS, and plots.*
- *Perform posterior predictive checks: do the simulations capture the characteristics in the data that we care about?*

- *Specify the control parameters of the ODE integrator and implement a PSIS diagnostic to check that the solver is sufficiently precise.*

Torsten supports alternatives to numerical integrators.



- It possible to combine multiple methods, e.g. solve the PK analytically and the PD numerically [Margossian and Gillespie, 2017].

Torsten function

```
matrix pmx_solve_twocpt(real[] time, real[] amt,
                        real[] rate, real[] ii,
                        real[] evid, int[] cmt,
                        int[] addl, int[] ss,
                        real[] theta)
```

- Returns a `matrix[nCmt, nEvent]` with the drug mass in each compartment at each event
- Takes in:
  - an event schedule
  - parameters for the ODEs

Analytical solutions are also available for the one compartment model.

```
matrix pmx_solve_onecpt(real[] time, real[] amt,
                        real[] rate, real[] ii,
                        real[] evid, int[] cmt,
                        int[] addl, int[] ss,
                        real[] theta)
```

- This time $\theta = (k_a, CL, V_{\text{cent}})$.

Exercise: Fit the one and two compartment models using analytical solutions.

- *Compare the posterior predictive checks obtain with each model.*
- *Estimate the loo-CV predictive score of both models.*

X
Population models

Usuaully we have multiple patients in our clinical trial.

With a hierarchical model, we can:

- estimate parameters for each patient,
- estimate population parameters and simulate new patients.

Population two compartment model

For the i^{th} patient, estimate

$$\theta_i = (\text{CL}_i, \text{Q}_i, \text{VC}_i, \text{VP}_i, k_{a_i})$$

Hierarchical prior:

$$\log \theta_i \sim \text{Normal}(\log \theta_{\text{pop}}, \Omega)$$

$$\Omega = \begin{pmatrix} \omega_1 & 0 & 0 & 0 & 0 \\ 0 & \omega_2 & 0 & 0 & 0 \\ 0 & 0 & \omega_3 & 0 & 0 \\ 0 & 0 & 0 & \omega_4 & 0 \\ 0 & 0 & 0 & 0 & \omega_5 \end{pmatrix}$$

Population two compartment model

Priors:

$$
\begin{aligned}
CL_{\text{pop}} &\sim \text{logNormal}(\log(10), 0.25) \\
Q_{\text{pop}} &\sim \text{logNormal}(\log(15), 0.5) \\
VC_{\text{pop}} &\sim \text{logNormal}(\log(35), 0.25) \\
VP_{\text{pop}} &\sim \text{logNormal}(\log(105), 0.5) \\
ka_{\text{pop}} &\sim \text{logNormal}(\log(2.5), 0.25) \\
\sigma &\sim \text{normal}^{+}(0, 1) \\
\omega_j &\sim \text{normal}^{+}(0, 0.2)
\end{aligned}
$$

Likelihood:

$$
\text{cObs} \sim \text{normal}\left(\log\left(\frac{y_2}{VC}\right), \sigma\right)
$$

Helpful bookkeeping

We now need to define parameters for each patient:

- `real theta[nSubjects, nTheta];`

We sequentially compute the concentration for each patient:

- `for (j in 1:nSubjects) {...}`

The `start` and `end` variables tell us which events belong to each patient. For the j^{th} patient, we need:

- `time[start[j]:end[j]], amt[start[j]:end[j]], ...`
- `theta[j, ]`

Helpful bookkeeping

We now need to define parameters for each patient:

- `real theta[nSubjects, nTheta];`

We sequentially compute the concentration for each patient:

- `for (j in 1:nSubjects) {...}`

The `start` and `end` variables tell us which events belong to each patient. For the j^{th} patient, we need:

- `time[start[j]:end[j]], amt[start[j]:end[j]], ...`
- `theta[j, ]`
- Such a `for` loop can be parallelized.

Exercise: *Build, fit, and criticize a hierarchical two compartment model.*

- *Make sure the inference is reliable. There should be no divergent transitions, $\widehat{R}$ should be close to 1 for all variables of interest, and the ESS sufficiently large.*
- *Perform posterior predictive checks:*
  - *Simulate data for existing patients.*
  - *Simulate data for new patients, drawn from the population distribution.*

Within-chain parallelization

Not every operation in Stan needs to be computed sequentially.

When there is conditional independence,

$$
\begin{aligned}
& \log p(\mathbf{y} \mid \theta) \\
= & \sum_{n=1}^{N} \log p(y_n \mid \theta) \\
= & \left(\sum_{n=1}^{I} \log p(y_n \mid \theta) \right) + \left(\sum_{n=I+1}^{J} \log p(y_n \mid \theta) \right) + \cdots
\end{aligned}
$$

and each sub-sum can be computed in parallel.

Within-chain parallelization

Can use the function

```
reduce_sum(F f, array[] T x, int grain_size, ···)
```

Within-chain parallelization

Can use the function

```
reduce_sum(F f, array[] T x, int grain_size, ···)
```

- `f` computes a partial sum, given a subset of `x`.

  Each thread may compute more than one term in the sum.

Within-chain parallelization

Can use the function

```
reduce_sum(F f, array[] T x, int grain_size, ···)
```

- `f` computes a partial sum, given a subset of `x`.

  Each thread may compute more than one term in the sum.

- `grain_size` is the recommended number of terms in the sum computed on each thread.

Within-chain parallelization

Can use the function

```
reduce_sum(F f, array[] T x, int grain_size, ···)
```

- `f` computes a partial sum, given a subset of `x`.

  Each thread may compute more than one term in the sum.

- `grain_size` is the recommended number of terms in the sum computed on each thread.
- $\cdots$ additional arguments passed to all subsums.

- The partial sum has the following signature

```
f (array[] int x,
   int start_subject, int end_subject, ···)
```

- The partial sum has the following signature

```
f (array[] int x,
   int start_subject, int end_subject, ···)
```

- We may pick `x` to be the index of the subject,

```
x = (1, 2, 3, 4, ···, n_subject)
```

- Then `start_subject` indexes the first subject and `end_subject` the last subject in the partial sum.

- The partial sum has the following signature

```
f (array[] int x,
   int start_subject, int end_subject, ···)
```

- We may pick `x` to be the index of the subject,

```
x = (1, 2, 3, 4, ···, n_subject)
```

- Then `start_subject` indexes the first subject and `end_subject` the last subject in the partial sum.
- For more guidance, see https://mc-stan.org/docs/stan-users-guide/reduce-sum.html.

Need to change our R script to enable multi-threading per chain:

```
mod <- cmdstan_model("model/twoCptPop_rs.stan"),
          cpp_options = list(stan_threads = TRUE))
```

```
n_chains <- 1
fit_rs <- mod$sample(data = data, chains = n_chains,
             init = init,
             parallel_chains = n_chains,
             threads_per_chain = 3,
             iter_warmup = 500, iter_sampling = 500,
             seed = 123, adapt_delta = 0.8)
```

Exercise: *Write the two compartment model using* `reduce_sum` *to parallelize the solving of the ODE and the evaluation of the log likelihood across patients.*

- *Make sure your posterior estimate is consistent with the previous model.*
- *Try running 1, 2, 3, 4+ threads per chain and examine the run time for a single chain.*

Torsten supports functions to do solve ODEs across patients using multiple threads,

`pmx_solve_ode_group_*`

- See https://metrumresearchgroup.github.io/Torsten/function/ode-group-integ/

XI
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, 2017]
- 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]

What do `Stan` and `Torsten` bring to the table?

What do `Stan` and `Torsten` bring to the table?

- A flexible and expressive language, with (in my view) the best user interface amongst probabilistic programming languages for specifying a model.

What do `Stan` and `Torsten` bring to the table?

- A flexible and expressive language, with (in my view) the best user interface amongst probabilistic programming languages for specifying a model.
- Algorithms that are efficient for full Bayesian inference, and that warn you when they fail.
- Many automatically deployed diagnostic tools.

What do `Stan` and `Torsten` bring to the table?

- A flexible and expressive language, with (in my view) the best user interface amongst probabilistic programming languages for specifying a model.
- Algorithms that are efficient for full Bayesian inference, and that warn you when they fail.
- Many automatically deployed diagnostic tools.
- Reasonable support for parallelization across cores and GPUs (other languages are better in some settings).

What do `Stan` and `Torsten` bring to the table?

- A flexible and expressive language, with (in my view) the best user interface amongst probabilistic programming languages for specifying a model.
- Algorithms that are efficient for full Bayesian inference, and that warn you when they fail.
- Many automatically deployed diagnostic tools.
- Reasonable support for parallelization across cores and GPUs (other languages are better in some settings).
- It's free and open-source.

Our goals:

- More expressive features.
- Improved computation for large systems of ODEs.
- Algorithms for fast approximate Bayesian inference
- BridgeStan: a package that makes it easy for users to specify their own inference algorithm and run them on Stan models (`https://github.com/roualdes/bridgestan`).

Stan by the people, for the people

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

Other probabilistic programming languages out there!

- `PyMC`:
  - Written in `Python`
- `Turing`
  - Written in `Julia`
  - Very clean autodiff and good support for ODE solvers.
- `TensorFlow Probability`
  - Interfaces with `JAX` and designed to work on GPUs.
  - Hackable inference algorithms.
  - Supports GPU-friendly samplers.
- `PyTorch`
  - Designed to work on GPUs.
  - Support for neural networks and optimization algorithms.

References I

[Baydin et al., 2018] Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2018).
Automatic differentiation in machine learning: a survey.
Journal of Machine Learning Research, 18:1 – 43.

[Betancourt, 2017] Betancourt, M. (2017).
A conceptual introduction to Hamiltonian Monte Carlo.
arXiv:1701.02434v1.

[Betancourt and Girolmi, 2015] Betancourt, M. and Girolmi, M. (2015).
Hamiltonian Monte Carlo for hierarchical models.
Current trends in Bayesian methodology with applications, 79.

[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.

References II

[Girolami et al., 2011] Girolami, M., Calderhead, B., and Chin, S. A. (2011).
Riemann manifold Langevin and Hamiltonian Monte Carlo methods.
Journal of the Royal Statistical Society, Series B, pages 123 – 214.

[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, 2019] Margossian, C. C. (2019).
A review of automatic differentiation and its efficient implementation.
WIREs Data Mining and Knowledge.

References III

[Margossian, 2023] Margossian, C. C. (2023).
General adjoint-differentiated Laplace approximation.
arXiv:2306.14976.

[Margossian and Betancourt, 2022] Margossian, C. C. and Betancourt, M. (2022).

Efficient automatic differentiation of implicit functions.
arXiv:2112.14217.

[Margossian and Gillespie, 2017] Margossian, C. C. and Gillespie, W. R. (2017).
Gaining efficiency by combining analytical and numerical methods to solve ODEs: Implementation in Stan and application to Bayesian PK/PD.
American Conference on Pharmacometrics.

[Margossian et al., 2023] Margossian, C. C., Hoffman, M. D., Sountsov, P., Riou-Durand, L., Vehtari, A., and Gelman, A. (2023).
Nested $\widehat{R}$: Assessing the convergence of Markov chain Monte Carlo when running many short chains.
Preprint. arXiv:2110.13017.

References IV

[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.

[Margossian et al., 2021] Margossian, C. C., Zhang, L., Weber, S., and Gelman, A. (2021).
Solving ODEs in a Bayesian context: challenges and opportunities.
Population Approach Group in Europe.

[Margossian et al., 2022] Margossian, C. C., Zhang, Y., and Gillespie, W. R. (2022).
Flexible and efficient Bayesian pharmacometrics modeling using Stan and Torsten, part I.
CPT: Pharmacometrics & Systems Pharmacology, 11:1151 – 1169.

[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.

References V

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

[Moins et al., 2022] Moins, T., Arbel, J., Dutfoy, A., and Girard, S. (2022).
On the use of a local $\widehat{R}$ to improve MCMC convergence diagnostic.
Bayesian Analysis.

[Neal, 2003] Neal, R. M. (2003).
Slice sampling.
Annals of Statistics, 31.

[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.

[Rue et al., 2009] Rue, H., Martino, S., and Chopin, N. (2009).
Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations.
Journal of Royal Statistics B, 71:319 – 392.

References VI

[Timonen et al., 2023] Timonen, J., Siccha, N., Bales, B., Lähdesmäki, H., and Vehtari, A. (2023).
An importance sampling approach for reliable and efficient inference in Bayesian ordinary differential equation models.
Stat.

[Vanhatalo et al., 2010] Vanhatalo, J., Pietiläinen, V., and Vehtari, A. (2010).
Approximate inference for disease mapping with sparse Gaussian processes.
Statistics in Medicine, 29(15):1580–1607.

[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 VII

[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.