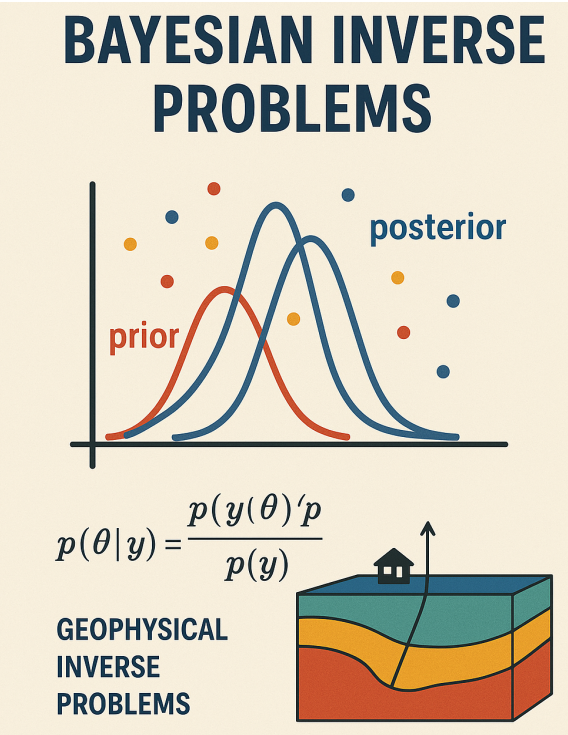
Posterior Estimation Practical: MC, McMC, HMC, V.I.

Mark Asch - MAKUTU/2025



MONTE CARLO

Recall: Monte Carlo Integration

Algorithm

1. Simulate uniform random variables $X_1, X_2, \ldots, X_n$ which can be done using software generated $U_1, U_2, \ldots, U_n$ independent and identically distributed (i.i.d.) uniform random variables on $[0, 1]$.

2. Then let $X_i = a + (b - a)U_i$ for $i = 1, 2, \ldots, n$. Thus, $X_1, X_2, \ldots, X_n$ are independent and identically distributed uniform random variables on $[a, b]$.

3. Evaluate $f(X_1), f(X_2), \ldots, f(X_n)$.

4. Take the average of $f(X_1), f(X_2), \ldots, f(X_n)$ by computing

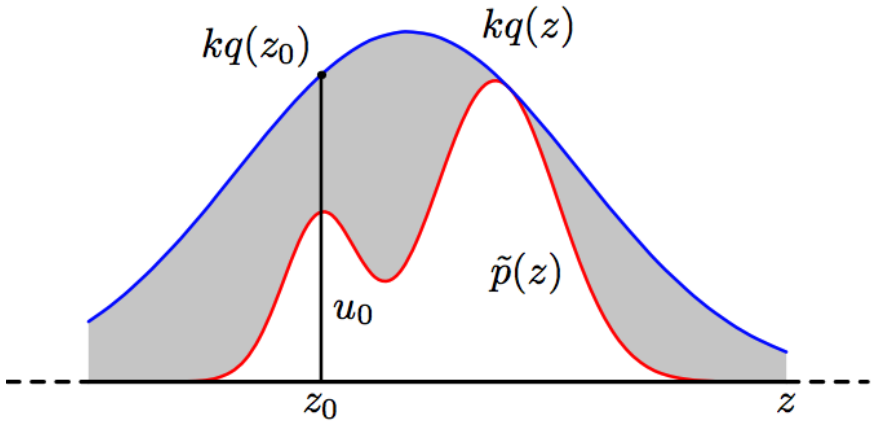
$$(b-a)\frac{1}{n}\sum_{i=1}^{n} f(X_i) = (b-a)\frac{f(X_1)+f(X_2)+\cdots+f(X_n)}{n}$$

and then by the strong law of large numbers this converges to

$$\begin{aligned}(b-a)\,\mathrm{E}(f(X_1)) &= (b-a)\int_a^b f(x)\frac{1}{b-a}\,\mathrm{d}x \\ &= \int_a^b f(x)\,\mathrm{d}x\end{aligned}$$

Recall: Monte Carlo Integration

Sampling Strategies: Rejection Sampling



- **Algorithm**:

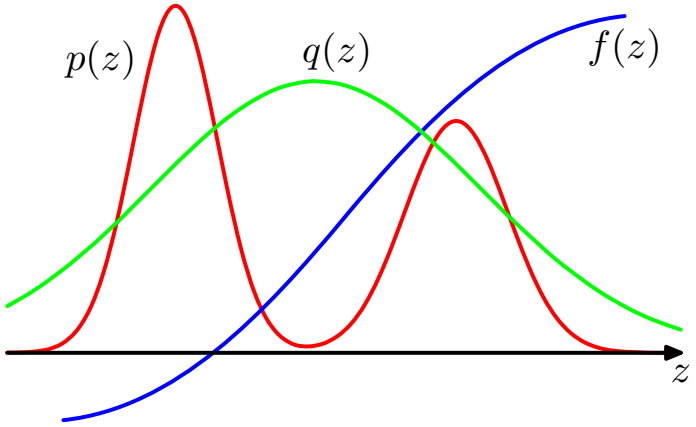
  $\Rightarrow$ Define a simple distribution $q(z)$ and find a k such that for all z

$$kq(z) \geq \tilde{p}(z).$$

  $\Rightarrow$ Draw $z_0 \sim q(z)$.
  $\Rightarrow$ Draw $u_0 \sim \mathcal{U}\left[0, kq(z_0)\right]$.
  $\Rightarrow$ Discard if $u_0 > \tilde{p}(z_0)$, otherwise retain u_0.

Recall: Monte Carlo Integration

Sampling Strategies: Importance Sampling



Input:

- Target distribution $p(x)$, Function of interest $f(x)$, Proposal distribution $q(x)$, Number of samples N

1. **Initialize**: Set sum $S = 0$
2. **For** $i = 1$ to N:
   (a) **Sample**: Generate $x_i \sim q(x)$

(b) **Compute weight** : $w_i = \frac{p(x_i)}{q(x_i)}$
(c) **Evaluate**: $y_i = f(x_i) \cdot w_i$
(d) **Accumulate**: $S = S + y_i$

3. **Estimate**: $\hat{\mu}_{\mathrm{IS}} = \frac{S}{N}$

Ex. 1: Monte Carlo Integration

Estimate π

This is a well-known, introductory exercise in Monte Carlo integration. Compute the area under the first quadrant of the unit circle, where the function to integrate is

$$f(x,y) = \begin{cases} 1 & \text{for } x^2 + y^2 \leq 1, \\ 0 & \text{otherwise,} \end{cases}$$

and the integral

$$I = \int_0^1 \int_0^1 f(x,y)\,\mathrm{d}x\mathrm{d}y.$$

Then $I \approx \pi/4$ and $4I \approx \pi$.

1. Write a python code to estimate π.

2. Study the accuracy of the approximation as a function of n, the number of samples and deduce the rate of convergence.

3. [Optional] Plot the sample points, red for those under the curve, blue for those above.

Ex. 2: Monte Carlo Integration

Rejection Sampling

Suppose that we want to sample from a multi-modal Gaussian

$$p(x) = \alpha_1 \mathcal{N}(\mu_1, \sigma_1^2) + \alpha_2 \mathcal{N}(\mu_2, \sigma_2^2),$$

but we have no way to accurately sample it because of the two modes.

1. A suitable proposal distribution $q(x)$ would be a simple, unimodal Gaussian, centered between the two modes and with a variance large enough to cover the range of p. Propose and plot a suitable $q(x)$ and superpose it on $p(x)$.

2. By trail and error, find a factor k that is large enough to provide an envelope, i.e. such that $kq(x) \geq p(x)$. Plot again the superposition, but this time with $kq(x)$ and $p(x)$.

3. Perform rejection sampling and plot the accepted sampling points (dots and histograms) for $n = 10^3$ and $n = 10^4$.

Ex. 3: Monte Carlo Integration

Importance Sampling

Suppose that we want to estimate a tail probability, or extreme event probability, of a standard Gaussian, $\mathcal{N}(0,1)$, for $\mathrm{P}(X > 5)$.

- Ordinary MC integration will not be very efficient since almost all the samples will be rejected.
- A better option would be to use an exponential density, truncated at $x = 5$, as the importance function for Importance Sampling

1. Show that ordinary Monte Carlo would result in approximately 3 samples out of 10,000,000 from $\mathcal{N}(0,1)$. to have a value greater than 5. Conclusion?
2. We can use the exponential density truncated at 5 as the importance (proposal) function and use importance sampling to estimate the probability.

(a) Use standard MC to estimate the probability.
(b) Plot the truncated exponential density and the standard Gaussian around the $x = 5$.
(c) Use importance sampling, with the same number of sample points, and compare the result with the theoretical value. Conclusions?

Ex. 4: MC - Rejection Sampling

Prove that rejection sampling draws samples from the desired distribution $p(\mathbf{z})$.

1. Suppose the proposal distribution is $q(\mathbf{z})$. Show that the probability of a sample value $\mathbf{z}$ being accepted is given by $\tilde{p}(\mathbf{z})/kq(\mathbf{z})$ where $\tilde{p}$ is any unnormalized distribution that is proportional to $p(\mathbf{z})$, and the constant k is set to the smallest value that ensures $kq(\mathbf{z}) \geq \tilde{p}(\mathbf{z})$ for all values of $\mathbf{z}$.

2. Note that the probability of drawing a value $\mathbf{z}$ is given by the probability of drawing that value from $q(\mathbf{z})$ times the probability of accepting that value given that it has been drawn. Make use of this, along with the sum and product rules of probability, to write down the normalized form for the distribution over $\mathbf{z}$, and show that it equals $p(\mathbf{z})$.

MARKOV CHAIN

Markov Chains

Principles and Basics

Markov chains are an essential component of Markov chain Monte Carlo (McMC) techniques. Under MCMC, the Markov chain is used to sample from some target distribution. To get a better understanding of what a Markov chain is, and further, how it can be used to sample form a distribution, the following exercises introduce some basic concepts.

- Recall: A Markov chain is defined by three elements:
  - ⇒ A state space x, which is a set of values that the chain is allowed to take.
  - ⇒ A transition operator $p(x^{(t+1)}|x^{(t)})$ that defines the probability of moving from state $x^{(t)}$ to $x^{(t+1)}$.
  - ⇒ An initial condition distribution$\pi^{(0)}$ which defines the probability of being in any one of the possible states at the initial iteration $t = 0$.

- If the transition operator for a Markov chain does not change across transitions, the Markov chain is called time homogenous, and as $t \to \infty$, the chain will reach an equilibrium that is called the chain's stationary distribution, $p(x^{(t+1)}|x^{(t)}) = p(x^{(t)}|x^{(t-1)})$

- The stationary distribution of a Markov chain is important for sampling from probability distributions, a technique that is at the heart of Markov Chain Monte Carlo (McMC) methods.

Ex. 5a: Markov Chains

Discrete

- Finite state-space (time homogenous) Markov chain

  $\Rightarrow$ If the state space of a Markov chain takes on a finite number of distinct values, and it is time homogenous, then the transition operator can be defined by a matrix P, where the entries of P are

$$p_{ij} = p(X^{(t+1)} = j | x^{(t)} = i)$$

  $\Rightarrow$ This means that if the chain is currently in the i-th state, the transition operator assigns the probability of moving to the j-th state by the entries of i-th row of P (i.e. each row of P defines a conditional probability distribution on the state space).

- Consider the problem of predicting the weather in Berkeley, CA. Suppose that

⇒ there are only 3 weather conditions: sunny (s), foggy (f), rainy (r), i.e. a state space that takes on 3 discrete values;

⇒ weather patterns are very stable there, so a Berkeley meteorologist (based on meteorological archives) can easily predict the weather next week based on the weather today with the following transition rules:

→ if it is sunny today, then it is highly likely that it will be sunny next week

$$p(X^{(t+1)} = \mathrm{s} | X^{(t)} = \mathrm{s}) = 0.8,$$

it is very unlikely that it will be raining next week

$$p(X^{(t+1)} = \mathrm{r} | X^{(t)} = \mathrm{s}) = 0.05,$$

and somewhat likely that it will foggy next week

$$p(X^{(t+1)} = \mathrm{f} | X^{(t)} = \mathrm{s}) = 0.15$$

→ if it is foggy today, then it is somewhat likely that it will be sunny next week

$$p(X^{(t+1)} = \mathrm{s} | X^{(t)} = \mathrm{f}) = 0.4,$$

it is slightly more likely that it will be foggy next week

$$p(X^{(t+1)} = \mathrm{f} | X^{(t)} = \mathrm{f}) = 0.5,$$

and fairly unlikely that it will rainy next week

$$p(X^{(t+1)} = \mathrm{r} | X^{(t)} = \mathrm{f}) = 0.1$$

→ if it is rainy today, then it is unlikely that it will be sunny next week

$$p(X^{(t+1)} = \mathrm{s} | X^{(t)} = \mathrm{r}) = 0.1,$$

it is somewhat likely that it will be foggy next week

$$p(X^{(t+1)} = \mathrm{f} | X^{(t)} = \mathrm{r}) = 0.3,$$

and fairly likely that it will be rainy next week

$$p(X^{(t+1)} = \mathrm{r} | X^{(t)} = \mathrm{r}) = 0.6.$$

1. Write the 3×3 transition matrix P, where each row of P corresponds to the weather at iteration t (today), and each column corresponds to the weather at iteration $t + 1$ (next week.)

2. Simulate the evolution of the Markov chain, starting from an initial state of rain, for 6 months (25 week, say).

3. At what point in time does the chain reach a steady equilibrium, and what are the 3 equilibrium probabilities?

4. [Optional] Compute the theoretical steady state by an eigenvector analysis and print the errors.

Ex. 5b: Markov Chains

Continuous

We can use the stationary distribution of a continuous state-space Markov chain in order to sample from a continuous probability distribution: we run a Markov chain for a sufficient amount of time so that it has reached its stationary distribution, then keep the states that the chain visits as samples from that stationary distribution.

In this exercise we define a continuous state-space Markov chain, by supposing:

- The transition operator is a Gaussian distribution with unit variance and a mean that is half the distance between zero and the previous state, and
- the distribution over initial conditions is a Gaussian distribution with zero mean and unit variance.

To ensure that the chain has moved sufficiently far from the initial conditions and that we are sampling

from the chain's stationary distribution, we will choose to throw away the first 50 burn-in states of the chain. We also run multiple chains simultaneously in order to sample the stationary distribution more densely. Here we will choose to run 5 chains simultaneously.

1. Write a code that simulates the 5 chains simultaneously for 1000 iterations.

2. Plot

   (a) a zoom of the first 100 iterations, and mark the burn-in cutoff,
   (b) the complete trace of the 5 chains,
   (c) the histogram of the retained states.

3. What is the stationary distribution? What are its estimated parameters?

4. [Optional] Identifying the chain as an AR(1) process, compute the theoretical variance and compare it with the value obtained from the simulations of the Markov chain.

McMC

McMC

Approaches & Methods

- Q: "How does the inference actually work? How do we get these magical samples from the posterior?"

- A: "Well it's easy, MCMC generates samples from the posterior distribution by constructing a reversible Markov-chain that has as its equilibrium distribution the target posterior distribution. Questions?". `https://twiecki.io/blog/2015/11/10/mcmc-sampling/`

  ⇒ Theory is quite clear, but requires deeper understanding of Markov chains and ergodicity.

  ⇒ Practice is somewhat more complex, and needs to be exercised, manipulated, tested, tuned.

- McMC flavors:

  ⇒ Metropolis Hastings is the simplest, and can be viewed as a random walk approach.

⇒ Gibbs Sampling factorizes a complex posterior into the product of simpler, tractable, known posteriors.

⇒ Hamiltonian MC attempts to avoid the "wandering" of the random walk, by introducing dynamics defined by a Hamiltonian system of equations.

Ex. 6: McMC

Metropolis Hastings - Basics

[Optional] Go to `https://twiecki.io/blog/2015/11/10/mcmc-sampling/`, follow the explanations and (re)code this very detailed and well-explained MH version of McMC.

Ex. 7: McMC

Metropolis Hastings - conjugate Gaussian

This instructive exercise will be in two parts:

1. We will use the Metropolis-Hasting algorithm to compute the posterior distribution of the mean of a Gaussian distribution with known variance, from a sequence of given observations.

2. We will study the influence of the proposal distribution on the convergence of the Markov Chain generated in the first part.

In the first part, we will start with a proposal that is far from the reality, and the likelihood function will also be very approximate. In the second part, we will investigate more closely the behavior of the chain as the proposal distribution changes.

Recall Bayes' Law that gives an expression for the

posterior conditional probability of a parameter θ

$$p(\theta \mid y) = \frac{p(\theta)p(y \mid \theta)}{\int p(\theta')p(y \mid \theta') \, \mathrm{d}\theta'},$$

where y is the observation, $p(\theta)$ is the prior probability, $p(y \mid \theta)$ is the likelihood function and the denominator is a normalization factor representing the total probability of y. Very often, in practical applications, this denominator is intractable (impossible, or too expensive to compute), whereas the likelihood and prior are known. This is an ideal instance for MCMC, since we can simulate the posterior without having to know the normalizing factor.

Consider a simple problem, where we have a sequence of 5 measurements (or samples), $\{y_1, \ldots, y_5\} = \{9.37, 10.18, 9.16, 11.60, 10.33\}$, depending on a parameter θ (the unknown mean) for which we would like to obtain the (posterior) probability distribution, $p(\theta \mid y)$.

For the Metropolis-Hasting algorithm, use the

Metropolis acceptance ratio

$$r = \frac{p\left(\theta^* \mid y\right)}{p\left(\theta^{(t)} \mid y\right)} = \frac{p\left(y \mid \theta^*\right) p(\theta^*)}{p\left(y \mid \theta^{(t)}\right) p(\theta^{(t)})},$$

where θ^* is a candidate value for the chain drawn from the symmetric Gaussian proposal distribution $\mathcal{N}(0, \delta^2)$, the actual value in the chain is denoted $\theta^{(t)}$, the known prior is $\mathcal{N}(\mu, \tau^2)$, and the likelihood is computed assuming that $y_i \sim \mathcal{N}(\theta, \sigma^2)$. The exact expressions for the sample mean and variance can be computed, and are given by

$$\mu_n = \frac{(n/\sigma^2)\bar{y} + (1/\tau^2)\mu}{n/\sigma^2 + 1/\tau^2}$$

and

$$\tau_n^2 = \frac{1}{n/\sigma^2 + 1/\tau^2}.$$

1. Suppose that the proposal distribution has variance $\delta^2 = 2$, and that the prior has parameters $\mu = 5$,

$\tau^2 = 10$. Fix a chain length of 10^4 and an initial value $\theta^{(0)} = 5$. Assume that the likelihood variance is known, with $\sigma^2 = 1$.

(a) Code and simulate the MH algorithm for these values.

(b) Plot the trace (θ as a function of the iteration number, subsampling one in every ten values) and the histogram of the Markov chain (after removing a burn-in period of the first 50 values, say) and compare it with the theoretical law.

(c) Conclusions?

2. We now study how the choice of the parameter δ in the proposal distribution affects the mixing, and hence the convergence of the simulated Markov chain. Consider a sequence of values

$$\delta^2 \in \left\{ \frac{1}{32}, \frac{1}{2}, 2, 32, 64 \right\}.$$

(a) For diagnostic purposes compute the autocorrelation function and record the the lag-1 autocor-

relations. For which value of δ^2 do we have a minimum (optimum)?

(b) Plot and compare the convergence of the 5 Markov chains for the first 500 iterations, one graphic for each value of δ^2. Observations and conclusions?

Ex. 8: McMC

Metropolis Hastings - 1D heat eq. inverse problem

- We want to estimate the conductivity (diffusivity) coefficient, D, in the 1D heat (diffusion) equation

$$\begin{cases} \dfrac{\partial u}{\partial t} - \dfrac{\partial}{\partial x}\left(D\dfrac{\partial u}{\partial x}\right) = 0, \quad x \in (0, L), \quad 0 < t \leq T, \\ u(x, 0) = f(x), \quad x \in (0, L), \\ u(0, t) = 0, \quad u(L, t) = 0, \quad t > 0, \end{cases} \tag{1}$$

  from noisy measurements (data)

$$y_i = u(x_i, T) + \epsilon_i$$

  at points x_i, $i = 1, \ldots, N$ and final time T, where the i.i.d. noise is taken as

$$\epsilon_i \sim \mathcal{N}(0, s^2).$$

- Suppose that the initial temperature distribution is concentrated in a small sub-interval,

$$u(x,0) = \begin{cases} 1, & 0.75L \leq x \leq 0.8L, \\ 0, & \text{elsewhere.} \end{cases}$$

- Suppose that the prior on the coefficient $p(D)$ is uniform, with $D \geq 0$. Then, by Bayes' Law, the posterior

$$p(D \mid \mathbf{y}) \propto p(\mathbf{y} \mid D) p(D),$$

where the likelihood $p(\mathbf{y} \mid D)$ is computed from the solution of (1) and the noise process.

- So, given a conductivity D and the initial conditions $u(x,0)$, we can solve the PDE, by finite differences or finite elements, and obtain the expected temperature $u(x_i, T; D)$ at locations x_i and time T.

- Then the likelihood can be computed from the measurements and the Gaussian noise model,

$$p(\mathbf{y} \mid D) = \prod_{i=1}^{N} p\left(\epsilon_i = y_i - u(x_i, T \mid D)\right)$$

$$\propto \exp\left[-\frac{1}{2}\sum_{i=1}^{N}\left(\frac{y_i - u(x_i, T \mid D)}{s}\right)^2\right].$$

- Assume an improper uniform prior,

$$p(D) = \begin{cases} 1 & \text{if } D \geq 0, \\ 0 & \text{otherwise,} \end{cases}$$

  though this could readily be changed to a proper, bounded uniform law by imposing an upper bound on D.

- Then, by Bayes' Law, we obtain the expression for

the posterior probability distribution of D,

$$p(D \mid \mathbf{y}) \propto \begin{cases} \exp\left[-\frac{1}{2}\sum_{i=1}^{N}\left(\frac{y_i - u(x_i,T|D)}{s}\right)^2\right], & \text{if } D \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Estimate this posterior by a random walk Metropolis-Hastings (MH) MCMC, with target distribution $p(D \mid \mathbf{y})$. For clarity, we recall the three-step algorithm.

1. Initialize: set $X_0 = D$, and for $n = 0, \ldots$ until convergence, compute X_{n+1} as follows.

2. Draw the random variable r from a $\mathcal{U}[0,1]$ distribution and set the candidate value

$$D' = D + w(2r - 1),$$

where w is the width of the proposal distribution.

3. With probability

$$\alpha(D \mid D') = \min\left\{1, \frac{\exp\left[-\frac{1}{2}\sum_{i=1}^{N}\left(\frac{y_i - u(x_i, T|D)'}{s}\right)^2\right]}{\exp\left[-\frac{1}{2}\sum_{i=1}^{N}\left(\frac{y_i - u(x_i, T|D)}{s}\right)^2\right]}\right\}$$

set $X_{n+1} = D'$, else keep $X_{n+1} = D$.

For the simulations, you can use the following values:

- $D = 0.5$, $L = 10$, $T = 2$.
- Try noise level $s = 0.03$ and proposal width $w = 0.1$. Initialize the Markov chain at $X_0 = 1$.
- Select reasonable discretization parameters for x and t.
- Solve the heat equation with the explicit FTCS finite difference scheme (though other methods/schemes can be used).

Output:

1. The trace of the Markov chain.

2. The histogram of the chain, with the estimated and true values indicated.

3. The posterior statistics: mean, median, standard deviation, 95% credible interval.

4. Compute the acceptance rate of the MH scheme.

McMC:
(Hamiltonian) HMC

McMC

Hamiltonian Dynamics - Theory

- Hamiltonian dynamics is one way that physicists describe how objects move throughout a system. Hamiltonian dynamics describe an object's motion in terms of its location $\mathbf{x}$ and momentum $\mathbf{p}$ (equivalent to the object's mass times its velocity) at some time t.

- For each location the object takes, there is an associated potential energy $U(\mathbf{x})$, and for each momentum there is an associated kinetic energy $K(\mathbf{p})$. The total energy of the system is constant and known as the Hamiltonian $H(\mathbf{x}, \mathbf{p})$ defined simply as the sum of the potential and kinetic energies,

$$H(\mathbf{x}, \mathbf{p}) = U(\mathbf{x}) + K(\mathbf{p}).$$

- Hamiltonian dynamics describe how kinetic energy is converted to potential energy (and vice versa) as

an object moves throughout a system in time. This description is implemented quantitatively via a set of differential equations known as the Hamiltonian equations,

$$\frac{\partial x_i}{\partial t} = \frac{\partial H}{\partial p_i} = \frac{\partial K(\mathbf{p})}{\partial p_i},$$
$$\frac{\partial p_i}{\partial t} = -\frac{\partial H}{\partial x_i} = -\frac{\partial U(\mathbf{x})}{\partial x_i}.$$

- Therefore, if we have expressions for $\partial U(\mathbf{x})/\partial x_i$ and $\partial K(\mathbf{p})/\partial p_i$ and a set of initial conditions (i.e. an initial position $\mathbf{x}_0$ and initial momentum $\mathbf{p}_0$ at time t_0), it is possible to predict the location and momentum of an object at any point in time $t = t_0 + T$ by simulating these dynamics for a duration T.

McMC

Hamiltonian Dynamics - Numerics

- There are a number of numerical methods for discretizing the dynamics in time, including Euler's method and the Leapfrog Method

- The Leapfrog method updates the momentum and position variables sequentially, starting by simulating the momentum dynamics over a small interval of time $\delta/2$, then simulating the position dynamics over a slightly longer interval in time δ, then completing the momentum simulation over another small interval of time $\delta/2$ so that $\mathbf{x}$ and $\mathbf{p}$ now exist at the same point in time.

  $\Rightarrow$ a half step in time to update the momentum variable
  $\Rightarrow$ a full step in time to update the position variable
  $\Rightarrow$ the remaining half step in time to finish updating

the momentum variable

$$p_i\left(t+\frac{\delta}{2}\right) = p_i(t) - \frac{\delta}{2}\frac{\partial U}{\partial x_i}(t)$$

$$x_i(t+\delta) = x_i(t) + \delta\frac{\partial K}{\partial p_i}\left(t+\frac{\delta}{2}\right)$$

$$p_i(t+\delta) = p_i\left(t+\frac{\delta}{2}\right) - \frac{\delta}{2}\frac{\partial U}{\partial x_i}(t+\delta)$$

$\Rightarrow$ this scheme is symplectic (area-preserving)... unlike Euler, Runge-Kutta, etc. Symplectic integrators exactly preserve phase space volume, just like the Hamiltonian trajectories (Liouville's Theorem) they are approximating.

Ex. 9: McMC

HMC - Harmonic Oscillator

The objective here is to demonstrates the leapfrog integration method for Hamiltonian dynamics, showing how the harmonic oscillator evolves in time, while conserving total energy in phase space. The code for the simulation of the dynamics can be utilized, as is, for Hamiltonian Monte Carlo—see below.

- The obtained results should clearly demonstrate the physics:

  $\Rightarrow$ the harmonic oscillator exhibits periodic motion,
  $\Rightarrow$ energy oscillates between kinetic and potential forms while total energy remains constant,
  $\Rightarrow$ and the phase space trajectory traces out an ellipse characteristic of conservative Hamiltonian systems.

- Suppose a ball of mass $m = 1$ is attached to a horizontal spring (no gravity). Then the spring

exerts a force on the ball

$$F = -kx,$$

where k is the spring constant, which we suppose equal to one.

- The potential energy is then

$$U(x) = \int F \, \mathrm{d}x = \int -x \, \mathrm{d}x = -\frac{x^2}{2}$$

and the kinetic energy

$$K(v) = \frac{1}{2}mv^2 = \frac{v^2}{2} = \frac{p^2}{2} = K(p).$$

Their partial derivatives are

$$\frac{\partial U(x)}{\partial x} = x, \quad \frac{\partial K(p)}{\partial p} = p.$$

1. Write down the Hamiltonian and the 3 steps of the leapfrog method for the Hamiltonian system.

2. Simulate the dynamics, with $\delta = 0.1$ for 70 time steps. Use the initial conditions: position $x(0) = -4$, momentum $p(0) = 1$.

3. Plot:

   (a) position as a function of time,
   (b) U, K and H as a function of time,
   (c) the phase plot, in the $x - p$ plane.

HMC

Theory and algorithm

- This is a foundational example that demonstrates how HMC combines the best aspects of Hamiltonian dynamics simulation with Bayesian (Metropolis) sampling, making it especially powerful for high-dimensional posterior distributions in modern statistical computing.

- Recall:

  $\Rightarrow$ The random-walk behavior of many Markov Chain Monte Carlo (McMC) algorithms makes Markov chain convergence to a target stationary distribution $p(x)$ inefficient, resulting in slow mixing.

  $\Rightarrow$ Hamiltonian/Hybrid Monte Carlo (HMC), is a McMC method that adopts physical system dynamics rather than a probability distribution to propose future states in the Markov chain.

⇒ This allows the Markov chain to explore the target distribution much more efficiently, resulting in faster convergence.

⇒ The main idea behind Hamiltonian/Hybrid Monte Carlo is to develop a Hamiltonian function $H(\mathbf{x}, \mathbf{p})$ such that the resulting Hamiltonian dynamics allow us to efficiently explore some target distribution $p(\mathbf{x})$.

- How can we choose such a Hamiltonian function? It turns out it is pretty simple to relate a $H(\mathbf{x}, \mathbf{p})$ to $p(\mathbf{x})$ using a basic concept adopted from statistical mechanics known as the canonical distribution.

  ⇒ For any energy function $E(\theta)$ over a set of variables θ, we can define the corresponding canonical distribution as

$$p(\theta) = \frac{1}{Z} e^{-E(\theta)}$$

  where we simply take the exponential of the negative of the energy function.

⇒ The variable Z is a normalizing constant called the partition function that scales the canonical distribution such that is sums to one, creating a valid probability distribution.

⇒ Now, as we saw above, the energy function for Hamiltonian dynamics is a combination of potential and kinetic energies

$$E(\theta) = H(\mathbf{x}, \mathbf{p}) = U(\mathbf{x}) + K(\mathbf{p})$$

⇒ Therefore the canonical distribution for the Hamiltonian dynamics energy function is

$$\begin{aligned} p(\mathbf{x}, \mathbf{p}) &\propto e^{-H(\mathbf{x},\mathbf{p})} \\ &= e^{-[U(\mathbf{x})+K(\mathbf{p})]} \\ &= e^{-U(\mathbf{x})} e^{-K(\mathbf{p})} \\ &\propto p(\mathbf{x})p(\mathbf{p}) \end{aligned}$$

⇒ Here we see that joint (canonical) distribution for $\mathbf{x}$ and $\mathbf{p}$ factorizes. This means that the two variables are independent, and the canoncial dis-

tribution $p(\mathbf{x})$ is independent of the analogous distribution for the momentum. Therefore, we can use Hamiltonian dynamics to sample from the joint canonical distribution over $\mathbf{p}$ and $\mathbf{x}$ and simply ignore the momentum contributions. Note that this is an example of introducing auxiliary variables to facilitate the Markov chain path. Introducing the auxiliary variable $\mathbf{p}$ allows us to use Hamiltonian dynamics, which are unavailable without them.

$\Rightarrow$ Because the canonical distribution for $\mathbf{x}$ is independent of the canonical distribution for $\mathbf{p}$, we can choose any distribution from which to sample the momentum variables. A common choice is to use a zero-mean, normal distribution with unit variance, that is

$$p(\mathbf{p}) \propto \frac{\mathbf{p}^\top \mathbf{p}}{2}.$$

Note that this is equivalent to having a quadratic

potential energy term in the Hamiltonian

$$K(\mathbf{p}) = \frac{\mathbf{p}^\top \mathbf{p}}{2},$$

which is the exact quadratic kinetic energy function (albeit in 1D) used in the harmonic oscillator example above, and is simple to differentiate.

$\Rightarrow$ Now that we have defined a kinetic energy function, all we have to do is find a potential energy function $U(\mathbf{x})$ that when negated and run through the exponential function, gives the target distribution $p(\mathbf{x})$ (or an unscaled version of it). In other words, we can simply define the potential energy function as

$$U(\mathbf{x}) = -\log p(\mathbf{x}),$$

and if we can calculate its partial derivative with respect to x, we will have all the ingredients for simulating Hamiltonian dynamics that can be used in an McMC technique.

- HMC: we use Hamiltonian dynamics as a proposal function for a Markov chain in order to explore the target (canonical) density $p(\mathbf{x})$ defined by $U(\mathbf{x})$ more efficiently than using a proposal probability distribution.

  $\Rightarrow$ Starting at an initial state$[\mathbf{x}_0, \mathbf{p}_0]$, we simulate Hamiltonian dynamics for a short time using the Leapfrog method.

  $\Rightarrow$ We then use the state of the position and momentum variables at the end of the simulation as our proposed state variables $\mathbf{x}^*$ and $\mathbf{p}^*$.

  $\Rightarrow$ The proposed state is accepted using an update rule analogous to the Metropolis acceptance criterion.

- For a given set of initial conditions, Hamiltonian dynamics will follow contours of constant energy in phase space (analogous to the circle traced out in phase space in the example above). Therefore we must randomly perturb the dynamics so as to explore all of $p(\mathbf{x})$. This is done by simply drawing a random momentum from the corresponding canon-

ical distribution $p(\mathbf{p})$ before running the dynamics prior to each sampling iteration t.

- Combining these steps:

  $\Rightarrow$ sampling random momentum,
  $\Rightarrow$ followed by Hamiltonian dynamics
  $\Rightarrow$ and Metropolis acceptance criterion,

- defines the HMC algorithm for drawing M samples from a target distribution.

HMC

Algorithm

1. set $t = 0$

2. generate an initial position state $\mathbf{x}^{(0)} \sim \pi^{(0)}$

3. repeat until $t = M$

   (a) set $t = t + 1$
   (b) sample a new initial momentum variable from the momentum canonical distribution $\mathbf{p}_0 \sim p(\mathbf{p})$
   (c) set $\mathbf{x}_0 = \mathbf{x}^{(t-1)}$.
   (d) run Leapfrog algorithm starting at $[\mathbf{x}_0, \mathbf{p}_0]$ for L steps and step-size δ to obtain proposed states $\mathbf{x}^*$ and $\mathbf{p}^*$
   (e) calculate the Metropolis acceptance probability:

$$\begin{aligned}\alpha &= \min(1, \exp(-U(\mathbf{x}^*) + U(\mathbf{x}_0) - K(\mathbf{p}^*) + K(\mathbf{p}_0))) \\ &= \min\left(1, \exp\left(H(\mathbf{x}_0) - H(\mathbf{x}^*)\right)\right)\end{aligned}$$

(f) draw a random number u from Unif$(0, 1)$
   i. if $u \leq \alpha$ accept the proposed state position $\mathbf{x}^*$ and set the next state in the Markov chain $\mathbf{x}^{(t)} = \mathbf{x}^*$
   ii. else set $\mathbf{x}^{(t)} = \mathbf{x}^{(t-1)}$

Ex. 10: McMC

HMC - Sampling from bivariate Gaussian

The target distribution $p(\mathbf{x})$ for this sampling exercise is a bivariate Gaussian with the following parameterization: $p(\mathbf{x}) = \mathcal{N}(\mu, \mathbf{\Sigma})$ with mean $\mu = [\mu_1, \mu_2] = [0, 0]$ and covariance

$$\mathbf{\Sigma} = \begin{bmatrix} 1 & \rho_{12} \\ \rho_{21} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}.$$

In order to sample from $p(\mathbf{x})$ (assuming that we are using a quadratic energy function), we need to determine the expressions for $U(\mathbf{x})$ and $\partial U(\mathbf{x})/\partial x_i$. Recall that the target potential energy function can be defined from the canonical form as $U(\mathbf{x}) = -\log(p(\mathbf{x}))$. So, taking the negative log of the Gaussian distribution above, we define the potential energy function

$$E(\mathbf{x}) = -\log\left(e^{-\frac{\mathbf{x}^\mathbf{T}\mathbf{\Sigma}^{-1}\mathbf{x}}{2}}\right) - \log Z,$$

where Z is the normalizing constant for a Gaussian distribution and can be ignored because it will cancel. The potential energy function is then simply,

$$U(\mathbf{x}) = \frac{\mathbf{x^T \Sigma^{-1} x}}{2}$$

with partial derivatives

$$\frac{\partial U(\mathbf{x})}{\partial x_i} = x_i.$$

We are now in possession of all the ingredients for implementing the HMC algorithm. The algorithm samples from a bivariate normal distribution with correlation coefficient 0.8. The HMC method is particularly effective for this type of problem because it can make large moves while maintaining high acceptance rates, avoiding the random walk behavior of simpler McMC methods.

1. Copy the notebook `HMC-2D-Gaussian.ipynb` and fully document it by adding markdown blocs. Iden-

tify all the mathematical formulas and their corresponding lines of code.

2. Adjust the hyperparameters, `NSamples`, `delta`, `L`, and study their effects.

3. [Optional] Compare the performance of HMC with Metropolis-Hastings.

4. [Optional] Run the exercise with HMCLab.

McMC for FWI

Ex 11: FWI using HMC

ADVANCED

- Fichtner's HMC lab paper in *GJI* (2023), 235.

  $\Rightarrow$ use HMCLab to reproduce the results

Ex. 12: FWI using VI

ADVANCED

- Zhang, Curtis review in *Advances in Geophysics*, 62 (2021).

  $\Rightarrow$ use VIP package to reproduce the results

  $\Rightarrow$ compare with HMC

McMC:
PPL

Ex. 13: McMC

pymc for Lotka-Volterra parameter estimation

All the above examples can be solved with probabilistic programming languages (PPL), such as

- pymc,
- numpyro,
- stan,
- etc.

These DSLs take care of a lot of the technical details, and are more efficient. However, they are much more "black-box" and thus more difficult to tune and debug, unless one already has a solid experience and understanding of McMC.

In this exercise, we will use `pymc` to solve a parameter estimation problem for the Lotka-Volterra (predator-prey) system

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \alpha x - \beta xy,$$
$$\frac{\mathrm{d}y}{\mathrm{d}t} = -\gamma y + \delta xy.$$

- The state vector $X(t) = [x(t), y(t)]$ comprises the densities of the prey and the predator species respectively.

- Parameters $\boldsymbol{\theta} = [\alpha, \beta, \gamma, \delta, x(0), y(0)]$ are the unknowns that we wish to infer from experimental observations.

  $\Rightarrow$ $x(0), y(0)$ are the initial values of the states needed to solve the ODE, and

  $\Rightarrow$ α, β, γ, and δ are unknown model parameters which represent the following:

  $\rightarrow$ α is the growing rate of prey when there is no predator,

  $\rightarrow$ β is the dying rate of prey due to predation,

  $\rightarrow$ γ is the dying rate of predator when there is no prey,

→ δ is the growing rate of predator in the presence of prey.

The Hudson's Bay Company data: the Lotka-Volterra predator prey model has been used to successfully explain the dynamics of natural populations of predators and prey, such as the lynx and snowshoe hare data of the Hudson's Bay Company between 1900 and 1921. This dataset is small, and can be entered by hand:

```
data = pd.DataFrame(dict(
  year = np.arange(1900., 1921., 1),
  lynx = np.array([4.0, 6.1, 9.8, 35.2, 59.4,
                   41.7, 19.0, 13.0, 8.3, 9.1, 7.4,
                   8.0, 12.3, 19.5, 45.7, 51.1,
                   29.7, 15.8, 9.7, 10.1, 8.6]),
  hare = np.array([30.0, 47.2, 70.2, 77.4, 36.3,
                    20.6, 18.1, 21.4, 22.0, 25.4,
                    27.1, 40.3, 57.0, 76.6, 52.3,
                    19.5, 11.2, 7.6, 14.6, 16.2,
                    24.7])))
```

The objective of this exercise is to estimate, without and with uncertainty, the parameters for the Lotka-Volterra model for the Hudson's Bay Company data from 1900 to 1920.

1. Formulate the deterministic and the stochastic inverse problems for the parameter estimation.

2. Load and plot the observational data.

3. Solve the system using `odeint`, for example, using the following reasonable values

$$\begin{aligned}\boldsymbol{\theta} &= [\alpha, \beta, \gamma, \delta, x(0), y(0)] \\ &= [0.52, 0.026, 0.84, 0.026, 34.0, 6.0]\,.\end{aligned}$$

   Conclusions?

4. Solve the deterministic inverse problem using scipy's `least_squares` "Given the residuals of $f(x)$ (an m-dimensional real function of n real

variables) and the loss function $\rho(s)$ (a scalar function), `least_squares` finds a local minimum of the cost function $F(x) = (1/2)\Sigma_{i=1}^{m}\rho(f_i(x)^2)$." It uses a Levenberg-Marquardt (unconstrained case), or trust-region type of algorithm (constrained case).

5. Plot the estimated solution and the measurements.

6. If we did not care about uncertainty, then we would be done. But we do care about uncertainty, so we will implement Bayesian inference on this parameter estimation problem. Use the code `McMC_ODE_LV.py` and the results of the least-squares calculation as priors. Note that all priors are (truncated) Gaussian distributions. The `DEMetropolis` sampler is the most efficient choice here, though others (including HMC) could be used.

7. Comment on the results and draw conclusions.

APPENDIX

Practical Advice for McMC Modeling of Inverse Problems

[See Predator-Prey model for full details.]

- For given values of the model parameters and initial state, the Lotka-Volterra (or any other mechanistic) model predicts population (or other) dynamics into the future (and into the past).

- But given noisy data about population dynamics, how do we solve the inverse problem, that of inferring the values of model parameters consistent with the data?

- Bayesian models require:

  $\Rightarrow$ a mathematical model of what we know about the parameters (i.e., a prior), $p(\theta)$, and

  $\Rightarrow$ a model of what we know about the data generating process given the parameters (i.e., a sampling distribution), $p(y|\theta)$, which considered as a function of θ is called the likelihood.

- Bayes' Law then gives us a general solution to the inverse problem, expressing the posterior $p(\theta|y)$ in terms of the prior and likelihood $p(y|\theta)$.

- (PPLs provide) a Markov chain Monte Carlo (MCMC) sampling (that) draws a sample $\theta^{(1)}, \ldots, \theta^{(M)}$ from the posterior to use for computational inference, and posterior quantities of interest may be expressed as derived random variables using functions $f(\theta)$ of parameters. Such functions may be used for posterior expectation calculations such as parameter estimates that minimize expected square error when $f(\theta) = \theta$, or event probabilities such as the probability of the hare population falling below some fraction of the lynx population, etc.

- L-V is a deterministic model (system of equations), but population measurements contain unexplained variation (weather, diseases, ...) as well as measurement errors—these are the source of uncertainty, or noise, and the measurements do NOT satisfy

the equations exactly—this gives rise to the need for a Bayesian approach. For example, in the lynx-hare case, we cannot even measure the population directly. Instead, pelts are used as a noisy proxy for population sizes.

- **Linear Regression Analogy**

  ⇒ Suppose the underlying deterministic model provides an expected population value, around which there will be variation due to both measurement error and simplifications in the scientific model.

  ⇒ Then, y_n is an observable scalar outcome, x_n is a row vector of unmodeled predictors (aka covariates, features), β is a coefficient vector parameter, and $\sigma > 0$ is the error scale, and the LR model is

$$y_n = x_n\beta + \epsilon_n \quad (2)$$

$$\epsilon_n \sim \mathcal{N}(0, \sigma) \quad (3)$$

  → The deterministic part of the equation is the

linear predictor $x_n\beta$ with predictor x_n (row n of the data matrix x) and coefficient (column) vector β..

→ The stochastic error term, ϵ_n, is assigned a normal distribution located at zero with scale parameter $\sigma > 0$.

→ Usually written as,

$$y_n \sim \mathcal{N}(x_n\beta, \sigma).$$

⇒ Solutions to the Lotka-Volterra equations replace the linear predictor $x_n\beta$, but maintain the error term to compensate for measurement error and unexplained variation in the data.

- To model positive-valued parameters we usually log-transform them, then they are no longer constrained to be positive... and an additive error (in log) becomes a multiplicative error, which can be considered as a percentage error of the total population—this is much better than plus/minus some

fixed value. We have

$$\log y_{n,k} = \log z_{n,k} + \epsilon_{n,k}$$
$$\epsilon_{n,k} \sim \mathcal{N}(0, \sigma_k)$$

where $z_{n,k}$ is the solution of the L-V system, with $k = 1, 2$ and thus

$$\begin{aligned} y_{n,k} &= \exp(\log z_{n,k} + \epsilon_{n,k}) \\ &= z_{n,k} \exp(\epsilon_{n,k}) \end{aligned}$$

- **Weakly informative priors**

  ⇒ We still need to model the priors for the parameters. This topic can be quite complex, but we base our modeling on the concept of weakly informative priors that inform the scale of the parameter, but have only a negligible effect on the exact value.

  ⇒ This makes exploration of outskirts of complicated distributions easier for the McMC sampler.

⇒ To formulate such priors, we need to know the rough scale on which to expect an answer. Because this model is well understood and widely used, as is the basic behavior of predator and prey species such as lynxes and hares, the parameter ranges leading to stable and realistic population oscillations are well known.

⇒ In the Lotka-Volterra case, we have

$$\frac{\mathrm{d}u}{\mathrm{d}t} = (\alpha - \beta v)u = \alpha u - \beta uv \quad (4)$$

$$\frac{\mathrm{d}v}{\mathrm{d}t} = (-\gamma + \delta\, u)\, v = -\gamma v + \delta uv \quad (5)$$

with four parameters $\alpha, \beta, \gamma, \delta \geq 0$, where factor α is the growth rate of the prey population, whereas β is the rate of shrinkage relative to the product of the population sizes. The factor γ is the shrinkage rate of the predator population and δ is the growth rate of the predator population as a factor of the product of the population sizes.

⇒ Since the scale of u and v is roughly 10 (in

thousands), reasonable weakly informative priors would be

$$\alpha, \gamma \sim \mathcal{N}(1, 0.5) \quad (6)$$

$$\beta, \delta \sim \mathcal{N}(0.05, 0.05) \quad (7)$$

$\Rightarrow$ Prior for the noise scale should be proportional. Considering a 95% interval of mean plus/minus two standard deviations for

$$\sigma \sim \mathsf{LogNormal}(-1, 1),$$

we get $\exp(-1-2) \approx 0.05$ and $\exp(-1+2) \approx 3$, so this would be a reasonable prior for σ with a median value of 0.4.

$\Rightarrow$ Finally, we require weakly informative priors for the initial predator and prey populations, which could be

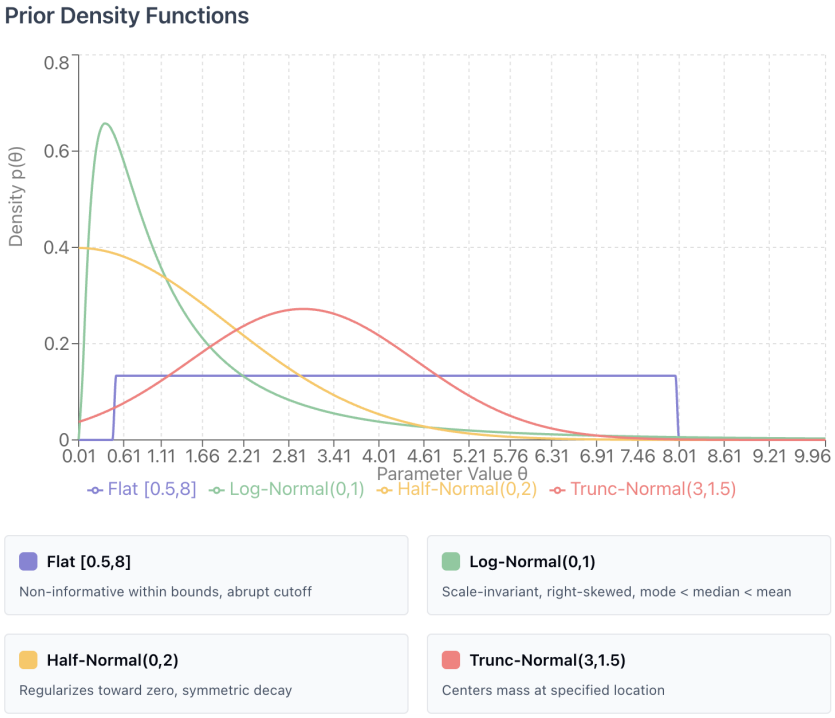
$$z_1^{init},\ z_2^{init} \sim \mathsf{LogNormal}(\log(10), 1)$$

that has a median of 2.3 and a 95% interval of

$$(1.2,\ 75).$$

- That's all folks...

Practical criteria for PRIOR selection



1. *Flat (Uniform) Priors*
   (a) Best for parameters with clear physical constraints (probabilities, fractions).
   (b) Requires domain knowledge to set bounds appropriately.
   (c) Not recommended when parameters could span multiple orders of magnitude.

(d) Example: Proportion of population vaccinated [0, 1].

2. *Log-Normal Priors* (often the best choice)

(a) Ideal for positive rate constants and scale parameters.
(b) Scale-invariant: treats 0.1→1 and 1→10 as equally likely.
(c) Natural for biological rates (growth, decay, transmission).
(d) Specify on log scale: median $= \exp(\mu)$, mode $= \exp(\mu - \sigma^2)$.

3. *Half-Normal Priors*

(a) Good for noise scale parameters (weakly informative).
(b) Provides gentle regularization toward zero.
(c) Commonly used in hierarchical models for variance components.
(d) Example: Observation noise σ when you expect moderate values.

4. *Truncated Normal Priors*

   (a) Most informative option - use when you have strong prior knowledge.
   (b) Centers posterior near specified location.
   (c) Can slow MCMC mixing due to truncation boundary.
   (d) Example: Parameter estimated as 0.5 ± 0.2 in previous studies

Note: Log-normal and half-normal priors generally have better geometry for gradient-based samplers (HMC, NUTS) compared to truncated normals.

Practical recommendations for BURN-IN (warmup)

1. For simple problems: 10-20% of chain length is usually sufficient.

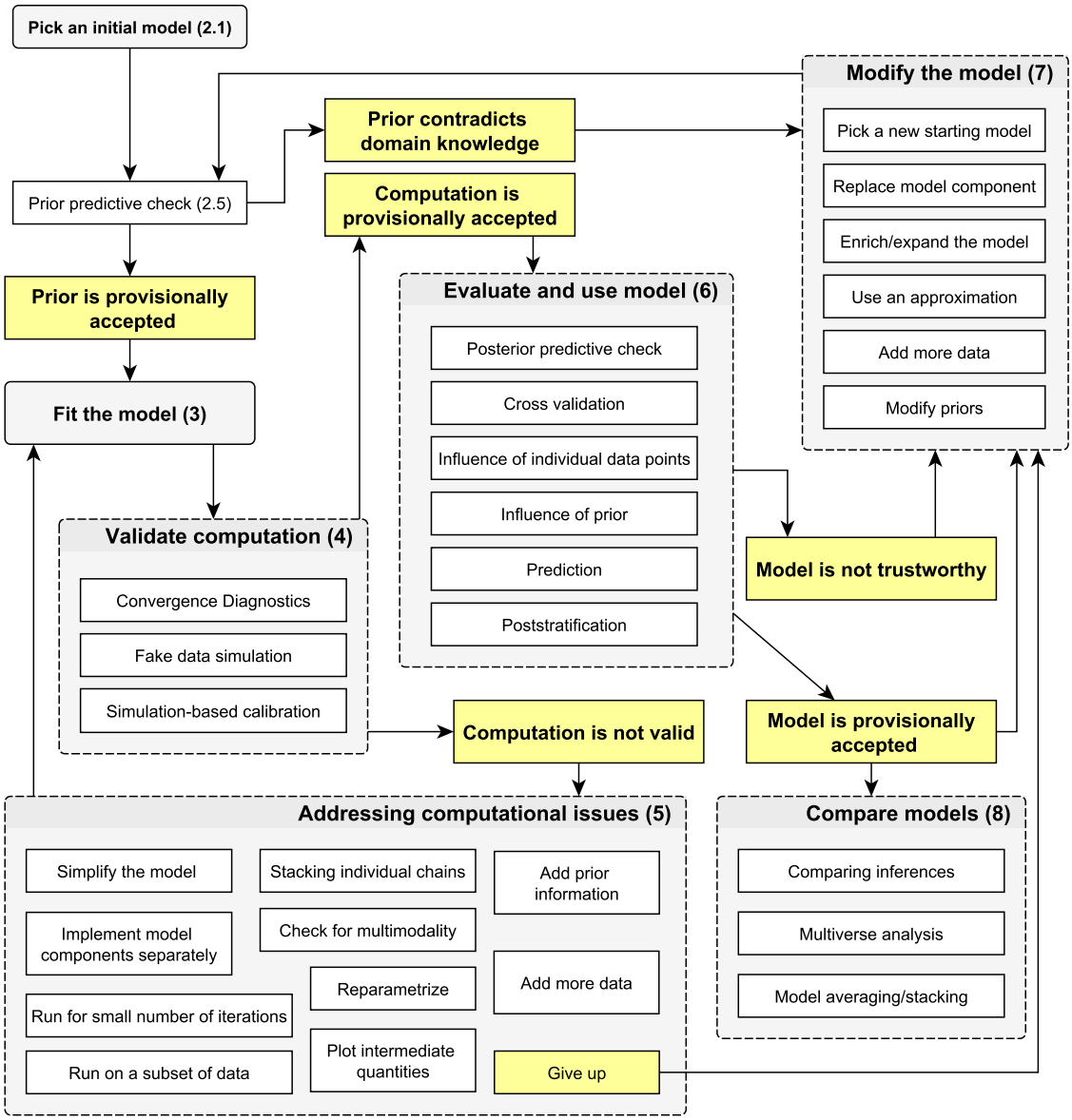
2. For complex problems: 50% can be conservative but safe.

3. Best practice: Run multiple chains and use Gelman-Rubin diagnostic $\hat{R}$.

4. Visual check: Always plot traces - if you see trending, increase burn-in.

5. Modern tools: Use PyMC/Stan which handle this automatically via tuning phases.

Bayesian Workflows



[See Gelman, et al. for full details.]

- Statistics is all about uncertainty: (statistics is hard...)

  ⇒ the usual uncertainties in the data and model parameters,
  ⇒ uncertain whether we are fitting our models correctly,
  ⇒ uncertain about how best to set up and expand our models, and
  ⇒ uncertain in their interpretation.

- Requires a formalized **workflow**:

  ⇒ (a) exploratory data analysis to aid in setting up an initial model;
  ⇒ (b) computational model checks using fake (synthetic) data simulation and the prior predictive distribution;
  ⇒ (c) computational checks to ensure that the inference algorithm works reliably;
  ⇒ (d) posterior predictive checks and other juxtapositions of data and predictions under the fitted model;

⇒ (e) model comparison via tools such as cross-validation.

- **Step 1**: before fitting a model

⇒ choose an initial model, usually from the literature (previous analyses); start simple and complexify, or start complex and simplify;

⇒ modular construction, especially for Bayesian models;

⇒ scale the parameters (just as in ML):

⇒ perform prior predictive checking to understand the implications of a prior distribution, using simulations from the model (NOT using the observed data); they evaluate what data sets would be consistent with the prior and are not be calibrated with actual data, but extreme values to help diagnose priors that are either too strong, too weak, poorly shaped, or poorly located;

$$y^{\text{sim}} \sim p(y) = \int p(y, \theta) \, \mathrm{d}\theta = \int p(\theta) p(y|\theta) \, \mathrm{d}\theta$$

→ We say that a prior leads to a weakly infor-

mative joint prior data-generating process if draws from the prior data-generating distribution $p(y)$ could represent any data set that could plausibly be observed.

→ As with the standard concept of weakly informative priors, it is important that this prior predictive distribution for the data has at least some mass around extreme but plausible data sets.

→ However, there should be no mass on completely implausible data sets.

→ We recommend assessing how informative the prior distribution on the data is by generating a 'flip book' (a series of visualizations to scroll through) of simulated data sets that can be used to investigate the variability and multivariate structure of the distribution.

- **Step 2**: fitting a model

⇒ initial values, adaptation, warmup

→ McMC simulation has 2 stages, starting with the warmup phase which is intended to move

the simulations from their possibly unrepresentative initial values to something closer to the region of parameter space where the log posterior density is close to its expected value

→ Initial values are not supposed to matter in the asymptotic limit, but they can matter in practice, and a wrong choice can threaten the validity of the results.

→ Then the sampling phase, which ideally is run until multiple chains have mixed sufficiently, as indicated by mixing diagnostics, in particular Rhat.

⇒ number of iterations

→ Recommended standard practice is to run at least until $\hat{R}$, the measure of mixing of chains, is less than 1.01 for all parameters and quantities of interest

→ Safe and conservative choice is to run MCMC until the effective sample size is in the thousands or Monte Carlo standard error is tiny in comparison to the required precision for parameter interpretation, but this should not

take too much time...

→ Effective variance reduction can also be obtained by increasing the number of parallel chains, usually between 4 and 8.

⇒ approximate algorithms and models

⇒ fit fast, fail fast

→ We want to fail fast when fitting bad models.

- **Step** 3: using constructed (synthetic) data

⇒ synthetic data simulation

→ The main tool we have for ensuring that the statistical computation is done reasonably well is to actually fit the model to some data and check that the fit is good.

→ Real data can be awkward for this purpose because modeling issues can collide with computational issues and make it impossible to tell if the problem is the computation or the model itself.

→ To get around this challenge, we first explore models by fitting them to simulated data.

→ Basic idea is to check whether our procedure

recovers the correct parameter values when fitting fake data. Typically we choose parameter values that seem reasonable a priori and then simulate a fake dataset of the same size, shape, and structure as the original data. We next fit the model to the fake data to check several things.

→ In case of failure, it is recommended to go simpler and simpler until we get the model to work. Then, from there, we can try to identify the problem.

→ Instability depending on parameter values is a common phenomenon in differential equation models.

→ Fitting the model to simulated data helps us better understand what the model can and cannot learn about the latent process in a controlled setting where we know the ground truth. If a model is able to make good inference on fake-data generated from that very model, this provides no guarantee that its inference on real data will be sensible. But

if a model is unable to make good inference on such fake data, then it's hopeless to expect the model to provide reasonable inference on real data.

⇒ simulation-based calibration and experimentation

→ Idea is that by performing Bayesian inference across a range of datasets simulated using parameters drawn from the prior, we should recover the prior. Simulation-based calibration is useful to evaluate how closely approximate algorithms match the theoretical posterior even in cases when the posterior is not tractable.

→ Simulation-based calibration requires fitting the model multiple times, which incurs a substantial computational cost, especially if we do not use extensive parallelization.

→ Experimentation using constructed data: suppose we are interested in the statistical properties of linear regression fit to data from alternative distributional forms. To start, we could

specify data $x_i, i = 1, ..., n$, draw coefficients a and b and a residual standard deviation σ from our prior distribution, simulate data from $y_i \sim \mathcal{N}(a + bx_i, \sigma)$, and fit the model to the simulated data. Repeat this 1000 times and we can check the coverage of interval estimates: that's a version of simulation-based calibration. We could then fit the same model but simulating data using different assumptions, for example drawing independent data points $y_i$from the $t(4)$ distribution rather than the normal.

→ Objective is to demonstrate how simulation of a statistical system under different conditions can give us insight, not just about computational issues but also about data and inference more generally.

- **Step** 4: computational issues

⇒ folk theorem: computational problems are often caused by model problems

⇒ simple vs complex models

⇒ monitoring, stacking, multimodality, reparametrization, marginalization, adding prior information, adding data

⇒ Posterior distributions with multimodality and other difficult geometries

- **Step** 5: evaluating and using a model

⇒ Posterior predictive checking: if a model is a good fit we should be able to use it to generate data that resemble the data that we observed.

⇒ To generate the data that are used for posterior predictive checks we simulate from the posterior predictive distribution

$$p(\tilde{y}|y) = \int p(\tilde{y}|\theta)p(\theta|y)\,\mathrm{d}\theta$$

⇒ Posterior predictive checking is mostly qualitative. By looking at some important features of the data and the replicated data, which were not explicitly included in the model, we may find a need to extend or modify the model.

⇒ Cross-validation

→ part of the data is left out, the model is fit to the remaining data, and predictive performance is checked on the left-out data—we usually use LOOCV (leave-one-out).

⇒ influence of prior information

- **Step** 6: modifying models

⇒ Model building is a language-like task in which the modeler puts together existing components (linear, logistic, and exponential functions; additive and multiplicative models; binomial, Poisson, and normal distributions; varying coefficients; and so forth) in order to encompass new data and additional features of existing data, along with links to underlying processes of interest.

⇒ Constructing a model for the data, Incorporating additional data,

⇒ Working with prior distributions

→ ladder of possibilities: (improper) flat prior; super-vague but proper prior; very weakly in-

formative prior; generic weakly informative prior; specific informative prior.

- **Step** 7: comparing models

  $\Rightarrow$ key aspect of Bayesian workflow is that we are fitting many models while working on a single problem.

  $\Rightarrow$ We fit multiple models for several reasons: big models, mistakes, getting more data, strengthening priors, comparison with simpler models, computational difficulties

  $\Rightarrow$ visualizing relations

  $\Rightarrow$ CV and model averaging

  $\Rightarrow$ comparing a large number of models: always choose the smallest model that has the same predictive performance as our expanded model. This leads to the problem of predictor (variable) selection.