STAT5003

Week 11 : Markov Chain Monte Carlo

Jaslene Lin
The University of Sydney

THE UNIVERSITY OF SYDNEY

CRICOS 00026A TEQSA PRV12057

Readings

::: {.callout-important}
Original papers

- Monte Carlo sampling methods using Markov chains and their applications
- Equation of State Calculations by Fast Computing Machines
:::

This presentation is based on the SOLES reveal.js Quarto template and is licensed under a Creative Commons Attribution 4.0 International License.

Markov Chain Monte Carlo

Recap Monte Carlo Simulation

Monte Carlo simulation refers to generating random samples **independently** from **a known distribution** to estimate expectations or probabilities.

- Example: Simulating stock prices, rolling dice, estimating π, etc.
- Disadvantages:
    1. Inefficient when targeting rare events or small regions of the space
    2. Scales poorly with dimensionality (curse of dimensionality)
    3. No adaptive as each sample is independent–no feedback or learning from previous samples.

Markov Chain Monte Carlo

- Markov Chain Monte Carlo (MCMC) is a Monte Carlo sampling technique for generating samples from **an arbitrary distribution**
- The samples are **not independent**, but from a **Markov chain** where the *future depends on upon the present*
- Two popular implementations of MCMC are
  - Metropolis-Hastings algorithm and generalization by Hastings
  - Gibbs samplers

Markov Chains

What are Markov Chains?

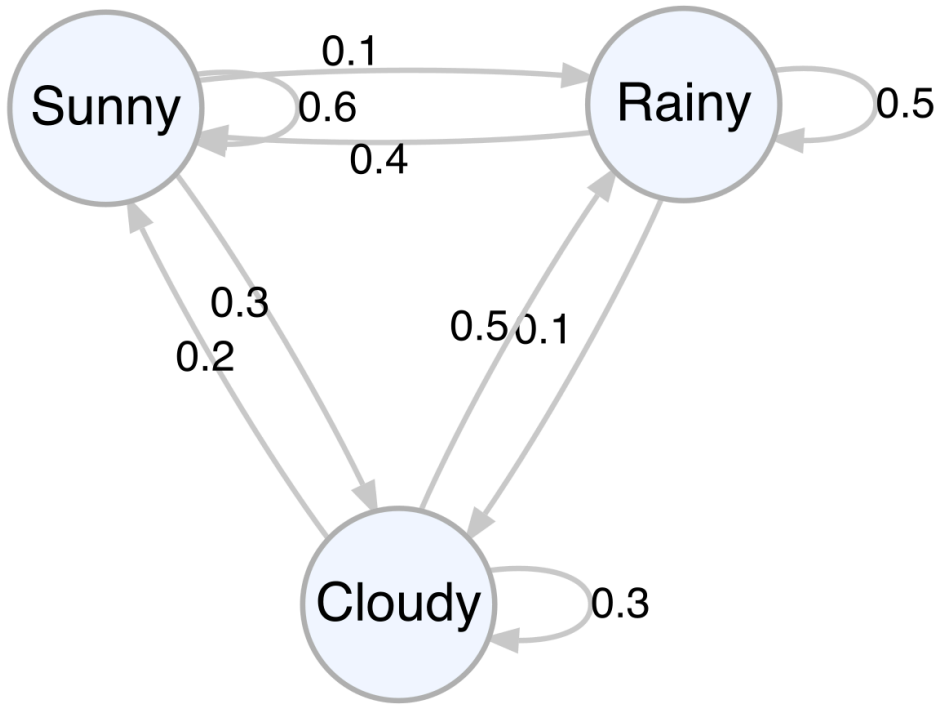
- Markov chain is a random (stochastic) process that follows the Markov property
- Markov property means that the future state of the process only depends on the current state
  - Consider a sequence $\{X_1, X_2, \ldots, X_n\}$
  - Probabilities $P(X_n|X_{n-1}, X_{n-2}, \ldots, X_1) = P(X_n|X_{n-1})$
- Three key components:

1. **State space**: Set of all possible states
2. **Transition matrix**: Probabilities of moving from one state to another
3. **Random sequence**: A sequence of random variables $(X_0, X_1, X_2, \ldots)$

Markov State Diagram

Modelling Future Weather Conditions

- The **state space** is $\mathcal{S} = \{\text{Sunny}, \text{Cloudy}, \text{Rainy}\}$, represented as the vertices of the graph
- Edges represent the probability of moving from one state to another state
  - ➠ Example: if it is sunny today, 10% chance of being rainy tomorrow

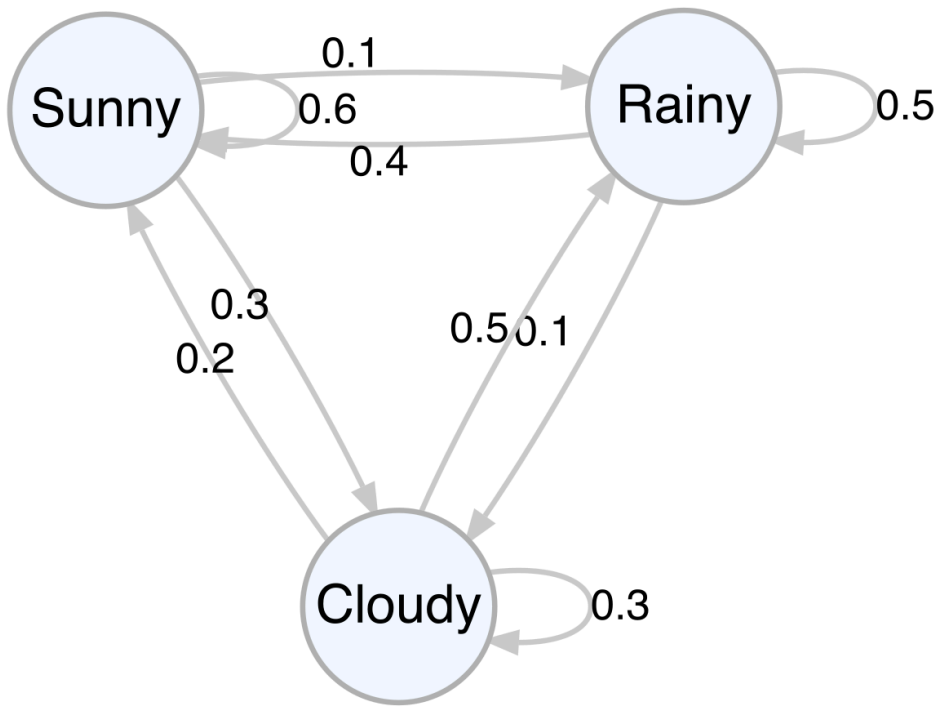


Transition Probability Matrix

- The **transition matrix** P is a 3×3 matrix:

$$P = \begin{pmatrix} 0.6 & 0.1 & 0.3 \\ 0.4 & 0.5 & 0.1 \\ 0.2 & 0.5 & 0.3 \end{pmatrix}$$

- Rows represent current state
- Columns represent next state
- P_{ij}: probability of transiting from state i to state j
- Row sums of P all equal one



Simulating Weather

- Start with an initial state (e.g., Sunny today).
- Use transition probabilities to simulate the next day's weather.
- Repeat the process to generate a sequence of weather conditions.

> **Note**
>
> This sequence is called a Markov Chain because each day's weather only depends on today's weather, not the entire history.

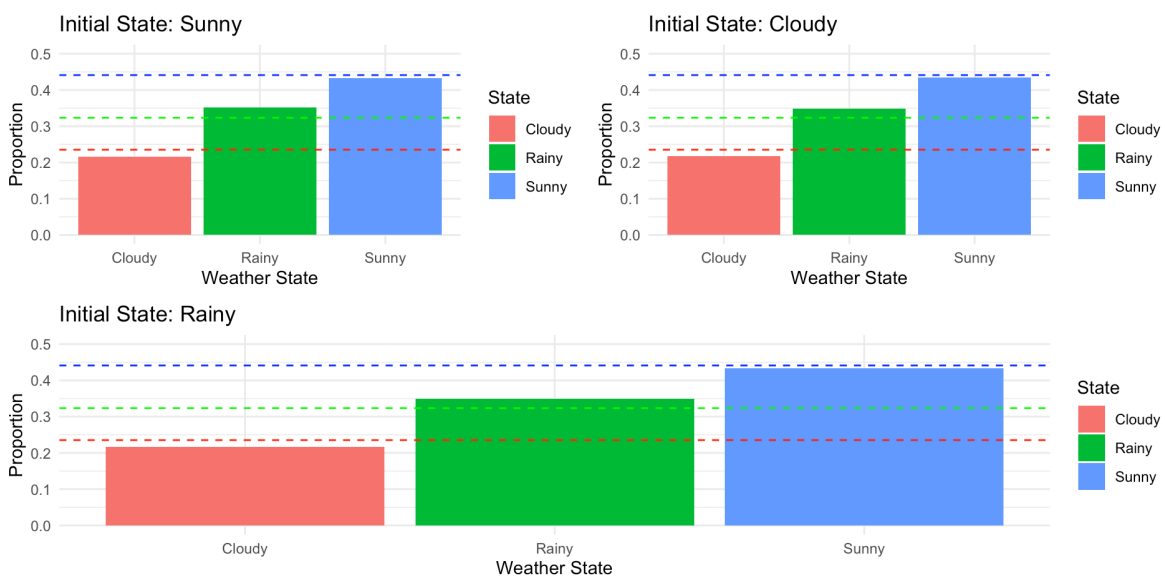
What Happens Over Time?

- As we simulate enough days, the weather sequence might start to stabilise.
- The system reaches a steady-state where the proportion of time spent in each state becomes constant.

```
weather
 Sunny  Rainy Cloudy
 0.432  0.352  0.216

weather
 Sunny  Rainy Cloudy
 0.434  0.348  0.218

weather
 Sunny  Rainy Cloudy
 0.434  0.350  0.216
```



Invariant (Stationary) Distribution

- An invariant distribution is a probability distribution π such that:

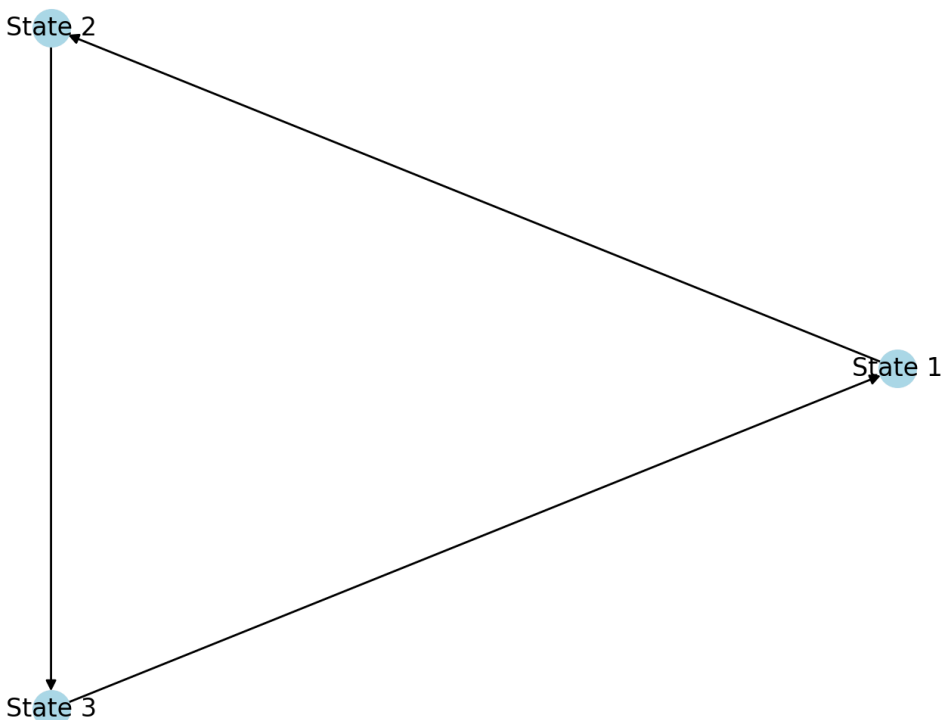
$\pi = \pi P$ and $\pi \mathbf{1} = 1$

- Interpretation:
  - ➠ If the system starts in distribution π, it stays in distribution π after any number of transitions. It represents the long-term behavior of the Markov Chain.
- This is satisfied if the Markov chain is:
  1. **Irreducible**
  2. **Aperiodic**

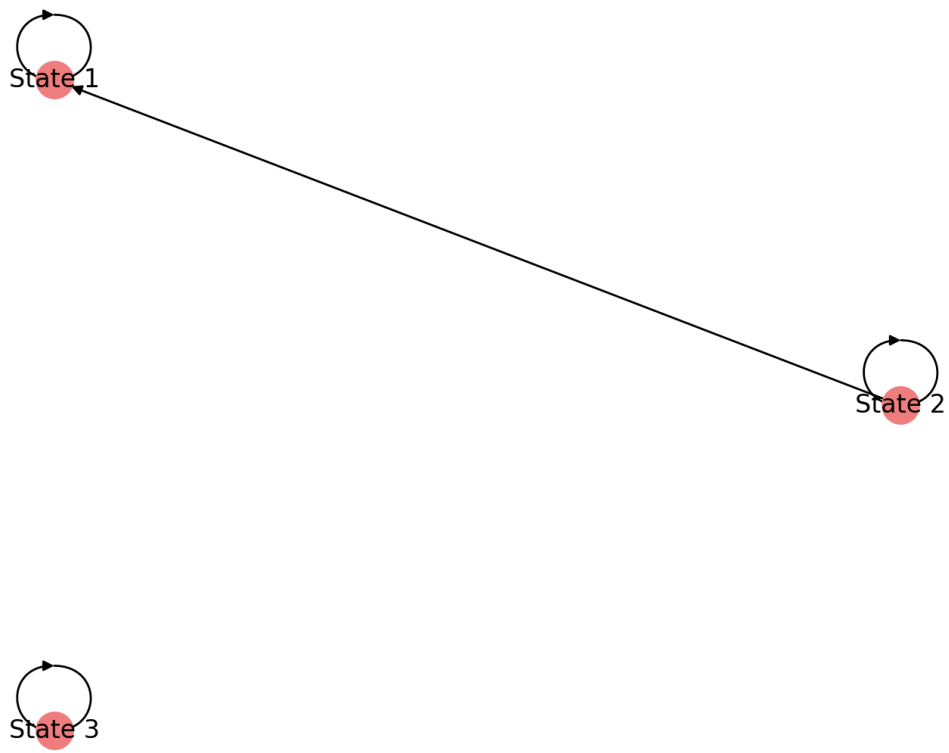
Irreducible

A Markov chain or transition matrix P is said to be irreducible if there is some way of getting from state i to state j, AND there is some way of getting from state j to state i, for all $i, j \in S$.

Communicating States



Non-Communicating States

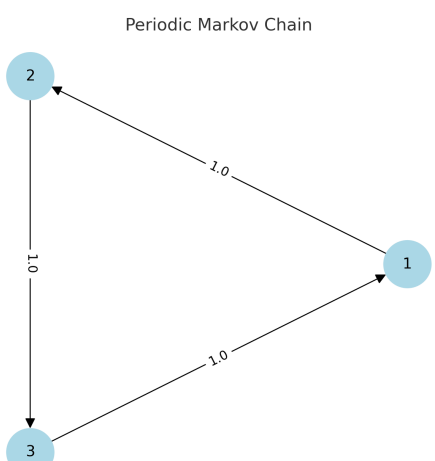


Periodic

Consider the following transition matrix:

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

- The chain cycles through: 1 → 2 → 3 → 1 → 2 → 3 → ...
- You can return to the starting state only after 3, 6, 9, ... steps.
- Thus, $\textbf{greatest common divisor } (\text{gcd})(3, 6, 9, ...) = 3$.
- $\Rightarrow$ All states are **periodic** with period 3.



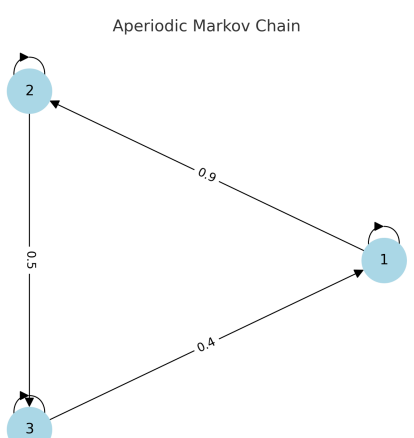
Aperiodic

Suppose we modify the transition matrix:

$$P = \begin{pmatrix} 0.1 & 0.9 & 0 \\ 0 & 0.5 & 0.5 \\ 0.4 & 0 & 0.6 \end{pmatrix}$$

- Now can return in 1, 2, 3, 4, ... steps.
- $\gcd(1, 2, 3, 4, \ldots) = 1$.
- $\Rightarrow$ States are now **aperiodic**.

The state is **aperiodic**, meaning returns can happen more irregularly without being locked into a repeating cycle.



The Use of Invariant Markov Chain

- Similar to Monte Carlo, our main interest here continues to be the simulation of $X_1, X_2, \cdots$, that follow (or approximately follow) the distribution with density $f(x)$ where directly simulating from $f(x)$ is difficult.
- This problem is ubiquitous in Bayesian statistics
  - Frequentist statistics: there exist true parameters ($\boldsymbol{\theta}$ is fixed but unknown)
  - Bayesian statistics: all model parameters are random variables
- For Bayesian statistics, the model parameters $\boldsymbol{\theta}$ has a prior distribution
- Given data $\mathbb{D}$, we want to compute the posterior distribution $p(\boldsymbol{\theta}|\mathbb{D})$

$$p(\boldsymbol{\theta}|\mathbb{D}) = \frac{p(\mathbb{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\mathbb{D}|\boldsymbol{\theta'})p(\boldsymbol{\theta'})d\boldsymbol{\theta'}}$$

  - Generally, it's very difficult to exactly compute $p(\boldsymbol{\theta}|\mathbb{D})$
  - Sample from $p(\boldsymbol{\theta}|\mathbb{D})$: using MCMC
  - Idea: construct a Markov chain such that its stationary distribution is the desired posterior distribution $p(\boldsymbol{\theta}|\mathbb{D})$

MCMC - Metropolis-Hasting algorithm

Metropolis-Hastings algorithm - Intuition

- Imagine you are a politician trying to visit all the town halls in your electorate.
- Imagine there are **four town halls**: A, B, C, and D.
- Each town hall has a **different number of voters**.
- Your goal: Spend time at each town hall **proportional** to its voter population.

Town Hall	Number of Voters	Proportion (Target $f(x)$)
A	500	0.25
B	1000	0.5
C	300	0.15
D	200	0.10

Metropolis-Hastings algorithm - Workflow

1. **Start** at a random town hall.
2. **Propose** moving to a randomly chosen new town hall.
3. **Accept or reject** the move:
    - If the new hall has **more voters ➔ Always accept**.
    - If fewer voters **➔ Accept with probability**

$$\text{Accept probability} = \min\left(1, \frac{\text{Number of people in new town hall}}{\text{Number of people in current town hall}}\right)$$

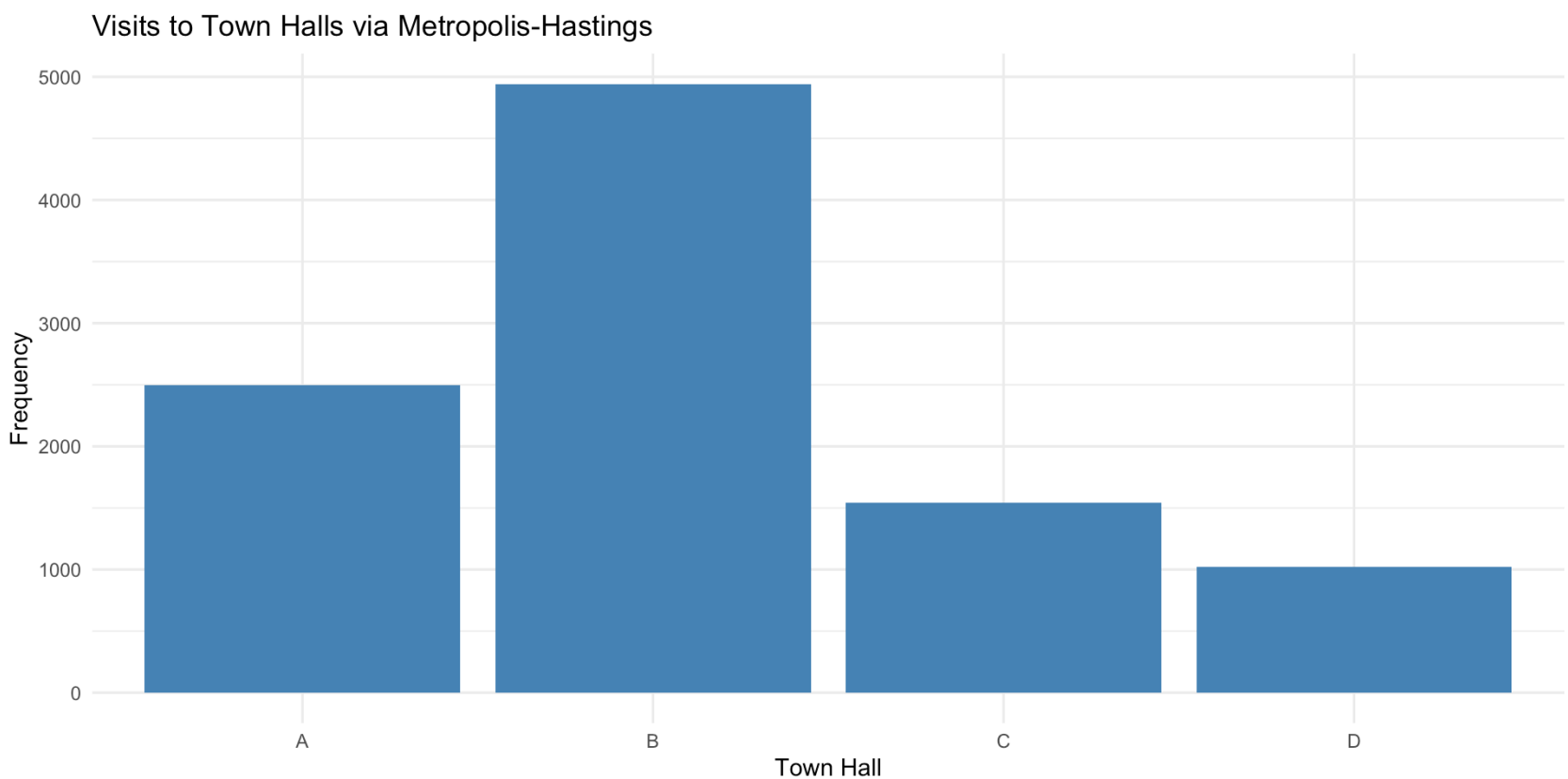
4. **Repeat** the process for a large number of steps (e.g., 10,000).
5. **Track** the proportion of time spent in each town hall.

R Simulation

▶ Code

```
     A      B      C      D 
0.2498 0.4941 0.1540 0.1021 
```

▶ Code



Metropolis-Hastings algorithm - Explanation

- Similar to the acceptance-rejection method
  - ➟ It simulates a trial state
  - ➟ Accepts or rejects the trial state according to some random mechanism
- Uses the Markov chain because each trial state depends on the previous state – Almost memory-less system
- Aim is to construct a Markov chain $\{X_t,\ t = 0, 1, \ldots\}$ such that the limiting distribution is $f(x)$

Metropolis-Hastings algorithm - Generalisation

We want to **simulate samples** from a complicated **target distribution** $f(x)$, which might be difficult to sample from directly.

Basic Steps

- **Start** with an initial state X_0.
- **Propose** a new point Y based on the current point X_t, using a **proposal distribution** $q(y|x)$.
- **Accept or reject** the proposal Y according to an acceptance probability $\alpha(X_t, Y)$.

This way, even if q isn't perfect, we still eventually generate samples that have the correct distribution $f(x)$ when it reaches stationarity.

Workflow

For $t = 0, 1, \ldots, N-1$ do:

- Draw $Y \sim q(x|X_t)$
- Calculate acceptance probability $\alpha(X_t, Y)$
- Define $\alpha(x, y) = \min\left\{\frac{f(y)q(x|y)}{f(x)q(y|x)}, 1\right\}$
- Draw $U \sim \text{Uniform}(0, 1)$
- If $U \leq \alpha$ then $X_{t+1} \leftarrow Y$ else $X_{t+1} \leftarrow X_t$

Return $X_1, X_2, \ldots, X_N$

Symmetric Proposal Function

- If the proposal density function is symmetric
  - $q(y|x) = q(x|y)$
  - The acceptance probability has a simpler form
  - The MCMC algorithm is also known as a **random walk sampler**
- One common choice of a symmetric proposal function is just the Gaussian function, i.e., $q(x) \sim \mathcal{N}(x_t, \sigma)$
- The choice of σ affects how quickly the state space is explored

Applications of MCMC

Where would you use MCMC

- One common application of MCMC is to draw from the posterior distribution in Bayesian statistical methods.

$$P(A|B) = \frac{P(B|A)}{P(B)} P(A)$$

- Posterior: The likelihood of A occurring given B has occurred.
- Likelihood ratio: The support B provides for A.
- Prior: The probability of A before any data are gathered.

The posterior distribution

- Can use the Bayes rule for modelling and data.

$$p(\boldsymbol{\theta}|\mathbb{D}) = \frac{p(\mathbb{D}|\boldsymbol{\theta})}{p(\mathbb{D})} p(\boldsymbol{\theta})$$

- Posterior: The likelihood of $\boldsymbol{\theta}$ occurring given the data $\mathbb{D}$.
- Likelihood ratio: The support $\mathbb{D}$ provides for $\boldsymbol{\theta}$.
- Prior: The probability of $\boldsymbol{\theta}$ before any data are gathered.
- Typically $\boldsymbol{p(\mathbb{D})}$ is a difficult integral to evaluate:

$$p(\mathbb{D}) = \int p(\mathbb{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) \, d\boldsymbol{\theta}.$$

MCMC for Estimating Regression Models

$$p(\boldsymbol{\theta}|\mathbb{D}) = \frac{p(\mathbb{D}|\boldsymbol{\theta})}{p(\mathbb{D})}p(\boldsymbol{\theta})$$

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \varepsilon; \qquad \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

- $\boldsymbol{\theta} = \{\beta_0, \beta_1, \ldots, \beta_p, \sigma^2\}$
- $\mathbb{D} = \{Y_1, Y_2, \ldots, Y_n, X_{11}, X_{12}, \ldots, X_{np}\}$
- $p(\mathbb{D}|\boldsymbol{\theta})$ will be Gaussian
- $p(\mathbb{D})$ is not so easy to compute
- $p(\boldsymbol{\theta})$ is the chosen prior

R Simulation

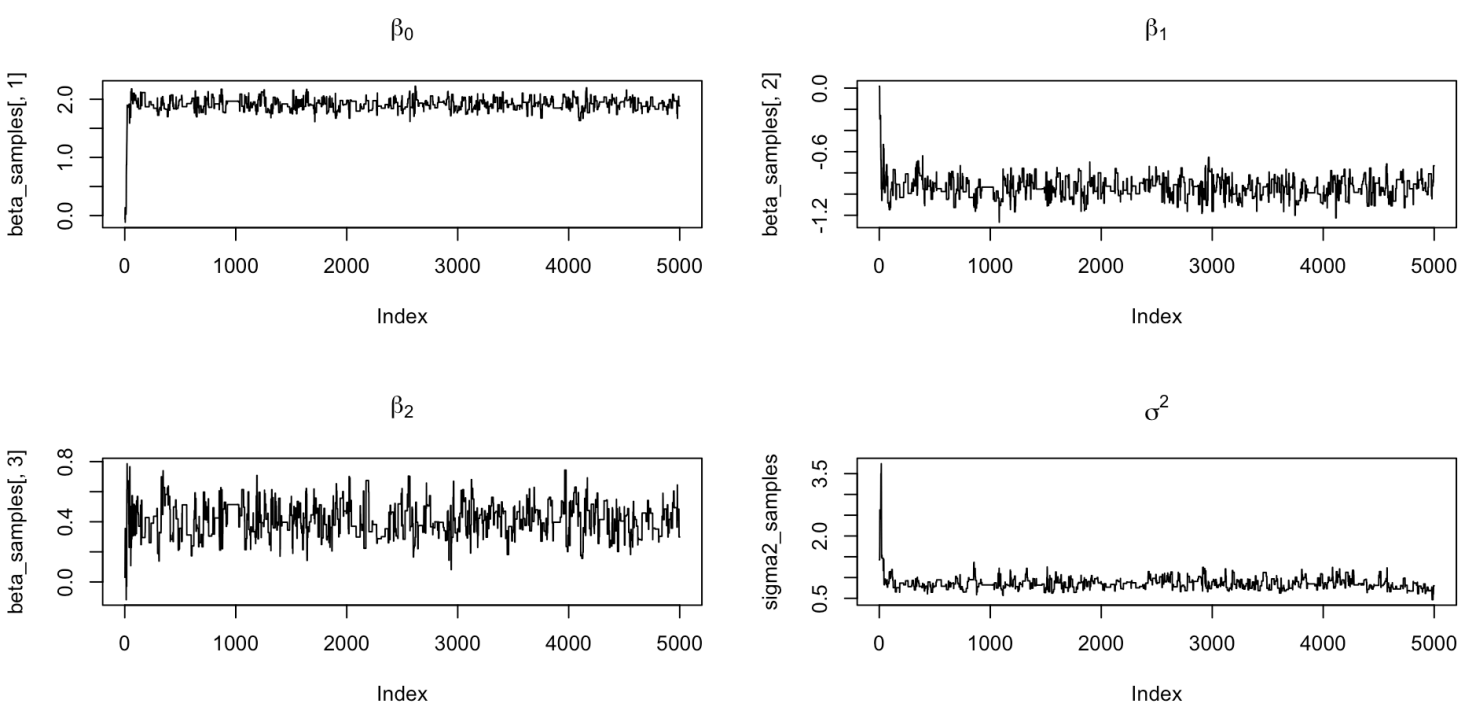
▶ Code

▶ Code

```
[1]  1.9086725 -0.9393679  0.4118723
```

▶ Code

```
[1] 0.859761
```



Estimating posterior with MCMC

- Recall the acceptance probability in the Metroplos-Hasting algorithm is

$$\alpha = \min\left\{\frac{f(y)q(x|y)}{f(x)q(y|x)}, 1\right\}$$

- Assume the proposal density is symmetric, i.e., $q(x|y) = q(y|x)$
- My target density f is now the posterior distribution, i.e., $p(\boldsymbol{\theta}|\mathbb{D})$
- Then in the Metropolis-Hastings algorithm, we only need to calculate

$$\alpha = \frac{p(\boldsymbol{\theta}'|\mathbb{D})}{p(\boldsymbol{\theta}|\mathbb{D})} = \frac{p(\mathbb{D}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')}{p(\mathbb{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}$$

- Since $p(\mathbb{D})$ doesn't depend on $\boldsymbol{\theta}$, it cancels out on the right hand side of the above formula and hence it isn't included in the formula

MCMC for Estimating $P(\text{Head})$ of a coin flip

Observe a series of coin flips

H, T, H, H, T, H, T, H, H, T, H, H, H, T, H, H, H, T, H, H, ...

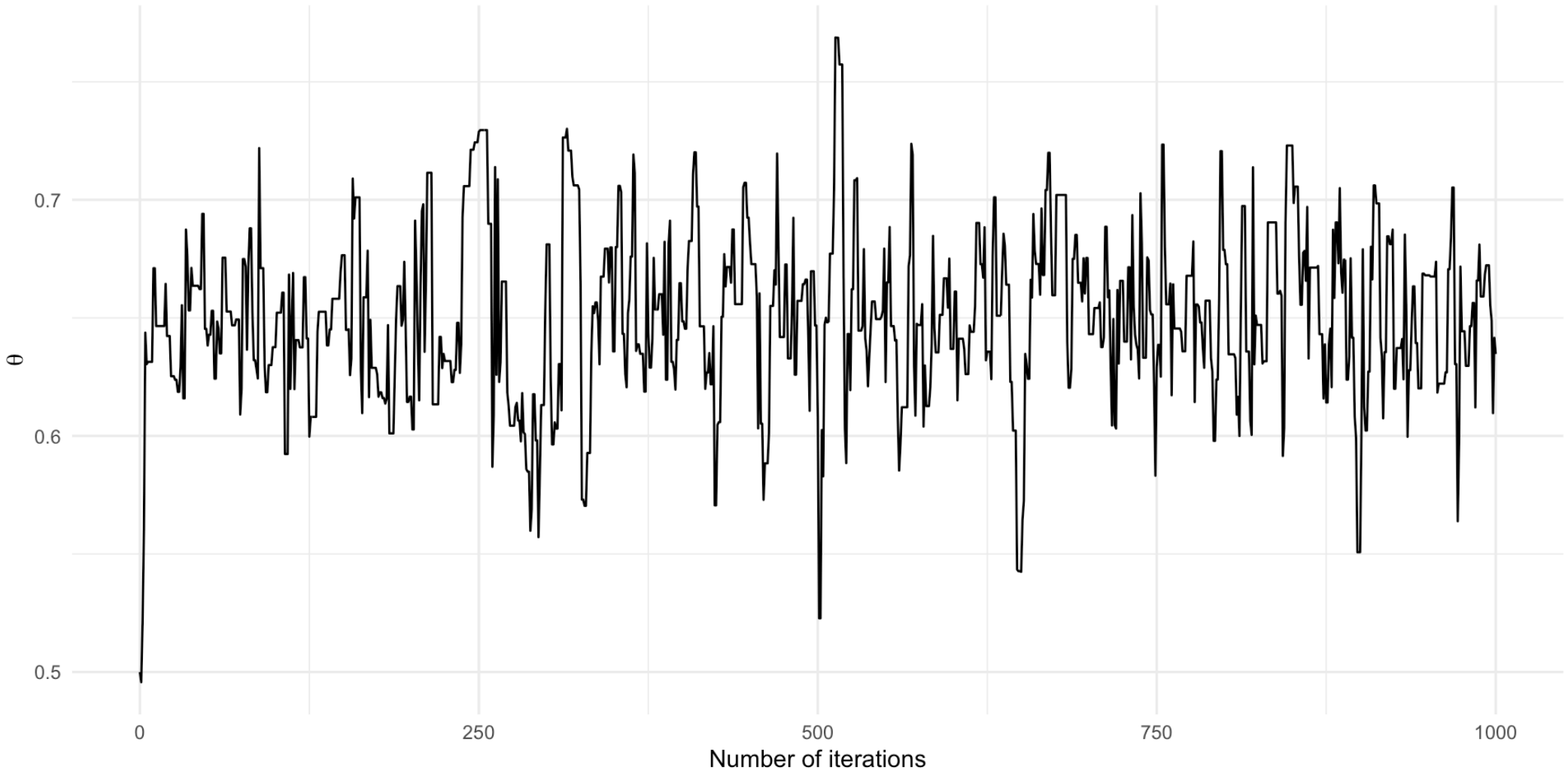
> Can you estimate the $P(\text{Head})$ of this coin?

> Assume you don't know anything about this coin and it could be biased!

- $\theta = P(\text{Head})$
- Frequentist perspective: θ is fixed but unknown
- Bayesian perspective: θ is a random variable; one can compute the likelihood of θ given the observed data
- θ_0 set as an initial point
- Compute θ_t as a Markov chain
- Use MCMC algorithm to estimate the posterior distribution of θ

MCMC for Estimating $P(\text{Head})$ of a coin flip

Assume we use a flat prior over $(0, 1)$ for θ



MCMC practical considerations

- The samples at the start of the MCMC, before the algorithm converges to the true distribution are known as the burn-in period
  - ➟ It should be discarded
- The samples generated by MCMC are correlated since they are from a Markov chain
  - ➟ Previously, many practitioners advocated **thinning** the samples by taking say every k^{th} sample
  - ➟ This was done for a few reasons historically
    - ➟ Reduce correlations and compute standard errors more easily
    - ➟ Less space needed to store the chain