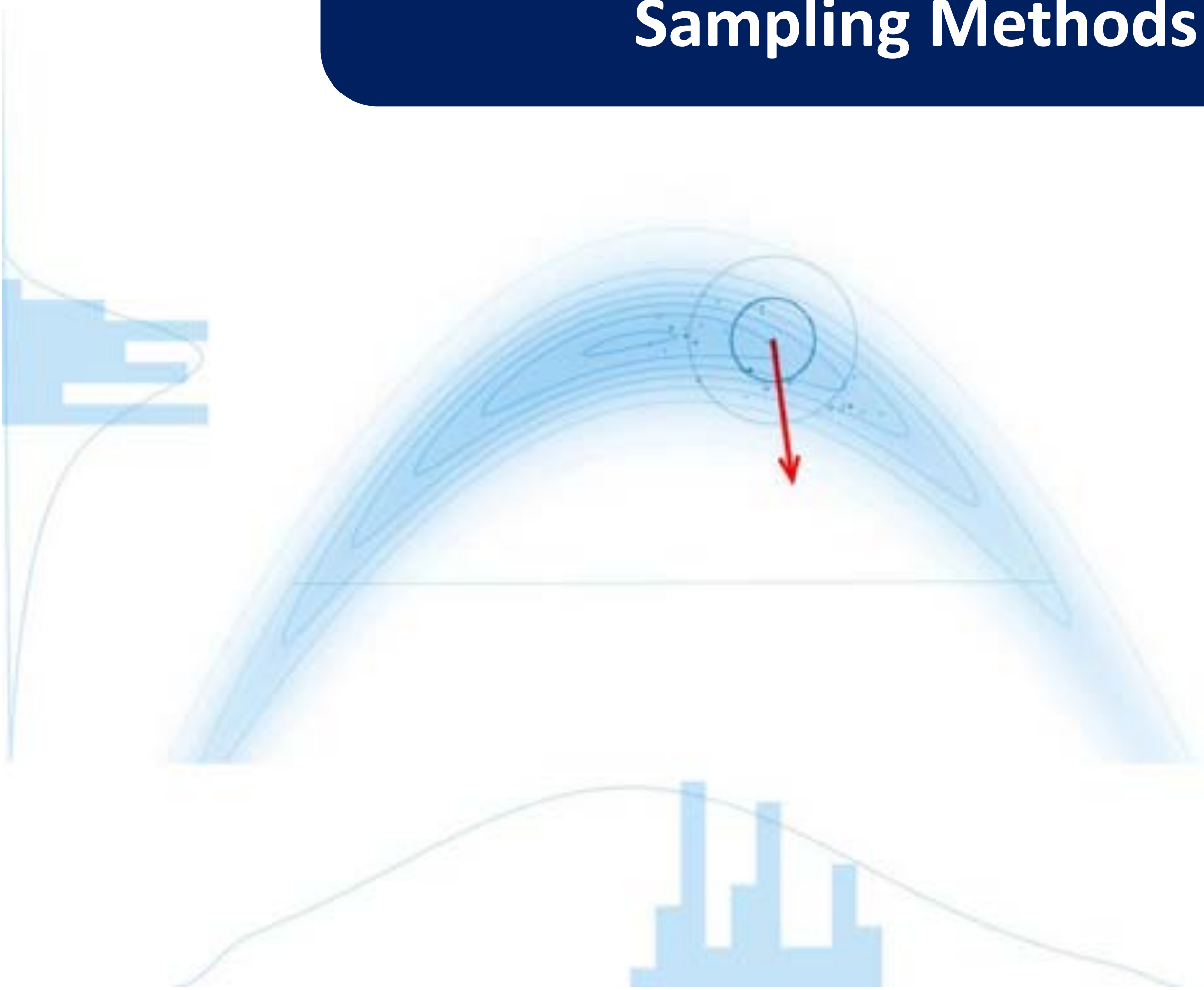
# An Introduction to Monte Carlo Statistical Sampling Methods



Zhe Wei **KHO**
Jiacheng **XU**

# The Bayesian Paradigm

**The Posterior Function**

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)\pi(\theta)}{\int_\Omega p(\mathcal{D}|\theta)\pi(\theta)\,\mathrm{d}\theta}$$

This is the starting point not the ending point for Bayesian modelling!

- Posterior mean (estimate parameter values)
- Credible Intervals (quantifying uncertainty)
- Predictive posterior (predict new observations)
- Bayesian Model Selection
- Evaluating Utility Functions

All involve calculating **integrals over the posterior**

**Integration is HARD!**

# What Do 'Real' Posterior Distributions Look Like?

## Example: Statistical Model of Neuronal Assemblies

**Likelihood**

$$P(t, \omega, s \,|\, \vec{p}, \vec{\lambda}, \vec{n}) = \left(\prod_{i=1}^{N} n_{t_i}\right) \cdot \left(\prod_{\mu=1}^{A}\prod_{k=1}^{M} p_\mu^{\omega_{k\mu}} (1 - p_\mu)^{1-\omega_{k\mu}}\right) \cdot \left(\prod_{i=1}^{N}\prod_{k=1}^{M} [\lambda_{t_i}(\omega_{kt_i})]^{s_{ik}} [1 - \lambda_{t_i}(\omega_{kt_i})]^{(1-s_{ik})}\right)$$

**Priors**

$$p_\mu \sim \mathrm{Beta}\left(\alpha_\mu^{(p)}, \beta_\mu^{(p)}\right)$$

$$\lambda_\mu(z) \sim \mathrm{Beta}\left(\alpha_{z,\mu}^{(\lambda)}, \beta_{z,\mu}^{(\lambda)}\right)$$

$$\{n_1, \cdots, n_A\} \sim \mathrm{Dir}\left(\alpha_1^{(n)}, \cdots, \alpha_A^{(n)}\right)$$

Unlike textbook examples, Bayesian models that show up in research are typically **analytically intractable** and **high dimensional**!

Diana G., Sainsbury T.T.J. & Meyer M.P. (2019). Comput Biol. 15 (10)

# The World of Monte Carlo

Throwing dice won't solve your financial woes…but maybe it can solve your mathematical ones?

# Sampling

Suppose that we want to calculate the **mean value** of the **roll of a fair six-sided die**

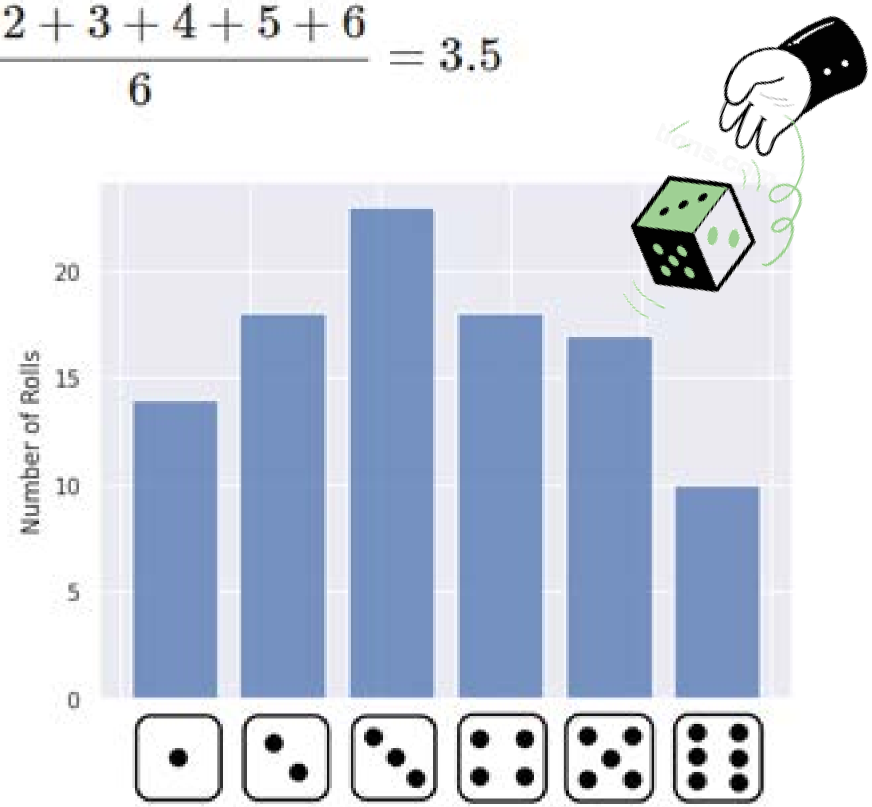
If we have **full information** about p(x), we can simply compute

$$\mathbb{E}_p(X) = \sum_{x \in \mathcal{X}} x p(x) = \frac{1+2+3+4+5+6}{6} = 3.5$$

If we don't, then we can **estimate** it by **drawing samples from p(x)**

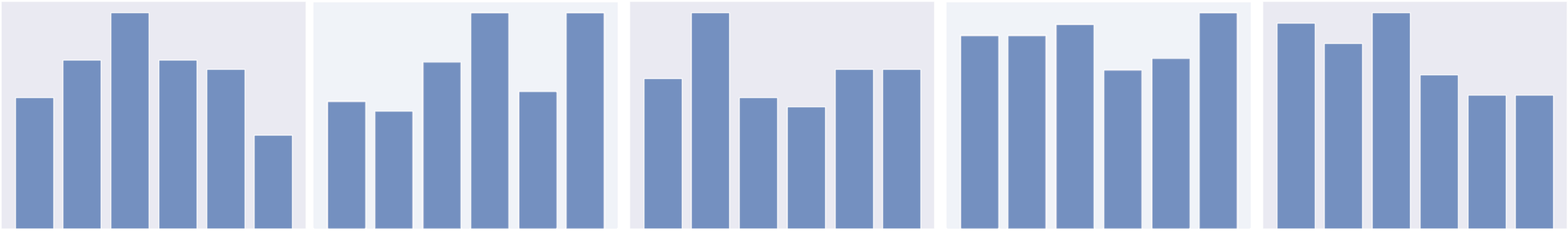
$$\bar{X} = \frac{1}{N} \sum_{i=1}^{N} x_i = 3.36$$

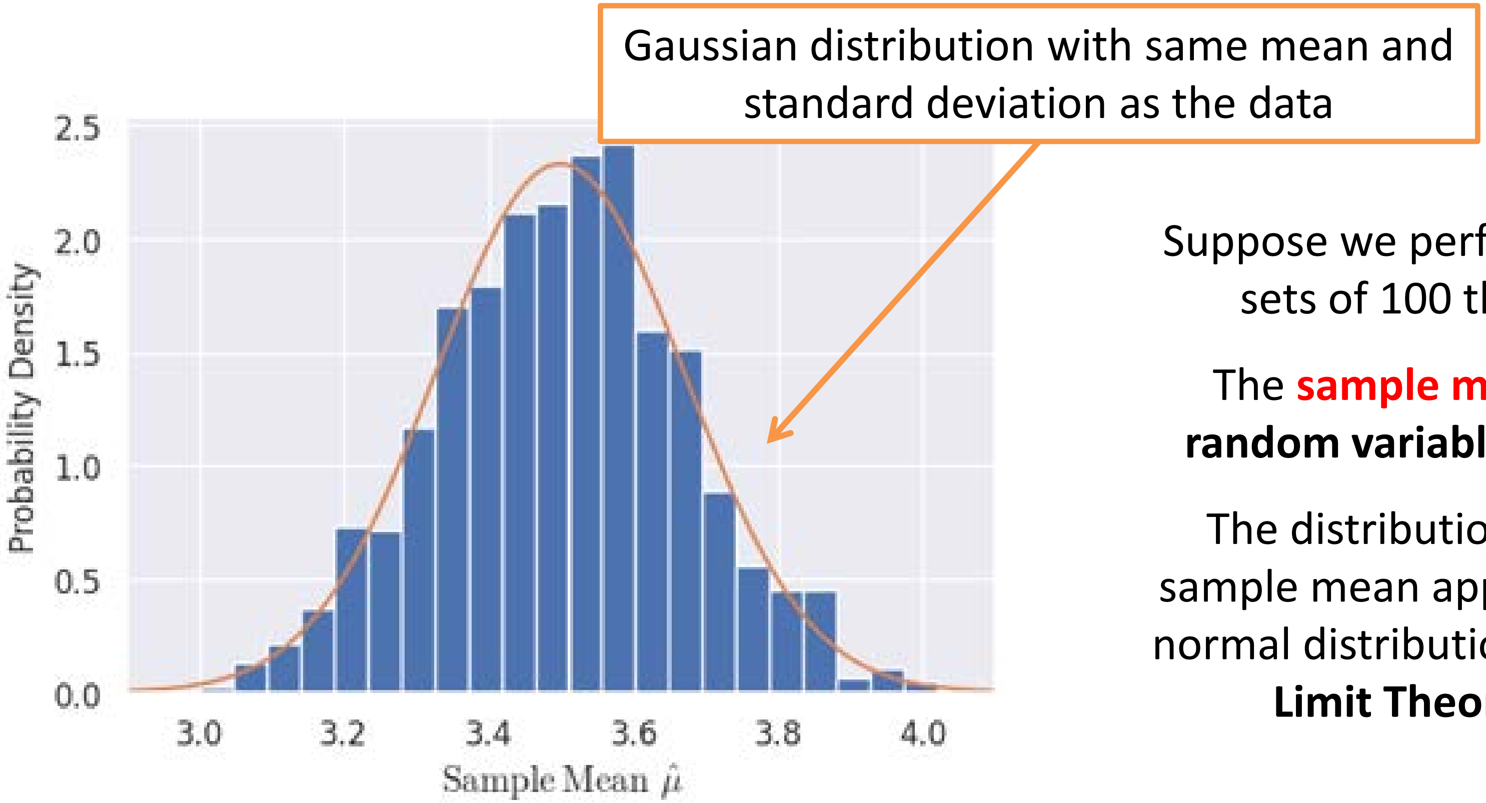
This is the **sample mean**




| | | | | | |
|---|---|---|---|---|---|
| 14 | 18 | 23 | 18 | 17 | 10 |

# Sampling






Suppose we perform many sets of 100 throws.

The **sample mean** is a **random variable** as well.

The distribution of the sample mean approaches a normal distribution **(Central Limit Theorem)**

# Law of the Unconscious Statistician

**Law of the Unconscious Statistician**

The expectation value of a function $g$ of a random variable $x$ is given by

$$\mathbb{E}\left[g(X)\right] = \sum_{x \in \mathcal{X}} g(x) p_X(x) \qquad \text{(discrete)}$$

$$\mathbb{E}\left[g(X)\right] = \int_{x \in \mathcal{X}} g(x) p_X(x)\, \mathrm{d}x \qquad \text{(continuous)}$$

(actually a nontrivial result but we will not prove it)

Strictly speaking, the expectation value of g(X) should be taken wrt the PDF of g(X).

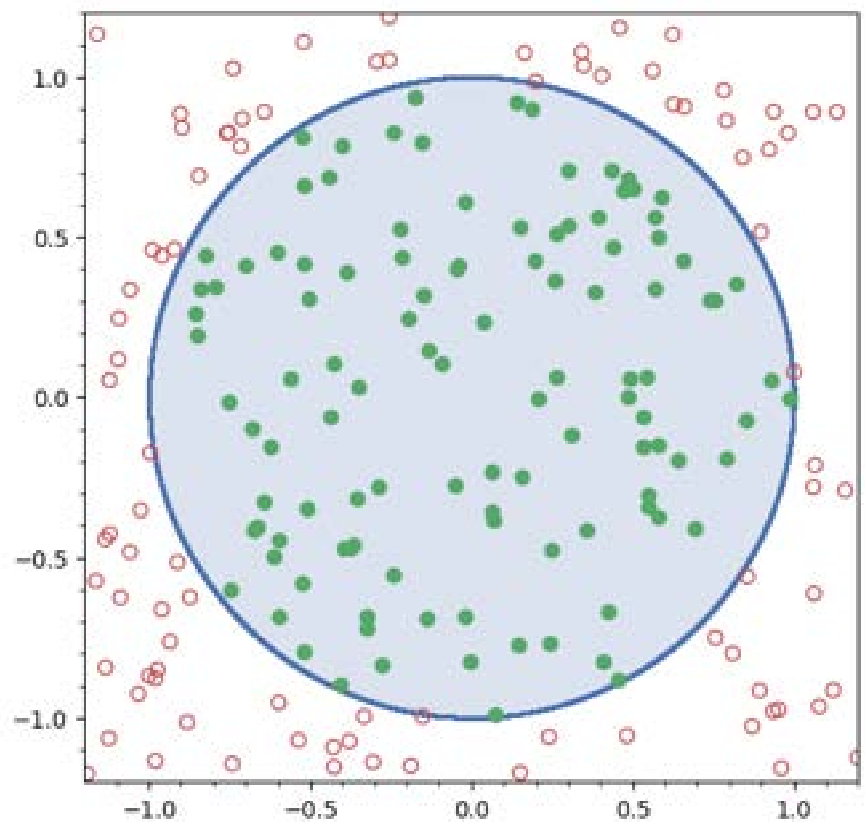
We use this result pretty much every time we compute the posterior expectation of some function

# Rejection Sampling

**IDEA:** As silly as it may sound, simply **prune the samples** until they fit the desired distribution!

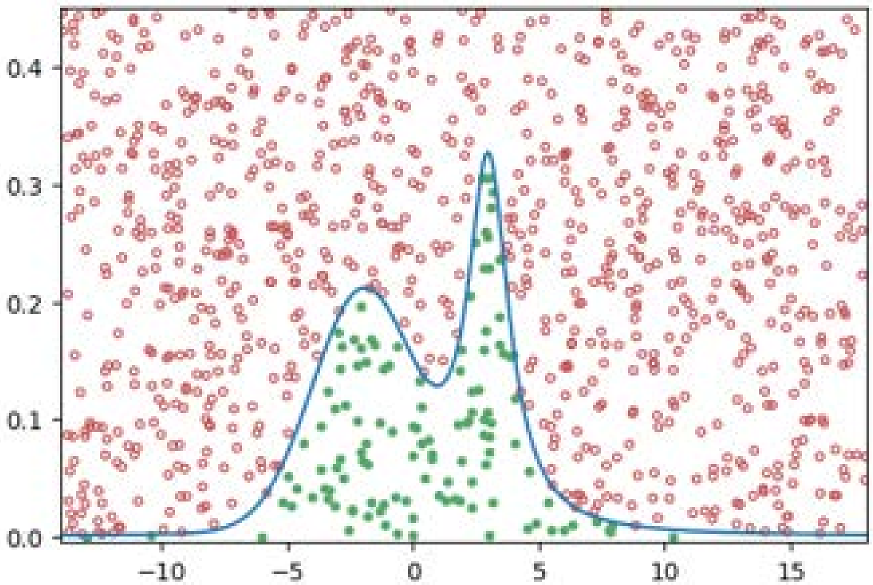
**Example: Selecting Samples Lying within the Unit Circle**

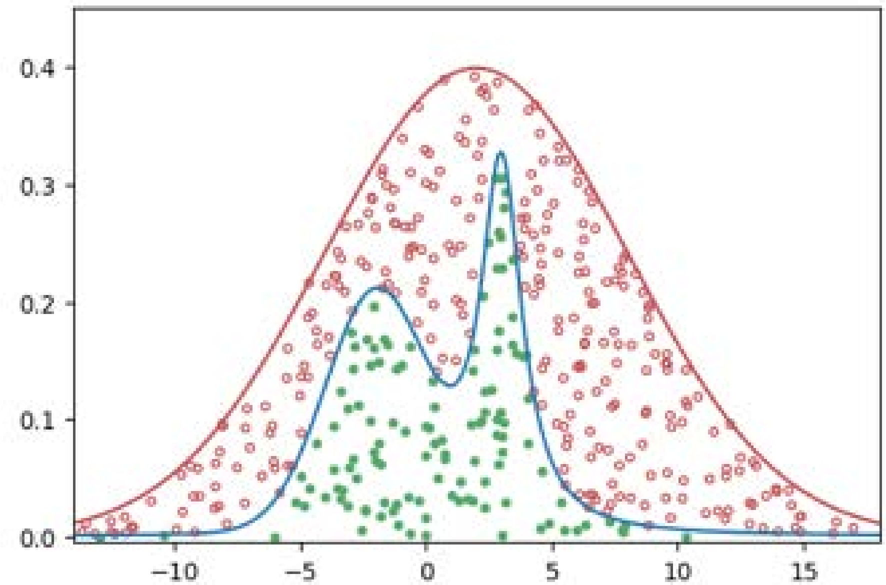
*"If it didn't hit the dartboard I didn't throw it. I didn't see anything and neither did you."*



# Rejection Sampling

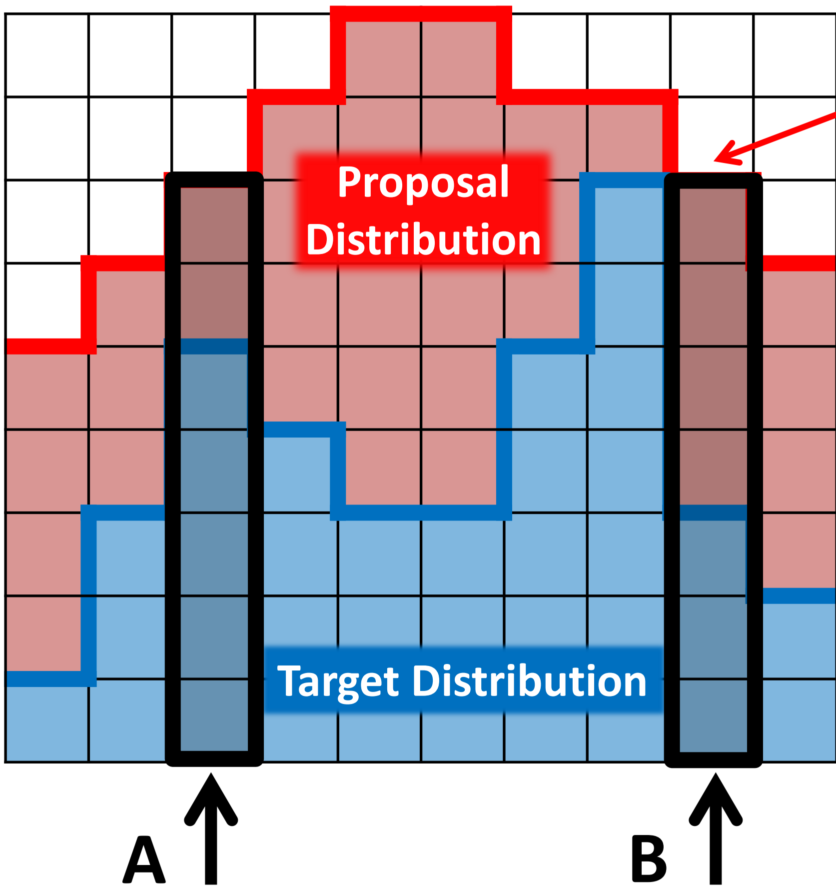
We can imagine doing the same thing by treating the **target PDF** as a **target region**




Choosing a **proposal distribution** that more tightly bounds the target results in more efficient ‘shooting’

# How Do We Actually ‘Shoot’?



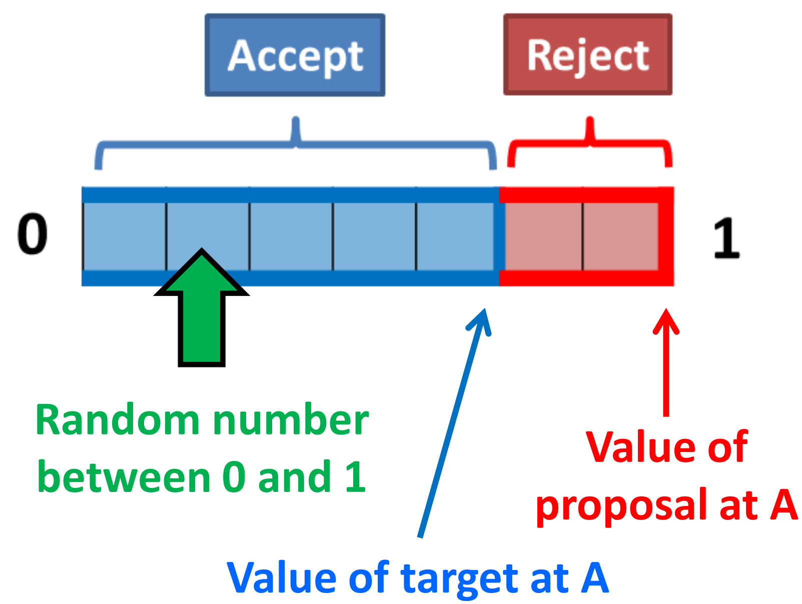

**Height** is proportionate to the **number of times** we land at that point

**SOLUTION**

**Only keep**:

5/7 of the samples at A

3/7 of the samples at B



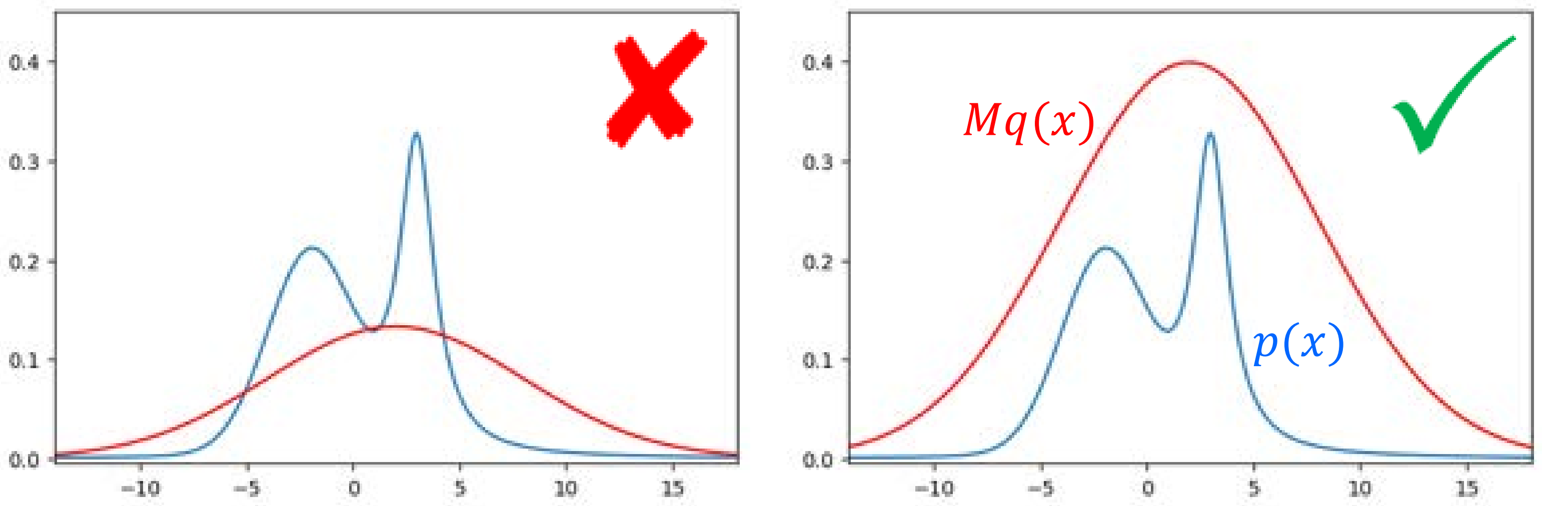

We **land** on A and B **equally often** but we **want** A and B to be in the **ratio 5:3**

We need to **reject** the **excess density**, and do so **probabilistically**

# Rejection Sampling

**Algorithm: Rejection Sampling**

① Choose an appropriate proposal distribution $q$ and scale factor $M$

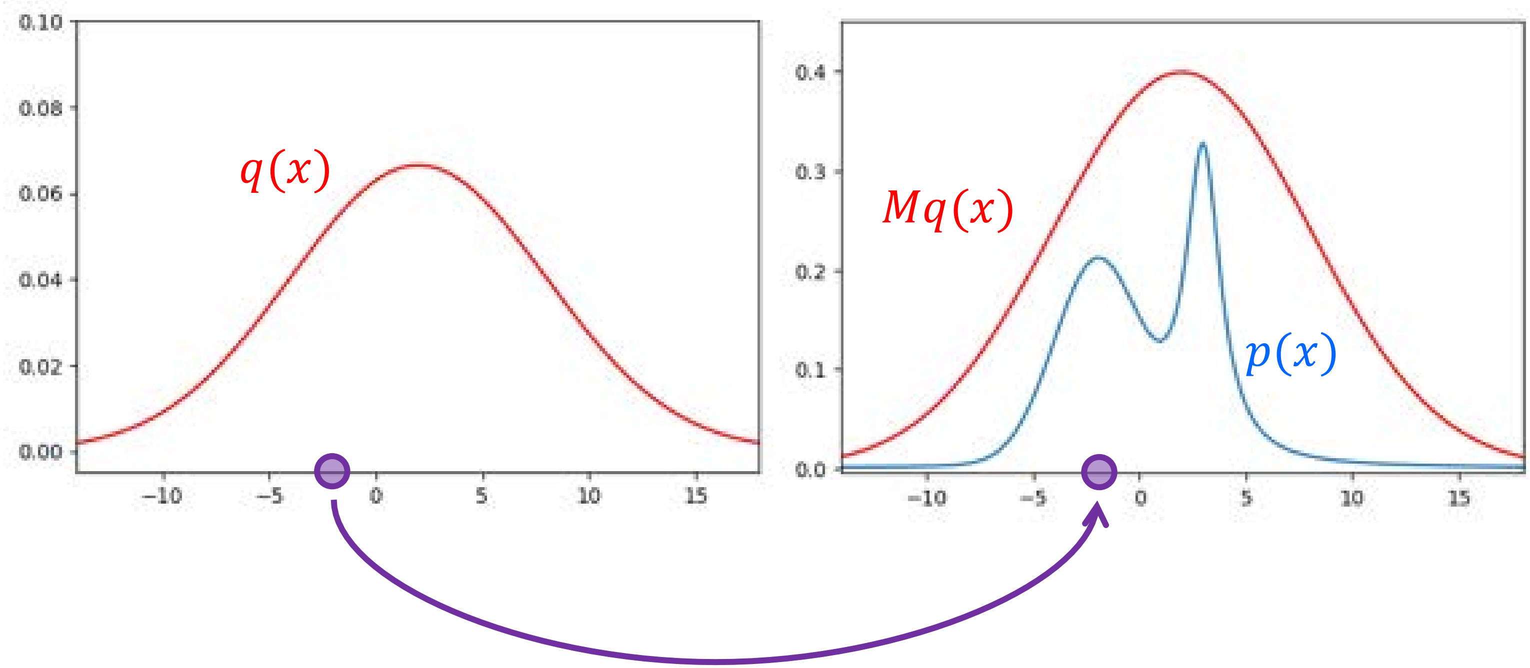



**Rescaled proposal distribution** should be **larger** than the **target** – we can discard but not add!

# Rejection Sampling

**Algorithm: Rejection Sampling**

② Simulate a candidate sample $x \sim q$ from the proposal density

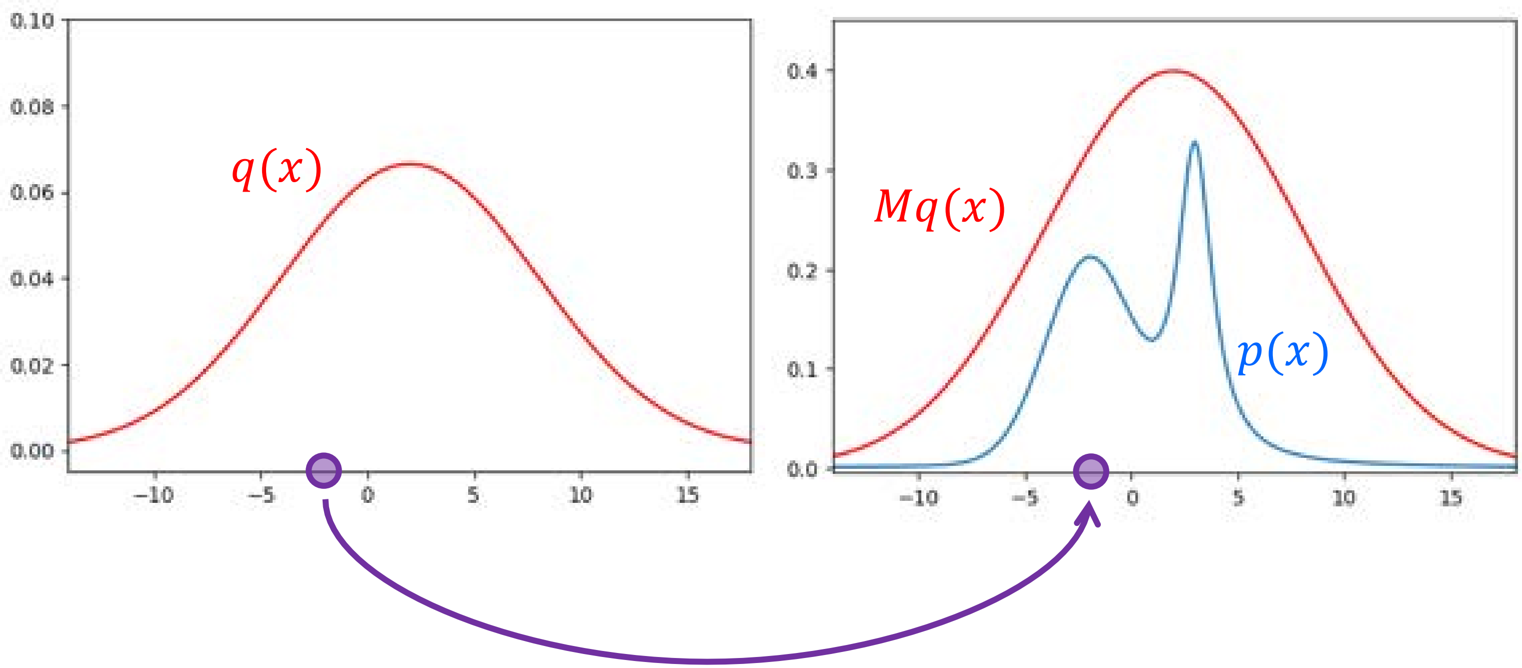


# Rejection Sampling

**Algorithm: Rejection Sampling**

① Choose an appropriate proposal distribution $q$ and scale factor $M$

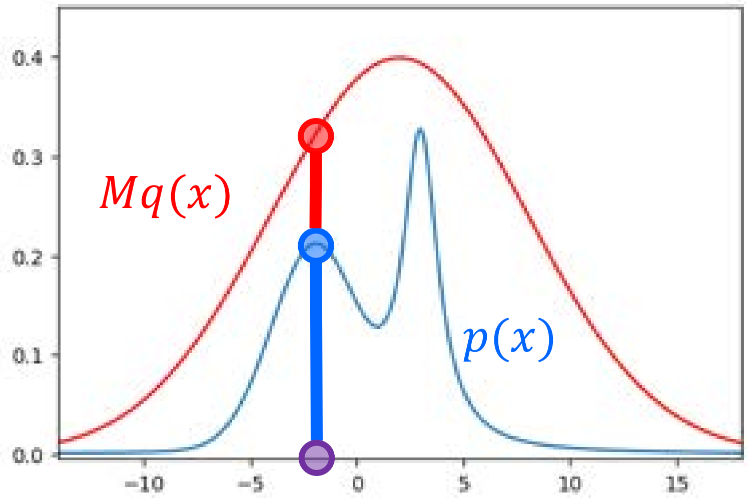
② Simulate a candidate sample $x \sim q$ from the proposal density

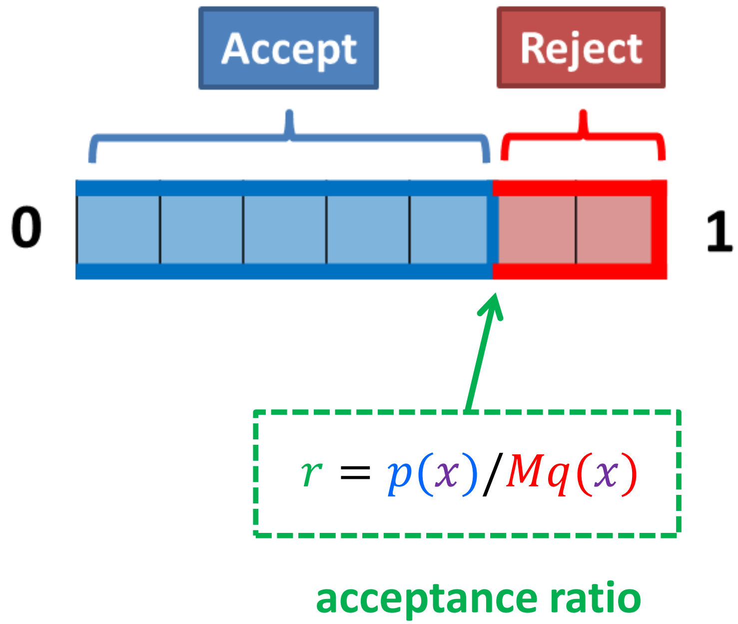


# Rejection Sampling

**Algorithm: Rejection Sampling**

③ Calculate the **acceptance ratio** $r = p(x)/Mq(x)$

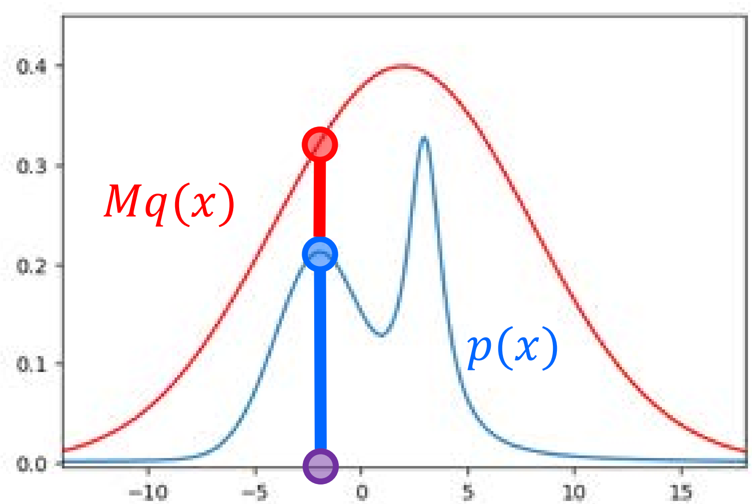


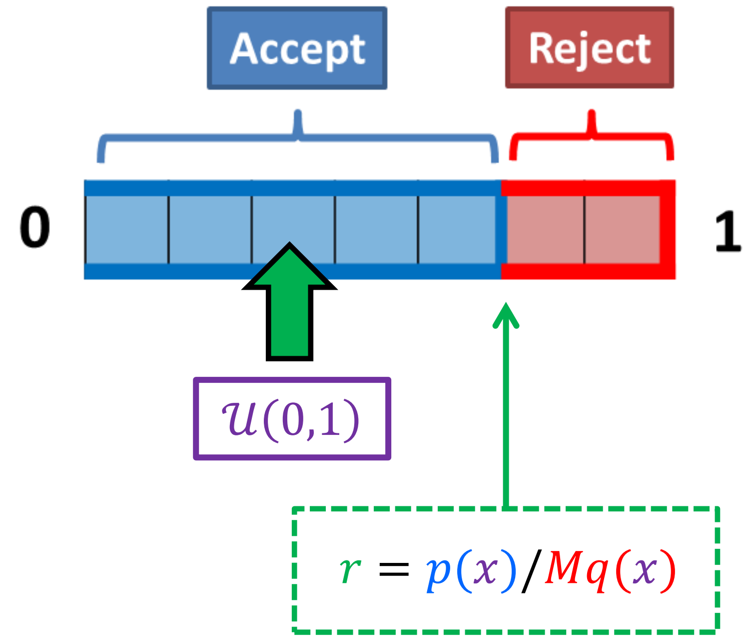



# Rejection Sampling

## Algorithm: Rejection Sampling

④ Simulate a random number $u \sim \mathcal{U}(0,1)$. If $u < r$, **accept** the sample. Otherwise, **reject** it.



# Rejection Sampling

**Algorithm: Rejection Sampling**

① Choose an appropriate proposal distribution $q$ and scale factor $M$

② Simulate a candidate sample $x \sim q$ from the proposal density

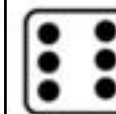
③ Calculate the **acceptance ratio** $r = p(x)/Mq(x)$

④ Simulate a random number $u \sim \mathcal{U}(0,1)$. If $u < r$, **accept** the sample. Otherwise, **reject** it.

⑤ Repeat until the desired number of samples are obtained.

# Importance Sampling – A Dice Throwing Example

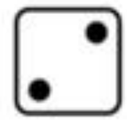
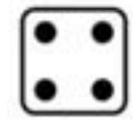
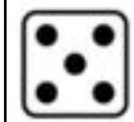

## Regular Six-Sided Dice

| |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| **# Rolls** | **14** | **18** | **23** | **18** | **17** | **10** |
| | | | | | | |
| | | | | | | |
| **Contribution** | **14** | **36** | **69** | **72** | **85** | **60** |

$$\mathbb{E}(X) = \frac{14 + 36 + 69 + 72 + 85 + 60}{14 + 18 + 23 + 18 + 17 + 10} = 3.36$$

# Importance Sampling – A Dice Throwing Example


## Regular Six-Sided Dice

| |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| **# Rolls** | **14** | **18** | **23** | **18** | **17** | **10** |
| **Weights** | **1** | **1** | **1** | **1** | **1** | **1** |
| **Effective # Rolls** | **14** | **18** | **23** | **18** | **17** | **10** |
| **Contribution** | **14** | **36** | **69** | **72** | **85** | **60** |

How **many times** we **count each dice roll**! We will see what tricks we can play with this soon…

# “Throwing” a Four-Sided Dice!


**Four-Sided Dice**

| | | | | | | |
|---|---|---|---|---|---|---|
| **# Rolls** | **14** | **18** | **23** | **18** | **17** | **10** |
| **Wei** | | | | **1** | **1** | **1** |
| **Effective # Rolls** | **14** | **18** | **23** | **18** | **17** | **10** |
| **Contribution** | **14** | **36** | **69** | **72** | **85** | **60** |

What if we simply don’t count rolls that land on 5 or 6?!

How can we ‘throw’ a **four-sided dice** when we only have a **six-sided dice**?

# “Throwing” a Four-Sided Dice!


## Four-Sided Dice

Change the weights to 0!

| | | | | | | |
|---|---|---|---|---|---|---|
| # Rolls | 14 | 18 | 23 | 18 | 17 | 10 |
| Weights | 1 | 1 | 1 | 1 | *0* | *0* |
| Effective # Rolls | 14 | 18 | 23 | 18 | *0* | *0* |
| Contribution | 14 | 36 | 69 | 72 | *0* | *0* |

$$\mathbb{E}(X) = \frac{14 + 36 + 69 + 72 + 0 + 0}{14 + 18 + 23 + 18 + 0 + 0} = 2.62$$

# “Throwing” a ‘Cheat’ Dice!


**Cheat Dice**

| | ⚀ | ⚁ | ⚂ | ⚃ | ⚄ | ⚅ |
|---|---|---|---|---|---|---|
| # Rolls | 14 | 18 | 23 | 18 | 17 | 10 |
| Weights | 1 | 1 | 1 | 1 | 1 | 1 |
| Effective # Rolls | 14 | 18 | 23 | 18 | 17 | 10 |
| Contribution | 14 | 36 | 69 | 72 | 85 | 60 |

Present 2 times

Not present

# “Throwing” a ‘Cheat’ Dice!

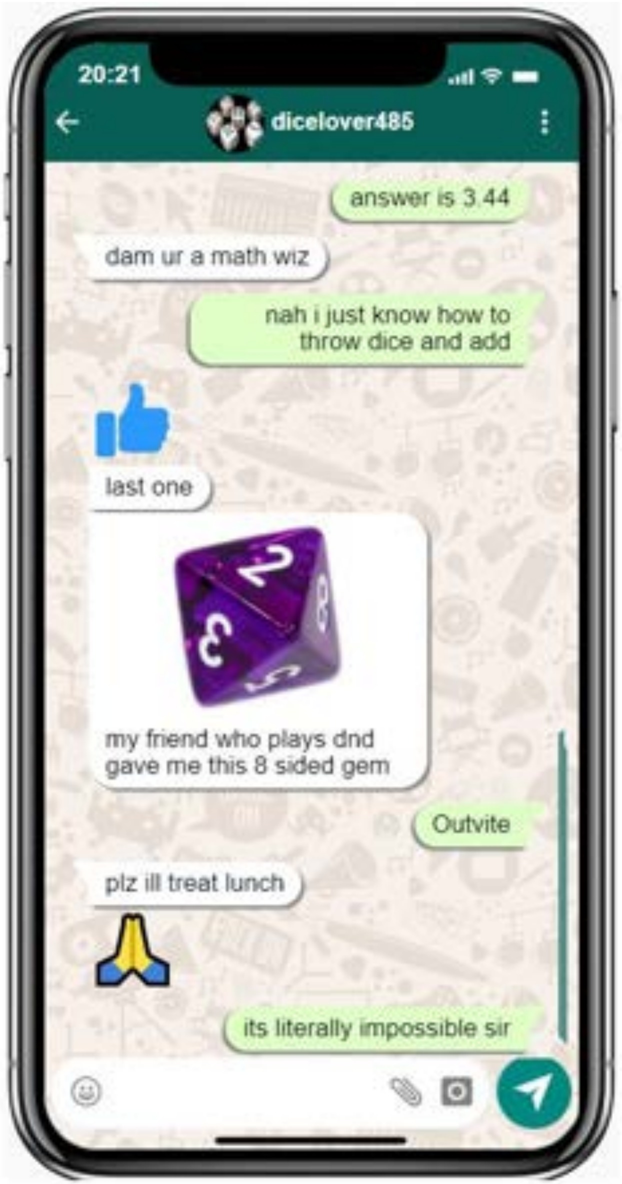

## Cheat Dice

Double the weights!

| | | | | | | |
|---|---|---|---|---|---|---|
| **# Rolls** | **14** | **18** | **23** | **18** | **17** | **10** |
| **Weights** | ***0*** | **1** | ***2*** | ***2*** | ***0*** | **1** |
| **Effective # Rolls** | ***0*** | **18** | ***46*** | ***36*** | ***0*** | **10** |
| **Contribution** | ***0*** | **36** | ***138*** | ***144*** | ***0*** | **60** |

$$\mathbb{E}(X) = \frac{0 + 36 + 138 + 144 + 0 + 60}{0 + 18 + 46 + 36 + 0 + 10} = 3.44$$

# Eight-Sided Dice...?




## Eight Sided Dice

| | ⚀ | ⚁ | ⚂ | ⚃ | ⚄ | ⚅ | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| # Rolls | 14 | 18 | 23 | 18 | 17 | 10 | *0* | *0* |
| Weights | 1 | 1 | 1 | 1 | 1 | 1 | *1?* | *1?* |
| Effective # Rolls | 14 | 18 | 23 | 18 | 17 | 10 | *0* | *0* |
| Contribution | 14 | 36 | 69 | 72 | 85 | 60 | *0* | *0* |

**Impossible** to get a sensible result if we **can never roll on them** to begin with!

# Importance Sampling

| Proposal Distribution q(X) | 1/6 | 1/6 | 1/6 | 1/6 | 1/6 | 1/6 |
|---|---|---|---|---|---|---|

| Target Distribution p(X) | 1/4 | 1/4 | 1/4 | 1/4 | 0 | 0 |
|---|---|---|---|---|---|---|
| Unnormalized Weights | 3/2 | 3/2 | 3/2 | 3/2 | 0 | 0 |
| Normalized Weights | 1/4 | 1/4 | 1/4 | 1/4 | 0 | 0 |

| Target Distribution p(X) | 0 | 1/6 | 1/3 | 1/3 | 0 | 1/6 |
|---|---|---|---|---|---|---|
| Unnormalized Weights | 0 | 1 | 2 | 2 | 0 | 1 |
| Normalized Weights | 0 | 1/6 | 1/3 | 1/3 | 0 | 1/6 |

Unnormalized Weights $\tilde{w}(x) = \frac{p(x)}{q(x)}$

If you are sampling a state **too often**, count it **fewer times**!

# Importance Sampling

**NOTE:** Importance Sampling is **NOT a sampling algorithm**. It is an **integration algorithm**

## Conventional Monte Carlo Integration

$$\mathbb{E}_p[f(X)] = \int f(x)\, p(x)\, \mathrm{d}x \approx \frac{1}{N}\sum_i^N f(x_i) \qquad x_i \sim p(x)$$

Draw samples $x_i$ from $p(x)$, calculate $p(x_i)$ and **average them**

Requires **being able to sample** from $p(x_i)$

If $p(x)$ is high where $f(x)$ is low, a lot of time is spent sampling **unimportant points**

# Importance Sampling

## Importance Sampling

Shifting **probability density** into **weights**

$$\mathbb{E}_p[f(X)] = \int f(x)\, p(x) \mathrm{d}x = \int f(x)\, \frac{p(x)}{q(x)}\, q(x)\, \mathrm{d}x$$

$$\approx \frac{1}{N} \sum_{i}^{N} f(x_i)\, \frac{p(x_i)}{q(x_i)}$$

$$= \frac{1}{N} \sum_{i}^{N} f(x_i)\, w(x_i)$$

Distribution we know to sample from

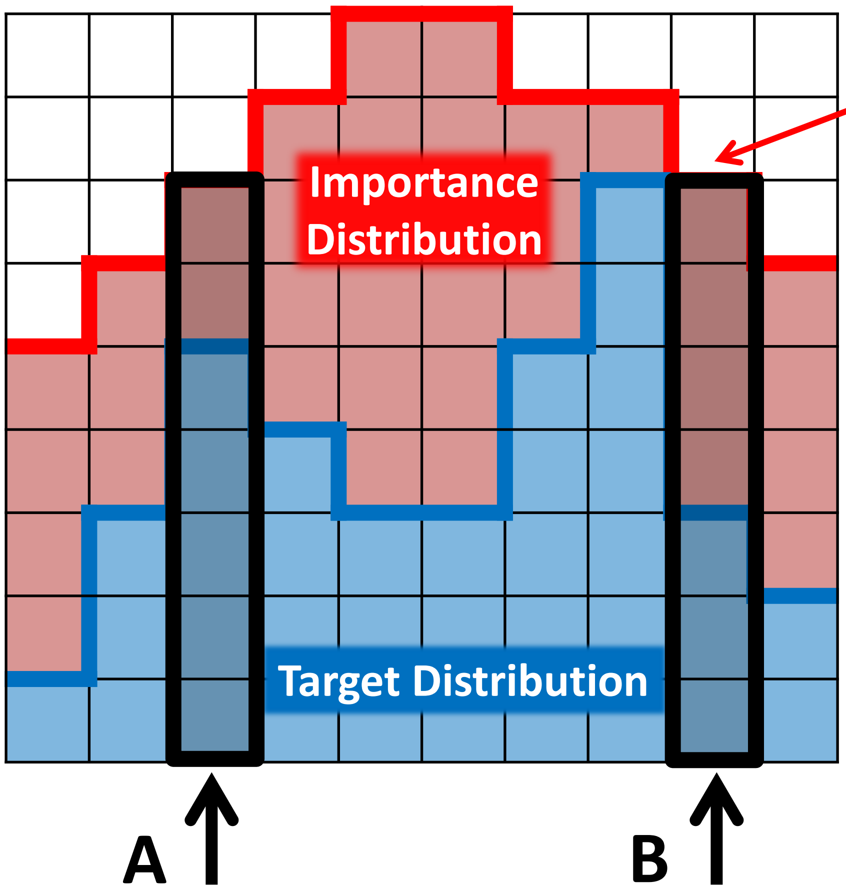
$$x_i \sim q(x)$$

Draw samples $\boldsymbol{x_i}$ from $\boldsymbol{q(x)}$, calculate $\boldsymbol{p(x_i)}$ and take a **weighted average**

**Note:** it is generally a good practice to **normalize** the weights (even if they are normalized in principle)

This also implies that $\boldsymbol{p(x)}$ does **NOT** need to be normalized

# Importance Sampling




We **land** on A and B **equally often** but we **want** A and B to be in the **ratio 5:3**

**Height** is proportionate to the **number of times** we land at that point

**IDEA:** If $q(x)$ samples a point too often compared to $p(x)$, simply "count" that point less often!

**SOLUTION**

**We count**:
Samples at A as **5/7** of a data point
Samples at B as **3/7** of a data point

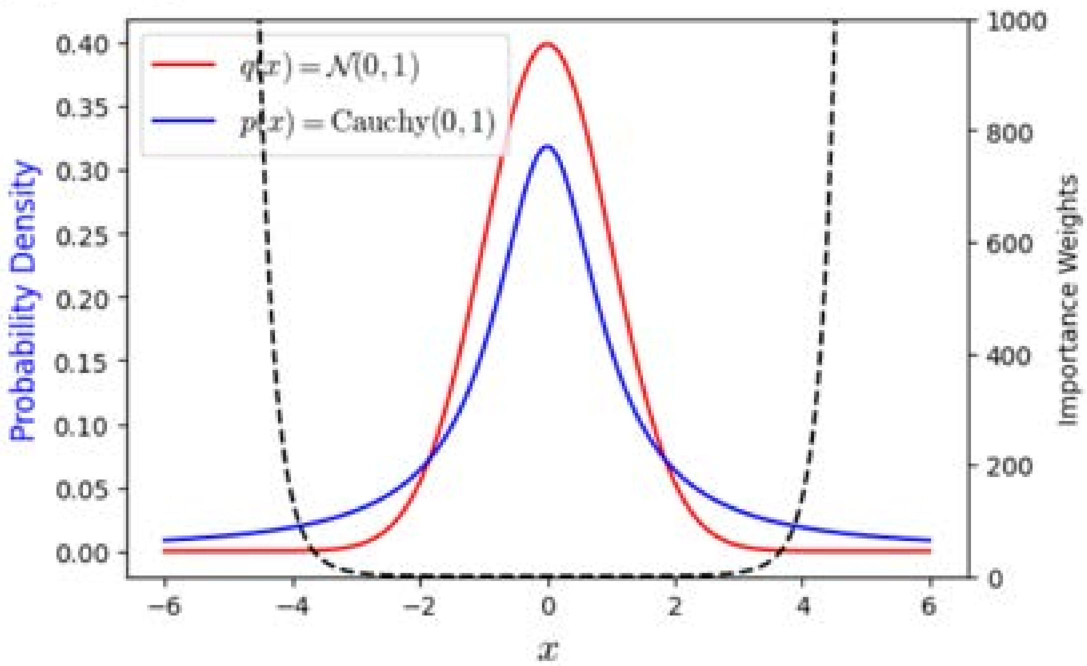
$$f(x_A) \to \frac{5}{7} f(x_A)$$

$$f(x_B) \to \frac{3}{7} f(x_B)$$

# Importance Sampling

**(HOMEWORK PROBLEM)**

**Example:** Importance Sampling from the Cauchy distribution using a Gaussian proposal




**Poor choice** of importance function can lead to **exponentially increasing** weights!

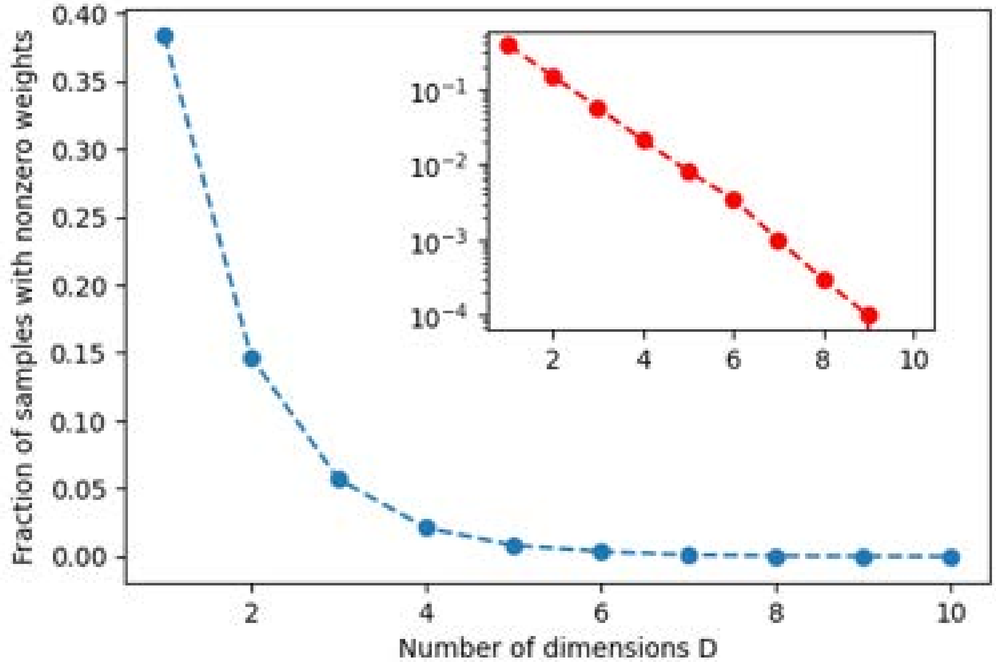
If the weights are too **unbalanced**, then the algorithm is effectively only sampling f(x) from points with high weights

Notion of **Effective Sample Size:**

$$n_{\mathrm{e}} = \frac{\left(\sum_{i=1}^{n} w_i\right)^2}{\sum_{i=1}^{n} w_i^2}$$

Related to variance of the weights

# Importance Sampling in D dimensions




Fraction of samples with **nonzero weights** **decreases exponentially** with **# of dimensions**

Most of our samples are **useless** for calculations!

Intuition about **geometry** in **high-dimensional space** is difficult

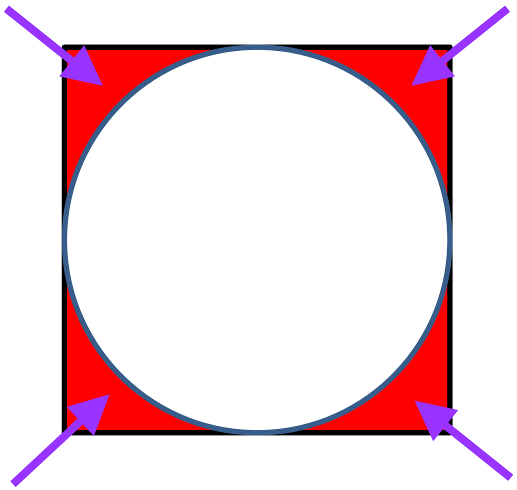
(Irrelevant) volume grows **exponentially** as **number of parameters** increases

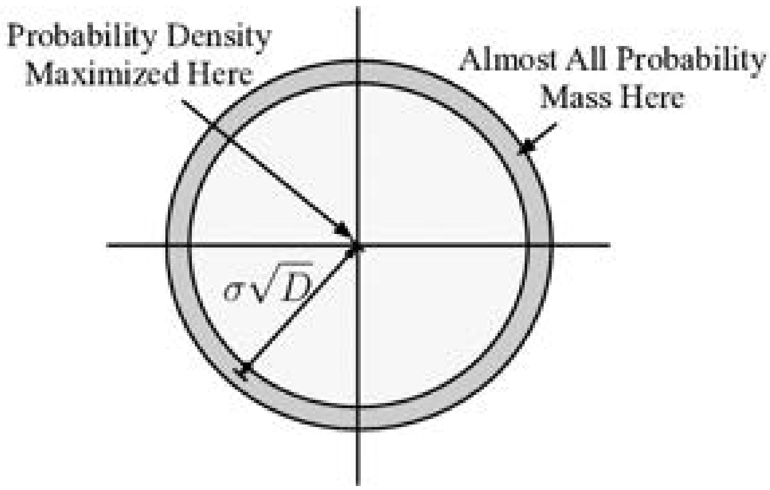
This is called **The Curse of Dimensionality**

# The Curse of Dimensionality

In **high dimensions**, most of the **volume** tends to be **concentrated** on the **edge of the sample space** (corners of the hypercube spanned by the variables)

| No. of Dimensions | $V_{sphere} / V_{cube}$ |
|---|---|
| 3 | 0.52360 |
| 4 | 0.30843 |
| 5 | 0.16450 |
| … | … |
| 10 | 0.00249 |
| … | … |
| 20 | 2.4611 x $10^{-8}$ |



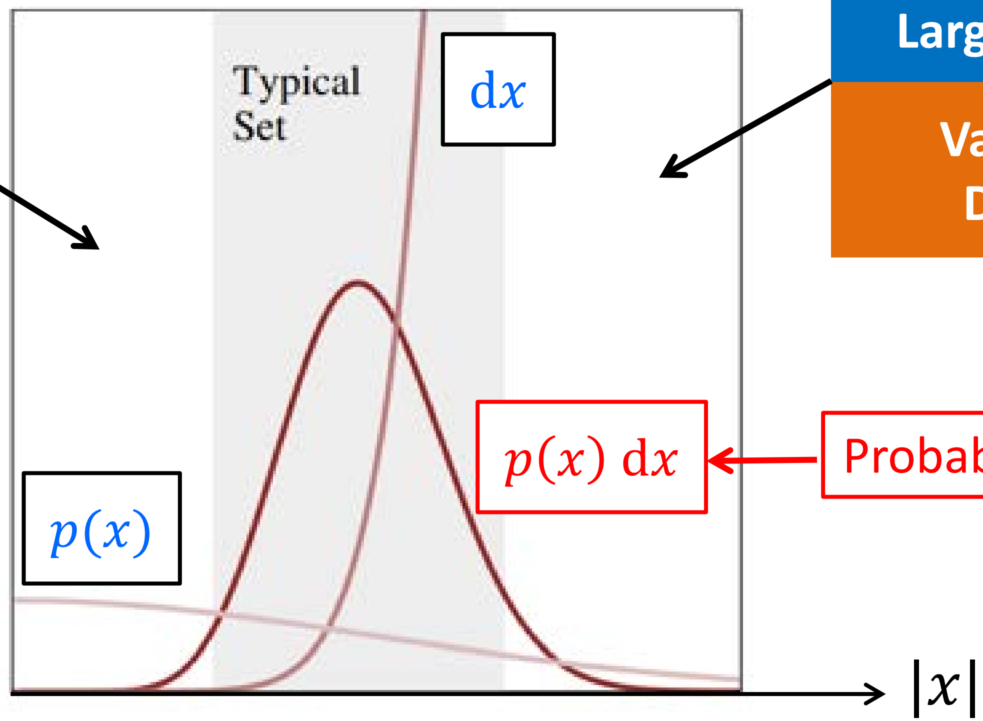


# The Typical Set – Where All The Probability Is

$$\mathbb{E}_p[f(X)] = \int f(x)\, \underline{p(x)\, \mathrm{d}x}$$

**Large (?) Densities**

**Exponentially Small Volume**




**Large Volume**

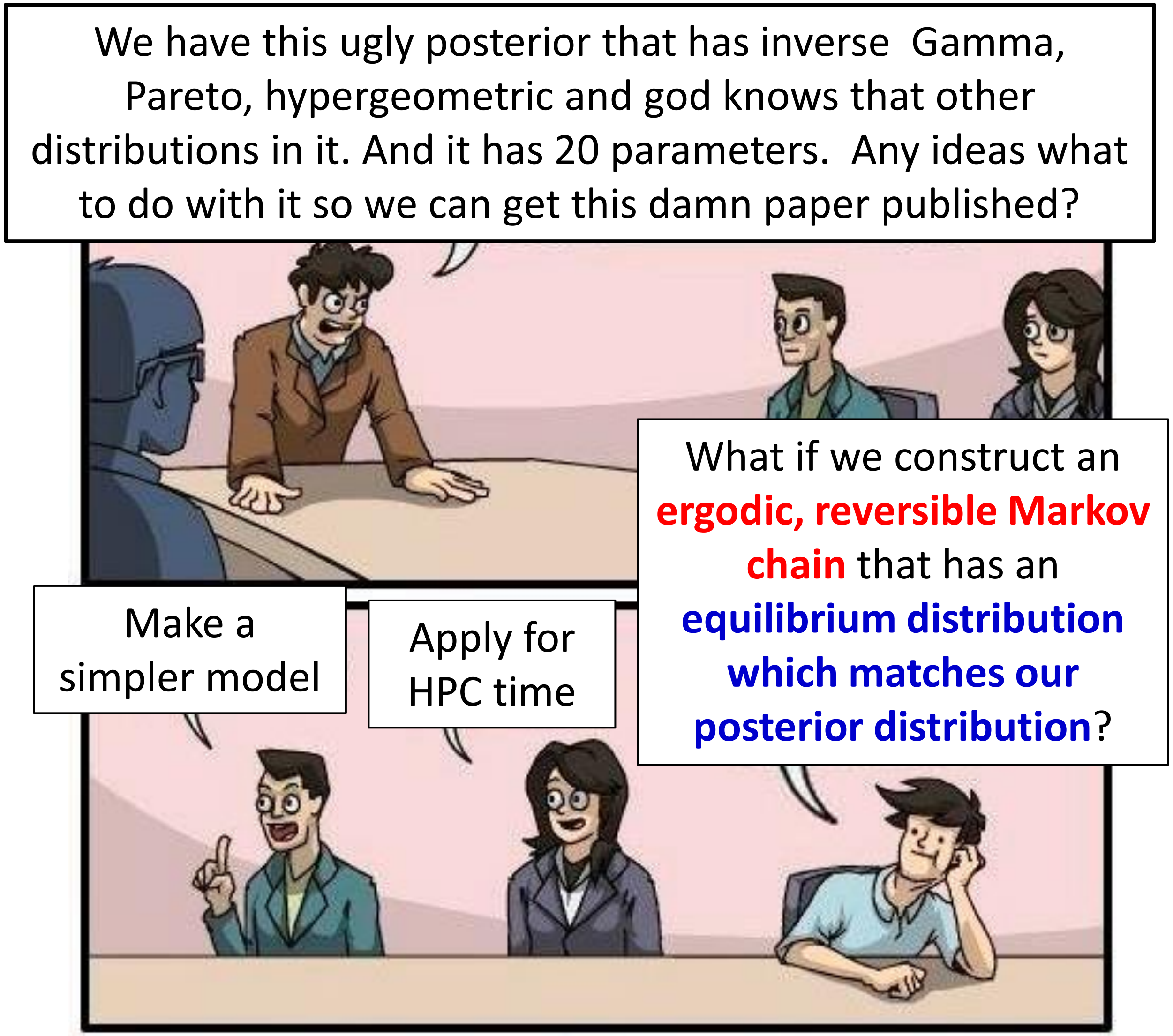
**Vanishing Density**

Probability Mass

Only the **typical set** has non-negligible contributions

Want to focus our sampling efforts there, but **HOW**?

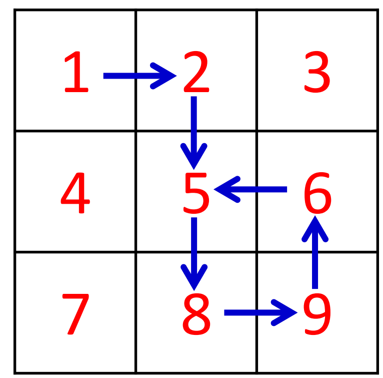
# The Markov Chain Monte Carlo Revolution




# The Blessing of Dimensionality?


**GOAL:** Design something that **moves towards** the **typical set** and **stays there**, exploring it **thoroughly**

**Random walk** in the **state space** of the **typical set**

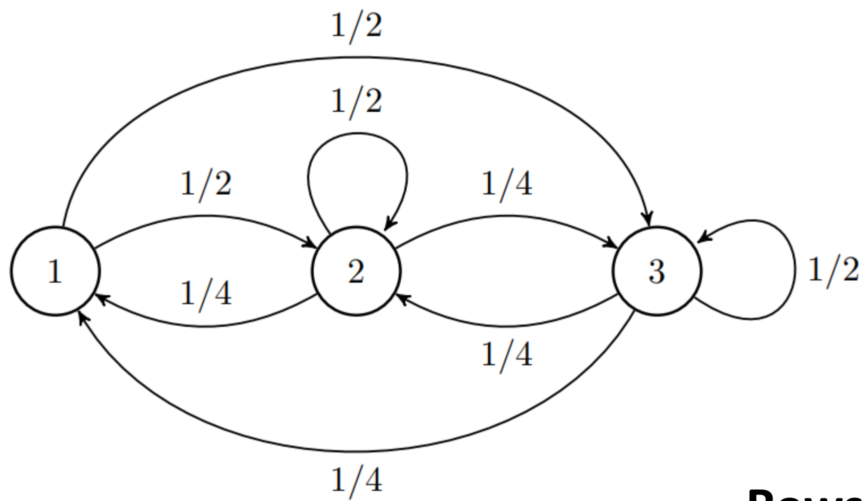



1 → 2 → 5 → 8 → 9 → 6 → 5 → …

**Typical set** is a **low(er) dimension manifold**!
*(Preview: Gibbs sampling makes use of this fact!)*

# Markov Chains

| Markov Chain | A system that experiences **transitions** from one state to another **probabilistically** & has **no memory** |
|---|---|

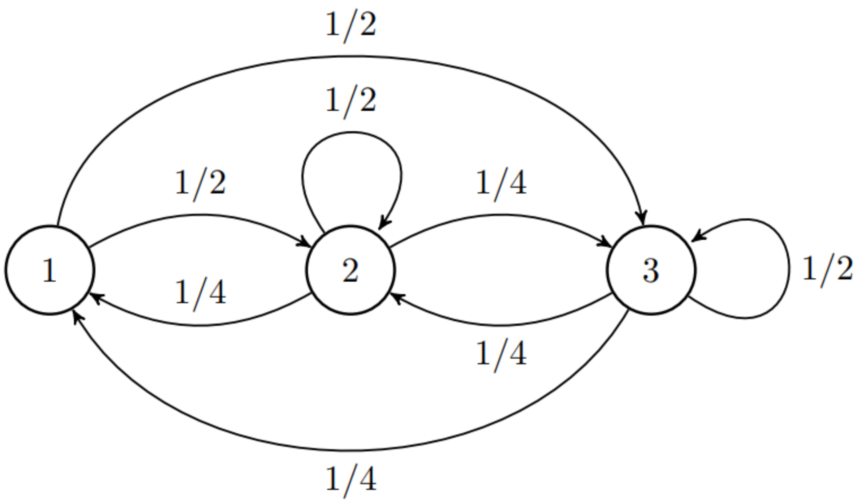



**Rows sum to one**

$$\mathcal{T} = \begin{bmatrix} \mathbb{P}(1 \to 1) & \mathbb{P}(1 \to 2) & \mathbb{P}(1 \to 3) \\ \mathbb{P}(2 \to 1) & \mathbb{P}(2 \to 2) & \mathbb{P}(2 \to 3) \\ \mathbb{P}(3 \to 1) & \mathbb{P}(3 \to 3) & \mathbb{P}(3 \to 2) \end{bmatrix} = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1/4 & 1/2 & 1/4 \\ 1/4 & 1/4 & 1/2 \end{bmatrix}$$

**Transition Matrix (Kernel for continuous case)**

# The Stationary Distribution of a Markov Chain




**Sequence of States:**

$$1 \to 2 \to 3 \to 3 \to 2 \to 3 \to$$
$$\cdots \to 1 \to 1 \to 2 \to 2 \to 3 \to 2$$

If we run the chain **long enough**, we will find that:

$$\mathbb{P}(X = 1) \to \frac{1}{5}$$

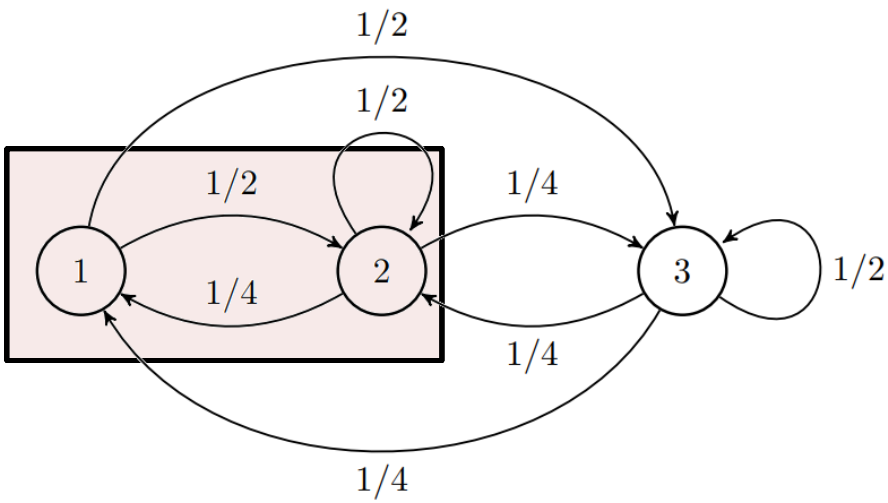
$$\mathbb{P}(X = 2) \to \frac{2}{5}$$

$$\mathbb{P}(X = 3) \to \frac{2}{5}$$

The system still **transitions** between the three states but the **asymptotic distribution** does not change

This is the **stationary state** of the system

# Detailed Balance & No Probability Flow




$$\mathbb{P}(X=1)\,\mathcal{T}(2\,|\,1) = \frac{1}{5}\cdot\frac{1}{2} = \frac{1}{10}$$

$$\mathbb{P}(X=2)\,\mathcal{T}(1\,|\,2) = \frac{2}{5}\cdot\frac{1}{4} = \frac{1}{10}$$

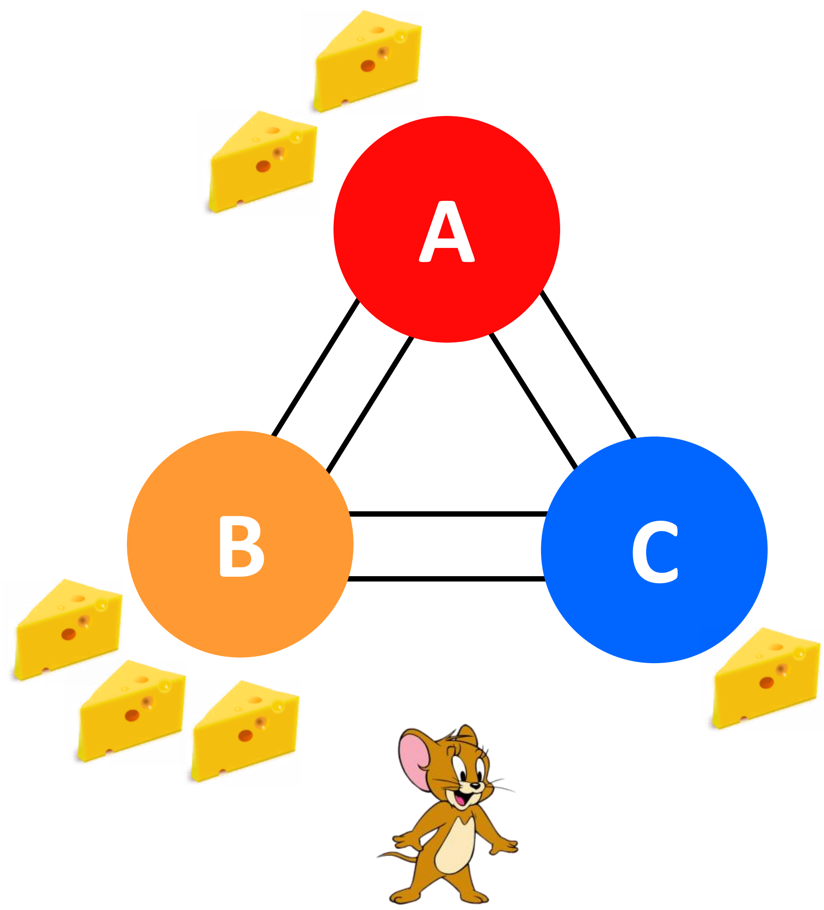
**Probability flow** from 1 → 2 is the **same** as that from 2 → 1!

**Detailed Balance**

$$p(\theta)\,\mathcal{T}(\theta^*\,|\,\theta) = p(\theta^*)\,\mathcal{T}(\theta\,|\,\theta^*)$$

**IDEA:** Play around with the **transition matrix** to **engineer** the **distribution** we want!

# Engineering Distributions




**PROBLEM:**

Jerry wants to spend time at the three locations **proportional to the amount of cheese** present.

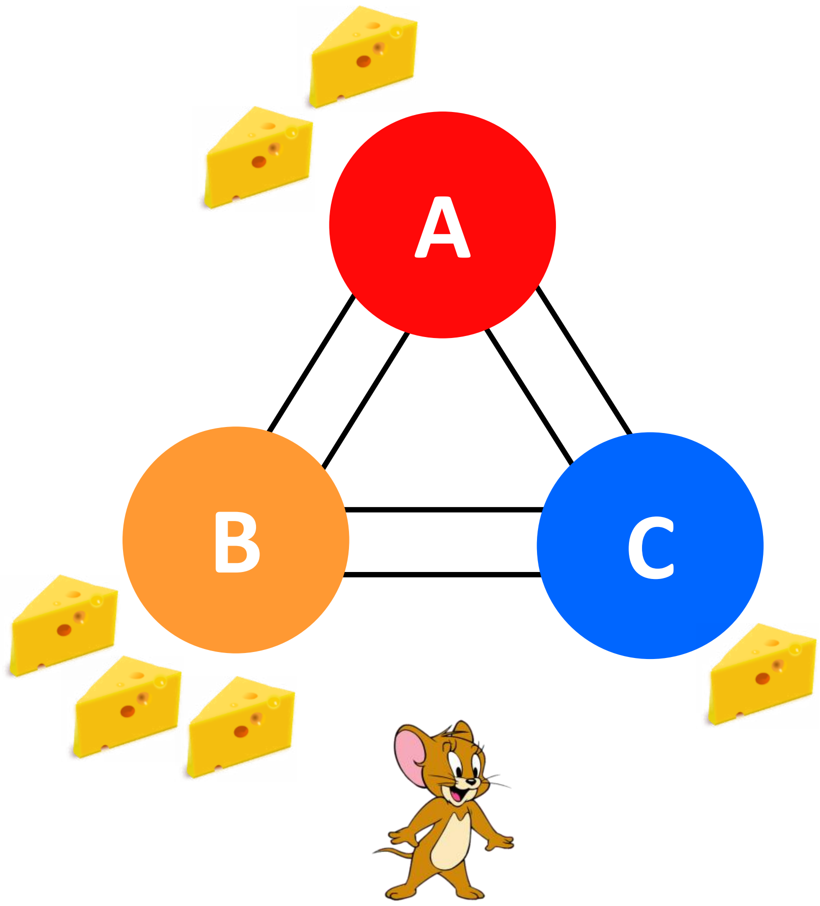
However, he has no **sense of direction** and **cannot remember**.

The only thing he knows is **counting the number of cheese present**

**TARGET:**

$$P(A) = \frac{1}{3} \quad P(B) = \frac{1}{2} \quad P(C) = \frac{1}{6}$$

# Engineering Distributions




If Jerry simply wanders around **randomly**, he would **spend equal time** at each location

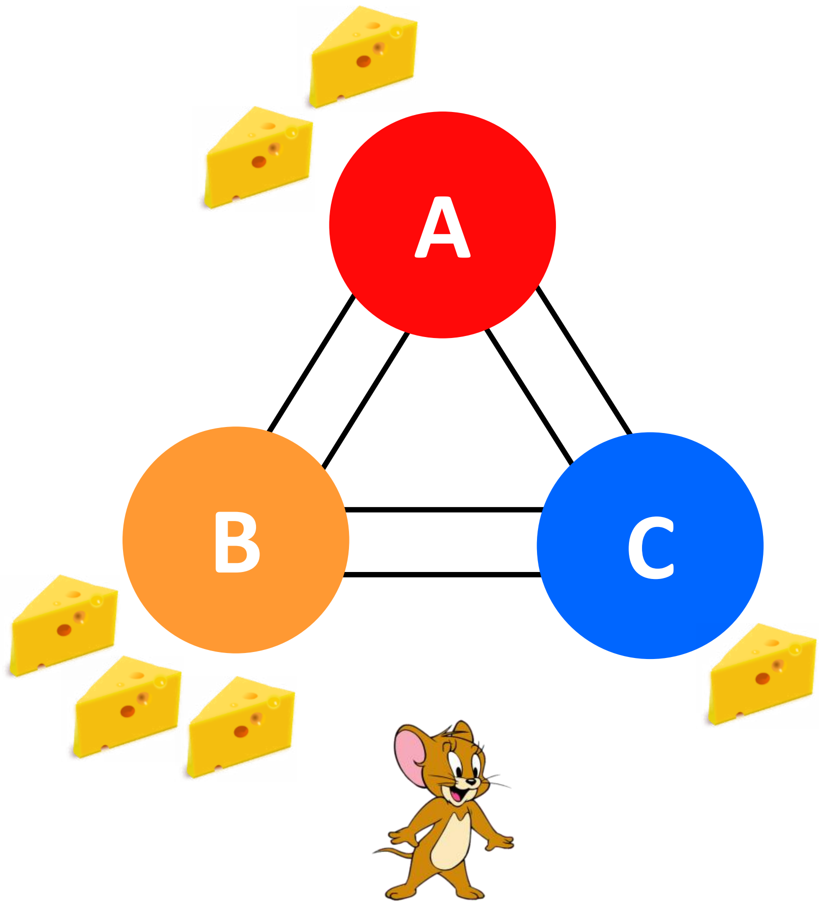
**IDEA:**
If Jerry **compares** the amount of cheese at his **current location** and the **new location**, can he use this information to do anything?

Recall **Detailed Balance**

$$\frac{P(A)}{P(C)} = \frac{T(C\,|\,A)}{T(A\,|\,C)} = 2$$

Example: Jerry wants to somehow move from A to C **half as often** as moving from C to A

# Engineering Distributions




Earlier, we *saw how to adjust distributions to fit what we want*. In this case, the base distribution is that of the **random walk** (which is uniformly distributed)

**SOLUTION**

If we have twice as many samples at C as what we wanted, simply **reject** half the moves from A to C (and keep all the moves from C to A)!!

If the new location has **more cheese**, Jerry **always moves to it**.

If the new location has **fewer cheese**, Jerry moves to it with **probability r** where r is the **ratio of the cheese at the new location to the old location**.

# Engineering Distributions

**Cheese Ratios**

$$P(A) \propto 2 \quad P(B) \propto 1 \quad P(C) \propto 3$$

$$\mathcal{T} = \begin{bmatrix} \mathbb{P}(A \to A) & \mathbb{P}(A \to B) & \mathbb{P}(A \to C) \\ \mathbb{P}(B \to A) & \mathbb{P}(B \to B) & \mathbb{P}(B \to C) \\ \mathbb{P}(C \to A) & \mathbb{P}(C \to B) & \mathbb{P}(C \to C) \end{bmatrix} = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1/2 & 0 & 1/2 \\ 1/2 & 1/2 & 0 \end{bmatrix}$$

Original Random Walk Matrix

# Engineering Distributions

**Cheese Ratios**

$$P(A) \propto 2 \quad P(B) \propto 1 \quad P(C) \propto 3$$

$$\mathcal{T} = \begin{bmatrix} \mathbb{P}(A \to A) & \boxed{\mathbb{P}(A \to B)} & \mathbb{P}(A \to C) \\ \mathbb{P}(B \to A) & \mathbb{P}(B \to B) & \mathbb{P}(B \to C) \\ \mathbb{P}(C \to A) & \mathbb{P}(C \to B) & \mathbb{P}(C \to C) \end{bmatrix} = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1/2 & 0 & 1/2 \\ 1/2 & 1/2 & 0 \end{bmatrix}$$

**Move to B half the time** and **stay at A** (move back to A) the **other half of the time**

# Engineering Distributions

**Cheese Ratios**

$$P(A) \propto 2 \quad P(B) \propto 1 \quad P(C) \propto 3$$

$$\mathcal{T} = \begin{bmatrix} \mathbb{P}(A \to A) & \boxed{\mathbb{P}(A \to B)} & \mathbb{P}(A \to C) \\ \mathbb{P}(B \to A) & \mathbb{P}(B \to B) & \mathbb{P}(B \to C) \\ \mathbb{P}(C \to A) & \mathbb{P}(C \to B) & \mathbb{P}(C \to C) \end{bmatrix} = \begin{bmatrix} 1/4 & 1/4 & 1/2 \\ 1/2 & 0 & 1/2 \\ 1/2 & 1/2 & 0 \end{bmatrix}$$

**Move to B half the time** and **stay at A** (move back to A) the **other half of the time**

# Engineering Distributions

**Cheese Ratios**

$$P(A) \propto 2 \quad P(B) \propto 1 \quad P(C) \propto 3$$

$$\mathcal{T} = \begin{bmatrix} \mathbb{P}(A \to A) & \mathbb{P}(A \to B) & \boxed{\mathbb{P}(A \to C)} \\ \boxed{\mathbb{P}(B \to A)} & \mathbb{P}(B \to B) & \boxed{\mathbb{P}(B \to C)} \\ \mathbb{P}(C \to A) & \mathbb{P}(C \to B) & \mathbb{P}(C \to C) \end{bmatrix} = \begin{bmatrix} 1/4 & 1/4 & 1/2 \\ 1/2 & 0 & 1/2 \\ 1/2 & 1/2 & 0 \end{bmatrix}$$

Moves are **always accepted**

# Engineering Distributions

**Cheese Ratios**

$$P(A) \propto 2 \quad P(B) \propto 1 \quad P(C) \propto 3$$

$$\mathcal{T} = \begin{bmatrix} \mathbb{P}(A \to A) & \mathbb{P}(A \to B) & \mathbb{P}(A \to C) \\ \mathbb{P}(B \to A) & \mathbb{P}(B \to B) & \mathbb{P}(B \to C) \\ \boxed{\mathbb{P}(C \to A)} & \mathbb{P}(C \to B) & \mathbb{P}(C \to C) \end{bmatrix} = \begin{bmatrix} 1/4 & 1/4 & 1/2 \\ 1/2 & 0 & 1/2 \\ 1/2 & 1/2 & 0 \end{bmatrix}$$

**Move to A 2/3 the time** and **stay at C** the **other 1/3 of the time**

# Engineering Distributions

**Cheese Ratios**

$$P(A) \propto 2 \quad P(B) \propto 1 \quad P(C) \propto 3$$

$$\mathcal{T} = \begin{bmatrix} \mathbb{P}(A \to A) & \mathbb{P}(A \to B) & \mathbb{P}(A \to C) \\ \mathbb{P}(B \to A) & \mathbb{P}(B \to B) & \mathbb{P}(B \to C) \\ \boxed{\mathbb{P}(C \to A)} & \mathbb{P}(C \to B) & \mathbb{P}(C \to C) \end{bmatrix} = \begin{bmatrix} 1/4 & 1/4 & 1/2 \\ 1/2 & 0 & 1/2 \\ 1/3 & 1/2 & 1/6 \end{bmatrix}$$

**Move to A 2/3 the time** and **stay at C** the **other 1/3 of the time**

# Engineering Distributions

**Cheese Ratios**

$$P(A) \propto 2 \quad P(B) \propto 1 \quad P(C) \propto 3$$

$$\mathcal{T} = \begin{bmatrix} \mathbb{P}(A \to A) & \mathbb{P}(A \to B) & \mathbb{P}(A \to C) \\ \mathbb{P}(B \to A) & \mathbb{P}(B \to B) & \mathbb{P}(B \to C) \\ \mathbb{P}(C \to A) & \boxed{\mathbb{P}(C \to B)} & \mathbb{P}(C \to C) \end{bmatrix} = \begin{bmatrix} 1/4 & 1/4 & 1/2 \\ 1/2 & 0 & 1/2 \\ 1/3 & 1/2 & 1/6 \end{bmatrix}$$

**Move to B 1/3 the time** and **stay at C** the **other 2/3 of the time**

# Engineering Distributions

## Cheese Ratios

$$P(A) \propto 2 \quad P(B) \propto 1 \quad P(C) \propto 3$$

$$\mathcal{T} = \begin{bmatrix} \mathbb{P}(A \to A) & \mathbb{P}(A \to B) & \mathbb{P}(A \to C) \\ \mathbb{P}(B \to A) & \mathbb{P}(B \to B) & \mathbb{P}(B \to C) \\ \mathbb{P}(C \to A) & \boxed{\mathbb{P}(C \to B)} & \mathbb{P}(C \to C) \end{bmatrix} = \begin{bmatrix} 1/4 & 1/4 & 1/2 \\ 1/2 & 0 & 1/2 \\ 1/3 & 1/6 & 1/2 \end{bmatrix}$$

**Move to B 1/3 the time** and **stay at C** the **other 2/3 of the time**

# Engineering Distributions

**Proposal Kernel**

$$\begin{bmatrix} 0 & 1/2 & 1/2 \\ 1/2 & 0 & 1/2 \\ 1/2 & 1/2 & 0 \end{bmatrix}$$

**Accept-Reject** based on the ratio of cheese (**Metropolis ratio**)

**Target Kernel**

$$\begin{bmatrix} 1/4 & 1/4 & 1/2 \\ 1/2 & 0 & 1/2 \\ 1/3 & 1/6 & 1/2 \end{bmatrix}$$

$$q(\theta^* \mid \theta)\, \alpha(\theta^* \mid \theta) = \mathcal{T}(\theta^* \mid \theta)$$

**Scheme for proposing new moves**

**Final transition probability**
(that has the target distribution as its stationary distribution!)

$$\min\left[1,\, p(\theta)/p(\theta^*)\right]$$

This is where the magic happens

**Accept if new move is 'better', otherwise accept with probability r**

# Metropolis Algorithm

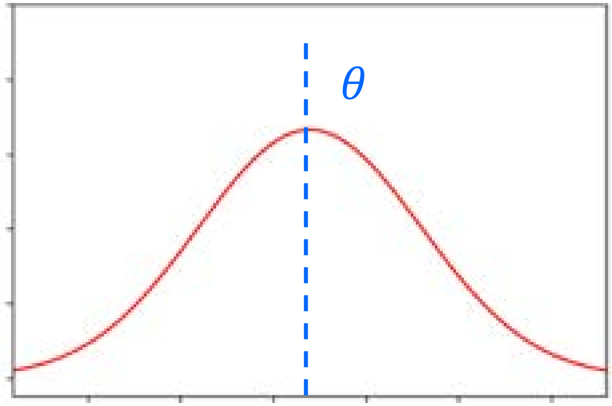
**Algorithm: Metropolis MCMC**

① Choose an appropriate transition kernel $q(\theta^*|\theta)$

This determines how **new moves** are **proposed**

A common choice is the Gaussian **centered around the current location** (**Random Walk** Metropolis algorithm)

The variance of the Gaussian should be carefully chosen




Note: in vanilla Metropolis, this is a **symmetric function**. Metropolis-Hastings generalizes this to an arbitrary kernel.

# Metropolis Algorithm

**Algorithm: Metropolis MCMC**

② Choose an appropriate initialization of the parameters

This is the **start location** of the algorithm

A poor choice far away from the typical set can lead to slow convergence.

A good guess is near the mode of the target distribution
(e.g. MAP of the posterior)

# Metropolis Algorithm

**Algorithm: Metropolis MCMC**

③ Draw a sample $\theta^*$ from $q(\theta^*|\theta)$ and propose it as the next move

It goes without saying that $q(\theta^*|\theta)$ should be something you know how to sample from

# Metropolis Algorithm

**Algorithm: Metropolis MCMC**

④ Calculate the Metropolis ratio $r = p(\theta)/p(\theta^*)$

This is basically comparing the relative probability densities of the current and proposed points

# Metropolis Algorithm

## Algorithm: Metropolis MCMC

⑤ Accept the proposed move with probability $\alpha(\theta^*|\theta) = \min[1, r]$

**Always** move to regions with **higher probability**

**Sometimes** move to regions with **lower probability**

**IMPORTANT:** Not moving means **sampling the current state again**

The acceptance probability should not be too high or too low.

**Too high:** proposed moves are likely to be nearby and system doesn't explore the sample space

**Too low:** system stays in same state for a long time and makes abrupt jumps

# Metropolis Algorithm

**Algorithm: Metropolis MCMC**

① Choose an appropriate transition kernel $q(\theta^*|\theta)$

② Choose an appropriate initialization of the parameters

③ Draw a sample $\theta^*$ from $q(\theta^*|\theta)$ and propose it as the next move

④ Calculate the Metropolis ratio $r = p(\theta)/p(\theta^*)$

⑤ Accept the proposed move with probability $\alpha(\theta^*|\theta) = \min[1, r]$

⑥ Repeat until the desired number of samples are obtained.

# Two algorithms

To get posterior,
normalization is hard:

$$p(\theta_1, \theta_2 \ldots | D) = p(D | \theta_1, \theta_2 \ldots) p(\theta_1, \theta_2 \ldots) / \underline{p(D)}$$

Ways around: 1. **Metropolis**:
compare two points
(candidate vs. current)

$$\frac{p(D | \theta_1, \theta_2 \ldots) p(\theta_1, \theta_2 \ldots) \cancel{/p(D)}}{p(D | \theta_1^*, \theta_2^* \ldots) p(\theta_1^*, \theta_2^* \ldots) \cancel{/p(D)}}$$

2. **Gibbs**: deal with
one para at a time
$p(\theta_i | \{\theta_{j \neq i}\}, D)$

To avoid the curse of
dimensionality

**Compare by an example**

# Two coins example

- Estimate the biases $\theta_1$, $\theta_2$ of two independent coins

Coin #1

Coin #2

- Observations D:

$$\frac{\text{The total \# of heads}}{\text{The total flips}} \Rightarrow \quad \frac{z_1}{N_1} \qquad \frac{z_2}{N_2}$$

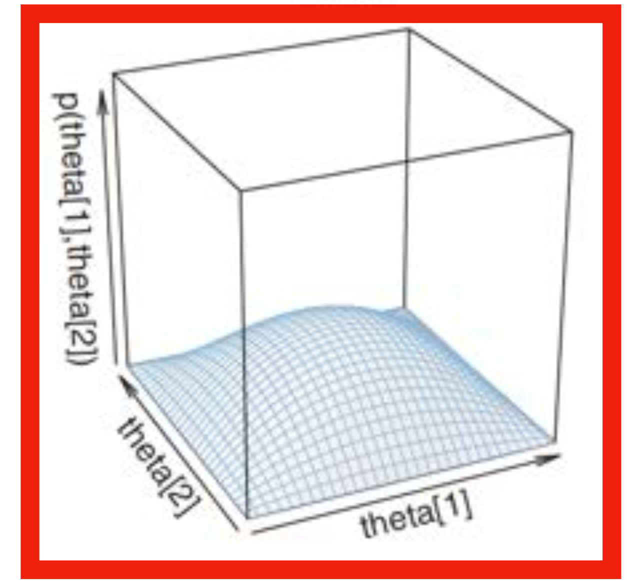
- How to estimate $\theta_1$, $\theta_2$?

  Bayesian:

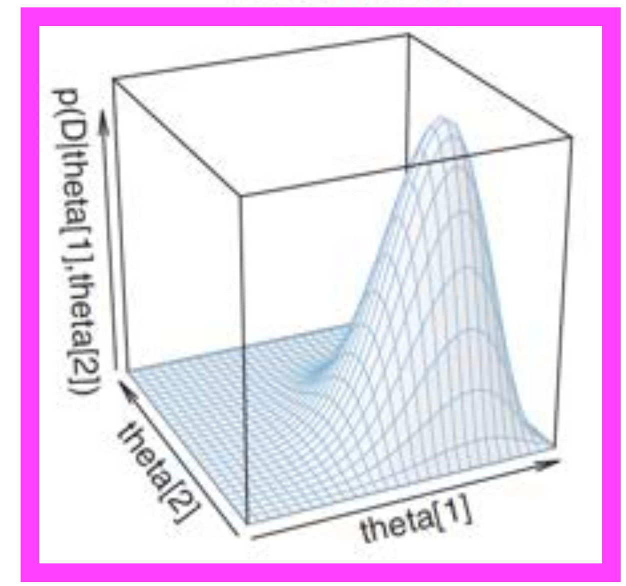
$$\underline{p(\theta_1, \theta_2 | D)} = \underline{p(D|\theta_1, \theta_2)}\,\underline{p(\theta_1, \theta_2)}/p(D)$$

Known up to a scale

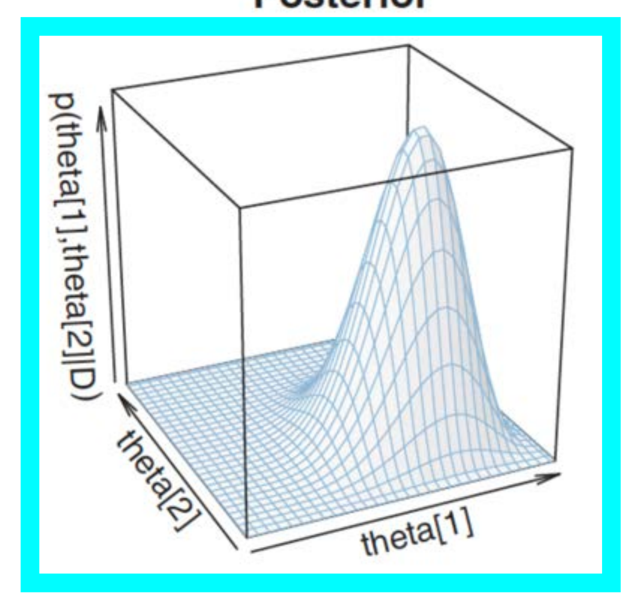
Prior




Likelihood




Posterior



# Two coins example

- Estimate the biases $\theta_1$, $\theta_2$ of two independent coins

Coin #1  Coin #2 

- Observations D:

$$\frac{\text{The total \# of heads}}{\text{The total flips}} \Rightarrow \frac{z_1}{N_1} \qquad \frac{z_2}{N_2}$$

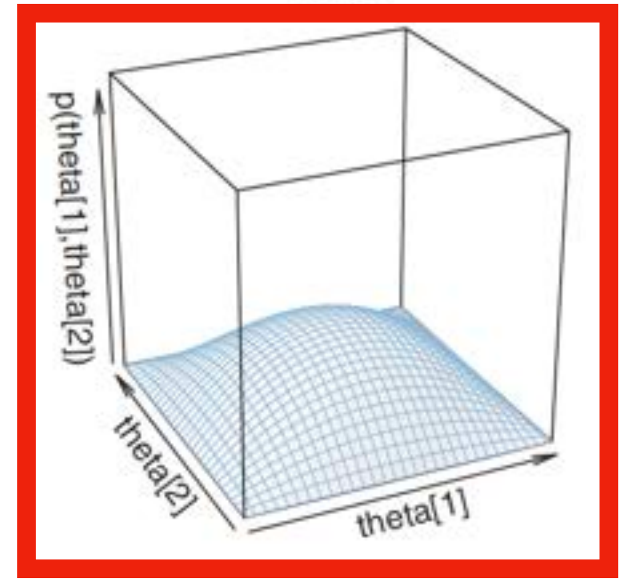
- How to estimate $\theta_1$, $\theta_2$?

  Bayesian:

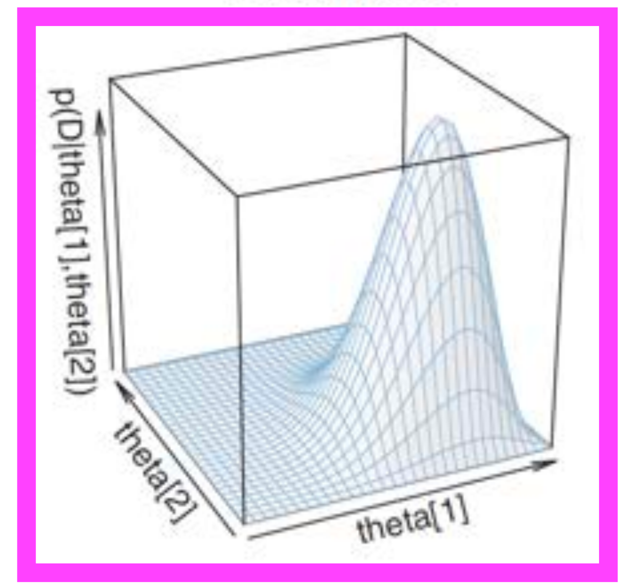
$$\underline{p(\theta_1, \theta_2 | D)} = \underline{p(D | \theta_1, \theta_2)}\,\underline{p(\theta_1, \theta_2)} / p(D)$$

Known up to a scale

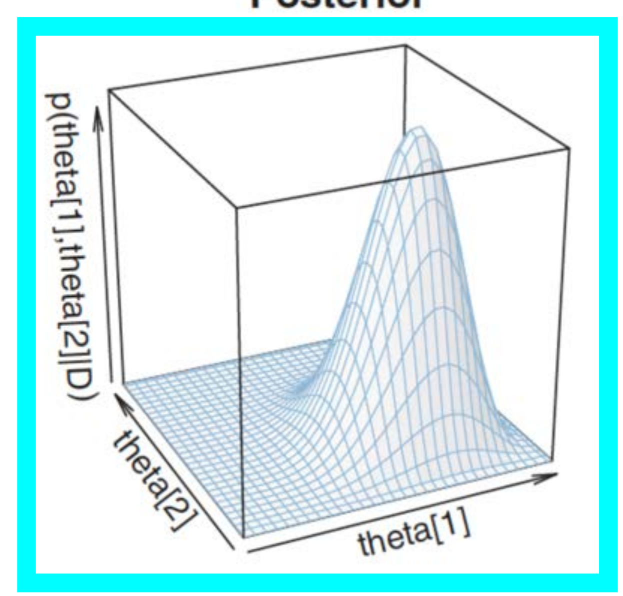
Prior




Likelihood




Posterior

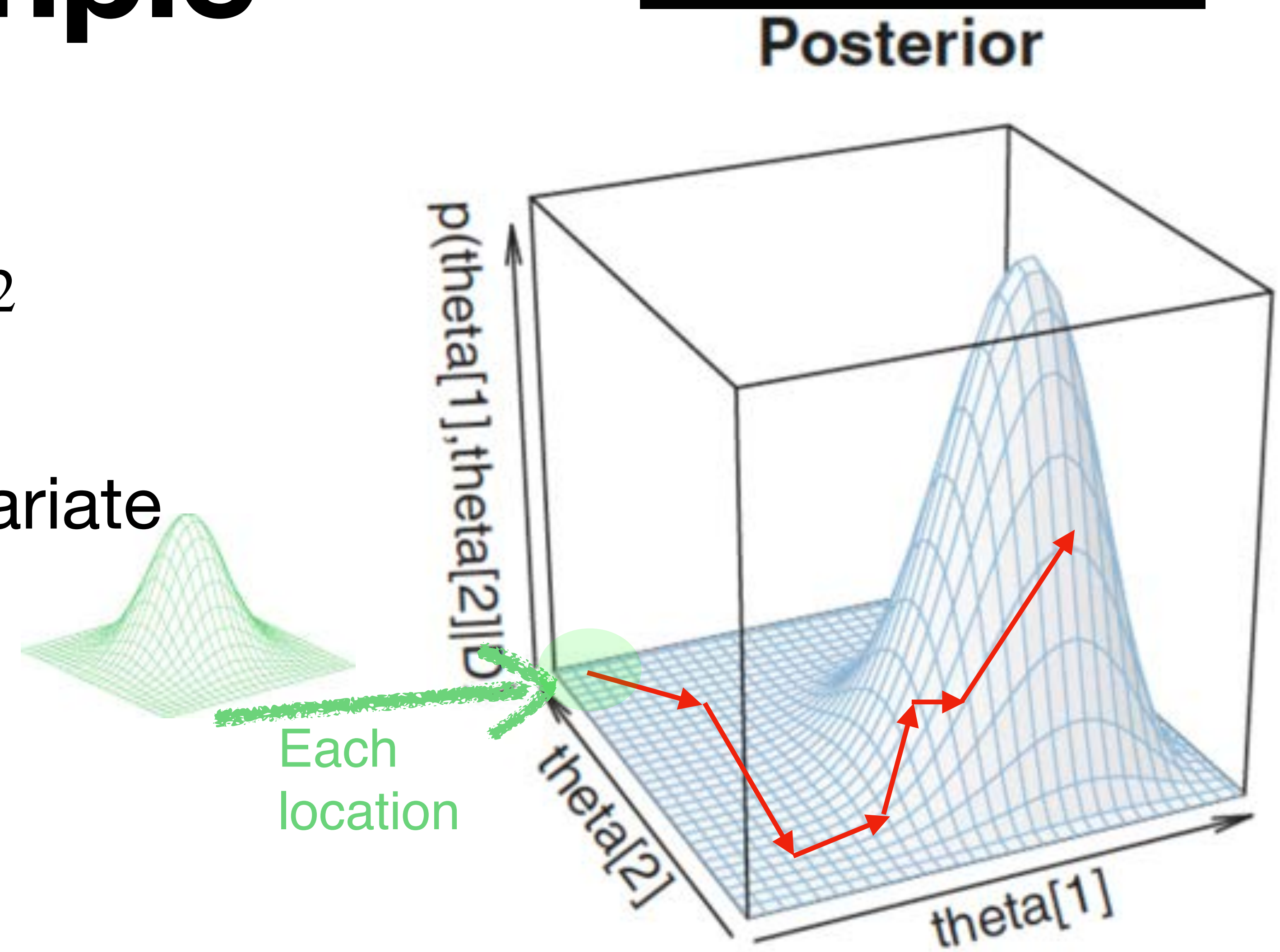


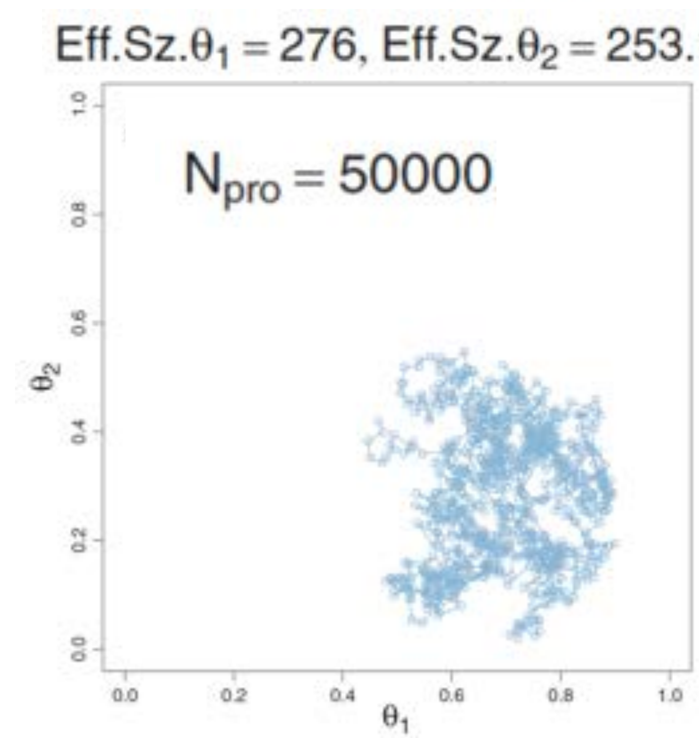
# Two coins example

## Metropolis

1. Random initialization: $\theta_1$, $\theta_2$
2. Draw a candidate from the proposal distribution (a bivariate normal)
3. Get acceptance rate
4. Accept or reject
5. Repeat step 2

Known up to a scale



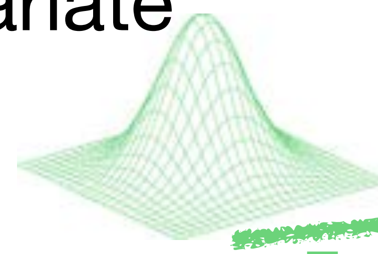




Low efficiency, why?

# Two coins example

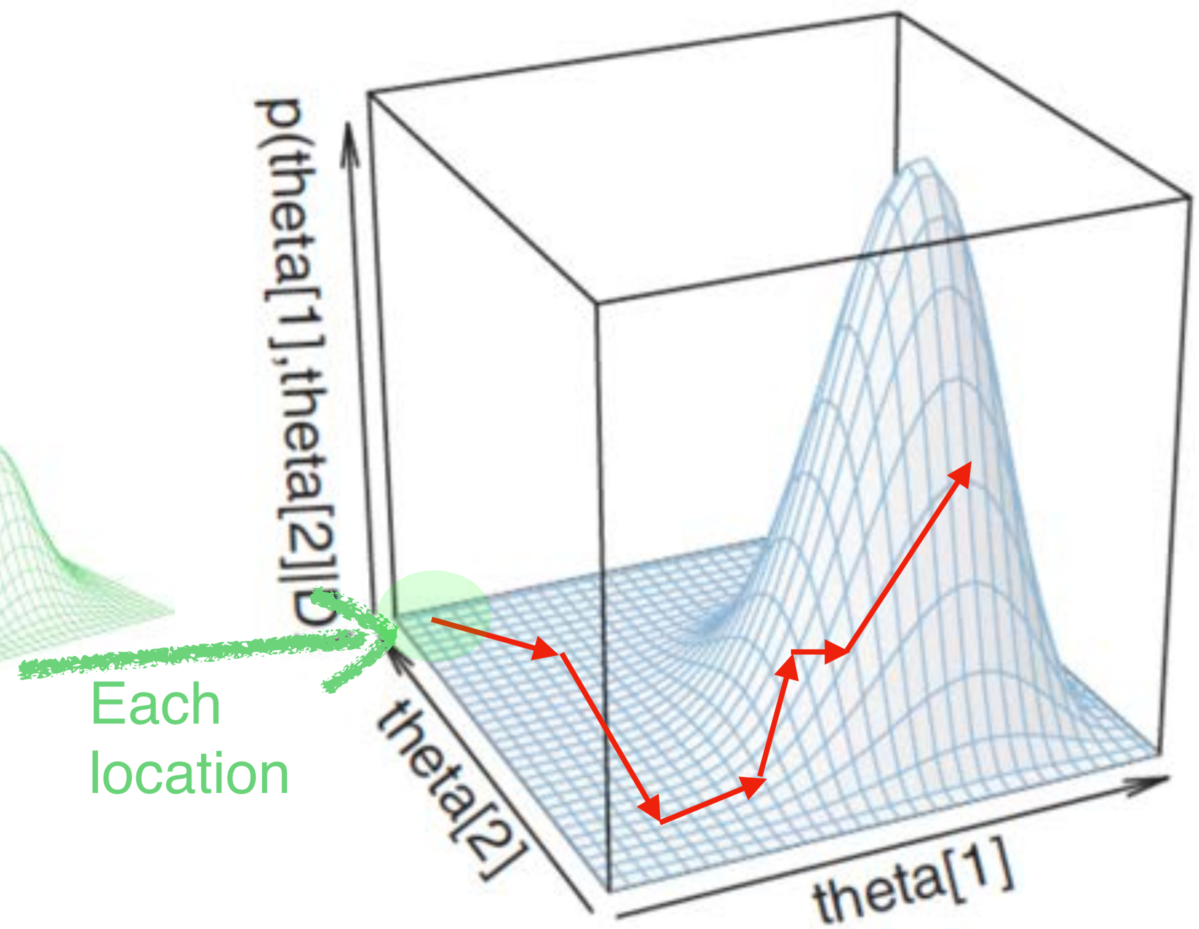
## Metropolis

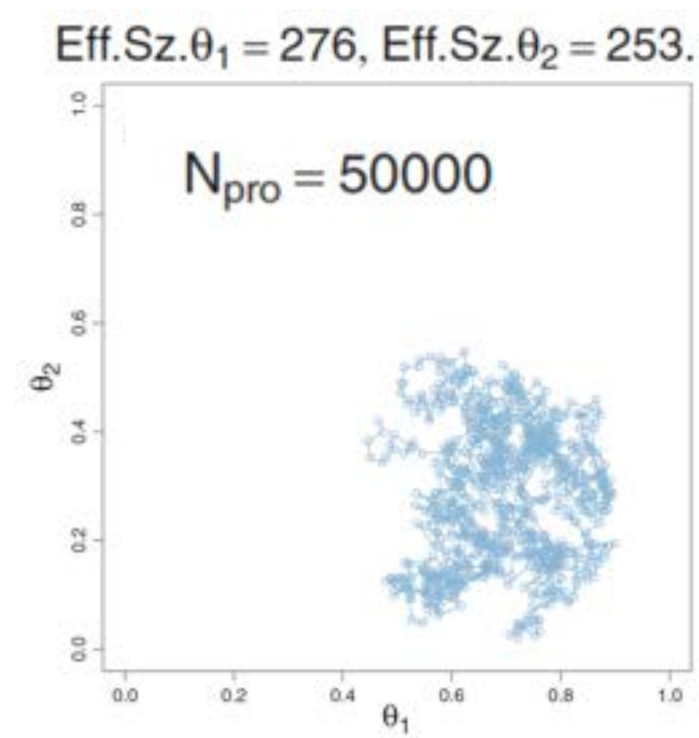
1. Random initialization: $\theta_1, \theta_2$
2. Draw a candidate from the proposal distribution (a bivariate normal)
3. Get acceptance rate
4. Accept or reject
5. Repeat step 2



Known up to a scale

Posterior

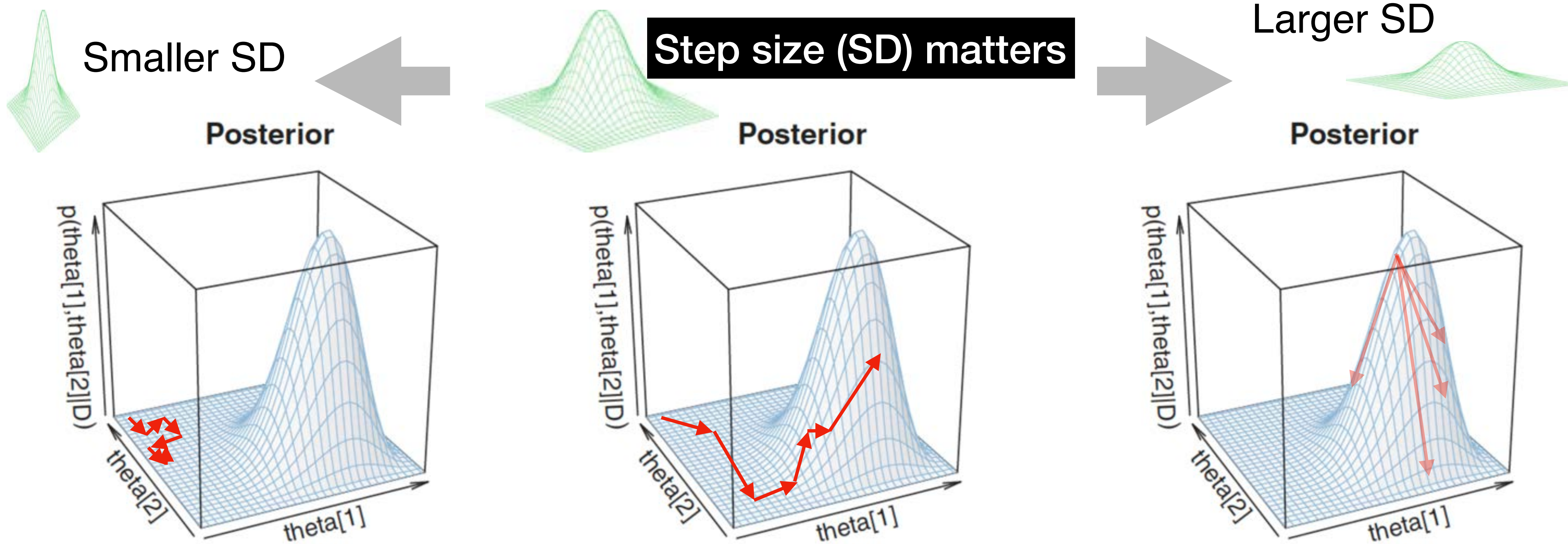



Low efficiency, why?

# Two coins example

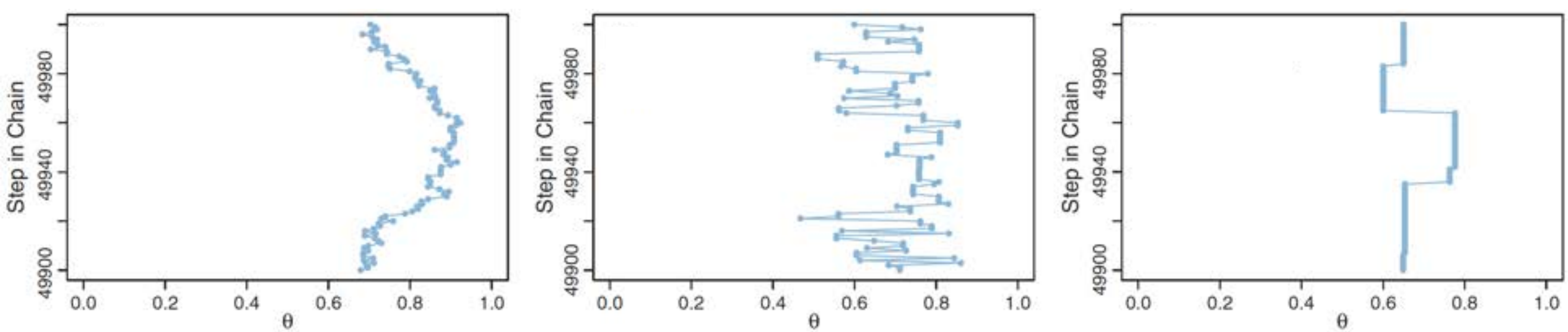
## **Metropolis** - limitations




Surrounding gets similar probability
Acceptance high, but explore slowly

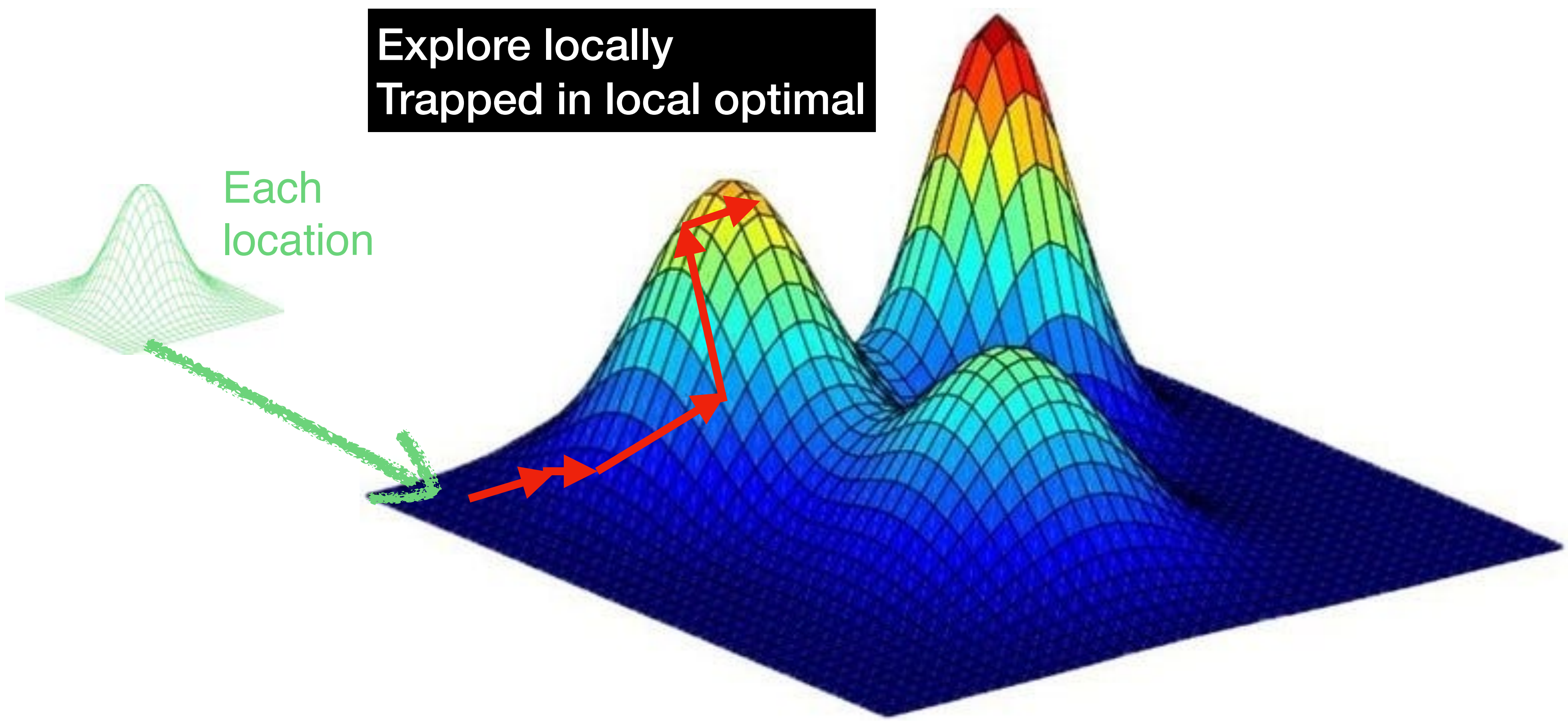
Some rejection

Surrounding gets different probability
Rejection high (waste time)



# Two coins example

## **Metropolis** - limitations



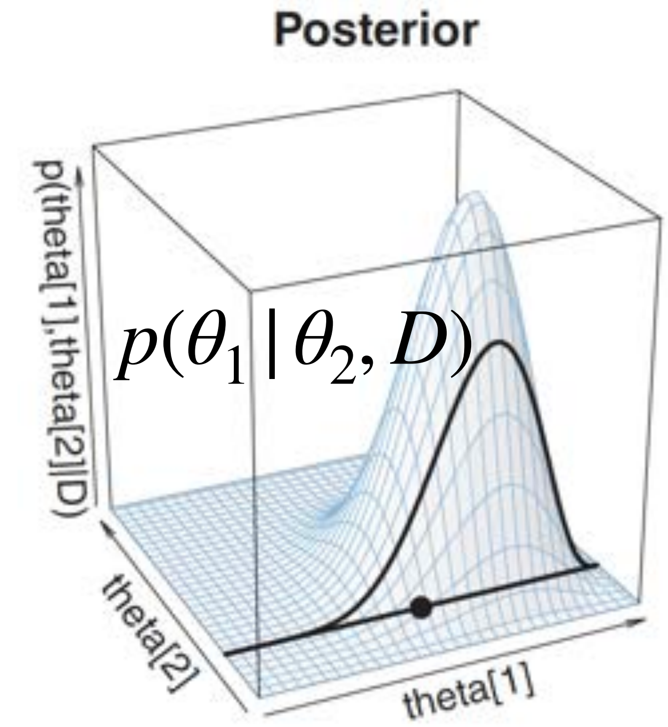
# Two coins example

## Gibbs

1. Random initialization: $\theta_1, \theta_2$

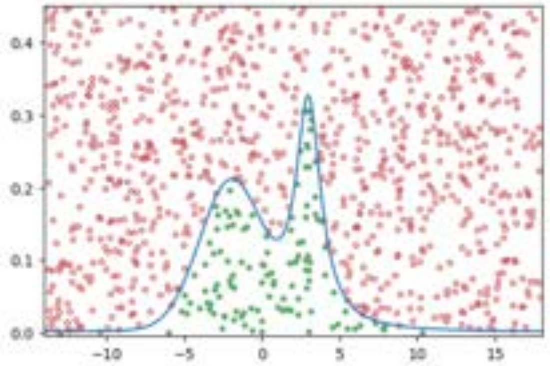
2. **Fix** $\theta_2$

3. Generate an **updated** $\theta_1$ from the proposal distribution $p(\theta_1 \,|\, \theta_2, D)$ (Always accept)

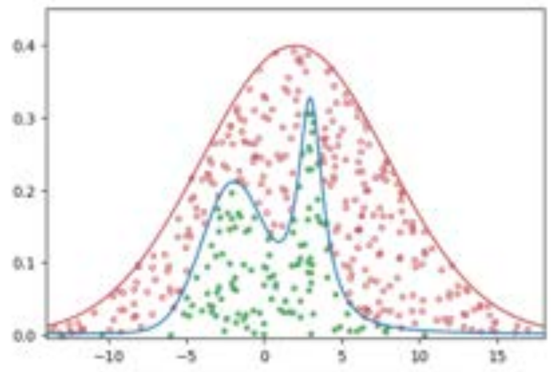



How to generate $\theta_1$ from $p(\theta_1 \,|\, \theta_2, D)$?

Rejection sampling



Importance sampling

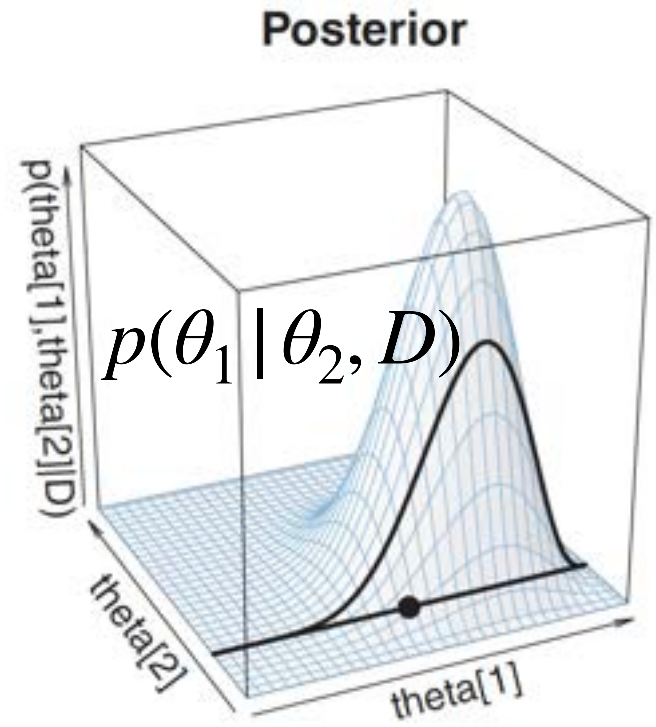


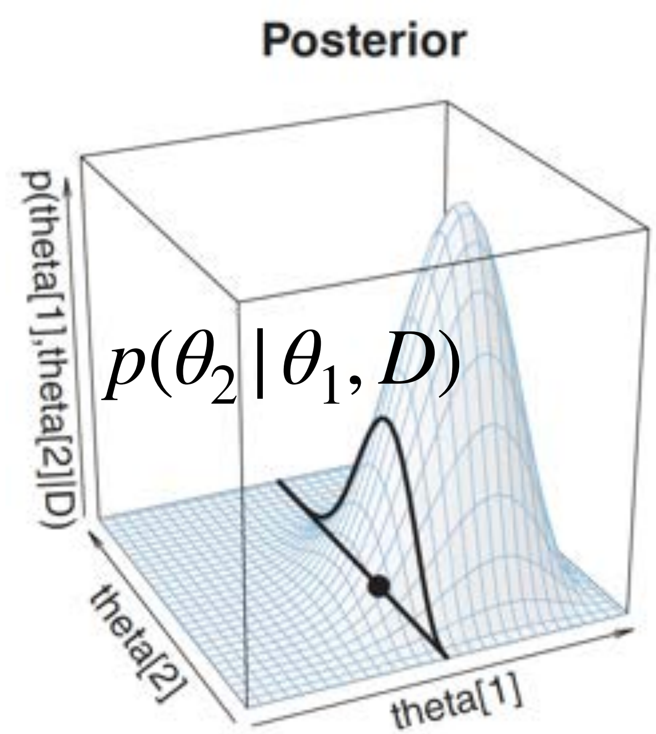
Other methods

# Two coins example

## Gibbs

1. Random initialization: $\theta_1, \theta_2$
2. **Fix $\theta_2$**
3. Generate an **updated $\theta_1$** from the proposal distribution $p(\theta_1 \mid \theta_2, D)$ (Always accept)
4. **Fix $\theta_1$**
5. Generate an **updated $\theta_2$** from the proposal distribution $p(\theta_2 \mid \theta_1, D)$ (Always accept)
6. Repeat step 2.



# Two coins example

## Gibbs

1. Random initialization: $\theta_1, \theta_2$

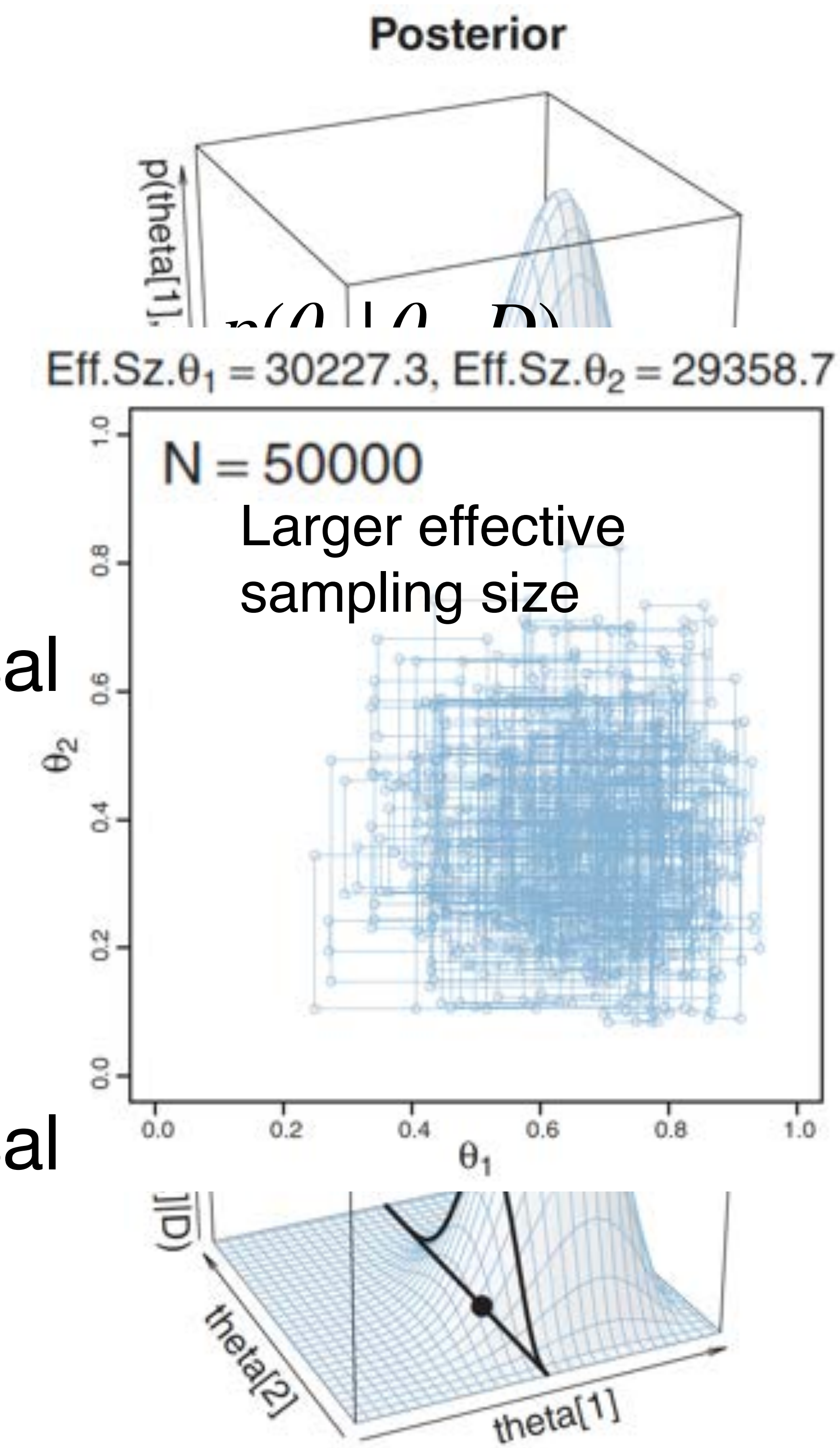
2. **Fix $\theta_2$**

3. Generate an **updated $\theta_1$** from the proposal distribution $p(\theta_1 \mid \theta_2, D)$ (Always accept)

4. **Fix $\theta_1$**

5. Generate an **updated $\theta_2$** from the proposal distribution $p(\theta_2 \mid \theta_1, D)$ (Always accept)

6. Repeat step 2.



## Gibbs

- Random initialization
- Cycle through each of $\theta_1$, $\theta_2$, $\theta_3$, $\theta_4$…… $\theta_1$, $\theta_2$, $\theta_3$, $\theta_4$…… fix the rest
- Draw a candidate from the **proposal distribution** $p(\theta_i \mid \{\theta_{j \neq i}\}, D)$
- **Always accepted**
- Repeat

**More efficient in high dimensions**

**Limit**

- Need conditional probabilities
- Bad for highly correlated parameters

## Metropolis

- Random initialization
- Draw a candidate from the **proposal distribution (e.g. a N-dimention normal)**
- **Get acceptance rate**
- **Determine to accept or not**
- Repeat

- Tune step size. (Proposal distribution similar to posterior distribution)
- rejection high
- Trapped in local optimal

# **Two coins example** - final question

- Observations D:

Coin #1

$\frac{z_1}{N_1}$

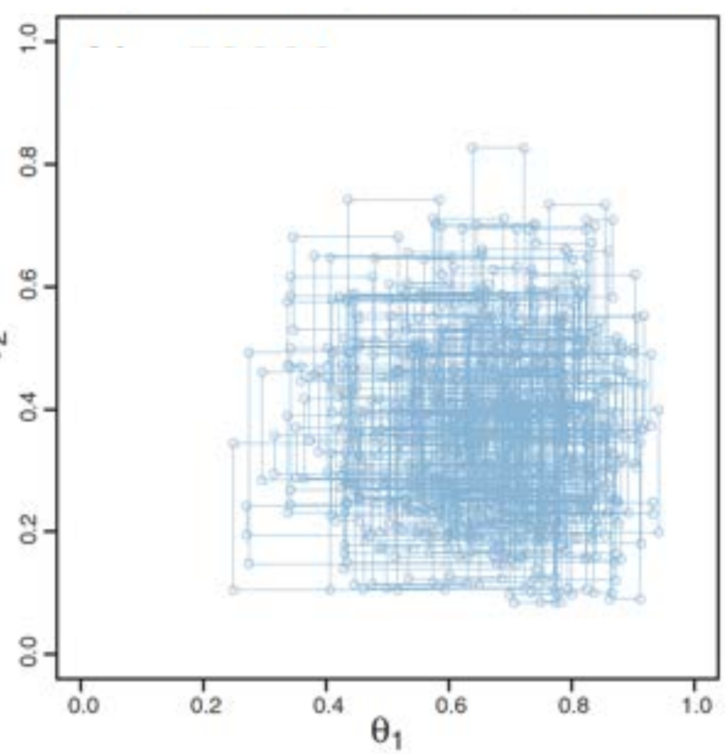
Coin #2

$\frac{z_2}{N_2}$

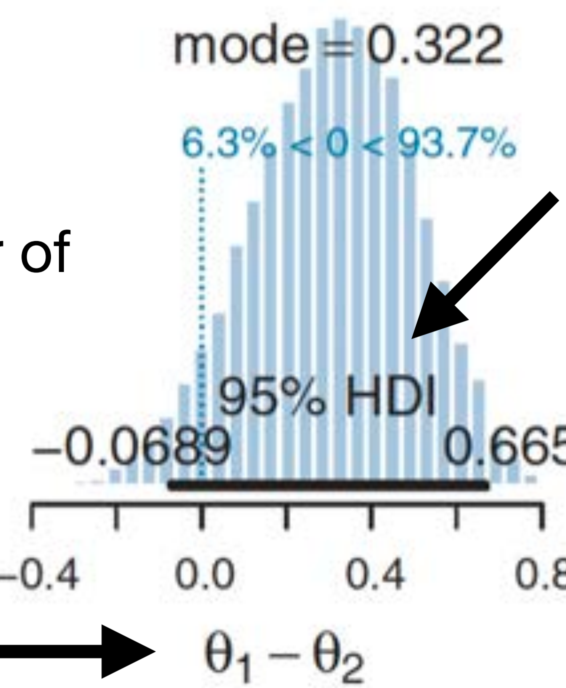
- Estimate the independent biases, $\theta_1, \theta_2$ by bayesian and MCMC

- **Do two biases differ?**

Raw data




For each pair of data, get the difference



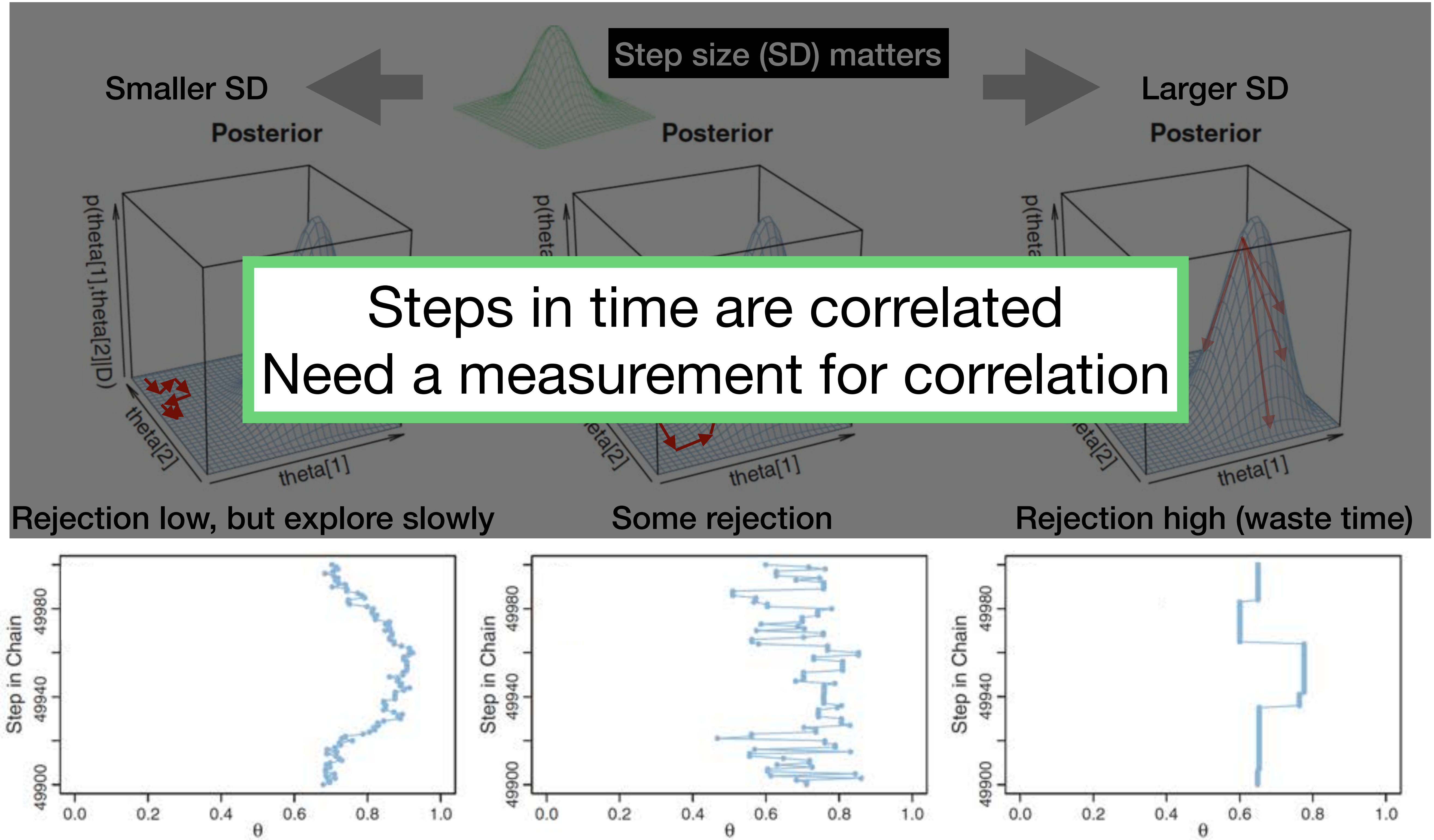

**Highest density interval**

**No significant difference**

How to evaluate the performance?

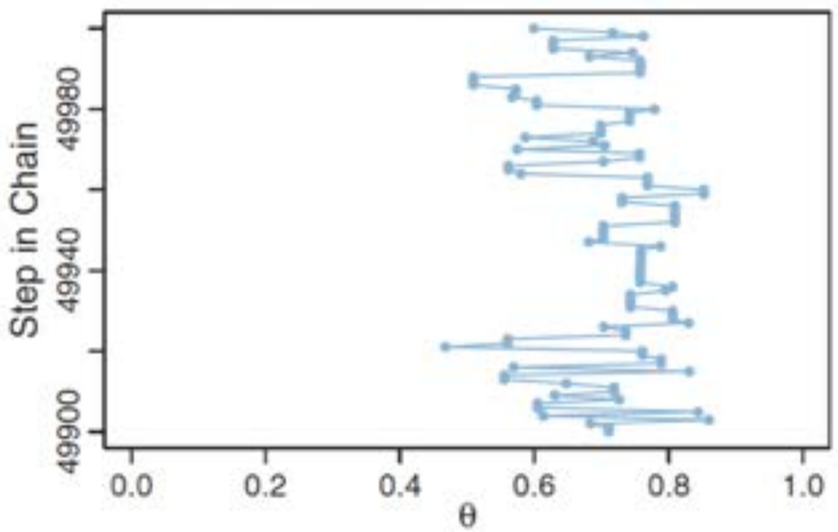
# Performance evaluation

## Sample more gives "Accuracy"



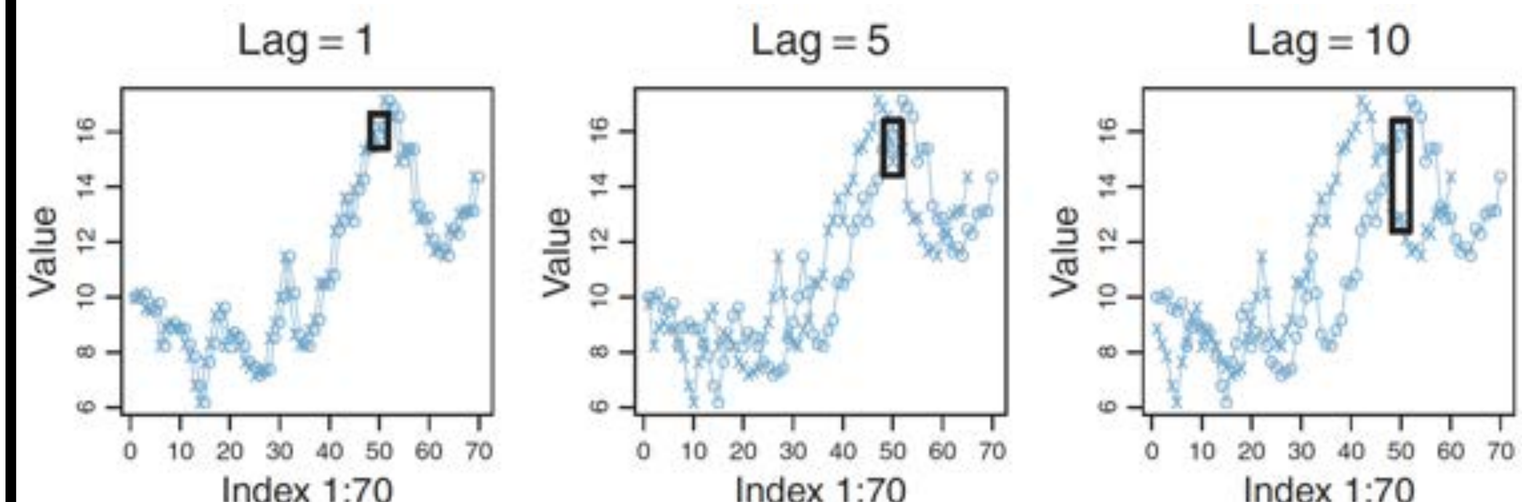
# Performance evaluation

## Sample more gives "Accuracy"

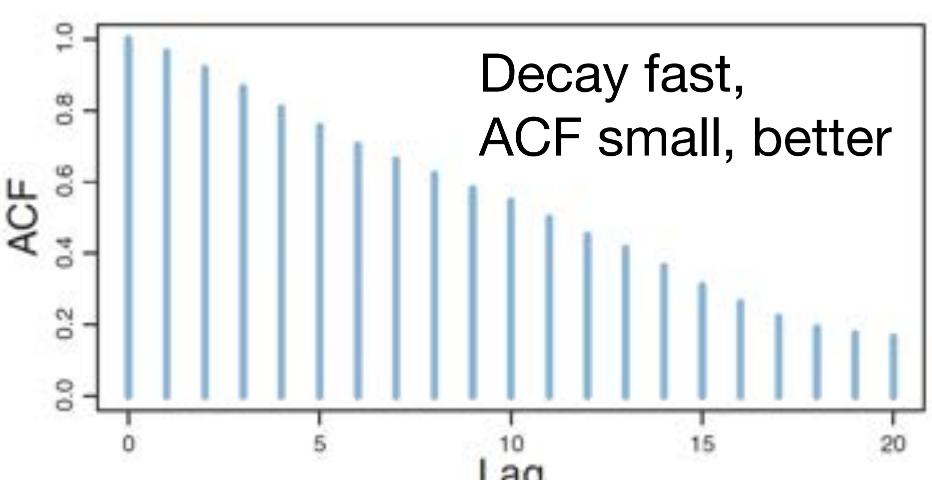



To quantify correlation

Autocorrelation ACF(k): how similar a trace is to itself with a time lag of k.

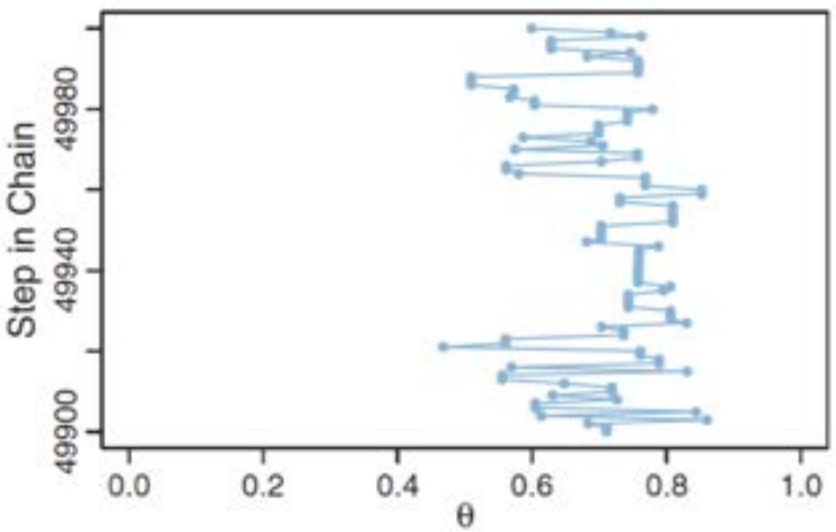



$$ACF(k) = \frac{\sum (x_t - \bar{x})(x_{t+k} - \bar{x})}{\sum (x_t - \bar{x})^2}$$



# Performance evaluation

## Sample more gives "Accuracy"




Lower autocorrelation, better sampling

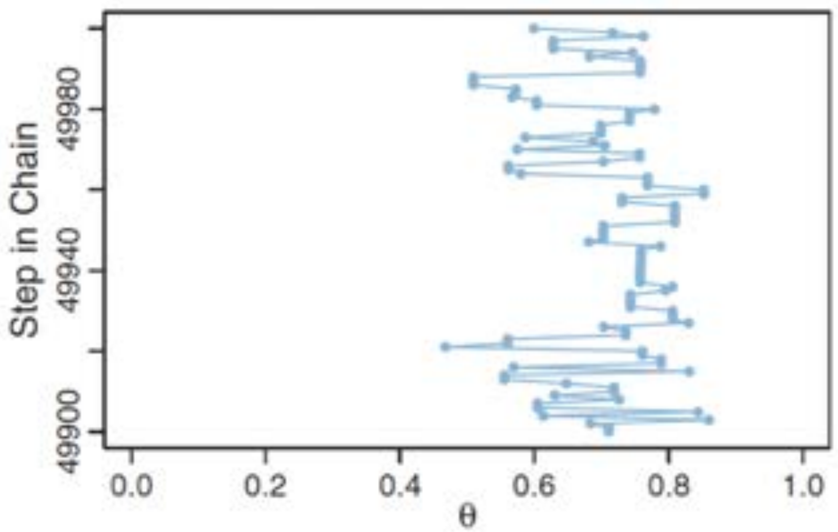
Quantify:
**effective sample size**

$$\mathrm{ESS} = N \Bigg/ \left(1 + 2\sum_{k=1}^{\infty} \mathrm{ACF}(k)\right)$$

Denominator: Total ACF from $-\infty$ and $\infty$

# Performance evaluation

## Sample more gives “Accuracy”




Intuitively, effective sample size (ESS) $\rightarrow \infty$, perfect estimation

Quantify: **Monte Carlo standard error** (MCSE)

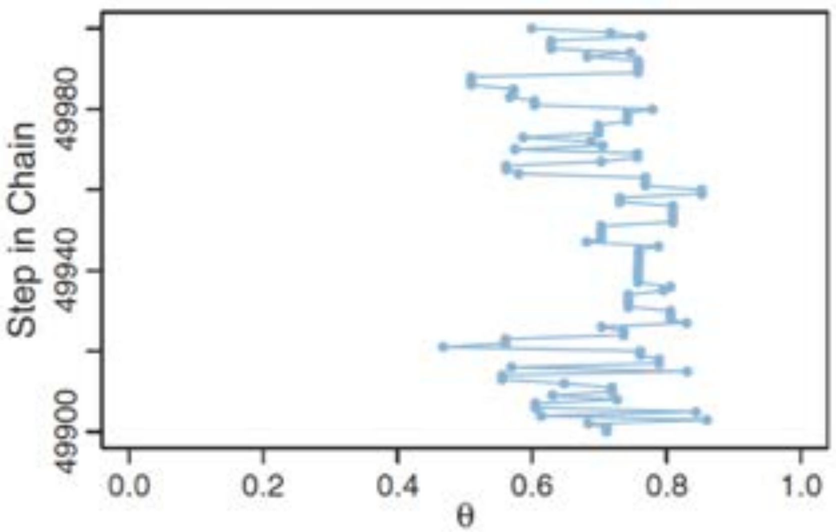
$$\mathrm{MCSE} = \mathrm{SD}/\sqrt{\mathrm{ESS}}$$

$$\mathrm{SE} = \mathrm{SD}/\sqrt{N}$$

Similar to conventional definition

# Performance evaluation

## Sample more gives “Accuracy”




**Quick review**

Raw data

↓

Autocorrelation

↓

Effective sample size

↓

Monte Carlo standard error

# Performance evaluation

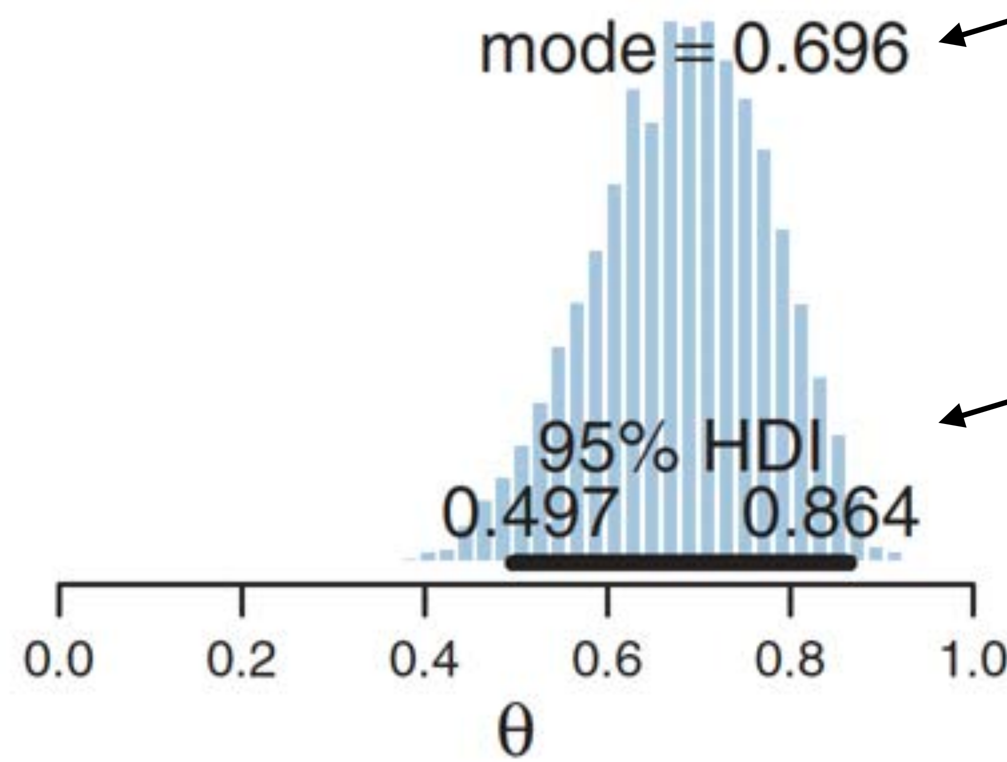
## Sample more gives “Accuracy”

The ‘accuracy’ also depends on which quantity wanted

**With the same sample size**

Mode is more accurate

Highest density interval (HDI)
Less accurate (less sampled)

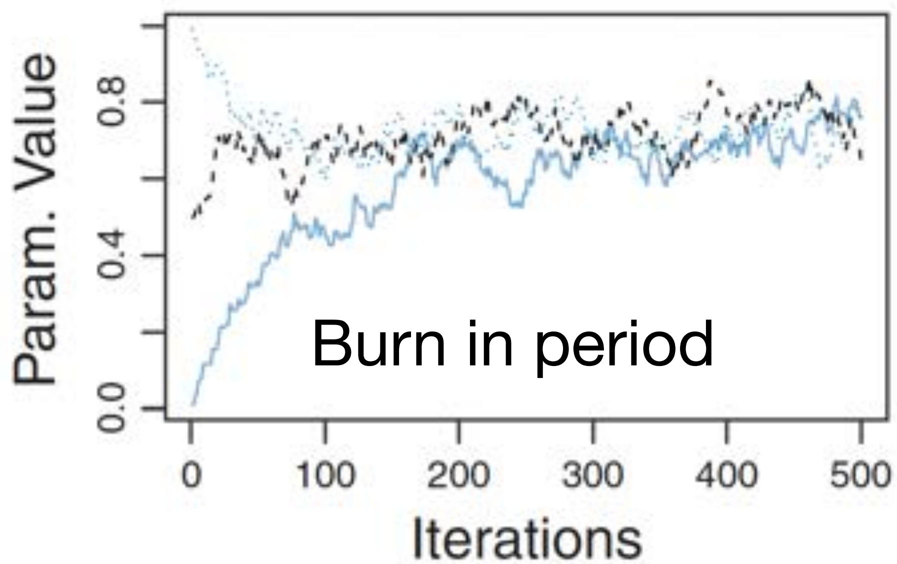


# Performance evaluation

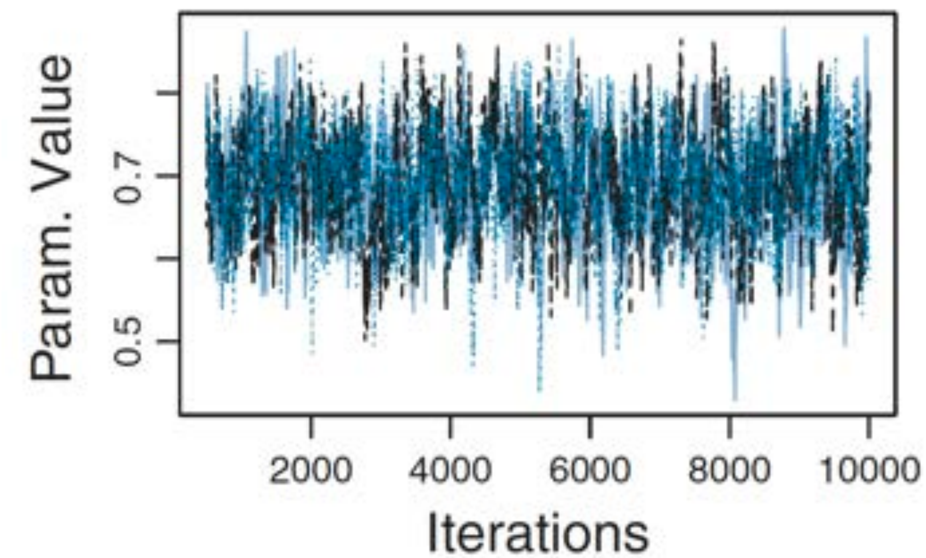
## Get rid of initialization effects. “Representativeness”

### Observe by eyes

Early in time (abandon)



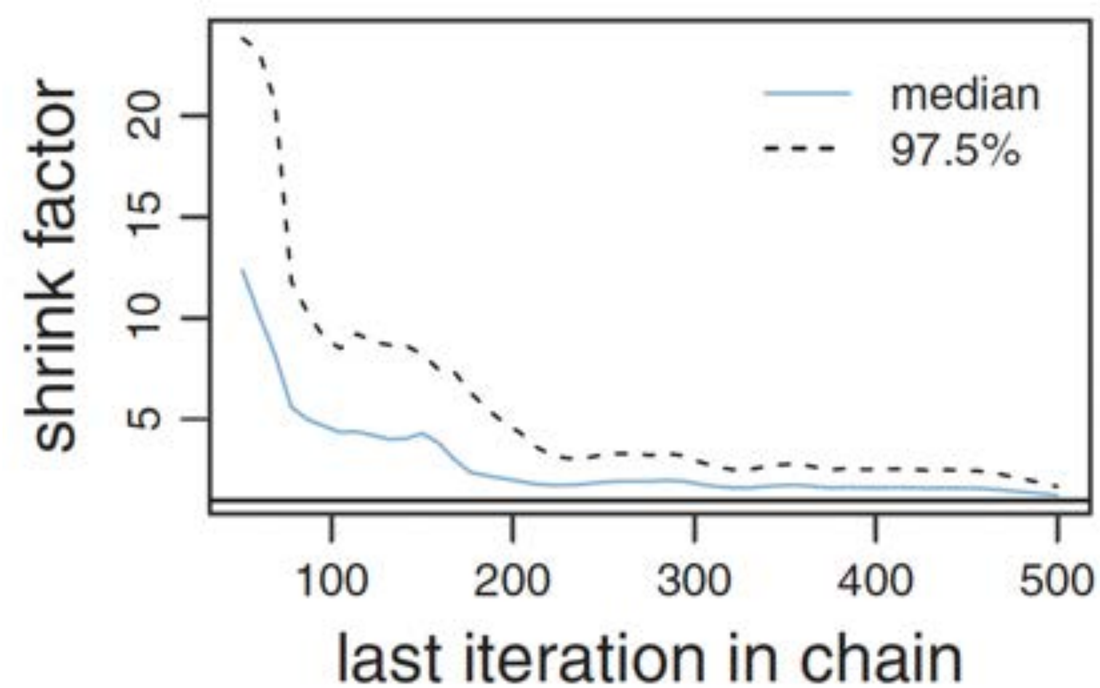

Later in time (data used)




### A formal measurement

Sinking factor (Gelman-Rubin statistic)

$$\frac{\text{Variance between traces}}{\text{Variance within traces}} \rightarrow \text{towards 1}$$



# Performance evaluation

## Practical technics. “Efficiency”

Parallel hardware

Adjust the sampling method

Change the parameterization

# MCMC Algorithms in Action

**ANIMATION**

https://chi-feng.github.io/mcmc-demo/

# APPENDICES

# How Do We Sample – A Quick Remark on U(0,1)

**Algorithm: A (Lousy) RNG – Coin Flipping**

① Flip a coin N times (e.g. 20) and record the results in sequence

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |

**Binary**

**(530969)**

**Decimal**

② Divide the (decimal) result by the maximum number possible

**Sampled Random Number:** $x = \frac{530969}{1048575} = \boxed{0.506373}$

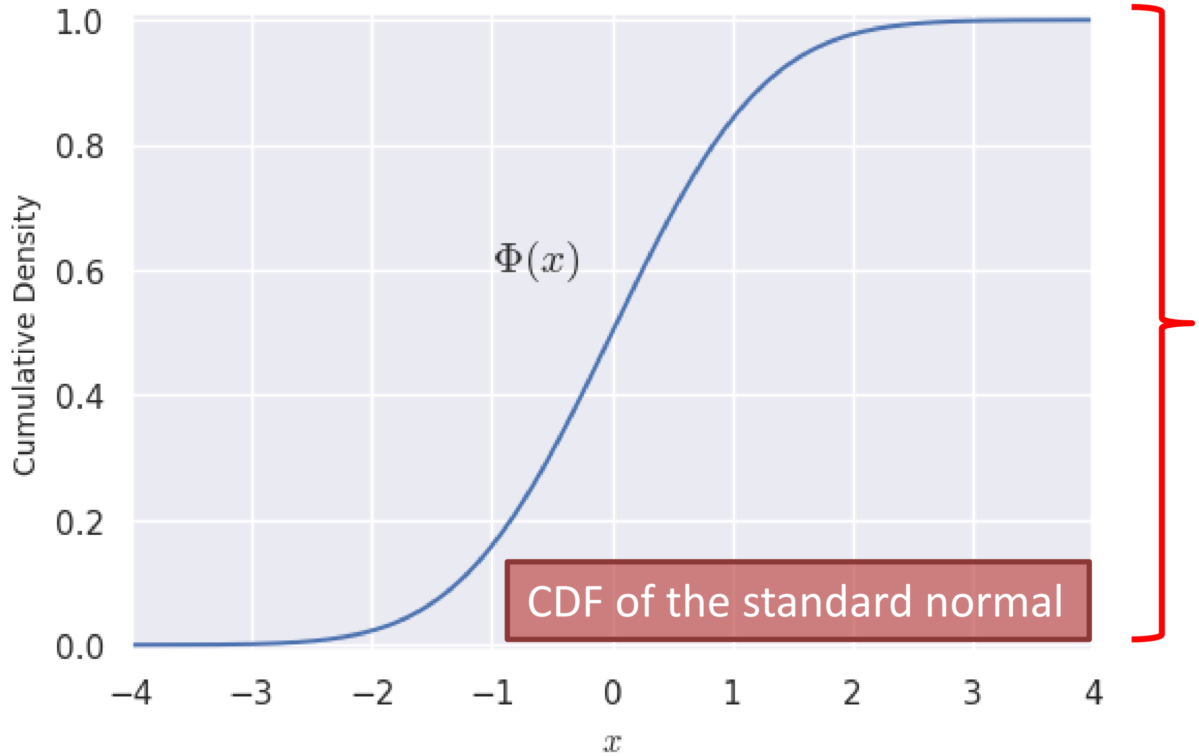
**All sampling is fundamentally constructed from a uniform RNG!**

# The Cumulative Density Function (CDF)

**RECALL:**

The cumulative density function is given by

$$F_X(\alpha) = \mathbb{P}(x \leq \alpha) = \int_{-\infty}^{\alpha} p_X(x)\,\mathrm{d}x$$



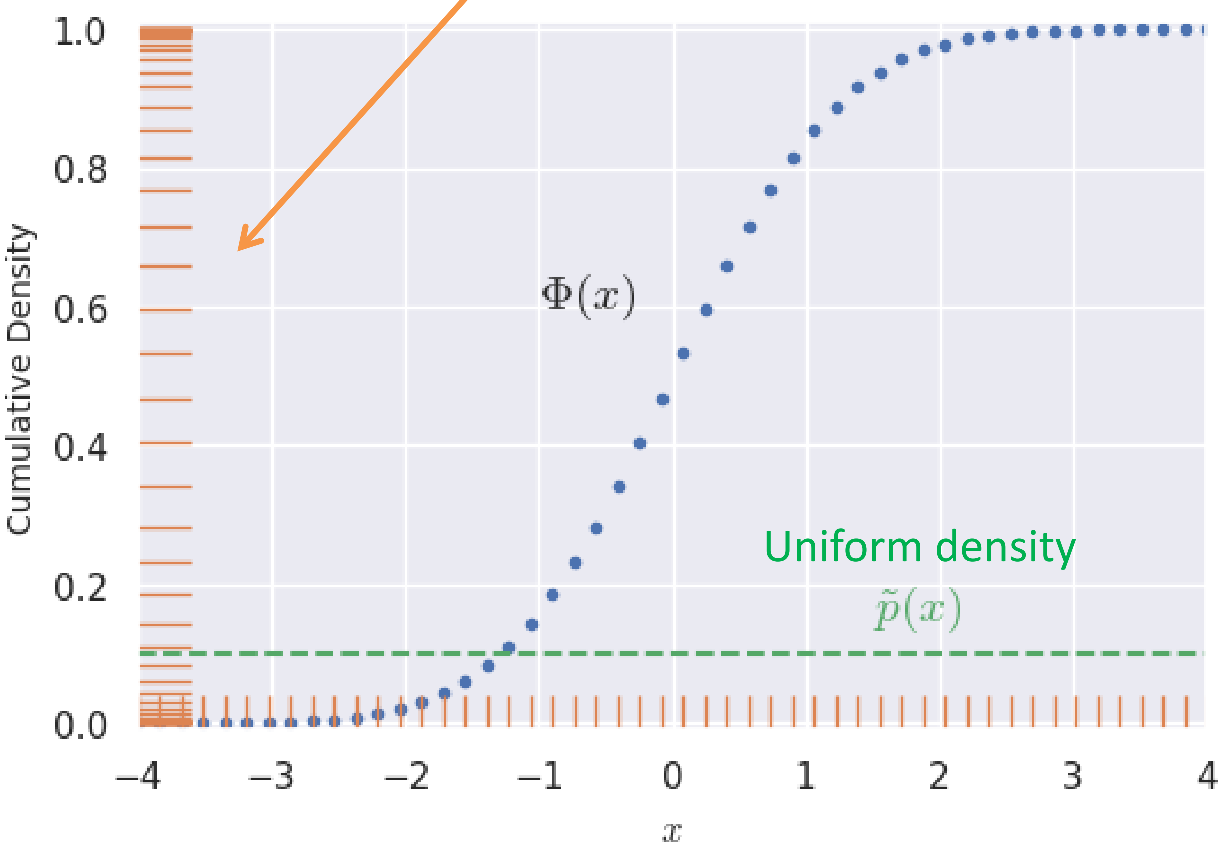

The range of the CDF conveniently lies **between 0 and 1**

What is the **density of points** along the y-axis?

# A First Attempt at Examining the y-axis Density

Let us begin by naively considering **evenly spaced points** along the x-axis…

This is called a "rug plot", commonly used to visualize the distribution of the data



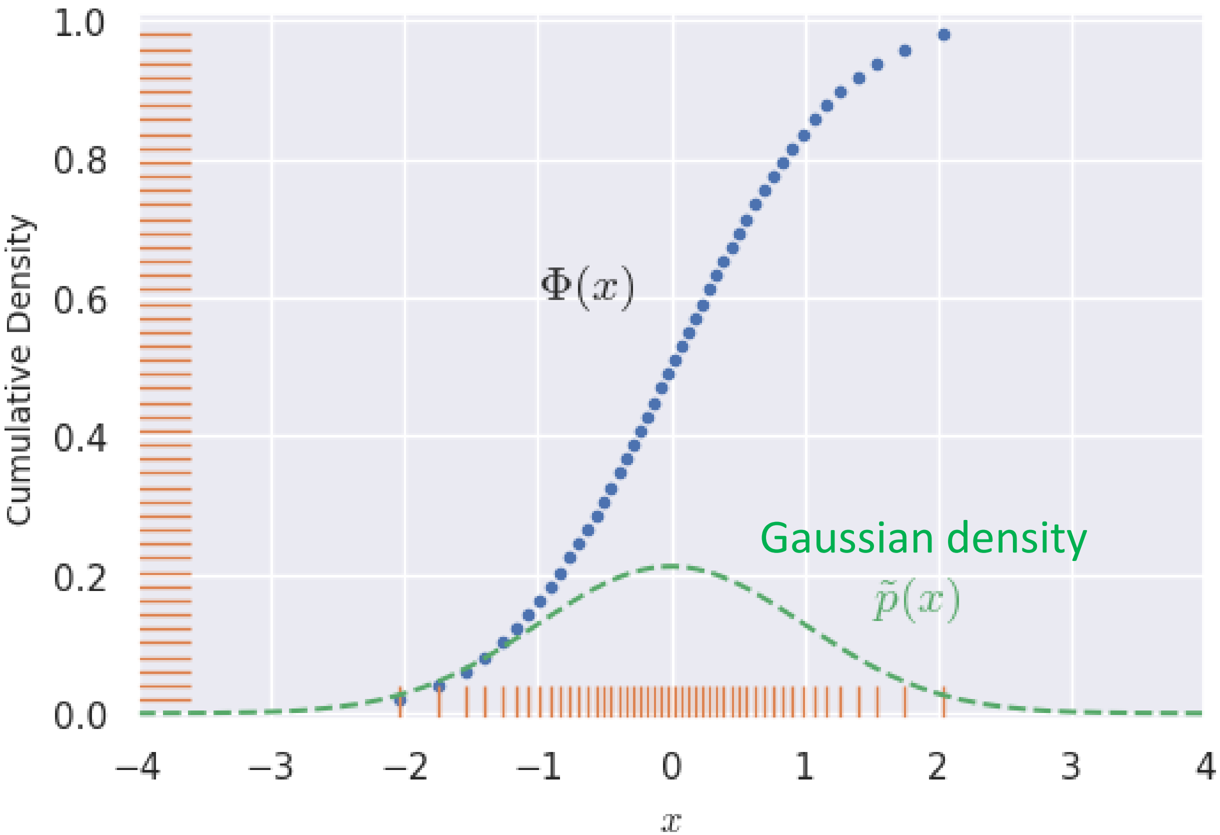

The points are clearly not evenly spaced along the y-axis

The spacing depends on how **'steep'** the curve is

**QUESTION:**
Is there something wrong here?

# A Correct Examination of the y-axis Density

**CORRECTION:** Since this is a CDF, the density of points along the x-axis should depend on the **PDF of x**!




The points seem to be evenly spaced along the y-axis!

Can we prove this? Can we use this result to do anything useful?

**(HOMEWORK EXERCISE)**

# Inverse CDF Transform Sampling

**IDEA:** Starting with samples drawn from the PDF of x, we end up with uniformly distributed samples when applying the CDF to them.

**Why not reverse the process?**

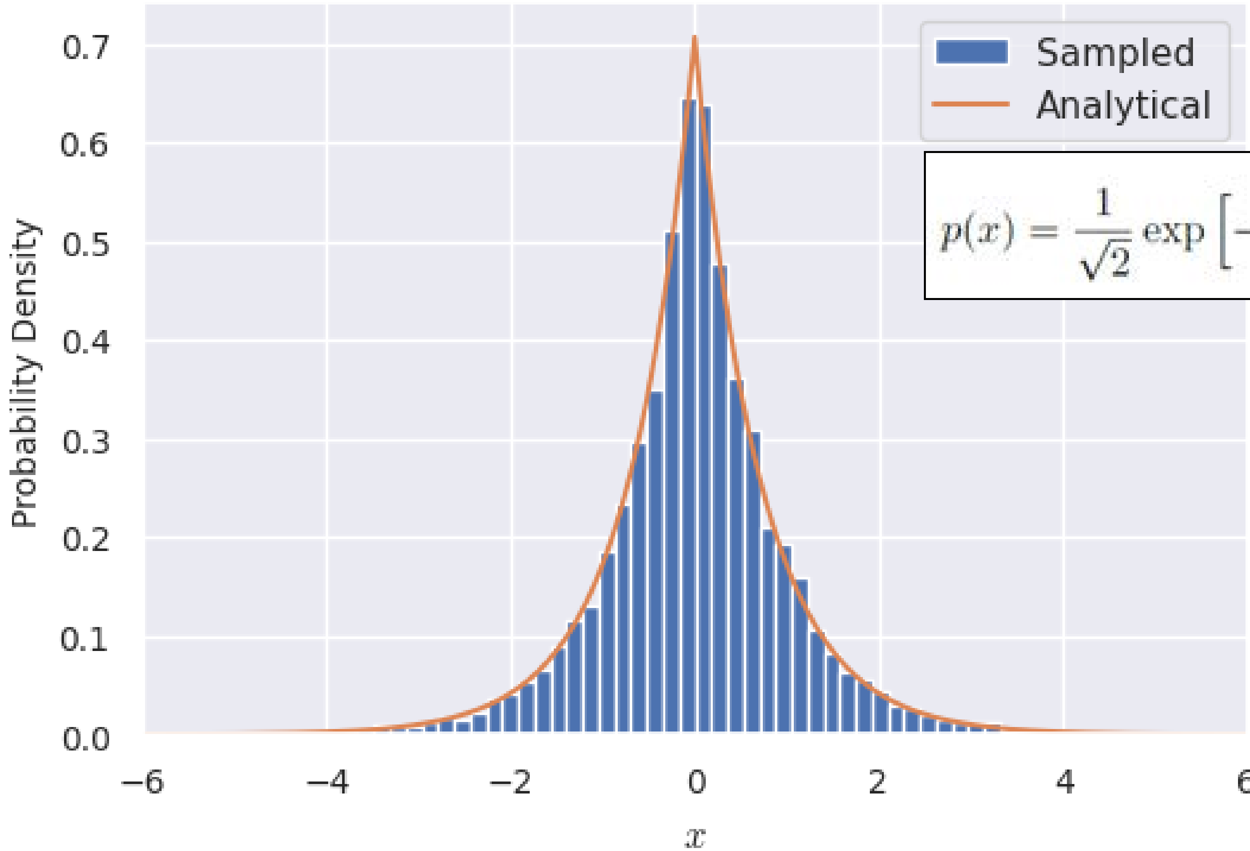
**Algorithm: Inverse CDF Sampling**

1. Calculate the inverse of the target CDF $F_X^{-1}$ — **Not possible in general**
2. Sample iid random variables from a uniform distribution $y^i \sim \mathcal{U}(0,1)$
3. Transforming each sample via $x^i = F_X^{-1}(y^i)$

# Inverse CDF Transform Sampling

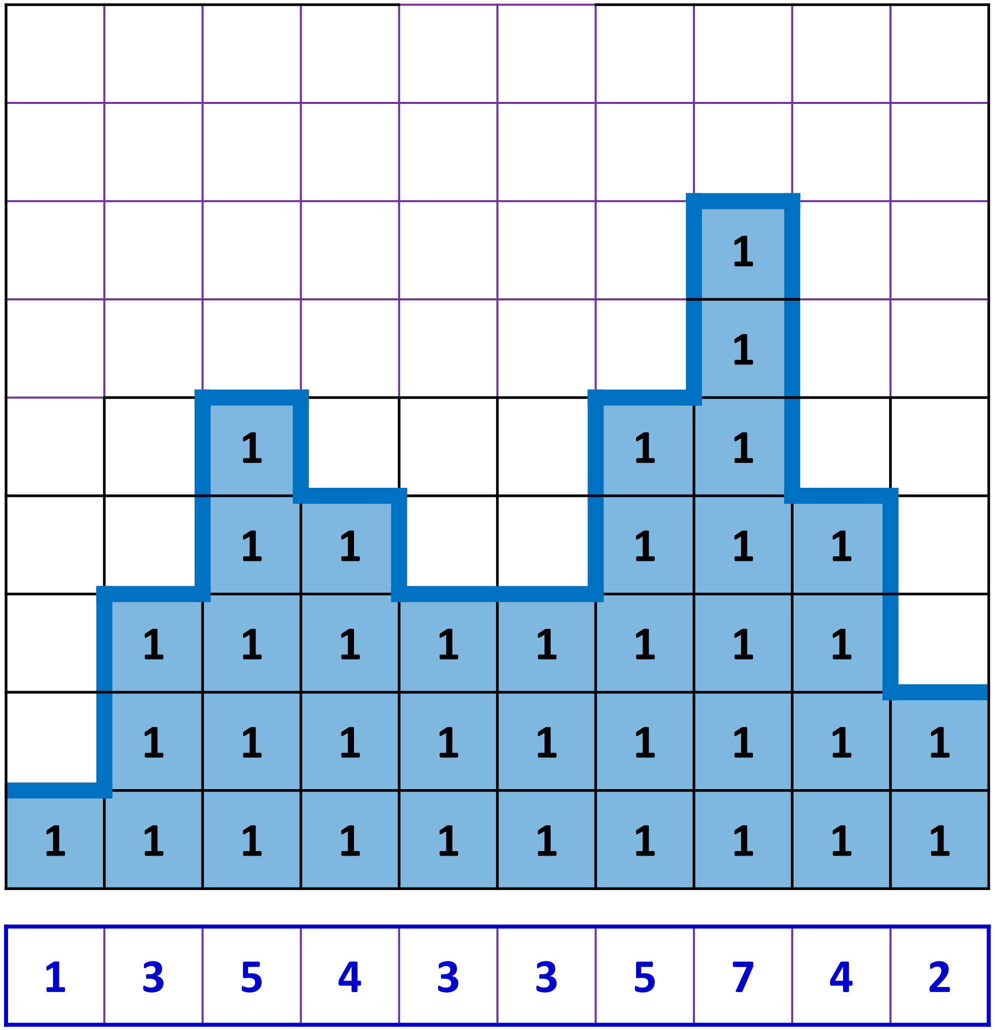
**(HOMEWORK EXERCISE)**

**Example:** Sampling from the Laplace distribution via inverse CDF transform sampling




$$p(x) = \frac{1}{\sqrt{2}} \exp\left[-\sqrt{2}\,|x|\right]$$

# Rejection Sampling: Why Can We Do "2D Shooting"?




We can 'smear out' the samples evenly in the y-direction – how much we can 'smear' depends on how much density we had at that point

This 'smearing out' makes every 2D square have **equal probability density**

Mathematically, the PDF of x can be considered as the **marginal distribution** of x wrt to the joint PDF

$$p(x,z) = \begin{cases} 1 & \text{if } 0 \le z \le p_X(x) \\ 0 & \text{otherwise} \end{cases}$$