

Stats 102C - Lecture 5-1: Introducing Markov Chains

Miles Chen, acknowledgements Michael Tsiang

Week 5 Monday

Section 1

Motivation

Big Picture

In the context of Bayesian statistics, we often want to find the expected value of a function $\mathbb{E}_f[h(X)]$. If solving the integral directly is too hard, we can estimate this value via Monte Carlo estimation. Monte Carlo estimation requires us to generate a sample of random values of X .

We have covered techniques for generating random values from distributions:

- inverse CDF
- convolutions
- rejection sampling

Inverse CDF is an effective and efficient method but requires us to find the CDF and inverse CDF, which can be difficult. Rejection sampling doesn't require finding F^{-1} , but as I will show, it can be inefficient.

Problems of efficiency with Rejection sampling

Consider the following example.

We want to generate values from the t-distribution with 3 df. We will fold the distribution in half at 0. It will be twice as tall as the t-distribution PDF with 3 df.

The PDF of the folded t-distribution is:

$$f(x) = 2 \cdot \frac{6\sqrt{3}}{\pi(3+x^2)^2} \text{ for } x \in (0, \infty)$$

Similar to the folded normal distribution, we can try to use the exponential distribution (with $\lambda = 1$) as our proposal distribution.

$$g(x) = e^{-x} \text{ for } x \in (0, \infty)$$

Both the desired PDF and the proposal PDF have the same support, so our design should work.

Problems of efficiency with Rejection sampling

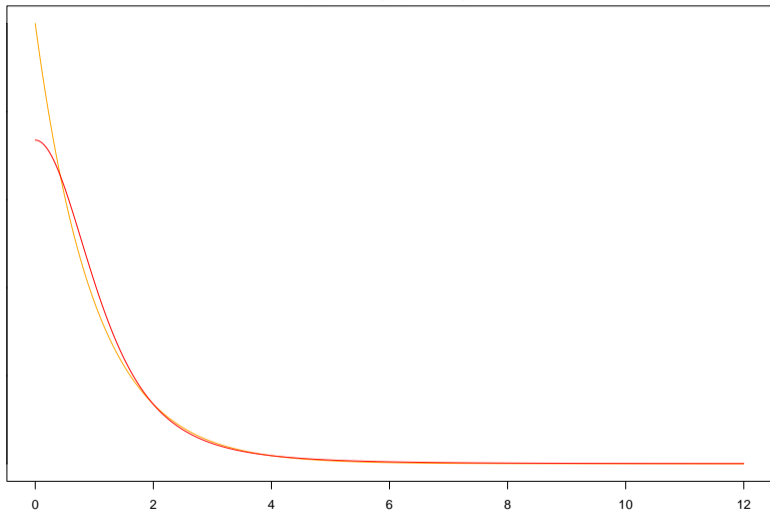
With rejection sampling, we need to find M so that $Mg(x) \geq f(x)$ for all x .

We search for $M = \max \frac{f(x)}{g(x)}$. When we do this, we run into a problem.

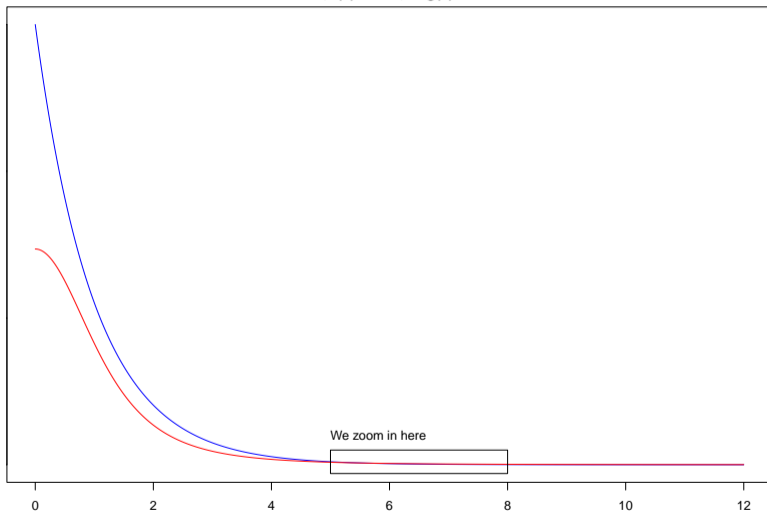
The tail of $f(x)$ decays on the order of x^4 in the denominator. Meanwhile the tail of $g(x)$ decays much faster: on the order of e^x in the denominator. ($f(x)$ is said to be 'heavy-tailed')

This means that no matter how big of a finite value of M you choose, eventually as $x \rightarrow \infty$, $Mg(x)$ will be less than $f(x)$.

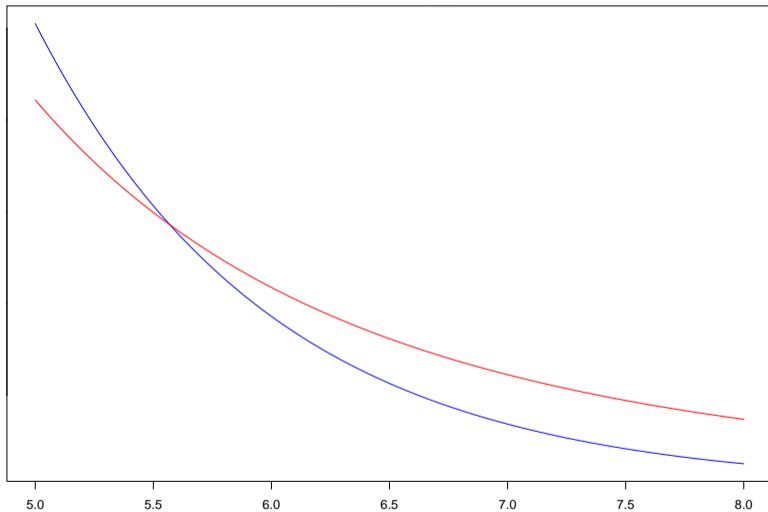
$f(x)$ in red, $g(x)$ in orange



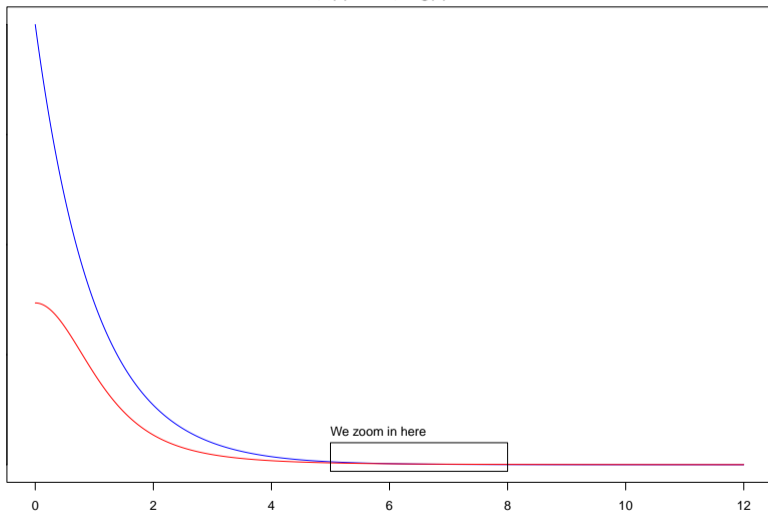
$M = 1.5$; $f(x)$ in red, $M^*g(x)$ in blue



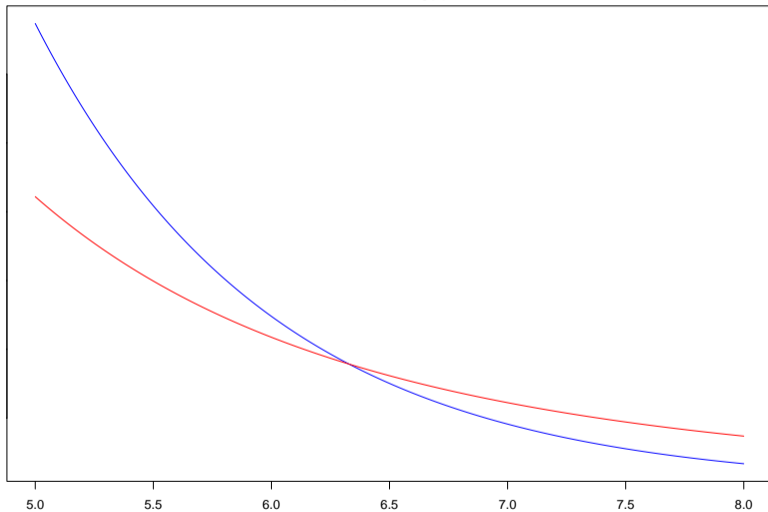
$M = 1.5$; $f(x)$ in red, $M^*g(x)$ in blue



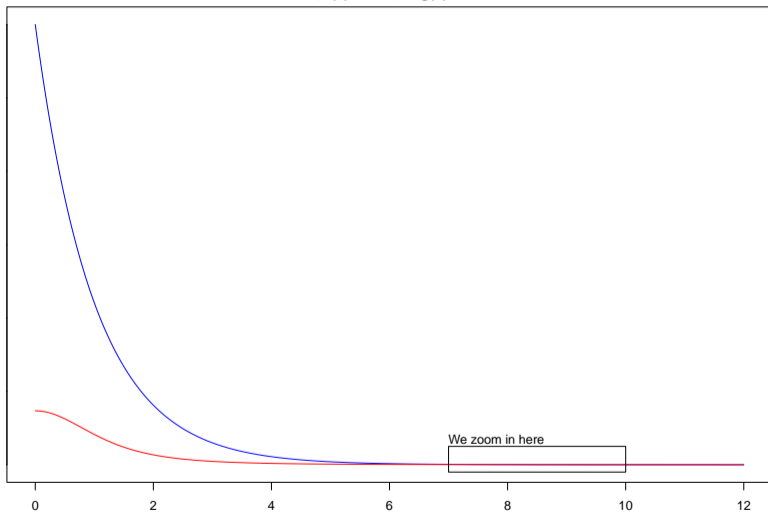
$M = 2$; $f(x)$ in red, $M^*g(x)$ in blue



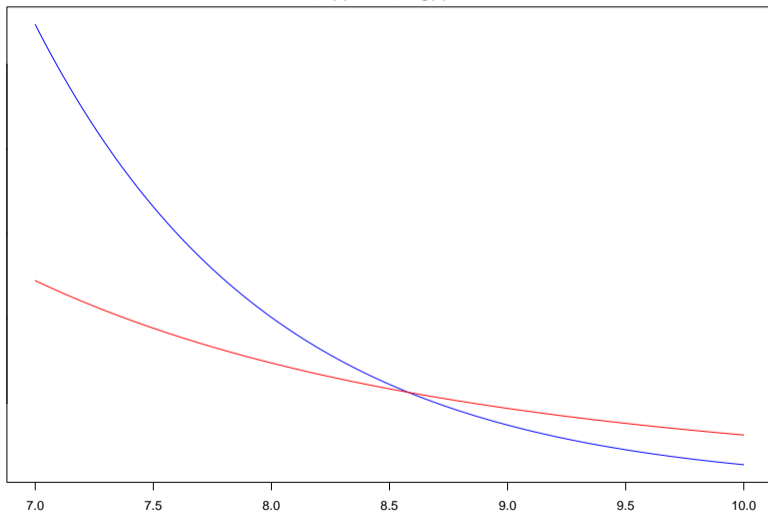
$M = 2$; $f(x)$ in red, $M^*g(x)$ in blue



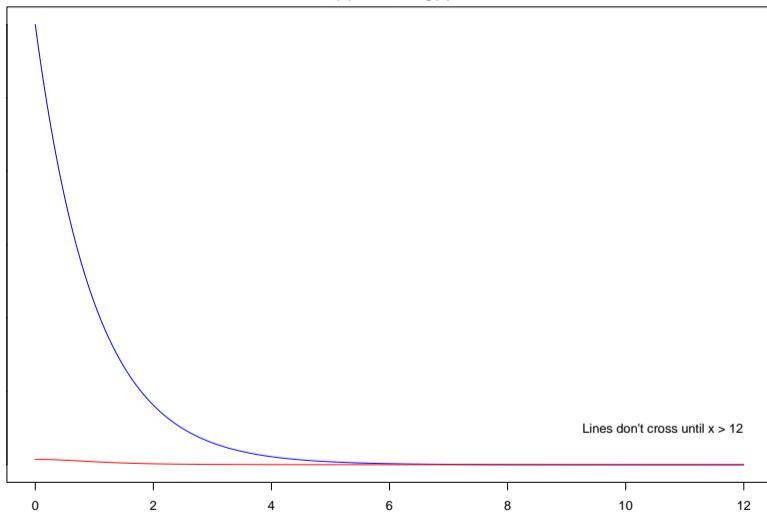
$M = 6$; $f(x)$ in red, $M^*g(x)$ in blue



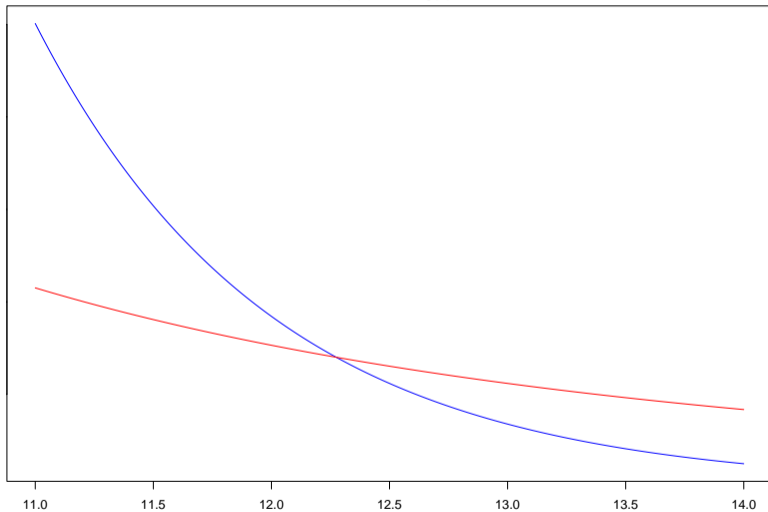
$M = 6$; $f(x)$ in red, $M^*g(x)$ in blue



$M = 60$; $f(x)$ in red, $M^*g(x)$ in blue



$M = 60$; $f(x)$ in red, $M^*g(x)$ in blue



Rejection sampling with the t-distribution

One could argue that with some arbitrarily large M , the rejection sampling algorithm would be accurate for values of X where $Mg(x) \geq f(x)$.

In the case of $M = 60$, $Mg(x) \geq f(x)$ for $X \leq 12.274$. This will be accurate for approx 0.995 of values.

However, if we do decide to implement this compromise of a solution, our algorithm will be very inefficient.

Using an $M = 60$ will lead us to accept only about 1 in 60 proposed values.

Rejection sampling with the t-distribution

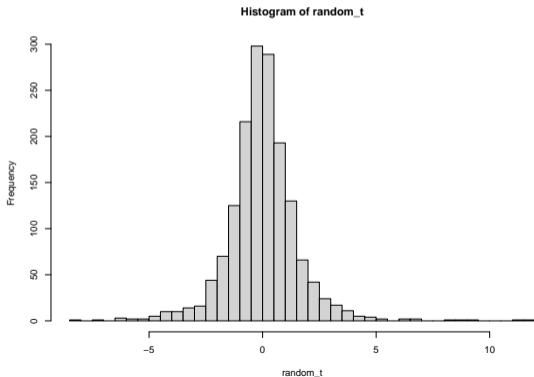
```
set.seed(1)
M <- 60
f <- function(x){ 2 * dt(x, df = 3)}
g <- function(x){ dexp(x) }
proposed_X <- rexp(10^5)
U <- runif(10^5)
r <- f(proposed_X) / (M * g(proposed_X))
accepted <- U < r
accepted_x <- proposed_X[accepted]
length(accepted_x) / length(proposed_X)
```

```
## [1] 0.01609
```

We proposed 100,000 values and accepted only 1609 of them.

To make our values from the folded t-distribution into values from the t-distribution, we can assign a positive or negative sign to the value with probability 0.5.

```
sign <- sample(c(1, -1), size = length(accepted_x), replace = TRUE)
random_t <- accepted_x * sign
hist(random_t, breaks = 30)
```



K-S test for our random sample

```
ks.test(random_t, pt, df = 3)
```

```
##  
##  Asymptotic one-sample Kolmogorov-Smirnov test  
##  
## data:  random_t  
## D = 0.017932, p-value = 0.6789  
## alternative hypothesis: two-sided
```

Our sample appears to pass the K-S test.

Our rejection sampling algorithm for sampling values from the t-distribution is not great.

- Pro: It seems to produce a sample that looks like it came from the t-distribution.
- Con: It's very inefficient. We're accepting only about 1.6% of proposed values
- Con: The algorithm will not produce the correct amount of values of X for $|X| > 12.275$.
(Our algorithm won't produce enough values in the extreme tails.)

There are better solutions for generating values from a t-distribution. One solution is a modification of the Box-Muller transform: <https://www.jstor.org/stable/2153537?seq=1>

The need for Markov Chains

While we have an elegant transform solution for the t-distribution, there are many other scenarios where rejection sampling is inefficient, and such a transform solution does not exist.

In higher dimensions, rejection sampling can become very inefficient.

In many of these cases, **Markov Chain Monte Carlo (MCMC)** can produce the sample we need. MCMC will simulate correlated samples that are (approximately) from a desired target distribution.

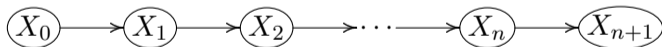
Section 2

Markov Chains

Definition:

A (discrete-time) Markov Chain $\{X_t : t = 1, 2, 3, \dots\}$ is a stochastic process (i.e. a sequence of random variables) that satisfy the Markov property:

$$\Pr(X_{n+1} = j | X_0 = i_0, \dots, X_{n-1} = i_{n-1}, X_n = i) = \Pr(X_{n+1} = j | X_n = i)$$



The **state space** of a Markov chain is the collection of all possible values for X_0, X_1, \dots

The **Markov property** means that the probability that the chain moves to state j on the next step only depends on the current state i , not on where the chain has been previously.

We will only consider Markov chains with countable or finite state spaces (i.e., discrete-state discrete-time Markov chains).

We can write the state space as $\{0, 1, 2, \dots\}$ (if countable) or $\{0, 1, 2, \dots, N\}$ (if finite).

Transition probability

The **transition probabilities** for a Markov chain $\{X_t : t = 0, 1, 2, \dots\}$ are defined as the conditional probabilities

$$P_{ij} := P(X_{n+1} = j | X_n = i),$$

for all n and all states i and j in the state space.

Transition Matrix

It is often convenient to describe the transition probabilities by a **transition matrix**

$$\mathbb{P} = \begin{bmatrix} P_{00} & P_{01} & P_{02} & \cdots \\ P_{10} & P_{11} & P_{12} & \cdots \\ \vdots & \vdots & \vdots & \\ P_{i0} & P_{i1} & P_{i2} & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix}$$

- All the entries are non-negative:

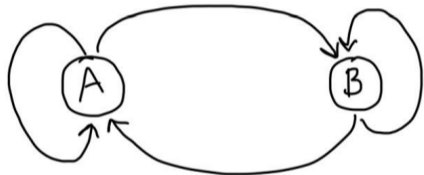
$$P_{ij} \geq 0, \text{ for all } i, j$$

- The sum of each row is 1. (Current state i is the row. Next state j is the column.)

$$\sum_{j=0}^{\infty} P_{ij} = \sum_{j=0}^{\infty} P(X_{n+1} = j | X_n = i) = 1, \text{ for all } i$$

Example - Two State Model

Here is a transition state diagram of Markov Chain with 2 possible states: A and B.

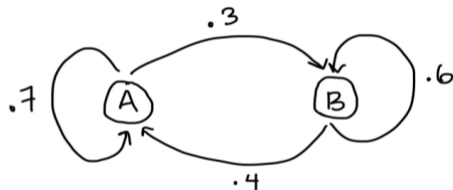


For example, A could represent sunny weather and B could represent not sunny (cloudy, rainy, etc.).

Example

We can add the probabilities of the transition matrix to the transition state diagram.

$$\mathbb{P} = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}$$



- If it is sunny (State A) today, there is a 70% probability it will be sunny (State A) tomorrow. There is a 30% chance it will not be sunny (State B).
- If it is not sunny (State B) today, there is a 40% probability it will be sunny (State A) tomorrow. There is a 60% chance it will not be sunny (State B).
- The probabilities in each row must add to 1. The probabilities of all arrows originating from a single node must add to 1.

Section 3

Example - Ehrenfest urn model

Example - Ehrenfest urn model

The **Ehrenfest urn model** is a classical mathematical model for diffusion of molecules through a membrane.

- Suppose we have two urns labeled A and B that contain a total of N balls (molecules).
- At each step, a ball is randomly chosen from the N balls and moved to the other urn (a molecule diffuses at random through the membrane).
- Let X_n denote the number of balls in urn A at step n .
- The possible values of X_n are $\{0, 1, 2, \dots, N\}$.
- The sequence $\{X_0, X_1, X_2, \dots, X_n, \dots\}$ is a Markov chain with state space $\{0, 1, 2, \dots, N\}$.

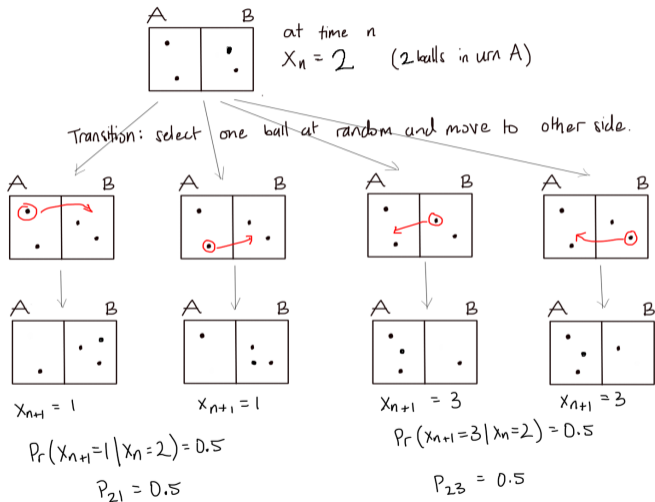
Example - Ehrenfest urn model

Let's say there are i balls in urn A at step n (i.e., $X_n = i$).

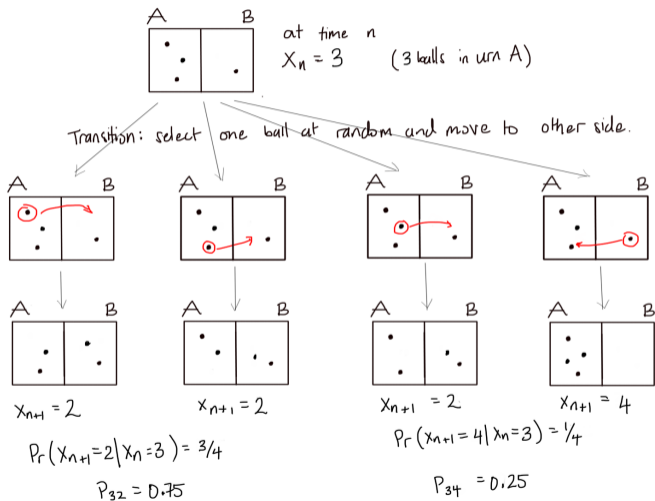
- If we know $X_n = i$, then at the next step, $X_{n+1} \in \{i - 1, i + 1\}$.
- If a ball is chosen from urn A , then $X_{n+1} = i - 1$.
- If a ball is chosen from urn B , then $X_{n+1} = i + 1$.

The ball is chosen randomly among all N balls (each has the same probability). The probability that a ball is chosen from a particular urn is the proportion of balls that are in that urn.

Ehrenfest urn model, $N = 4$



Ehrenfest urn model, $N = 4$

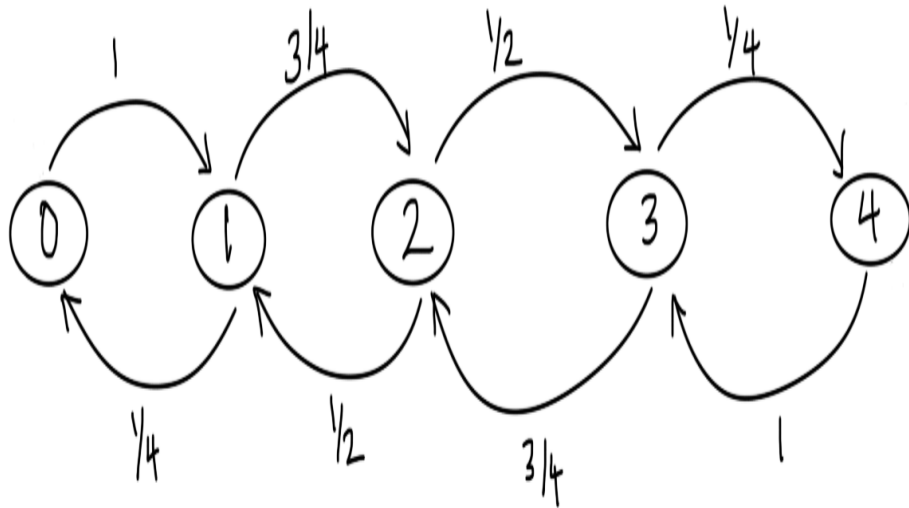


Example - Ehrenfest urn model

For $N = 4$, the transition matrix will be:

$$\mathbb{P} = \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} & P_{04} \\ P_{10} & P_{11} & P_{12} & P_{13} & P_{14} \\ P_{20} & P_{21} & P_{22} & P_{23} & P_{24} \\ P_{30} & P_{31} & P_{32} & P_{33} & P_{34} \\ P_{40} & P_{41} & P_{42} & P_{43} & P_{44} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1/4 & 0 & 3/4 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 3/4 & 0 & 1/4 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Example - Ehrenfest urn model, $N = 4$



Ehrenfest urn model

More generally, for N balls, the transition probabilities are:

- $P(X_{n+1} = i - 1 | X_n = i) = \frac{i}{N}$
- $P(X_{n+1} = i + 1 | X_n = i) = 1 - \frac{i}{N}$
- $P(X_{n+1} = j | X_n = i) = 0$, for $j \notin \{i - 1, i + 1\}$

And the values in the transition matrix \mathbb{P} are:

$$P_{ij} = P(X_{n+1} = j | X_n = i) = \begin{cases} \frac{i}{N} & \text{if } j = i - 1, \\ 1 - \frac{i}{N} & \text{if } j = i + 1, \\ 0 & \text{otherwise.} \end{cases}$$

Section 4

Example - Random Walk

Example - Random Walk

A **random walk** is a discrete-time stochastic process that is widely used to model the path an object or particle takes as it moves through space.

Some applications:

- The path a particle takes as it moves through a liquid or gas (this is a continuous-time process called Brownian motion)
- The path an insect takes as it searches for food
- A gambler's winnings or losses
- Stock prices

One-dimensional Random Walk

Let's look at one-dimensional random walks. (Stock prices, a gambler's winnings or losses)

Definition:

A **random walk** is a stochastic process

$$\{X_0, X_1, X_2, \dots, X_n, \dots\},$$

defined on the integers \mathbb{Z} , such that:

- The walk starts at 0: $X_0 = 0$
- At each step, the random walk moves to the right 1 unit with probability p and moves to the left 1 unit with probability $q = 1 - p$

One-dimensional Random Walk

Since $X_0 = 0$, then

$$X_1 = \begin{cases} 1 & \text{with probability } p, \\ -1 & \text{with probability } q. \end{cases}$$

If the current state is $X_n = i$, for $i \in \mathbb{Z}$. Then

$$X_{n+1} = \begin{cases} i + 1 & \text{with probability } p, \\ i - 1 & \text{with probability } q. \end{cases}$$

The random walk is a Markov chain, with transition probabilities

$$P(X_{n+1} = j | X_n = i) = \begin{cases} p & \text{if } j = i + 1, \\ q & \text{if } j = i - 1, \\ 0 & \text{otherwise.} \end{cases}$$

One-dimensional Random Walk

The random walk can also be expressed as a sum of iid random variables.

- Let $Y_1, Y_2, \dots, Y_n, \dots$ be iid random variables such that

$$\Pr(Y_i = 1) = p \quad \text{and} \quad \Pr(Y_i = -1) = q,$$

where $p + q = 1$.

- Define the Markov chain $\{X_0, X_1, X_2, \dots, X_n, \dots\}$ by:

$$\begin{aligned} X_0 &= 0 \\ X_n &= X_{n-1} + Y_n, \quad \text{for } n = 1, 2, \dots \end{aligned}$$

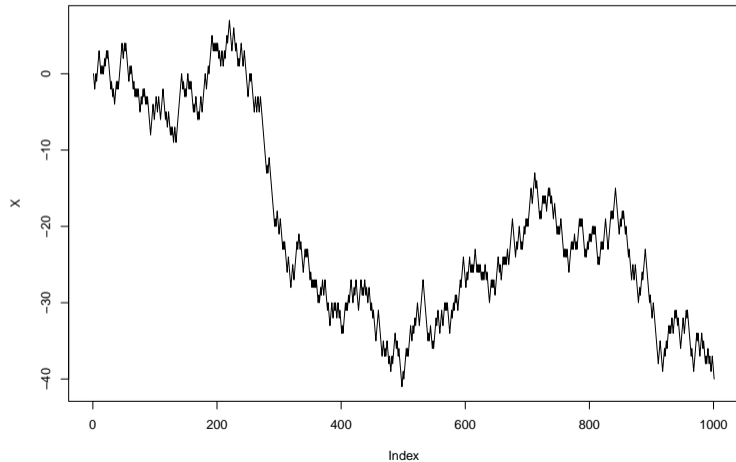
Random Walk

```
set.seed(1)
n <- 1000
p <- 0.5
q <- 1 - p

# Generate n iid samples from Y
Y <- sample(c(1, -1), size = n, replace = TRUE, prob = c(p, q))

# Compute the random walk X
X <- c(0, cumsum(Y))
```

```
plot(X, type="l")
```



```
set.seed(3)
Y <- sample(c(1, -1), size = n, replace = TRUE, prob = c(p, q))
X <- c(0, cumsum(Y))
plot(X, type="l")
```

