Monte Carlo Simulation

SYS-611: Simulation and Modeling

Paul T. Grogan, Ph.D.
Assistant Professor
School of Systems and Enterprises

Agenda

1. Review of Process Generators
2. Monte Carlo Simulation
3. Buffon's Needle Activity

Reading: S.M. Ross, "Statistical Analysis of Simulated Data," Ch. 8 in Simulation, 2012.

J.V. Farr, "Review of Probability and Statistics," Ch. 3 in Simulation of Complex Systems and Enterprises, Stevens Institute of Technology, 2007.

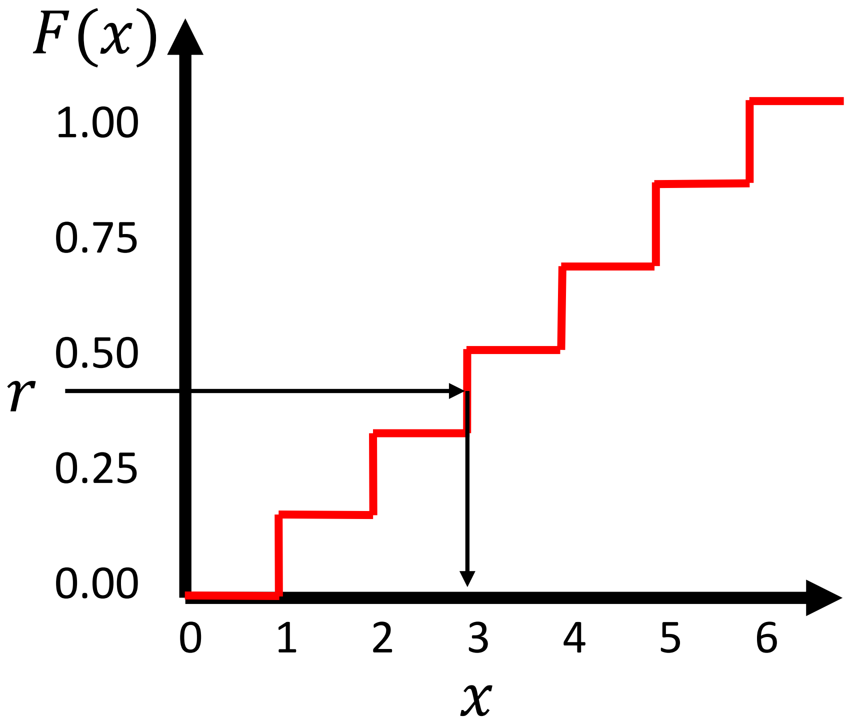
Review of Process Generators

Process Generator

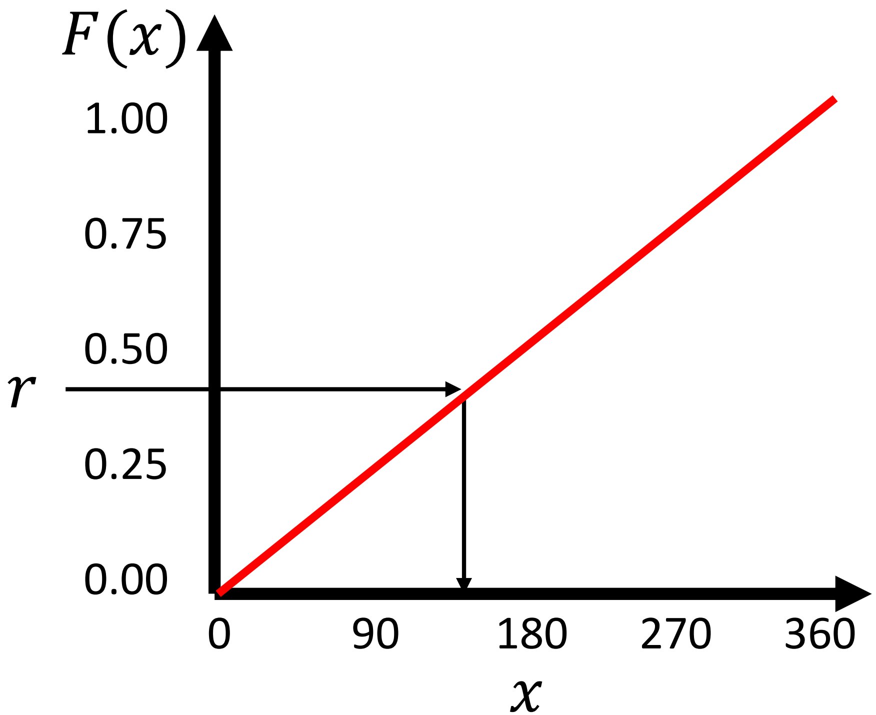
Process generators are algorithms that produce random variable values following a known distribution

- Built-in process generators exist in software for common distributions (Uniform, Binomial, etc.)
- Uniform(0,1) generator most useful in this class
- Two methods to generate arbitrary processes:
  - **Inverse transform method** – requires complete CDF
  - **Accept-reject method** – only requires PMF/PDF

Inverse Transform Method



$$r = F(x)$$
$$\rightarrow x = F^{-1}(r)$$



$$r = F(x)$$
$$\rightarrow x = F^{-1}(r)$$

Inverse Transform for Discrete Processes

```
import numpy as np

x = [1, 2, 3, 4, 5, 6]
cdf = [1./6, 2./6, 3./6,
       4./6, 5./6, 6./6]

def gen_roll_ivt():
    r = np.random.rand()
    for i in range(6):
        if r <= cdf[i]:
            return x[i]
```

	A	B	C
1	cdf	x	
2	0.00	1	
3	0.17	2	
4	0.33	3	
5	0.50	4	
6	0.67	5	
7	0.83	6	
8			
9	0.897527	=VLOOKUP(A9,A2:B7,2)	
10			
11			

- CDF lower bounds
- Random variable (X) values
- Uniform (0,1) (`=RAND()`)
- `VLOOKUP` function

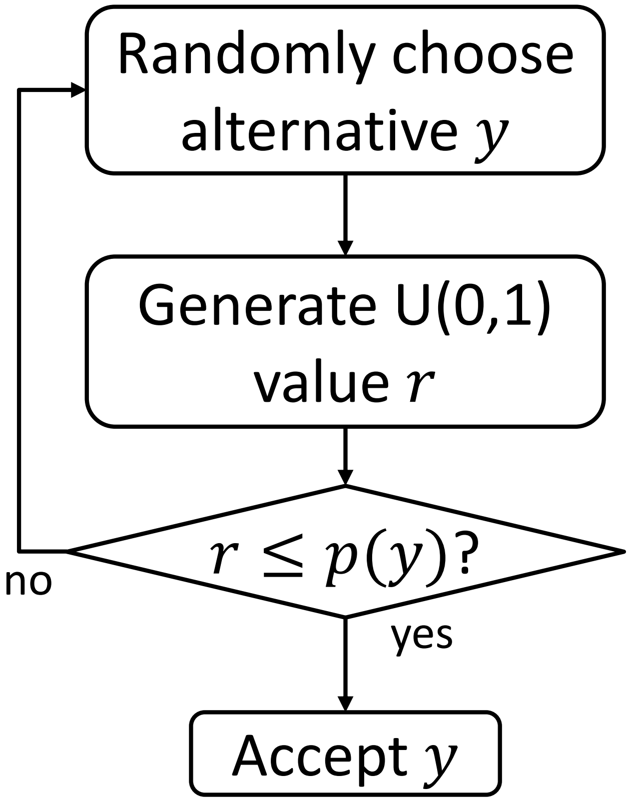
Inverse Transform for Continuous Processes

```
import numpy as np

def gen_spin_ivt():
    r = np.random.rand()
    return 360*r
```

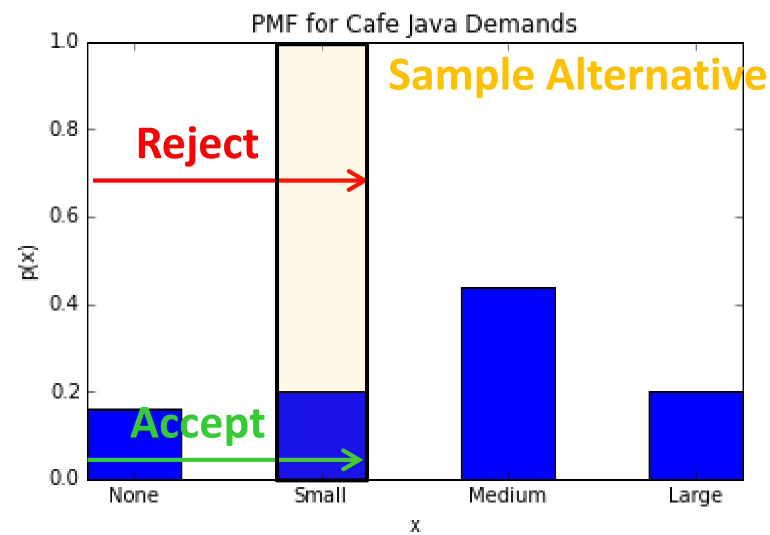
	A	B	C
1	0.828353	=360*A1	
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			

- Uniform (0,1) (`=RAND()`)
- Inverse CDF

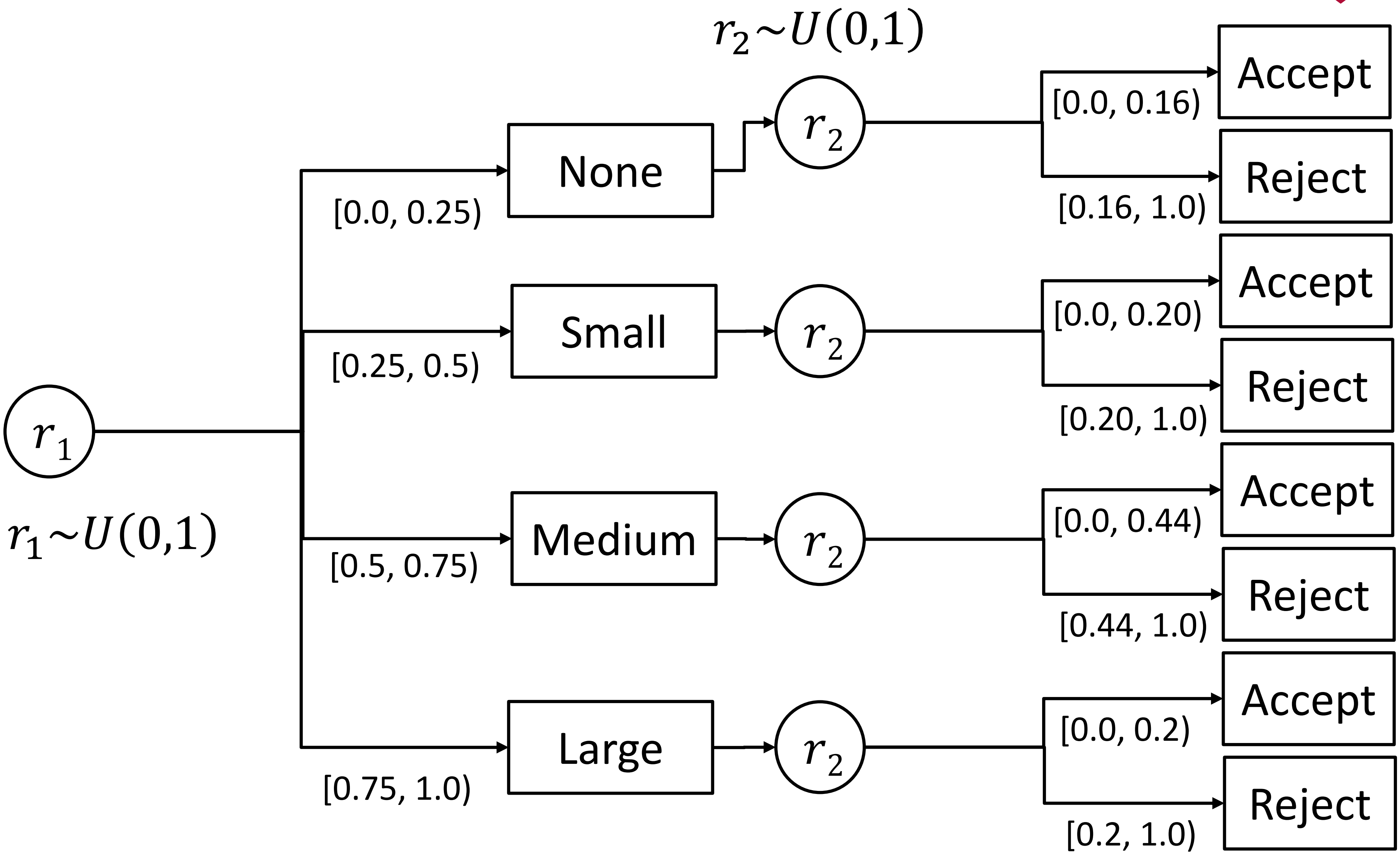
Accept-Reject Method for Discrete Processes



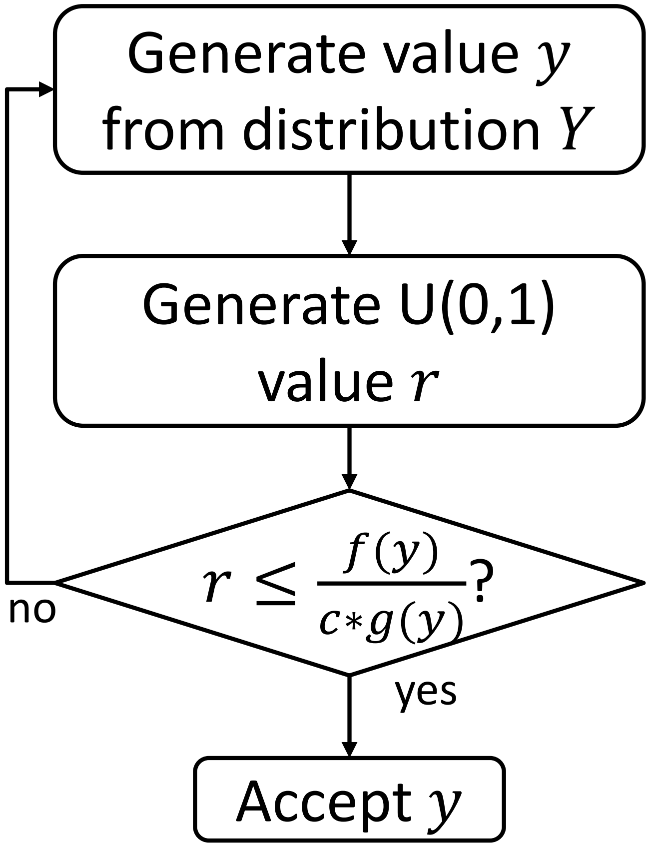
- Some CDFs are not easy to quantify or express
- Rely only on PMFs



Discrete ARM Flow Chart



Accept-Reject Method for Continuous Processes



- Some CDFs do not have closed-form equations
- Rely only on PDFs
  - Use a simpler "enveloping" distribution Y with PDF $g(y)$ where $c * g(y) \ge f(y) \; \forall \; y$
  - Simplest: $Y \sim \text{uniform}(a, b)$
  - Find maximum $f(x)$ and assign c appropriately

Accept-Reject (Ross p. 73)

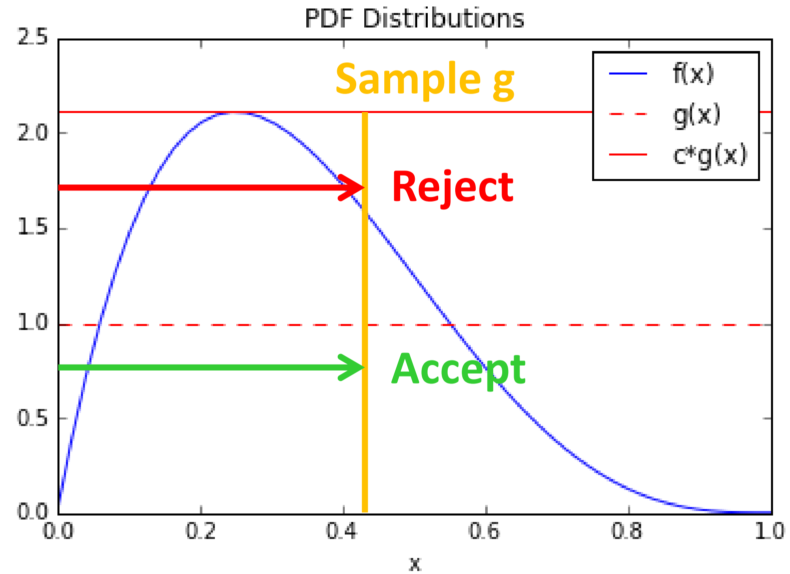
- PDF: $f(x) = 20 \cdot x(1-x)^3,\ 0 < x < 1$
- Proposed PDF: $Y \sim \text{uniform}(0,1),\ g(y) = 1,\ 0 < y < 1$
- What is the max value of $f(x)$ to ensure enveloping?

$$0 = f'(x) = 20(1-x)^3 - 60x(1-x)^2$$

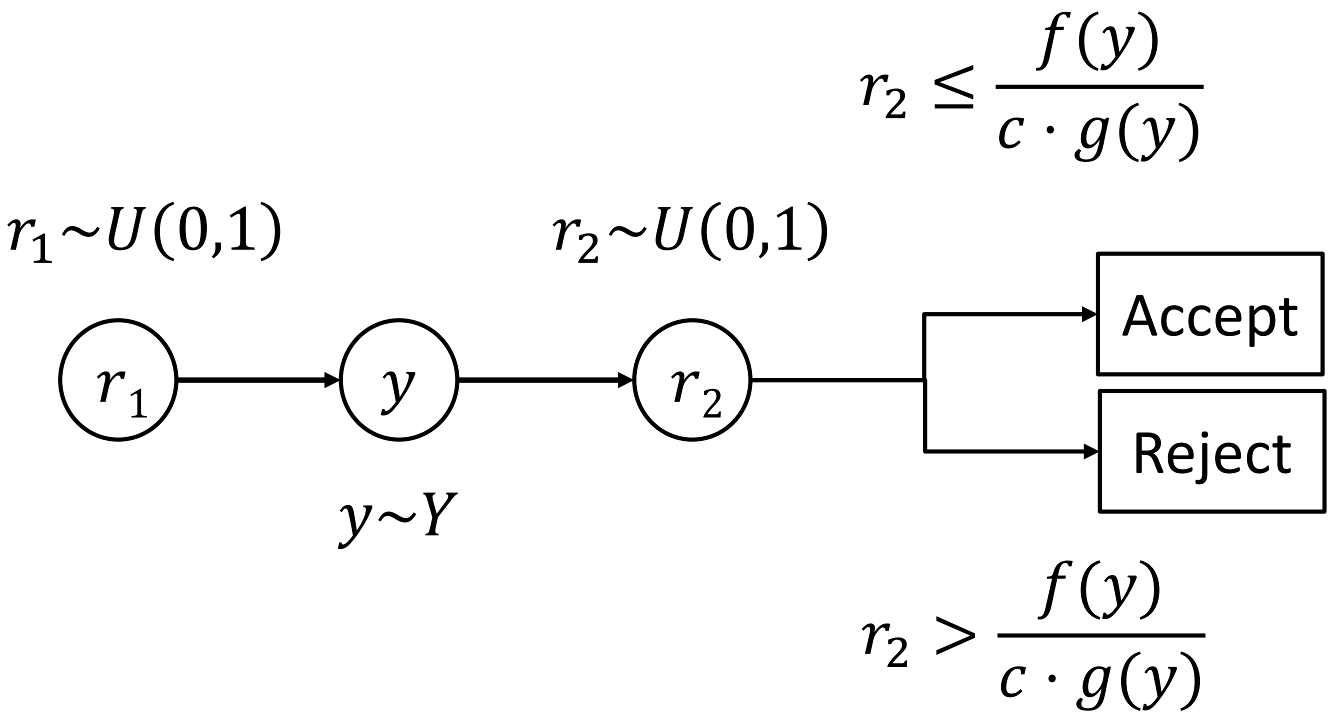
$$= -20(x-1)^2(4x-1)$$

$$\rightarrow f(0.25) = \frac{135}{64} \rightarrow c = \frac{135}{64} \approx 2.1$$

- $r \leq \frac{f(y)}{c \cdot g(y)} = \frac{256}{27} y(1-y)^3$
- Equivalently: $r * c \leq f(y)$



Continuous ARM Flow Chart



Monte Carlo Simulation

Monte Carlo Simulation

Monte Carlo simulation solves a problem (possibly deterministic in nature) by statistically analyzing samples from a stochastic model

- Developed in 1940s classified research by von Neumann, Ulam, Fermi, Metropolis (& others)
- Code named by Ulam and Metropolis in reference to *Monte Carlo* casino in Monaco
- Early applications limited due to computation (ENIAC: 1st general-purpose computer in 1946)

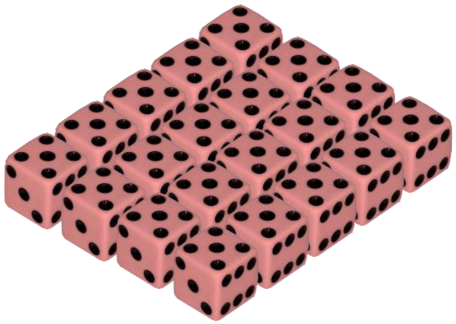
Monte Carlo Approach

1. Identify **elementary state variables** and **random variables** with **probability distributions**
2. Identify **derived state variables** and their functional form
3. Determine **number of samples** required or other convergence criteria
4. For each sample, **generate RVs** and compose and **record derived state variables**
5. Compute/visualize statistics from results

Dice Fighters Exercise

Red Team:

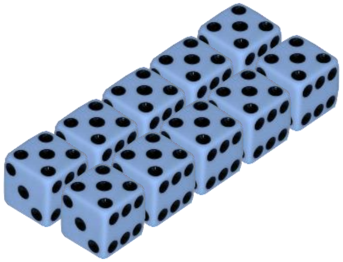
- 2x fighting force size
- Simple weapons



- Roll 6 to hit target

Blue Team:

- Small fighting force
- 3x effective weapons



- Roll 4|5|6 to hit target

Q: What is the probability of Red winning?

Modeling Dice Fighters

- Elementary random variables

$$h_R\text{: number of red hits}$$

$$h_R \sim \text{binomial}\left(p = \frac{1}{6}, n = R\right)$$

$$h_B\text{: number of blue hits}$$

$$h_B \sim \text{binomial}\left(p = \frac{2}{3}, n = B\right)$$

- Other state variables

$$R_t\text{: red team size at time } t$$

$$B_t\text{: blue team size at time } t$$

- Derived state variable

$$W = \begin{cases} 2 & \text{if } R_f > 0 \\ 1 & \text{if } B_f > 0 \\ 0 & \text{otherwise} \end{cases}$$

- Initial conditions

$$R_0 = 20, \qquad B_0 = 10$$

- State changes

$$R_{t+1} = R_t - h_B$$

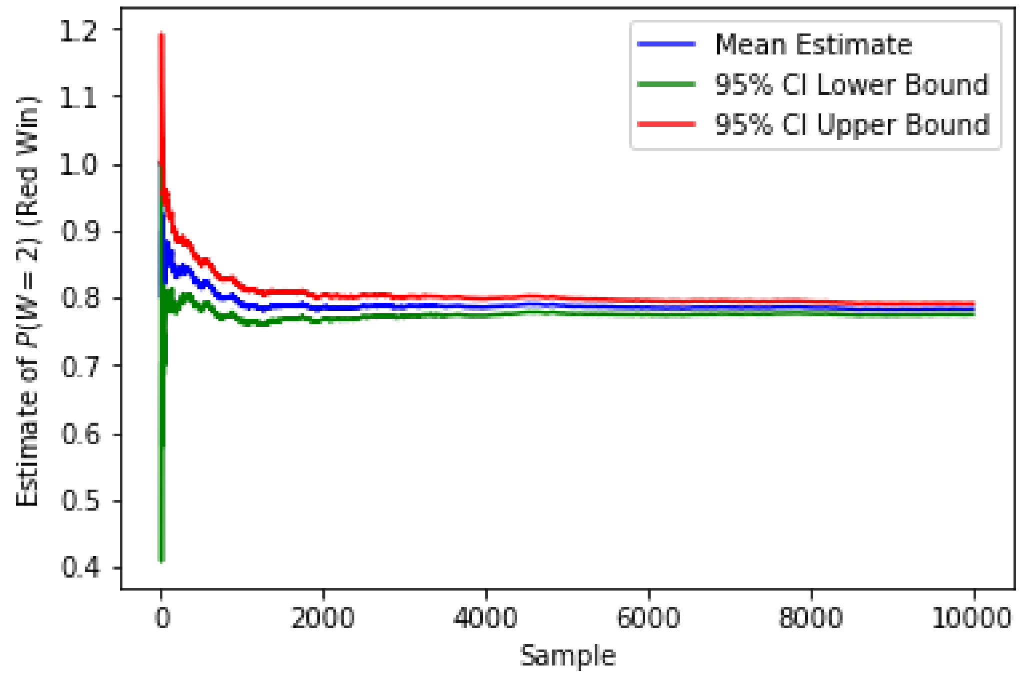
$$B_{t+1} = B_t - h_R$$

- Terminal condition

$$R_t \leq 0 \text{ or } B_t \leq 0$$

Dice Fighters M.C. Simulation

```
def gen_battle():
  global red_size, blue_size

  red_size = 20
  blue_size = 10

  while not is_complete():
    red_hits = gen_red_hits()
    blue_hits = gen_blue_hits()
    red_size -= blue_hits
    blue_size -= red_hits

  if red_size > 0:
    return 2
  elif blue_size > 0:
    return 1
  else:
    return 0

samples = np.array([gen_battle()
         for i in range(10000)])

print np.mean(samples==2)
print 1.96*stats.sem(samples==2)
```



$$P(W = 2) = 0.783 \pm 0.008 \text{ (95\% CI)}$$

$$P(W = 1) = 0.213 \pm 0.008 \text{ (95\% CI)}$$

$$P(W = 0) = 0.004 \pm 0.001 \text{ (95\% CI)}$$

Buffon’s Needle Activity

Example: Buffon's Needle

Suppose the floor is made of parallel strips of wood, each the same width t, and we drop a needle of length l onto the floor.

What is the probability that the needle will lie across a line between two strips?

George-Louis Leclerc, Compte de Buffon, c. 1733

Consider "short needle" cases with $l \leq t$

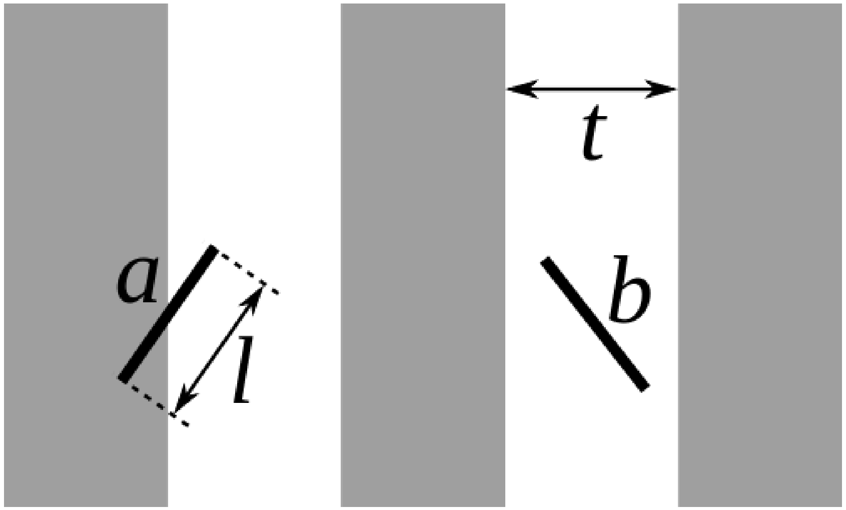
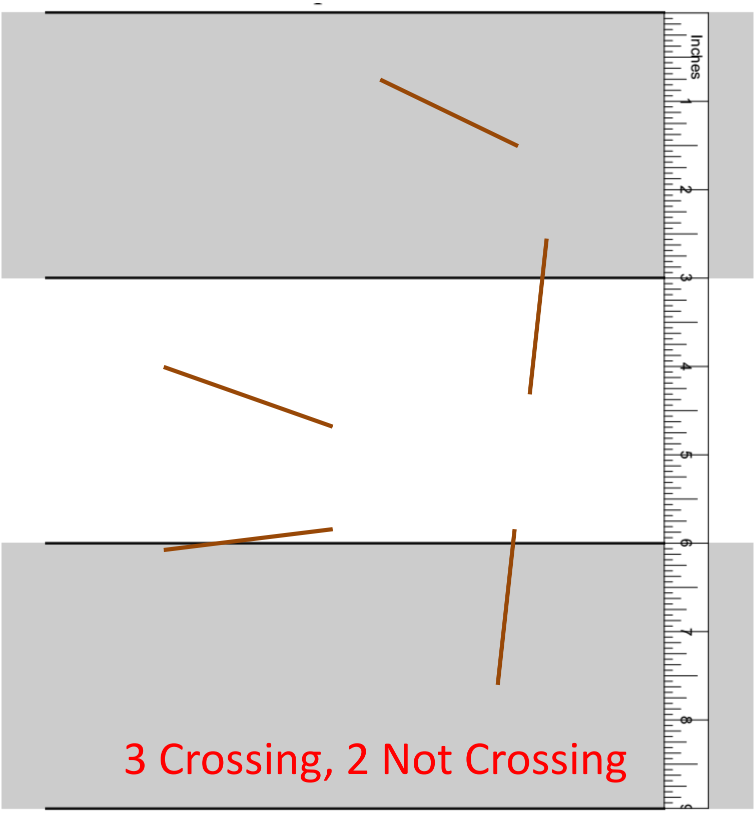


Image: Claudio Rocchini / Wikimedia

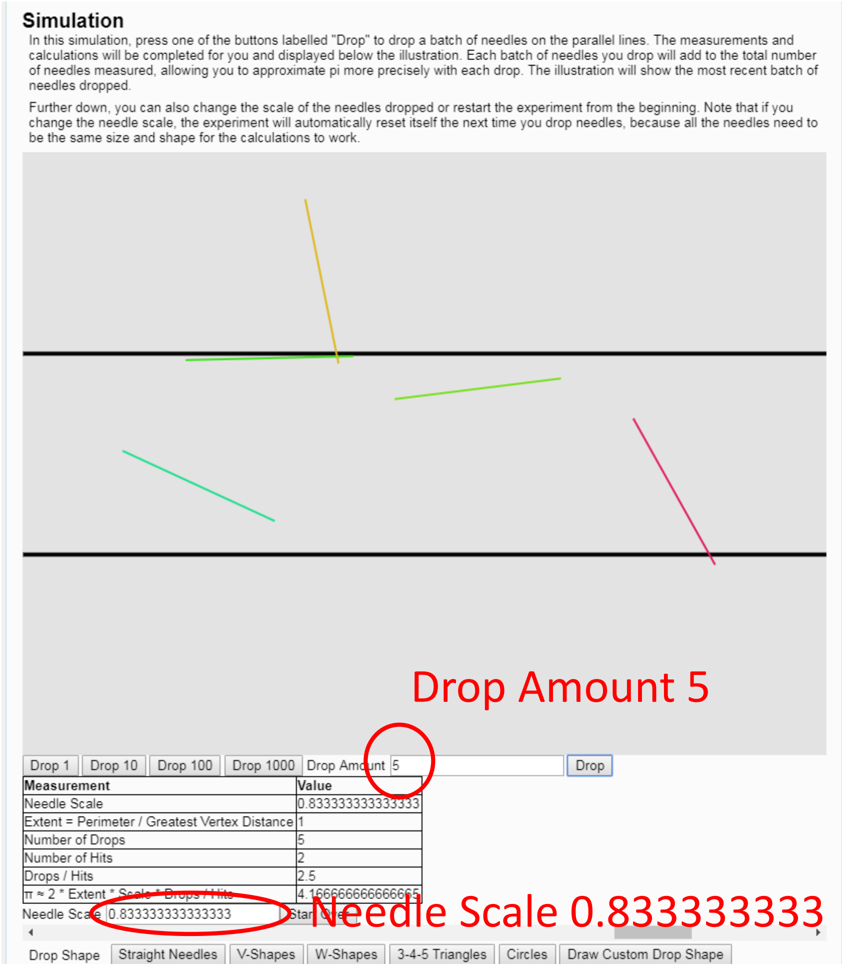
Buffon's Needle Experiment

Lines are $t = 3$ in. wide
Needles are $l = 2.5$ in. long



Submit at goo.gl/U9Awqj

mste.illinois.edu/activity/buffon/



Modeling Buffon's Needle (1)

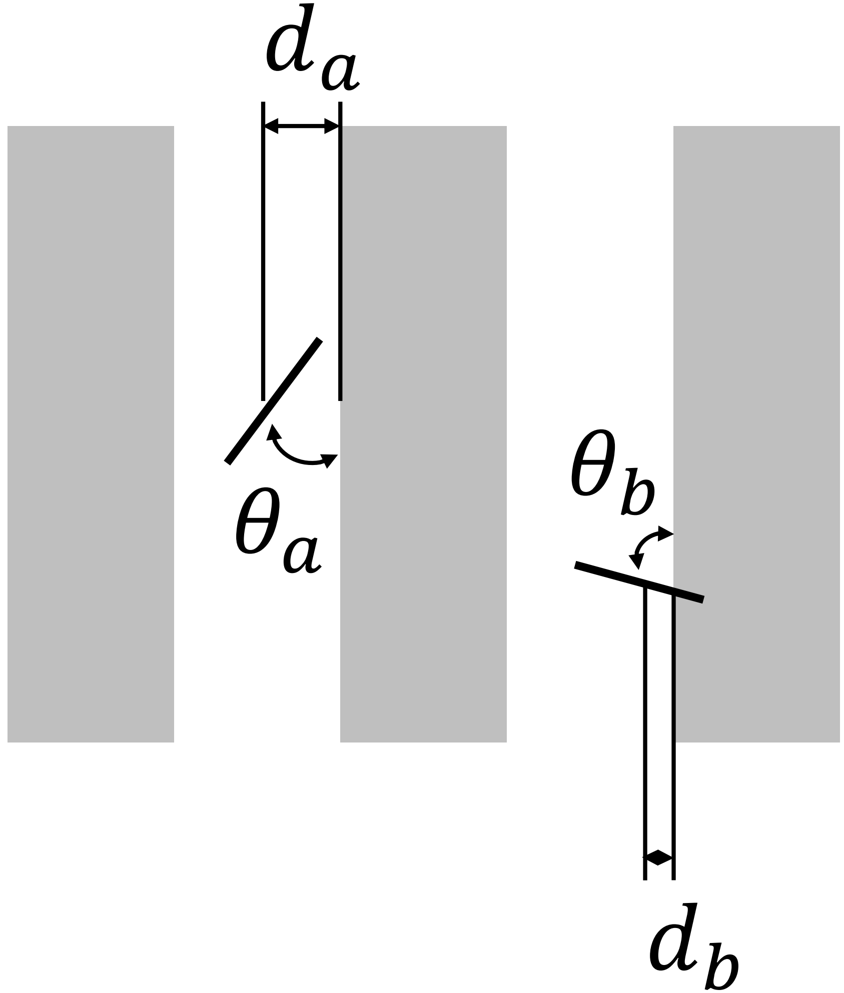
1. Identify elementary state variables with probability distributions

- Distance from needle midpoint to nearest line

  $d \sim U(0, t/2) \Rightarrow f(d) = 2/t$

- Acute angle between needle and nearest line

  $\theta \sim U(0, \pi/2) \Rightarrow f(\theta) = 2/\pi$

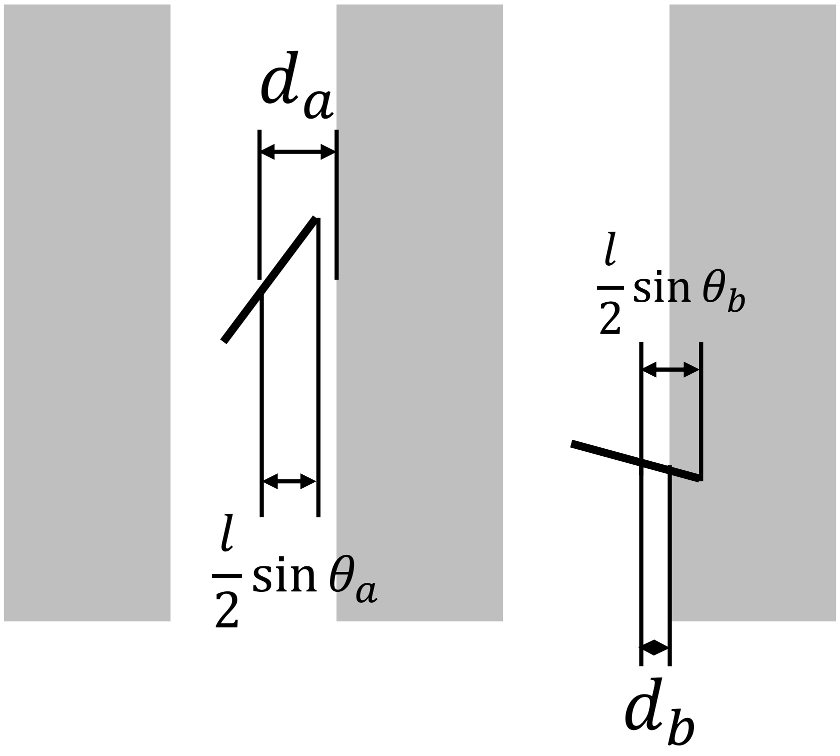


Modeling Buffon's Needle (2)

2. Identify derived state variables and their functional form

- X: Needle crosses line

$$X = \begin{cases} 1 & \text{if d} \leq \dfrac{l}{2}\sin\theta \\ 0 & \text{otherwise} \end{cases}$$



Modeling Buffon's Needle (3)

3. Determine number of samples required or other convergence criteria

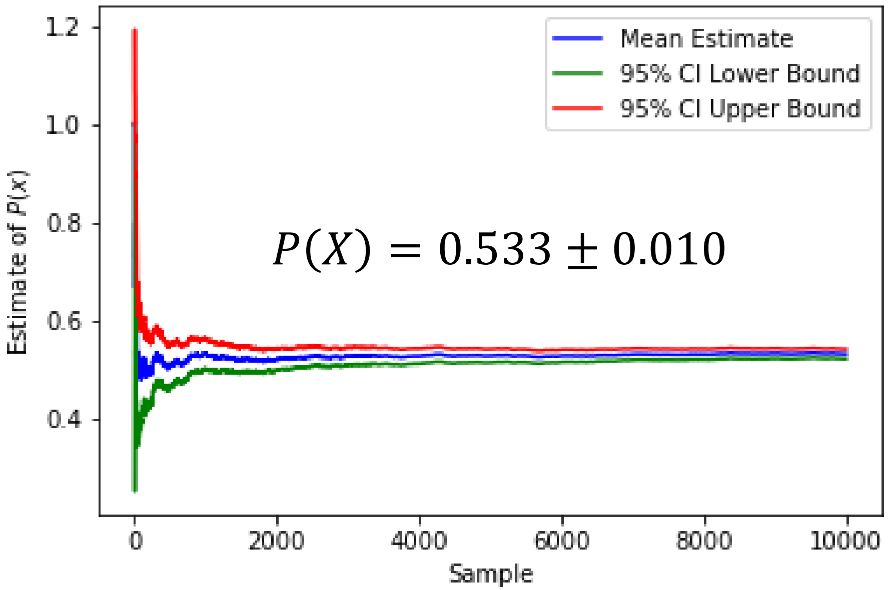
- Estimate $P(X)$ with 95% confidence and ± 0.01 accuracy
- Apply Central Limit Theorem:
  - $(1-\alpha) * 100\%$ confidence interval: $\bar{x} \pm z_{1-\alpha/2} \frac{s_x}{\sqrt{n}}$
  - Critical z-score: $z_{1-\alpha/2} = z_{0.975} = 1.96$
  - $0.01 = z_{1-\alpha/2} \frac{s_x}{\sqrt{n}} \Rightarrow n = \left(\frac{z_{1-\alpha/2} s_x}{0.01}\right)^2 = 38416 \cdot {s_x}^2$

Buffon's Needle Simulation

4. For each sample, generate primary RVs and compose and record derived state variables

5. Visualize statistics from derived state variables

```
import numpy as np
import scipy.stats as stats

line_width = 3.0
needle_length = 2.5

def drop_needle():
    d = np.random.rand()*line_width/2
    theta = np.random.rand()*np.pi/2
    if d < needle_length/2*np.sin(theta):
        return True
    else:
        return False

samples = [drop_needle()
           for i in range(10000)]

print np.mean(samples)
print 1.96*stats.sem(samples)
```



Buffon's Needle: Analytical

$$P(X) = \int_{\theta=0}^{\pi/2} \int_{d=0}^{\frac{l}{2}\sin\theta} f(d \cap \theta)\, dd\, d\theta$$

$$= \int_{\theta=0}^{\pi/2} \int_{d=0}^{\frac{l}{2}\sin\theta} \frac{2}{t} \cdot \frac{2}{\pi}\, dd\, d\theta$$

$$= \int_{\theta=0}^{\pi/2} \frac{2 \cdot l}{t \cdot \pi} \cdot \sin\theta\, d\theta$$

$$= -\frac{2 \cdot l}{t \cdot \pi} \cos\theta \Big|_{\theta=0}^{\frac{\pi}{2}}$$

$$= \frac{2 \cdot l}{t \cdot \pi} = \frac{5}{3\pi} = 0.5305$$

$$\Rightarrow \boldsymbol{\pi = \frac{2 \cdot l}{t \cdot P(X)}}$$