Statistical Thinking: Monte Carlo Simulation

Industrial AI Lab.

Prof. Seungchul Lee

Probability of Having Head with a Fair Coin

- Head 1 and Tail 0

```python
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```

```python
n_trials = 100
n_H = 0

for i in range(n_trials):
    flip = np.random.randint(2)
    if flip == 1:
        n_H += 1

print(n_H/n_trials)
```

```
0.49
```

Expectation

- Q: The Expected Number of Trials up to the First Hitting H ?

$$\text{coin}\begin{cases} H : \frac{1}{2} \\ T : \frac{1}{2} \end{cases}$$

$$\begin{array}{lll} 1 & H & \frac{1}{2} \\ 2 & TH & \frac{1}{2}\frac{1}{2} \\ 3 & TTH & \frac{1}{2}\frac{1}{2}\frac{1}{2} \\ & \vdots & \end{array}$$

$$\sum_{n=1}^{\infty} n\left(\frac{1}{2}\right)^n$$

```
val = 0

for n in range(1,20):
    val += n*(1/2)**n

print(val)
```

```
1.999959945678711
```

Expectation

- Q: The Expected Number of Trials up to the First Hitting H ?

```
n_trials = 1000

NUM = []

for i in range(n_trials):
    num = 1
    while np.random.randint(2) != 0:
        num += 1
    NUM.append(num)

print(np.mean(NUM))
```

```
1.956
```

Expectation

- Remark: how to compute

$$\sum_{n=1}^{\infty} n\left(\tfrac{1}{2}\right)^n = 1\tfrac{1}{2} + 2\left(\tfrac{1}{2}\right)^2 + 3\left(\tfrac{1}{2}\right)^3 + \cdots$$

$$\frac{d}{dx}\sum_{n=1}^{\infty}(1-x)^{n+1} = \frac{d}{dx}\frac{(1-x)^2}{1-(1-x)} = \frac{d}{dx}\frac{(1-x)^2}{x}$$

$$\sum_{n=1}^{\infty}(n+1)(1-x)^n = \sum_{n=1}^{\infty} n(1-x)^n + \sum_{n=1}^{\infty}(1-x)^n$$

$$= \frac{(1-x)^2 + 2(1-x)x}{x^2}$$

$$x = \frac{1}{2} \implies \sum_{n=1}^{\infty} n\left(\frac{1}{2}\right)^n + \frac{1-\frac{1}{2}}{\frac{1}{2}} = \frac{\frac{1}{4} + 2\cdot\frac{1}{2}\cdot\frac{1}{2}}{\frac{1}{4}}$$

$$\implies \sum_{n=1}^{\infty} n\left(\frac{1}{2}\right)^n = 2$$

Expectation

- Or

$$y = 1\frac{1}{2} + 2\left(\frac{1}{2}\right)^2 + 3\left(\frac{1}{2}\right)^3 + \cdots$$

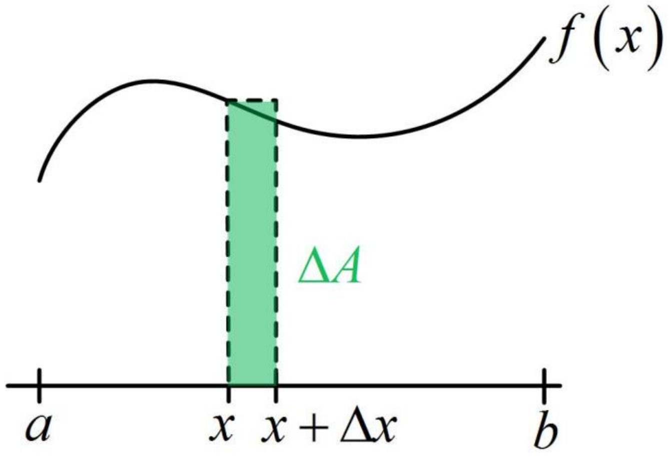
$$2y = 1 + 2\left(\frac{1}{2}\right) + 3\left(\frac{1}{2}\right)^2 + \cdots$$

$$\implies y = 1 + \left(\frac{1}{2}\right) + \left(\frac{1}{2}\right)^2 + \cdots \qquad = \frac{1}{1-\frac{1}{2}} = 2$$

Integration

$$\int_0^1 x^2 dx = \frac{1}{2}x^3\Big|_0^1 = \frac{1}{3}$$

- Question : how to solve integration with computers ?



$$\Delta A = f(x)\Delta x$$

$$A \approx \sum \Delta A = \sum f(x_k)\Delta x$$

Integration

```python
# by summing up

dx = 0.001
x = np.arange(0,1,dx)

A = 0
for xk in x:
    A += (xk**2)*dx

print(A)
```

$$\Delta A = f(x)\Delta x$$

$$A \approx \sum \Delta A = \sum f(x_k)\Delta x$$

```
0.3328335
```

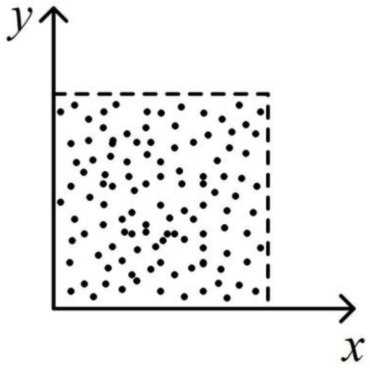
```python
# shortened version

A = np.sum(x**2)*dx
print(A)
```

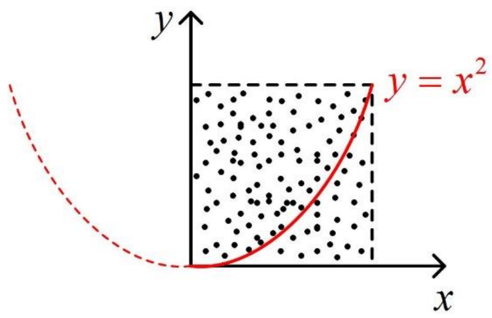
```
0.3328335
```

Integration

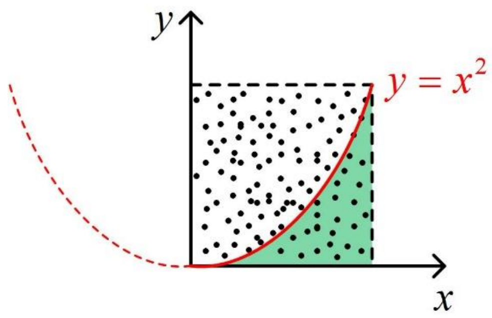
- Question: another method ? (use randomness)

$$x = \mathrm{rand}(n, 1)$$
$$y = \mathrm{rand}(n, 1) \implies \mathrm{plot}(x, y)$$





$$\frac{\#\ \text{under}\ y = x^2}{\#\ \text{total}}$$



$$\frac{\text{area under}\ y = x^2}{\text{total area}}$$

Monte Carlo Simulation

- It is known as **Monte Carlo simulation**

$\Longrightarrow$ extremely powerful

$\Longrightarrow$ can apply to many, many, many (engineering) problems

Integration: Monte Carlo Simulation

```
# the number of points below curve out of the total number is a fraction of area

n = 10000

# generate n random numbers x and y

x = np.random.rand(n,1)
y = np.random.rand(n,1)

count = 0
for i in range(n):
    # compute y to f(x)
    if y[i,0] < x[i,0]**2:
        count += 1

# result normalized by total #
print(count/n)
```

```
0.3349
```

$$\frac{\#\text{ under } y = x^2}{\#\text{ total}}$$

```
# shortened version

A = np.sum(y < x**2)/n
print(A)
```

```
0.3349
```

Compute π statistically

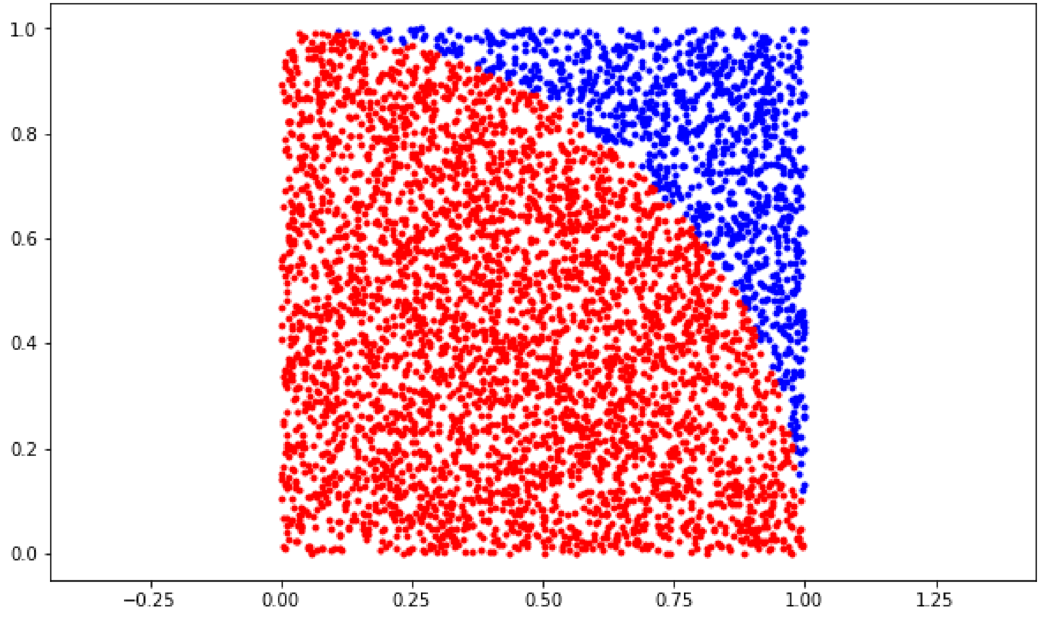
```
n = 5000

y = np.random.rand(n,1)
x = np.random.rand(n,1)

idx = np.empty((n,1))
count = 0

for i in range(n):
    if np.sqrt(x[i]**2 + y[i]**2) < 1:
        count += 1
        idx[i] = 1
    else:
        idx[i] = 0

print((count/n)*4)
```

```
3.1288
```

$$\boxed{\frac{\#\text{ under } y = x^2}{\#\text{ total}}}$$

Compute π statistically

```
plt.figure(figsize=(10,6))
plt.plot(x[idx == 0], y[idx == 0], 'b.')
plt.plot(x[idx == 1], y[idx == 1], 'r.')
plt.axis('equal')
plt.show()
```



Compute π statistically

```python
# shortened version

pi = np.sum(np.abs(x + 1j*y) < 1)/n * 4

print(pi)
```

```
3.1288
```