

Lecture A6. Simulation 3

Sim, Min Kyu, Ph.D., mksim@seoultech.ac.kr



서울과학기술대학교 데이터사이언스학과

- 1 I. Recap on estimation of π
- 2 II. Running estimate approach

I. Recap on estimation of π

Implementation - basic

- Following is the most basic example of Monte Carlo simulation.

```
set.seed(1234) # fix the random seed
MC_N <- 10^3
x <- runif(MC_N)*2-1 # runif() generates U(0,1)
y <- runif(MC_N)*2-1
t <- sqrt(x^2+y^2)
pi_hat <- 4*sum(t<=1)/MC_N
pi_hat
```

```
## [1] 3.188
```

- One caveat with this vectorized programming style is that we may not observe the intermediate results.
- This may become an issue when each MC repetition takes a long time. Often, observing intermediate result is beneficial.

Implementation - “The first-timer would write”

- It is mentioned that the first timer would write as below.

```
set.seed(1234)
MC_N <- 10^6
count <- 0
for (MC_i in 1:MC_N) {
  x_i <- runif(1)*2-1
  y_i <- runif(1)*2-1
  t_i <- sqrt(x_i^2+y_i^2)
  if (t_i <= 1) count <- count + 1
}
pi_hat <- 4*count/MC_N
```

- This note will rewrite the above code in the way that the estimate for π is iteratively updated as MC_i increases toward MC_N.

II. Running estimate approach

Development

- Remind that from A4, p8,

$$\hat{\pi} = 4 \times \frac{\text{number of } \{t_i \leq 1\}}{N} = 4 \times \frac{\sum_{i=1}^N I_{\{t_i \leq 1\}}}{N}$$

- This is complete estimate after N number of Monte Carlo repetitions are completed.
- Assume that we have currently completed $n(\leq N)$ number of Monte Carlo repetition, then using the completed n samples, we can still generate an estimate as follows.

$$\hat{\pi}_n = \frac{\sum_{i=1}^n 4 \cdot I_{\{t_i \leq 1\}}}{n}$$

- Note that the notation $\hat{\pi}_n$ speaks itself as “estimate of π using n samples”.
- In other words, the estimate of π is updated as $\hat{\pi}_1, \hat{\pi}_2, \hat{\pi}_3, \dots, \hat{\pi}_n, \dots, \hat{\pi}_N$.

- From the expression for $\hat{\pi}_n$, the numerator implies the running sum up to n repetition, while the denominator counts the number of repetition so far.
- Let's notate the individual result of i -th experiment as A_i , and the running sum by n repetition as S_n , as follows.

$$\hat{\pi}_n = \frac{\sum_{i=1}^n 4 \cdot I_{\{t_i \leq 1\}}}{n} =: \frac{\sum_{i=1}^n A_i}{n} =: \frac{S_n}{n}$$

- Now, we can rewrite the above as a recursive form.

$$\begin{aligned} \hat{\pi}_n &= \frac{A_1 + A_2 + \cdots + A_{n-1} + A_n}{n} = \frac{S_{n-1} + A_n}{n} \\ &= \frac{\frac{n-1}{n-1} \cdot S_{n-1} + A_n}{n} = \frac{(n-1)\hat{\pi}_{n-1} + A_n}{n} \\ &= \left(\frac{n-1}{n}\right) \hat{\pi}_{n-1} + \left(\frac{1}{n}\right) A_n \end{aligned}$$

- The last expression tells us that, the Monte Carlo updating occurs, from $\hat{\pi}_{n-1}$ to $\hat{\pi}_n$, in a way that the old estimate ($\hat{\pi}_{n-1}$) is updated with the feed of new information A_n as a weighted average of the two.

Implementation

- Another possible benefit of this approach is that we can halt this experiment when $\hat{\pi}_{n-1}$ and $\hat{\pi}_n$ are close enough. (within similar vein with *early stopping* in deep learning domain)
- How would you impose an early stopping condition in this experiment?

$$\hat{\pi}_n = \left(\frac{n-1}{n} \right) \hat{\pi}_{n-1} + \left(\frac{1}{n} \right) A_n$$

```
set.seed(1234)
beg_time <- Sys.time() # to time
old_est <- 0
n <- 1
MC_N <- 10^6
repeat{
  x_i <- runif(1)*2-1
  y_i <- runif(1)*2-1
  t_i <- sqrt(x_i^2+y_i^2)
  A_n <- 4*(t_i <= 1)
  new_est <- ((n-1)/n)*old_est + (1/n)*A_n
  if (n > MC_N) break
  n <- n+1
  old_est <- new_est
}
print(new_est)

## [1] 3.138345

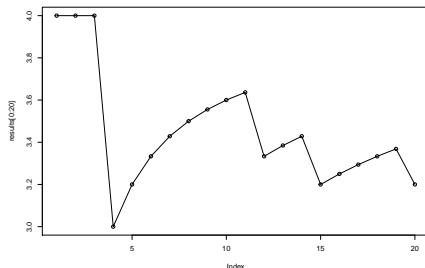
end_time <- Sys.time() # to time
print(end_time-beg_time) # to time

## Time difference of 2.953222 secs
```

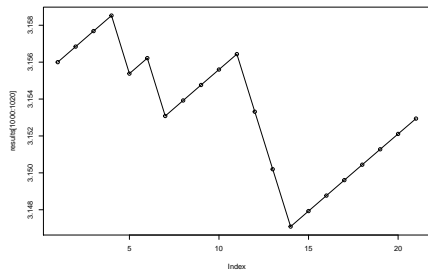
Convergence trajectory

```
set.seed(1234)
beg_time <- Sys.time() # to time
old_est <- 0
n <- 1
MC_N <- 10^6
results <- rep(0, MC_N) # to save
repeat{
  x_i <- runif(1)*2-1
  y_i <- runif(1)*2-1
  t_i <- sqrt(x_i^2+y_i^2)
  A_n <- 4*(t_i <= 1)
  new_est <- ((n-1)/n)*old_est + (1/n)*A_n
  results[n] <- new_est # to save
  if (n > MC_N) break
  n <- n+1
  old_est <- new_est
}
```

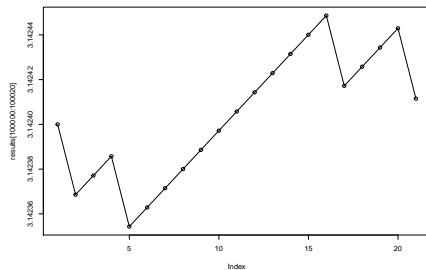
```
plot(results[0:20], type='o')
```



```
plot(results[1000:1020], type='o')
```



```
plot(results[10000:10020], type='o')
```



Discussion

$$\hat{\pi}_n = \left(\frac{n-1}{n}\right) \hat{\pi}_{n-1} + \left(\frac{1}{n}\right) A_n \quad (1)$$

- The above discussion can be generalized into:

$$\hat{\pi}_n = (1 - \alpha) \hat{\pi}_{n-1} + \alpha A_n \quad (2)$$

- The previous implementation had $\alpha = \frac{1}{n}$.
 - The quantity $\alpha \in (0, 1)$ implies the importance of the last observation.
 - The quantity $1 - \alpha$ implies the importance of the previous estimate.
- Oftentimes, setting $\alpha > \frac{1}{n}$ makes sense. When?

Discussion

- It can be also written as

$$\begin{aligned}\hat{\pi}_n &= \left(\frac{n-1}{n}\right) \hat{\pi}_{n-1} + \left(\frac{1}{n}\right) A_n \\ &= \left(\frac{n-1}{n}\right) \hat{\pi}_{n-1} + \left(\frac{1}{n}\right) \hat{\pi}_{n-1} + \left(\frac{1}{n}\right) A_n - \left(\frac{1}{n}\right) \hat{\pi}_{n-1} \\ &= \hat{\pi}_{n-1} + (1/n)(A_n - \hat{\pi}_{n-1})\end{aligned}$$

- Likewise,

$$\begin{aligned}\hat{\pi}_n &= (1 - \alpha) \hat{\pi}_{n-1} + \alpha A_n \\ &= \hat{\pi}_{n-1} + \alpha(A_n - \hat{\pi}_{n-1})\end{aligned}$$

$$\hat{\pi}_n = \hat{\pi}_{n-1} + \alpha(A_n - \hat{\pi}_{n-1}) \quad (3)$$

• How would you interpret each term?

- $\hat{\pi}_n$
- $\hat{\pi}_{n-1}$
- α
- $(A_n - \hat{\pi}_{n-1})$
- A_n

Exercise 1

Write a python code that produces results in page 10 and 11 using Eq (3). Use the variable names of `new_est`, `old_est`, `alpha`, `MC_tgt`, `MC_err` to demonstrate your understanding.

"If I only had an hour to chop down a tree, I would spend the first 45 minutes sharpening my axe.
- A. Lincoln"