# Lecture A4. Simulation 1

Bong Seok Kim

2021-01-02

## 차 례



### Simulation Approach,Implementation-basic p.11

```
import numpy as np
import time
import pandas as pd

beg_time=time.time()

np.random.seed(seed=1234) #fix the random seed

MC_N=10**3

x= np.random.uniform(low=-1, high=1, size=MC_N) # genrates U(-1,1)

y= np.random.uniform(low=-1, high=1, size=MC_N)

t=np.sqrt(x**2+y**2)

df=pd.DataFrame({'x':x,'y':y,'t':t})

df.head(6) # always display and check !
```

```
##          x         y         t
## 0 -0.616961 -0.197787  0.647889
## 1  0.244218  0.861229  0.895186
## 2 -0.124545  0.030672  0.128266
## 3  0.570717  0.619164  0.842070
## 4  0.559952  0.763544  0.946861
## 5 -0.454815  0.525336  0.694863
```

## Simulation Approach, Vectorized Programming p.12

### From the previous slide

```
import numpy as np
import time

beg_time=time.time()

np.random.seed(seed=1234)

MC_N=10**6

x= np.random.uniform(low=-1, high=1, size=MC_N)

y= np.random.uniform(low=-1, high=1, size=MC_N)

t=np.sqrt(x**2+y**2)

pi_hat=4*np.sum(t<=1)/MC_N

end_time=time.time()

print("Time diffrence of",end_time-beg_time,"secs")
```

```
## Time diffrence of 0.052806854248046875 secs
```

### What first-timer would write

```
import random
import math
import time

beg_time=time.time()

random.seed(1234)

MC_N=10**6

count=0

for i in range(MC_N):
```

```
    x_i=random.uniform(-1,1)
    y_i=random.uniform(-1,1)
    t_i=math.sqrt(x_i**2+y_i**2)

    if(t_i<=1):
        count+=1

pi_hat= 4*count/MC_N

end_time=time.time()

print("Time diffrence of",end_time-beg_time,"secs")
```

```
## Time diffrence of 1.0900709629058838 secs
```

## Simulation approach, Implementation - varying number of trials p.13

### Approach with a custom function

```
def pi_simulator(MC_N):
    np.random.seed(seed=1234)

    x= np.random.uniform(low=-1, high=1, size=MC_N)

    y= np.random.uniform(low=-1, high=1, size=MC_N)

    t=np.sqrt(x**2+y**2)

    pi_hat=4*np.sum(t<=1)/MC_N

    return(pi_hat)

print(pi_simulator(100))
```

```
## 2.96
```

```
print(pi_simulator(1000))
```

```
## 3.06
```

```
print(pi_simulator(10000))
```

```
## 3.1352
```

```
print(pi_simulator(100000))
```

```
## 3.13976
```

### How many repetition is necessary to get closer?

```
import pandas as pd

num_trials=list(map(lambda x: 10**x, range(2,8)))
outcomes=list(map(pi_simulator,num_trials))
results=pd.DataFrame({'num_trials': num_trials,'outcomes':outcomes})

results
```

```
##    num_trials  outcomes
## 0         100  2.960000
## 1        1000  3.060000
## 2       10000  3.135200
## 3      100000  3.139760
## 4     1000000  3.142876
## 5    10000000  3.142289
```

## Discussion, Computation time p.17

```
import numpy as np
import time

def pi_simulator2(MC_N): # name change

    beg_time=time.time() # newly added

    np.random.seed(seed=1234)

    x= np.random.uniform(low=-1, high=1, size=MC_N)

    y= np.random.uniform(low=-1, high=1, size=MC_N)

    t=np.sqrt(x**2+y**2)

    pi_hat=4*np.sum(t<=1)/MC_N

    end_time=time.time() # newly added

    print(MC_N)

    print("Time diffrence of",end_time-beg_time,"secs") # newly added

    return(pi_hat)


print(*list(map(pi_simulator2,num_trials)))
```

```
## 100
## Time diffrence of 0.0 secs
## 1000
## Time diffrence of 0.0 secs
## 10000
## Time diffrence of 0.0 secs
## 100000
## Time diffrence of 0.0029916763305664062 secs
## 1000000
## Time diffrence of 0.022935152053833008 secs
## 10000000
```

```
## Time diffrence of 0.248335599899292 secs
## 2.96 3.06 3.1352 3.13976 3.142876 3.1422888
```

### Confdence interval, Repetitive simulation experiments p.22

```
import numpy as np

def pi_simulator3(MC_N): # name change

    # np.random.seed(seed=1234) # seed must not be fixed

    x= np.random.uniform(low=-1, high=1, size=MC_N)

    y= np.random.uniform(low=-1, high=1, size=MC_N)

    t=np.sqrt(x**2+y**2)

    pi_hat=4*np.sum(t<=1)/MC_N

    return pi_hat
```

```
import numpy as np

n=100 # number of experiments to repeat

MC_N=1000 # number of simulation repetition in a single experiment

np.random.seed(seed=1234)

samples=np.zeros(n)

for i in range(n):
    samples[i]=pi_simulator3(MC_N)

print(*samples[0:6])
```

```
## 3.06 3.184 3.12 3.228 3.124 3.092
```

```
from scipy import stats
import numpy as np

X_bar=np.mean(samples)

s=np.sqrt(np.sum((X_bar-samples)**2)/(n-1))
```

```
t=stats.t(df=n-1).ppf((0.975))

print(X_bar)
```

```
## 3.1412000000000004
```

```
print(s)
```

```
## 0.05271305973538882
```

```
print(t)
```

```
## 1.9842169515086827
```

## Confdence interval,Excercise 1 p.24

Do the above experiment with MC_N increased by the fator of ten, and present the confdence interval. (Use set.seed(1234))

```
from scipy import stats
import numpy as np

n=100 # number of experiments to repeat

MC_N=10000 # number of simulation repetition in a single experiment

np.random.seed(seed=1234)

samples=np.zeros(n)

for i in range(n):
    samples[i]=pi_simulator3(MC_N)

X_bar=np.mean(samples)

s=np.sqrt(np.sum((X_bar-samples)**2)/(n-1))

t=stats.t(df=n-1).ppf((0.975))

lb=X_bar-t*s/np.sqrt(n)

ub=X_bar+t*s/np.sqrt(n)
```

```
print(lb)
```

```
## 3.13840259759299
```

```
print(ub)
```

```
## 3.1443254024070098
```

```
print(ub-lb)
```

```
## 0.005922804814019855
```

### Confdence interval,Excercie 2 p.25

Do the Exercise 1 above with ◎ increased by the factor of ten, and present the confdence interval. (Use set.seed(1234))

```
from scipy import stats
import numpy as np

n=1000 # number of experiments to repeat

MC_N=10000 # number of simulation repetition in a single experiment

np.random.seed(seed=1234)

samples=np.zeros(n)

for i in range(n):
    samples[i]=pi_simulator3(MC_N)

X_bar=np.mean(samples)

s=np.sqrt(np.sum((X_bar-samples)**2)/(n-1))

t=stats.t(df=n-1).ppf((0.975))

lb=X_bar-t*s/np.sqrt(n)

ub=X_bar+t*s/np.sqrt(n)
```

```
print(lb)
```

```
## 3.141465340511527
```

```
print(ub)
```

```
## 3.1434762594884726
```

```
print(ub-lb)
```

```
## 0.002010918976945497
```

```r
"Done, Lecture A4. Simulation 1 "
```

```
## [1] "Done, Lecture A4. Simulation 1 "
```