Lecture A4.Simulation 1

Baek, Jong min

2021-01-05

Implementation - basic(Page 11)

```
np.random.seed(1234)
N = 10**3
x = np.random.uniform(low=0,high=1,size=N)*2-1
y = np.random.uniform(low=0,high=1,size=N)*2-1
t = np.sqrt(x**2+y**2)
result = np.c_[x,y,t]
pi_hat = 4*np.sum(t<=1)/N
print(f'x : {x[:5]}')
```

```
## x : [-0.6169611   0.24421754 -0.12454452  0.57071717  0.55995162]
```

```
print(f'y : {y[:5]}')
```

```
## y : [-0.19778718  0.8612288   0.03067229  0.61916404  0.76354446]
```

```
print(f't : {t[:5]}')
```

```
## t : [0.64788947 0.8951856  0.12826585 0.84207018 0.9468611 ]
```

```
print(result[:5])
```

```
## [[-0.6169611  -0.19778718  0.64788947]
##  [ 0.24421754  0.8612288   0.8951856 ]
##  [-0.12454452  0.03067229  0.12826585]
##  [ 0.57071717  0.61916404  0.84207018]
##  [ 0.55995162  0.76354446  0.9468611 ]]
```

```
print('pi_hat : \n {}'.format(pi_hat))
```

```
## pi_hat :
##  3.06
```

Vectorized programming(Page 12)

```
start_time = time.time()
np.random.seed(1234)
N = 10**6
x = np.random.uniform(low=0,high=1,size=N)*2-1
y = np.random.uniform(low=0,high=1,size=N)*2-1
t = np.sqrt(x**2+y**2)
pi_hat = 4*np.sum(t<=1)/N
end_time = time.time()
print(end_time - start_time)
```

```
## 0.14860081672668457
```

Implementation - varying number of trials

1) Approach with a custom function

```
def pi_simulator(N):
  np.random.seed(1234)
  x = np.random.uniform(low=0,high=1,size=N)*2-1
  y = np.random.uniform(low=0,high=1,size=N)*2-1
  t = np.sqrt(x**2+y**2)
  pi_hat = 4*np.sum(t<=1)/N
  return pi_hat
```

```
pi_simulator(100)
```

```
## 2.96
```

```
pi_simulator(1000)
```

```
## 3.06
```

```
pi_simulator(10000)
```

```
## 3.1352
```

```
pi_simulator(100000)
```

```
## 3.13976
```

2) How many repetition is necessary to get closer?

```
num_trials =[10 ** N for N in range(2,7)]
print('num_trail : {}'.format(num_trials))
```

```
## num_trail : [100, 1000, 10000, 100000, 1000000]
```

```
outcomes = np.array(list(map(pi_simulator,num_trials)))
print('outcomes : {}'.format(outcomes))
```

```
## outcomes : [2.96     3.06     3.1352   3.13976  3.142876]
```

```
result = np.c_[np.array(num_trials).T,outcomes.T]
print('result : \n {}'.format(result))
```

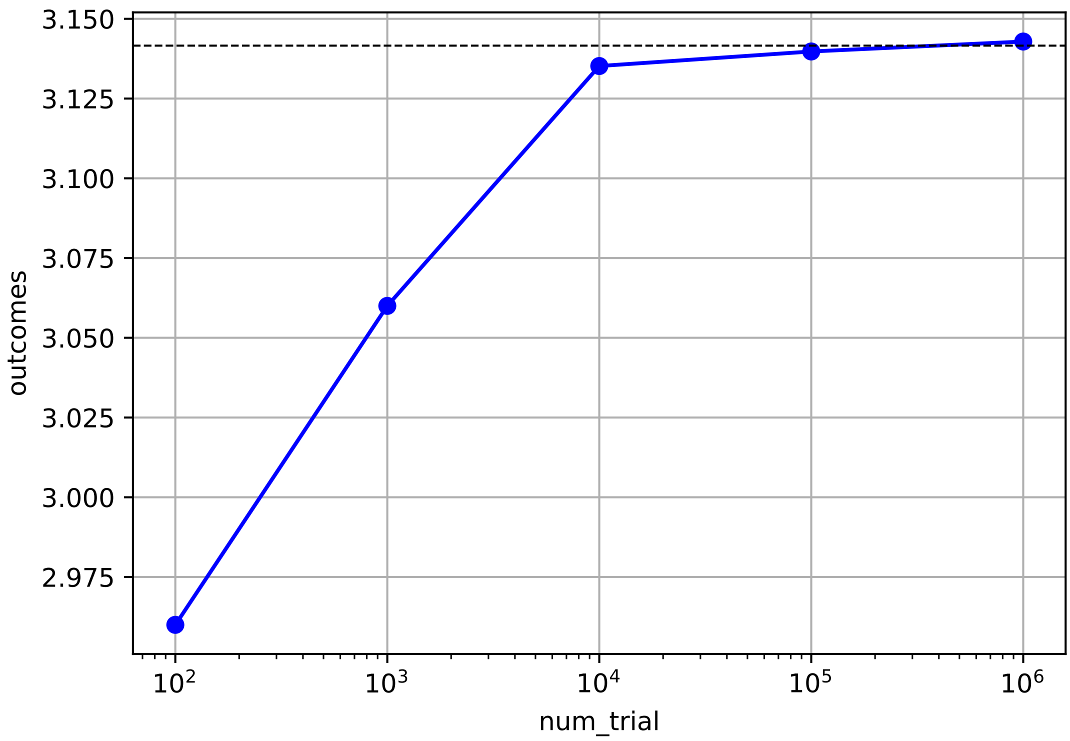
```
## result :
##  [[1.000000e+02 2.960000e+00]
##  [1.000000e+03 3.060000e+00]
##  [1.000000e+04 3.135200e+00]
##  [1.000000e+05 3.139760e+00]
##  [1.000000e+06 3.142876e+00]]
```

```
results = pd.DataFrame(result)
print(results)
```

```
##            0         1
## 0      100.0  2.960000
## 1     1000.0  3.060000
## 2    10000.0  3.135200
## 3   100000.0  3.139760
## 4  1000000.0  3.142876
```

3) How many repetition is necessary to get closer?(Visualization)

```
plt.plot(results[0],results[1],'bo-')
plt.axhline(y=3.14159, color='black', linestyle='--',linewidth=0.8)
plt.xscale('log')
plt.xlabel('num_trial')
plt.ylabel('outcomes')
plt.grid(True)
plt.show()
```



#$ Computation Time(Page 17)

```
def pi_simulator2(N):
  start_time = time.time()

  np.random.seed(1234)
  x = np.random.uniform(low=0,high=1,size=N)*2-1
  y = np.random.uniform(low=0,high=1,size=N)*2-1
  t = np.sqrt(x**2+y**2)
  pi_hat = 4*np.sum(t<=1)/N

  end_time = time.time()

  print('N : {}'.format(N))
  print('end_time - start_time : {} '.format(end_time - start_time))

  return pi_hat
```

```
outcomes = np.array(list(map(pi_simulator2,num_trials)))
```

```
## N : 100
## end_time - start_time : 0.0
## N : 1000
## end_time - start_time : 0.0
## N : 10000
## end_time - start_time : 0.0
## N : 100000
## end_time - start_time : 0.004987478256225586
## N : 1000000
## end_time - start_time : 0.0728306770324707
```

```
print('outcomes : {}'.format(outcomes))
```

```
## outcomes : [2.96     3.06     3.1352   3.13976  3.142876]
```

Repetitive simulation experiments(Page 22)

```
def pi_simulator3(N):

  x = np.random.uniform(low=0,high=1,size=N)*2-1
  y = np.random.uniform(low=0,high=1,size=N)*2-1
  t = np.sqrt(x**2+y**2)
  pi_hat = 4*np.sum(t<=1)/N

  return pi_hat
```

```
n = 100
N = 1000
np.random.seed(1234)
samples = np.zeros(n)
for i in range(0,n):
  samples[i] = pi_simulator3(N)

samples[:5]
```

```
## array([3.06 , 3.184, 3.12 , 3.228, 3.124])
```

From LN.A4.p13(Page 23)

```
from scipy.stats import t

X_bar = np.mean(samples)
s = np.sqrt(np.sum(X_bar-samples)**2)/(n-1)
tt = t.ppf(q=0.975, df = n-1)
print(X_bar)
```

```
## 3.1412000000000004
```

```
print(s)
```

```
## 3.2297397080004555e-16
```

```
print(tt)
```

```
## 1.9842169515086827
```

Exercise 1

Don the above experiment with MN-N increased by the factor of ten, and present the confidence interval(Use set.seed(1234))

```
from scipy.stats import t

n = 100
N = 1000
np.random.seed(1234)
samples = np.zeros(n)
for i in range(0,n):
  samples[i] = pi_simulator3(N)

X_bar = np.mean(samples)
s = np.sqrt(np.sum((X_bar-samples)**2)/(n-1))
tt  = t.ppf(q=0.975,df=n-1)

lb = X_bar - tt*s/np.sqrt(n)
ub = X_bar + tt*s/np.sqrt(n)

print(lb)
```

```
## 3.1307405853307158
```

```
print(ub)
```

```
## 3.151659414669285
```

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

```
## 0.020918829338569367
```

Exercise 2

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

```
from scipy.stats import t


n =100
N = 10000
np.random.seed(1234)
samples = np.zeros(n)
for i in range(0,n):
  samples[i] = pi_simulator3(N)
X_bar = np.mean(samples)
s = np.sqrt(np.sum(X_bar-samples)**2/(n-1))
tt = t.ppf(q=0.975,df = n-1)
lb = X_bar-tt*s/np.sqrt(n)
ub = X_bar+tt*s/np.sqrt(n)

print(lb)
```

```
## 3.1413639999999994
```

```
print(ub)
```

```
## 3.1413640000000003
```

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

```
## 8.881784197001252e-16
```

A4.Rmd

```r
"No pain No gain"
```

```
## [1] "No pain No gain"
```