A4 Python Code

Kang, Eui Hyeon

2021-01-03

차 례

Implementation - basic (11p)

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

# 11p
np.random.seed(1234)
N=10**3
x=np.random.uniform(0,1,size=N)*2-1
y=np.random.uniform(0,1,size=N)*2-1
t=np.sqrt(x**2+y**2)
bind=pd.DataFrame({'x':x,'y':y,'t':t})
bind.head(6)
```

```
##           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
```

```
pi_hat=4*sum(t<=1)/N
pi_hat
```

```
## 3.06
```

Vectorized Programming (12p-1)

```python
import time
import numpy as np

# 12p-1
beg_time=time.time()
np.random.seed(1234)

N=10**6
x=np.random.uniform(0,1,size=N)*2-1
y=np.random.uniform(0,1,size=N)*2-1
t=np.sqrt(x**2+y**2)
pi_hat=4*sum(t<=1)/N
end_time=time.time()
print('Time difference of ',end_time-beg_time,' secs')
```

```
## Time difference of  2.7521777153015137  secs
```

Vectorized Programming (12p-2)

```
import numpy as np
import time

# 12p-2
beg_time=time.time()
np.random.seed(1234)

N=10**6
count=0

for i in range(N):
    x_i=np.random.uniform(0,1,size=1)*2-1
    y_i=np.random.uniform(0,1,size=1)*2-1
    t_i=np.sqrt(x_i**2+y_i**2)
    if t_i <=1:
        count+=1

pi_hat=4*count/N
end_time=time.time()
print('Time difference of ',end_time-beg_time,' secs')
```

```
## Time difference of  18.96697235107422  secs
```

Implementation - varying number of trials (13p-1)

```
import numpy as np


# 13p-1
def pi_simulator(N):
    np.random.seed(1234)

    x=np.random.uniform(0,1,size=N)*2-1
    y=np.random.uniform(0,1,size=N)*2-1
    t=np.sqrt(x**2+y**2)

    pi_hat=4*sum(t<=1)/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))
```

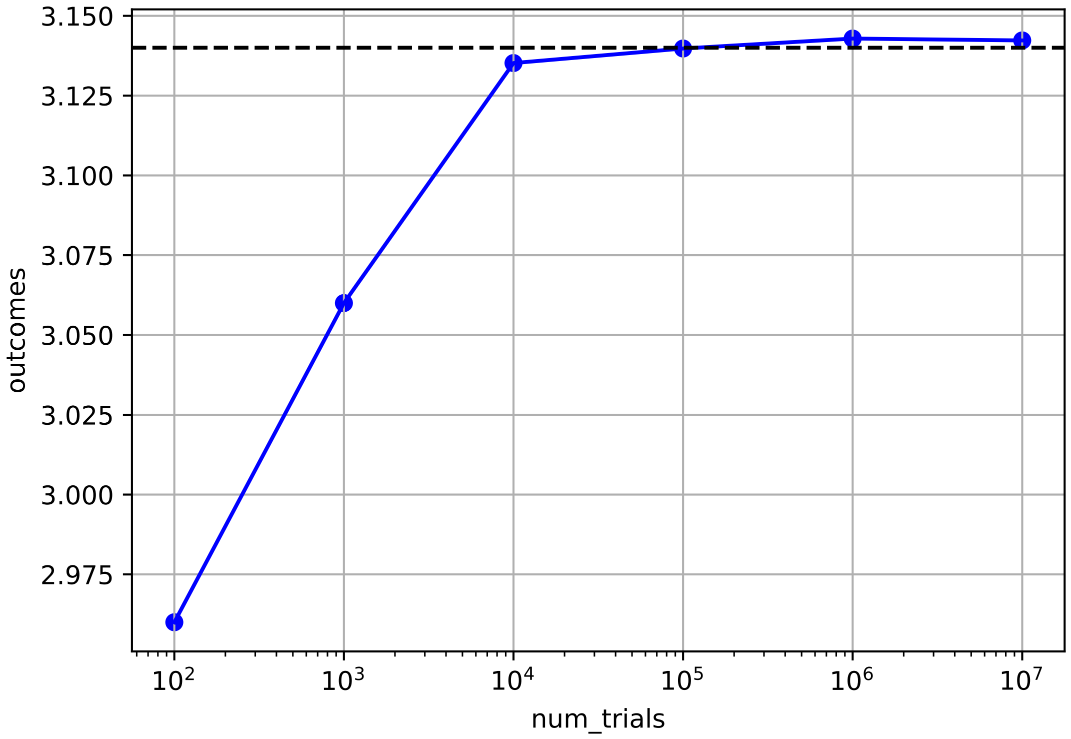
```
## 3.13976
```

Implementation - varying number of trials (13p-2)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 13p-2
def pi_simulator(N):
    np.random.seed(1234)

    x=np.random.uniform(0,1,size=N)*2-1
    y=np.random.uniform(0,1,size=N)*2-1
    t=np.sqrt(x**2+y**2)

    pi_hat=4*sum(t<=1)/N
    return pi_hat

num_trials=10**np.arange(2,8)
outcomes=[pi_simulator(i) for i in num_trials]

outcomes=np.asarray(outcomes)
results=pd.DataFrame({'num_trials':num_trials,'outcomes':outcomes})
results

# 15p
```

```
##    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
```

```
plt.scatter(results['num_trials'],results['outcomes'], c='blue')
plt.plot(results['num_trials'],results['outcomes'], c='blue')
plt.axhline(3.14,0,1,color='black',linestyle='--')
plt.xscale('log')
plt.grid(True,axis='both')
plt.xlabel('num_trials')
plt.ylabel('outcomes')
plt.show()
```


3.150
3.125
3.100
3.075
3.050
3.025
3.000
2.975
outcomes
10^2
10^3
10^4
10^5
10^6
10^7
num_trials

Computation time (17p)

```
import time
import numpy as np

# 17p
def pi_simulator2(N):
    beg_time=time.time()
    np.random.seed(1234)

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

    print(N)
    print('Time difference of ',end_time-beg_time, 'secs')
    return pi_hat

num_trials=10**np.arange(2,8)
[pi_simulator2(i) for i in num_trials]
```

```
## 100
## Time difference of  0.0 secs
## 1000
## Time difference of  0.0030279159545898438 secs
## 10000
## Time difference of  0.028923988342285156 secs
## 100000
## Time difference of  0.293947696685791 secs
## 1000000
## Time difference of  2.7065541744232178 secs
## 10000000
## Time difference of  26.426914930343628 secs
## [2.96, 3.06, 3.1352, 3.13976, 3.142876, 3.1422888]
```

Repetitive simulation experiments (22-23p)

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

# 22p
def pi_simulator3(N):
    x=np.random.uniform(0,1,size=N)*2-1
    y=np.random.uniform(0,1,size=N)*2-1
    t=np.sqrt(x**2+y**2)
    pi_hat=4*sum(t<=1)/N

    return pi_hat

n=100
N=1000
np.random.seed(1234)

samples=np.zeros((n,))
for i in range(n):
    samples[i]=pi_simulator3(N)

samples[:6]
```

```
## array([3.06 , 3.184, 3.12 , 3.228, 3.124, 3.092])
```

```
x_bar=np.mean(samples)
s=np.sqrt(sum((x_bar-samples)**2)/(n-1))
t_=t(n-1).ppf(0.975)

x_bar
```

```
## 3.1412000000000004
```

```
s
```

```
## 0.05271305973538881
```

```
t
```

```
## <scipy.stats._continuous_distns.t_gen object at 0x000000000D45F588>
```

Exercise 1 (24p)

```
import numpy as np
from scipy.stats import t
# 24p

def pi_simulator3(N):
    x=np.random.uniform(0,1,size=N)*2-1
    y=np.random.uniform(0,1,size=N)*2-1
    t=np.sqrt(x**2+y**2)
    pi_hat=4*sum(t<=1)/N

    return pi_hat

n=100
N=10000
np.random.seed(1234)

samples=np.zeros((n,))
for i in range(n):
    samples[i]=pi_simulator3(N)

samples[:6]
```

```
## array([3.1352, 3.1276, 3.1396, 3.1548, 3.1388, 3.1652])
```

```
x_bar=np.mean(samples)
s=np.sqrt(sum((x_bar-samples)**2)/(n-1))
t_=t(n-1).ppf(0.975)
lb=x_bar-t_*s/np.sqrt(n)
ub=x_bar+t_*s/np.sqrt(n)

lb
```

```
## 3.13840259759299
```

```
ub
```

```
## 3.1443254024070098
```

```
ub-lb
```

```
## 0.005922804814019855
```

Exercise 2 (25p)

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

# 25p
def pi_simulator3(N):
    x=np.random.uniform(0,1,size=N)*2-1
    y=np.random.uniform(0,1,size=N)*2-1
    t=np.sqrt(x**2+y**2)
    pi_hat=4*sum(t<=1)/N

    return pi_hat

n=1000
N=10000
np.random.seed(1234)

samples=np.zeros((n,))
for i in range(n):
    samples[i]=pi_simulator3(N)

samples[:6]
```

```
## array([3.1352, 3.1276, 3.1396, 3.1548, 3.1388, 3.1652])
```

```
x_bar=np.mean(samples)
s=np.sqrt(sum((x_bar-samples)**2)/(n-1))
t_=t(n-1).ppf(0.975)
lb=x_bar-t_*s/np.sqrt(n)
ub=x_bar+t_*s/np.sqrt(n)

lb
```

```
## 3.141465340511527
```

```
ub
```

```
## 3.1434762594884726
```

```
ub-lb
```

```
## 0.002010918976945497
```