A4

Tae Hyeon Kwon, undergrad(ITM)

2021-01-03

#Implementation-basic

```
import numpy as np

np.random.seed(1234)
N = 10**3
x = np.random.uniform(-1, 1, size=N)
y = np.random.uniform(-1, 1, size=N)
t = np.sqrt(x**2+y**2)
pi_hat = 4*sum(t<=1)/N

print(pi_hat)
```

```
## 3.06
```

#Time difference

```
import numpy as np
import time

start = time.time()

np.random.seed(1234)
N = 10**6
x = np.random.uniform(-1, 1, size=N)
y = np.random.uniform(-1, 1, size=N)
t = np.sqrt(x**2+y**2)
pi_hat = 4*sum(t<=1)/N

end = time.time()

print(end-start)
```

```
## 3.646251678466797
```

#custom function

```
import numpy as np

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

print('pi_simulator(100):',pi_simulator(100))
```

```
## pi_simulator(100): 2.96
```

```
print('pi_simulator(1000):',pi_simulator(1000))
```

```
## pi_simulator(1000): 3.06
```

```
print('pi_simulator(10000):',pi_simulator(10000))
```

```
## pi_simulator(10000): 3.1352
```

```
print('pi_simulator(100000):',pi_simulator(100000))
```

```
## pi_simulator(100000): 3.13976
```

#num_trials

```
import numpy as np
import pandas as pd

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


num_trials = 10**np.arange(2,8)
outcome = [pi_simulator(i) for i in num_trials]
result = np.vstack((num_trials,outcome)).T
result = pd.DataFrame(result,columns=['num_trials','outcome'])
print(result)
```

```
##    num_trials   outcome
## 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
## 5  10000000.0  3.142289
```

#pi_simulator2

```
import numpy as np
import time


def pi_simulator2(N):
    start = time.time()
    np.random.seed(1234)
    x = np.random.uniform(-1, 1, size=N)
    y = np.random.uniform(-1, 1, size=N)
    t = np.sqrt(x**2+y**2)
    pi_hat = 4*sum(t<=1)/N

    end = time.time()
    print(N)
    print('Time difference of ',end-start,'secs')
    return (pi_hat)

print([pi_simulator2(100)])
```

```
## 100
## Time difference of  0.0 secs
## [2.96]
```

```
print([pi_simulator2(1000)])
```

```
## 1000
## Time difference of  0.002991199493408203 secs
## [3.06]
```

```
print([pi_simulator2(10000)])
```

```
## 10000
## Time difference of  0.03989100456237793 secs
## [3.1352]
```

```
print([pi_simulator2(100000)])
```

```
## 100000
## Time difference of  0.34407734870910645 secs
## [3.13976]
```

```
print([pi_simulator2(1000000)])
```

```
## 1000000
## Time difference of  3.413872718811035 secs
## [3.142876]
```

```
print([pi_simulator2(10000000)])
```

```
## 10000000
## Time difference of  34.47481989860535 secs
## [3.1422888]
```

#pi_simulator3

```
import numpy as np


def pi_simulator3(N):
    #np.random.seed(1234)
    x = np.random.uniform(-1, 1, size=N)
    y = np.random.uniform(-1, 1, size=N)
    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 = [0]*n
for i in range(0,n):
    samples[i] = pi_simulator3(N)

print(samples[:6])
```

```
## [3.06, 3.184, 3.12, 3.228, 3.124, 3.092]
```

#confidence interval

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


def pi_simulator3(N):
    #np.random.seed(1234)
    x = np.random.uniform(-1, 1, size=N)
    y = np.random.uniform(-1, 1, size=N)
    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 = [0]*n
for i in range(0,n):
    samples[i] = pi_simulator3(N)
samples[:6]
```

```
## [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)

print('X_bar:',X_bar)
```

```
## X_bar: 3.1412000000000004
```

```
print('s:',s)
```

```
## s: 0.05271305973538881
```

```
print('T:',T)
```

```
## T: 1.9842169515086827
```

#confidence interval length (Excercise 1)

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


def pi_simulator3(N):
    #np.random.seed(1234)
    x = np.random.uniform(-1, 1, size=N)
    y = np.random.uniform(-1, 1, size=N)
    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 = [0]*n
for i in range(0,n):
    samples[i] = pi_simulator3(N)
samples[:6]
```

```
## [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)

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

print('lb:',lb)
```

```
## lb: 3.1307405853307158
```

```
print('ub:',ub)
```

```
## ub: 3.151659414669285
```

```
print('ub-lb:',ub-lb)
```

```
## ub-lb: 0.020918829338569367
```

#confidence interval length (Excercise 2)

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

```
def pi_simulator3(N):
    #np.random.seed(1234)
    x = np.random.uniform(-1, 1, size=N)
    y = np.random.uniform(-1, 1, size=N)
    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 = [0]*n
for i in range(0,n):
    samples[i] = pi_simulator3(N)
samples[:6]
```

```
## [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)

print('lb:',lb)
```

```
## lb: 3.141465340511527
```

```
print('ub:',ub)
```

```
## ub: 3.1434762594884726
```

```
print('ub-lb:',ub-lb)
```

```
## ub-lb: 0.002010918976945497
```