A4_python_Jeong,wonryeol

Jeong, wonryeol

1/3/2021

Implementation _basic p11

```
import numpy as np
import pandas as pd

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})

print(bind.head(5))
```

```
##           x         y         t
## 0  0.035665  0.434506  0.435967
## 1  0.184331 -0.961762  0.979267
## 2 -0.245737  0.566441  0.617448
## 3 -0.663885 -0.019262  0.664165
## 4  0.098914  0.045291  0.108790
```

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

print(pi_hat)
```

```
## 3.148
```

Vectorized programming p12

From the previous slide

```
import time

beg_time = time.time()
np.random.seed(1234)
N = 10**6
x = np.random.uniform(0,1,N)*2-1
y = np.random.uniform(0,1,N)*2-1
t = np.sqrt(x**2 + y**2)

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

end_time = time.time()

print(end_time - beg_time)
```

```
## 2.6600239276885986
```

What first_timer would write

```
import time

beg_time = time.time()
np.random.seed(1234)
N = 10**6
count = 0

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

pi_hat = 4*count/N

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


end_time = time.time()

print(end_time - beg_time)
```

```
## 17.474950075149536
```

Implementation - varying number of trials p13

Approach with a custom function

```
def pi_simulator(N):
  np.random.seed(1234)

  x = np.random.uniform(0,1,N)*2-1
  y = np.random.uniform(0,1,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
```

How many repetition is necessary to get closer?

```
num_trials=10**np.arange(2,8)

outcomes = np.vectorize(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
```

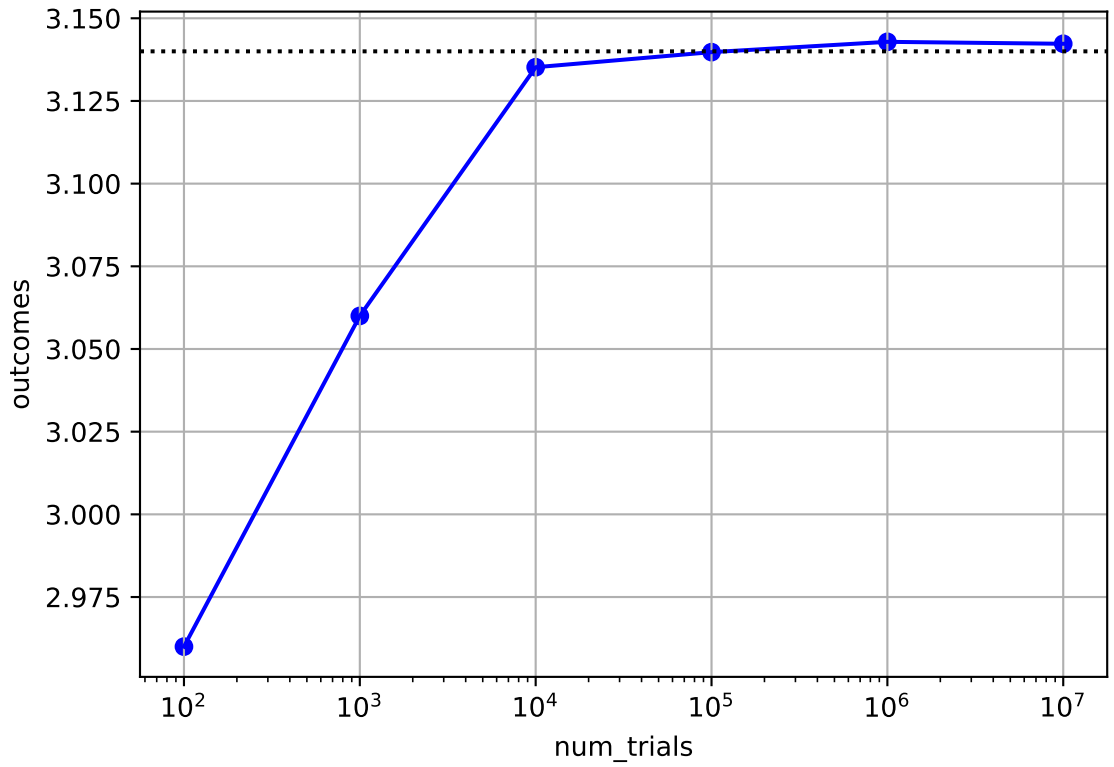
The previous figure was plotted by the following code.

```
import matplotlib.pyplot as plt



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()
```



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

  np.random.seed(1234)

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

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

  end_time = time.time()
  print(N)
  print(end_time - beg_time)

  return pi_hat



outcomes = np.vectorize(pi_simulator2)(num_trials)
```

```
## 100
## 0.00032210350036621094
## 100
## 0.00028514862060546875
## 1000
## 0.002547025680541992
## 10000
## 0.030738115310668945
## 100000
## 0.2889716625213623
## 1000000
## 2.845777988433838
## 10000000
## 23.0815110206604
```

Repetitive simulation experiments

```
def pi_simulator3(N):


  #np.random.seed(1234)

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

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

  end_time = time.time()

  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)

print(samples[:6])
```

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

Exercise1

Do the Exercise above with n increased by the factor of ten, and present the confidence interval. p24

```
from scipy.stats import t


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

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

print(samples[:6])
```

```
## [3.1352 3.1276 3.1396 3.1548 3.1388 3.1652]
```

```
X_bar = np.mean(samples)

s = np.sqrt(np.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
```

```
## 3.1413639999999994
```

```
ub
```

```
## 3.1413640000000003
```

Exercise2

Do the Exercise1 above with n increased by the factor of ten, and present the confidence interval. p24

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

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

print(samples[:6])
```

```
## [3.1352 3.1276 3.1396 3.1548 3.1388 3.1652]
```

```
X_bar = np.mean(samples)

s = np.sqrt(np.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
```

```
## 3.1424708
```

```
ub
```

```
## 3.1424708
```

‘‘‘