A4_Exercises

Kwon do yun

2021-01-04

Implementation - basic (p. 11)

```
import numpy as np
np.random.seed(1234)
N = 10**3
x = np.random.rand(N, 1)*2-1
y = np.random.rand(N, 1)*2-1
t = np.sqrt(x**2+y**2)

vec = np.concatenate((x,y,t), axis=1)
print(vec[: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 ]]
```

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

```
## [3.06]
```

Vectorized programming (p. 12)

```
from datetime import datetime
import numpy as np

beg_time = datetime.now()
N = 10**6
x = np.random.rand(N, 1)*2-1
```

```
y = np.random.rand(N, 1)*2-1
t = np.sqrt(x**2+y**2)
pi_hat = 4*sum(t<=1)/N
end_time = datetime.now()
print("Time difference of " ,end_time - beg_time, "secs")
```

```
## Time difference of  0:00:01.325292 secs
```

```
from datetime import datetime
import numpy as np

beg_time = datetime.now()
N = 10**6
count = 0

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

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

```
## Time difference of  0:00:13.893203 secs
```

Implementation - varying number of trials (p. 13)

```
import numpy as np
def pi_simulator(N):
    np.random.seed(1234)
    x = np.random.rand(N,1)*2-1
    y = np.random.rand(N,1)*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
```

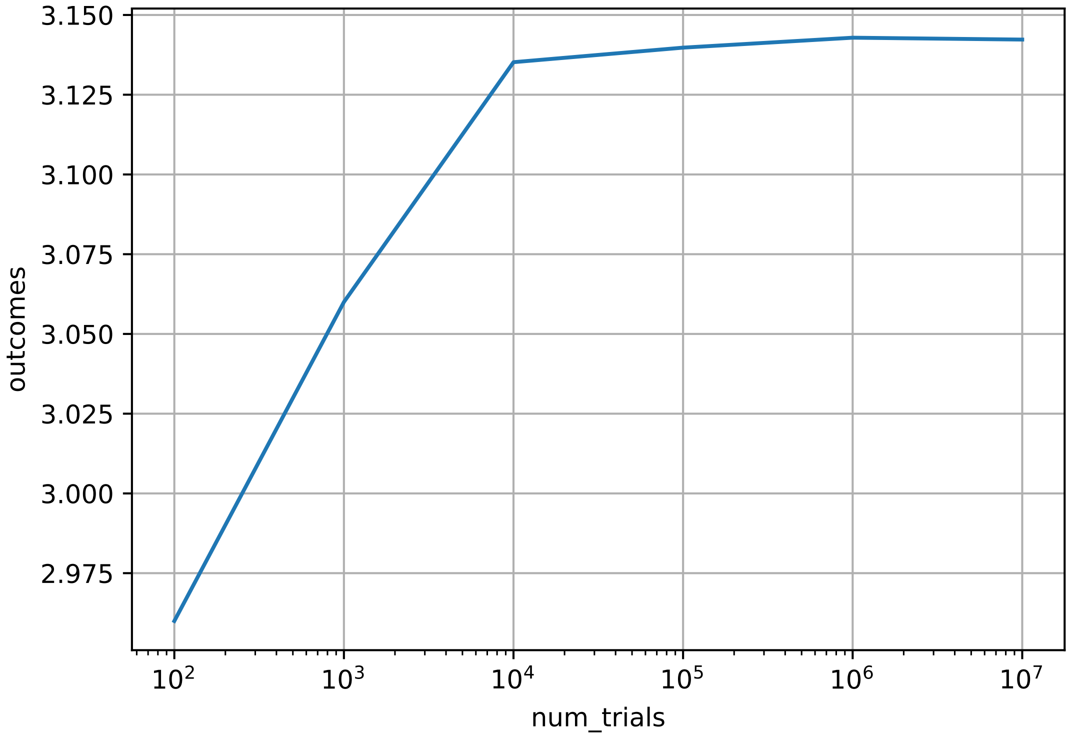
```
import numpy as np
def pi_simulator(N):
    np.random.seed(1234)
    x = np.random.rand(N,1)*2-1
    y = np.random.rand(N,1)*2-1
    t = np.sqrt(x**2+y**2)
    pi_hat=4*np.sum(t <= 1)/N
    return pi_hat

num_trials = [10**i for i in range(2,8)]
outcomes=list(map(pi_simulator, num_trials))
print(outcomes)
```

```
## [2.96, 3.06, 3.1352, 3.13976, 3.142876, 3.1422888]
```

How many repetition is neceessary to get closer? (p. 14)

```
import matplotlib.pyplot as plt
plt.plot(num_trials,outcomes)
plt.grid(True,axis='both')
plt.xscale('log')
plt.xlabel("num_trials")
plt.ylabel("outcomes")
plt.show()
```



Computation time (p. 17)

```
import numpy as np
from datetime import datetime
def pi_simulator_2(N):
    beg_time = datetime.now()
    np.random.seed(1234)
    x = np.random.rand(N,1)*2-1
    y = np.random.rand(N,1)*2-1
    t = np.sqrt(x**2+y**2)
    pi_hat=4*np.sum(t <= 1)/N
    end_time = datetime.now()
    print(N)
    print(end_time - beg_time)
    return pi_hat
```

```
num_trials = [10**i for i in range(2,8)]
list(map(pi_simulator_2, num_trials))
```

```
## 100
## 0:00:00.001000
## 1000
## 0:00:00
## 10000
## 0:00:00
## 100000
## 0:00:00.003000
## 1000000
## 0:00:00.049011
## 10000000
## 0:00:00.466106
## [2.96, 3.06, 3.1352, 3.13976, 3.142876, 3.1422888]
```

Repetitive simulation experiments (p. 22)

```
import numpy as np
def pi_simulator_3(N):
    x = np.random.rand(N,1)*2-1
    y = np.random.rand(N,1)*2-1
    t = np.sqrt(x**2+y**2)
    pi_hat=4*np.sum(t <= 1)/N
    return pi_hat
```

```
n = 100
MC_N = 1000
np.random.seed(1234)
samples = list(range(0,n))
for i in range(n):
    samples[i] = pi_simulator_3(MC_N)
print(samples[:5])
```

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

p. 23

```
import scipy as sp
import scipy.stats

X_bar = np.mean(samples)
s = np.sqrt(sum((X_bar-samples)**2)/(n-1))
t_ = sp.stats.t.ppf(0.975, n-1) # ppf means inverse cumulative distribution function
print("X_bar :",X_bar)
```

```
## X_bar : 3.1412000000000004
```

```
print("s :",s)
```

```
## s : 0.05271305973538881
```

```
print("t :",t_)
```

```
## t : 1.9842169515086827
```

Exercise 1 (p. 24)

```
n = 100
MC_N = 10000
np.random.seed(1234)
samples = list(range(0,n))
for i in range(n):
    samples[i] = pi_simulator_3(MC_N)
X_bar = np.mean(samples)
s = np.sqrt(sum((X_bar-samples)**2)/(n-1))
t_ = sp.stats.t.ppf(0.975, n-1)
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
```

Exercise 2 (p. 25)

```
n = 1000
MC_N = 10000
np.random.seed(1234)
samples = list(range(0,n))
for i in range(n):
    samples[i] = pi_simulator_3(MC_N)
X_bar = np.mean(samples)
s = np.sqrt(sum((X_bar-samples)**2)/(n-1))
t_ = sp.stats.t.ppf(0.975, n-1)
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
```