MPI-accelerated Bootstrap Resampling for Hypothesis Testing in Small Sample Investigation

Parallel Computing
Final Report

M112040036 孫瑞鴻
M112040034 李祐瑄
2023-06-06

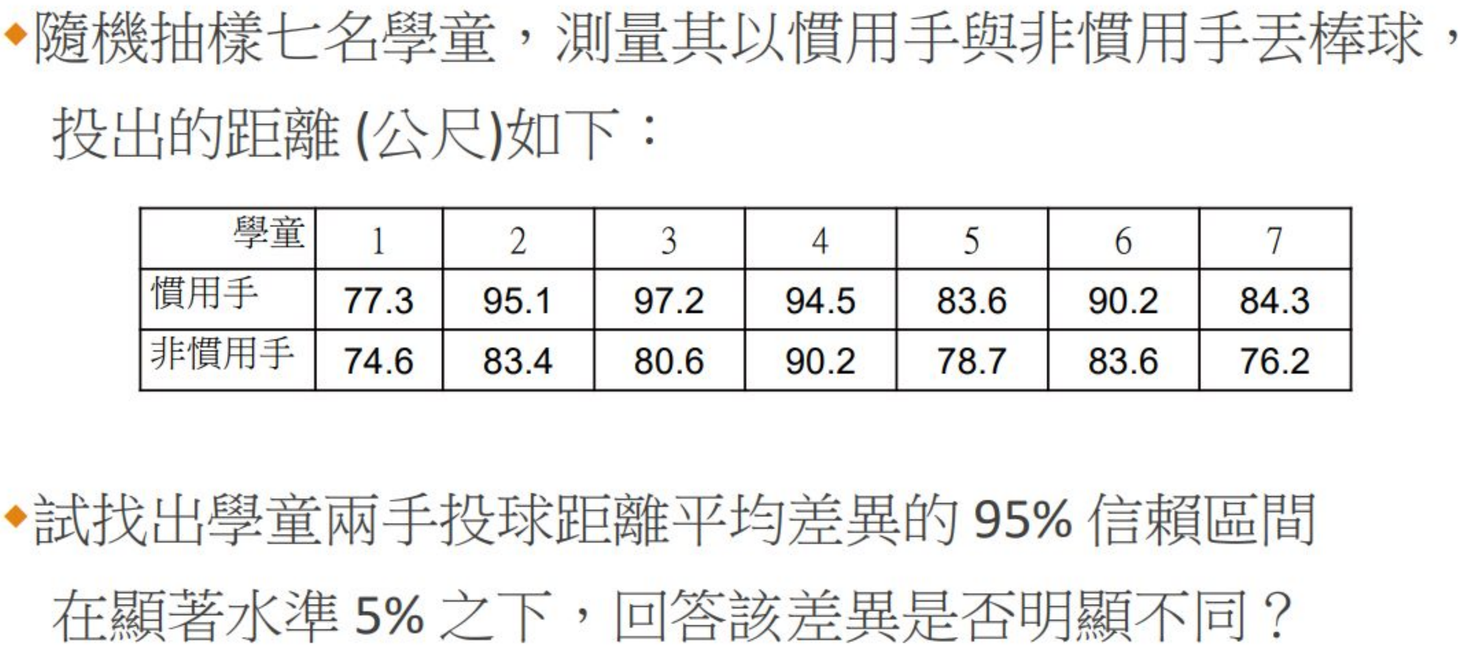
CONTENTS

- **Data**
- **Bootstrap**
- **MPI Application**
- **Conclusion**
- **Extra try — Remote**

01

Data

The data on throwing distances for dominant and non-dominant hand in pitching.



- 隨機抽樣七名學童，測量其以慣用手與非慣用手丟棒球，投出的距離 (公尺)如下：

學童	1	2	3	4	5	6	7
慣用手	77.3	95.1	97.2	94.5	83.6	90.2	84.3
非慣用手	74.6	83.4	80.6	90.2	78.7	83.6	76.2

- 試找出學童兩手投球距離平均差異的 95% 信賴區間在顯著水準 5% 之下，回答該差異是否明顯不同？

虛無假設:兩手投球距離無明顯差異

H0 = mean(X_1) - mean(X_2) = 0

source：STATISTICAL INFERENCE(NSYSU) w12_permutation test by ShengLi Tzeng

02

Bootstrap

parametric Bootstrap

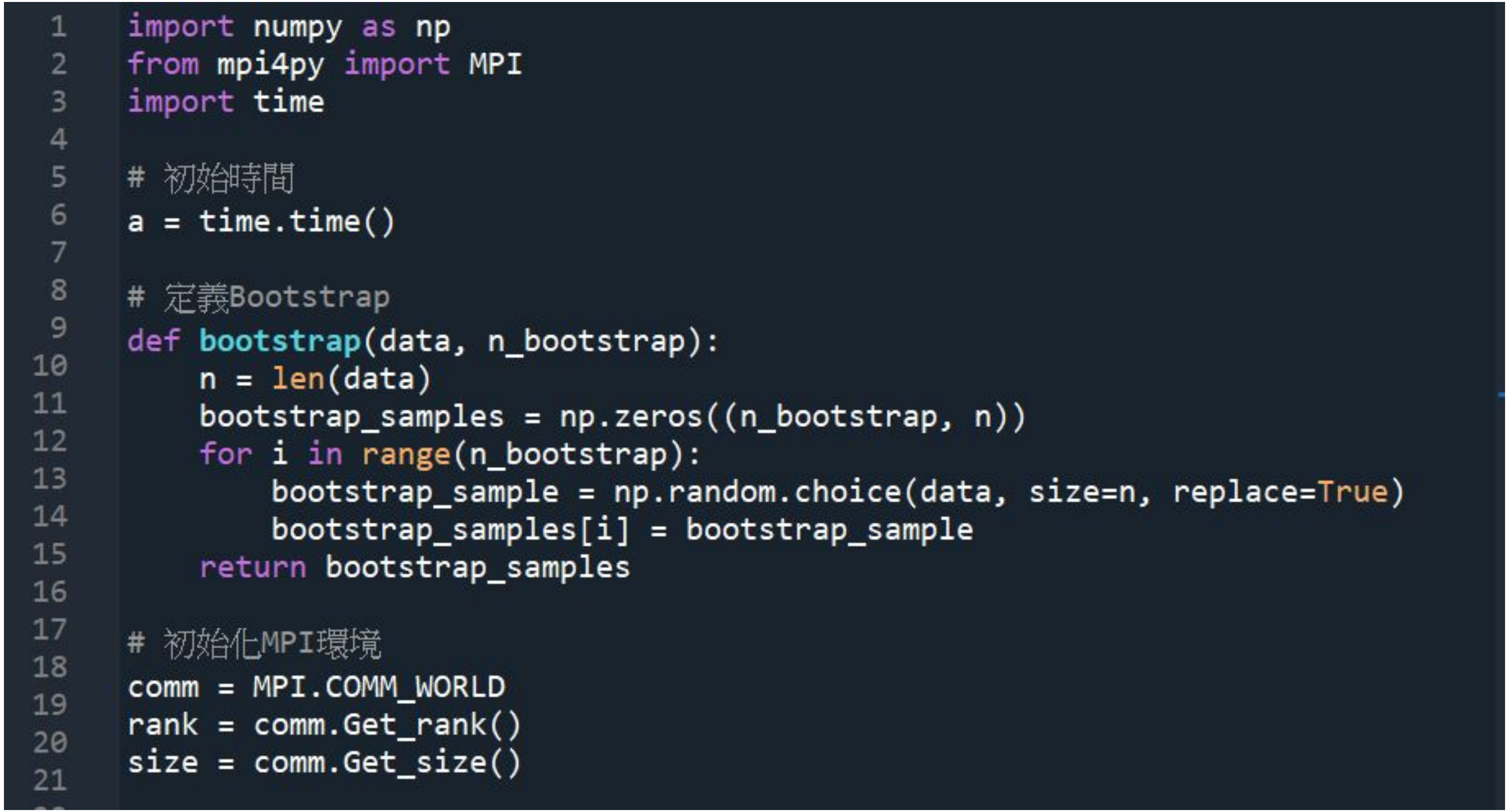
- 觀測到的資料不知來自何分布
  - 計算經驗分布 $\hat{F}$，即每個資料點 $x_i, i=1,2,\ldots,n$ 的機率值都是 $1/n$
  - 將經驗分布 $\hat{F}$ 當作母體實際的分布
  - 從「母體」產生B 組樣本數為n 的 Bootstrap 樣本
  - 每個Bootstrap 樣本各自計算統計量 Bootstrap replications 的數值
  - 蒐集B個Bootstrap replications，得到統計量的抽樣分布

source：STATISTICAL INFERENCE(NSYSU) w06_simulation&inference by ShengLi Tzeng

03

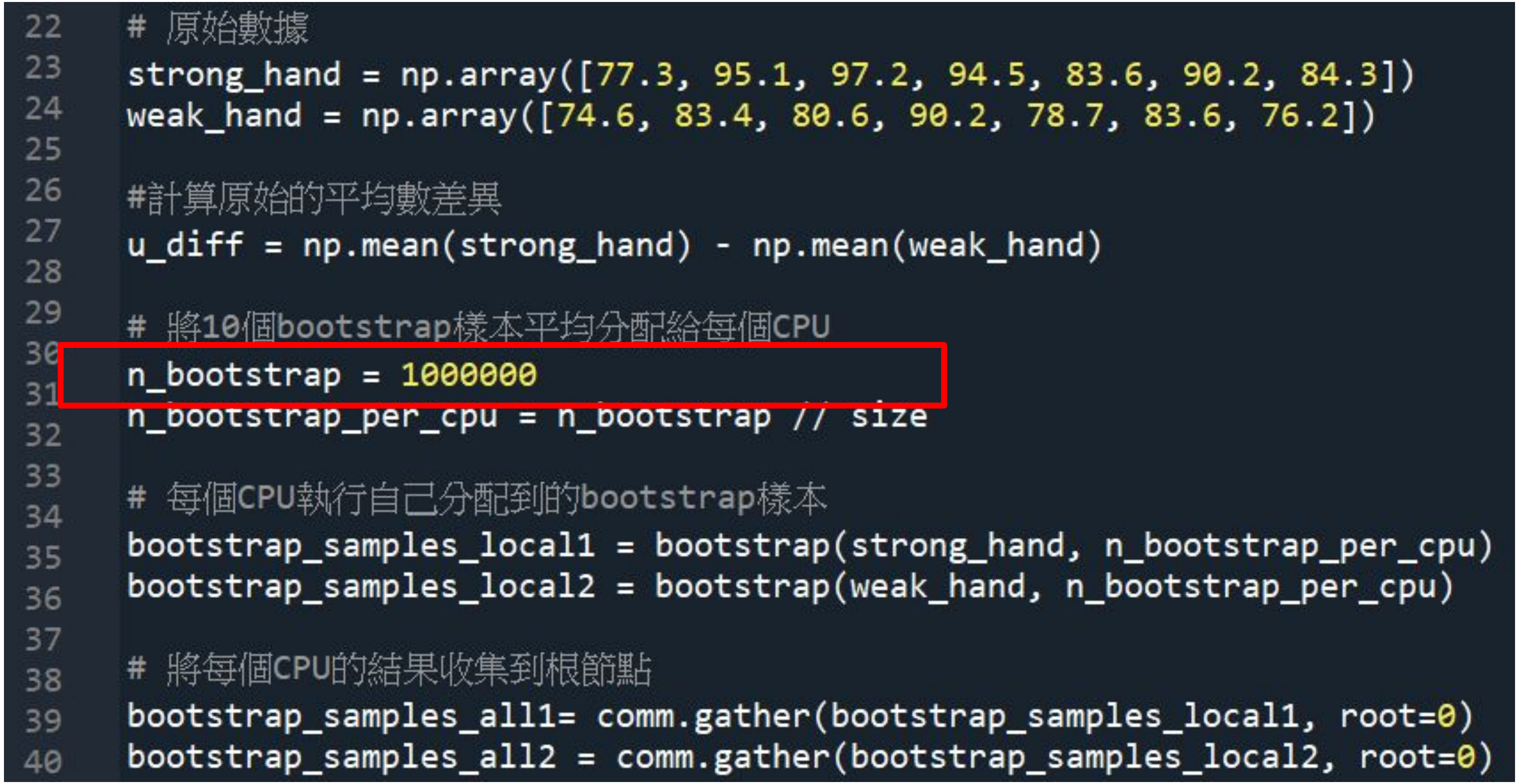
MPI Application

MPI Application



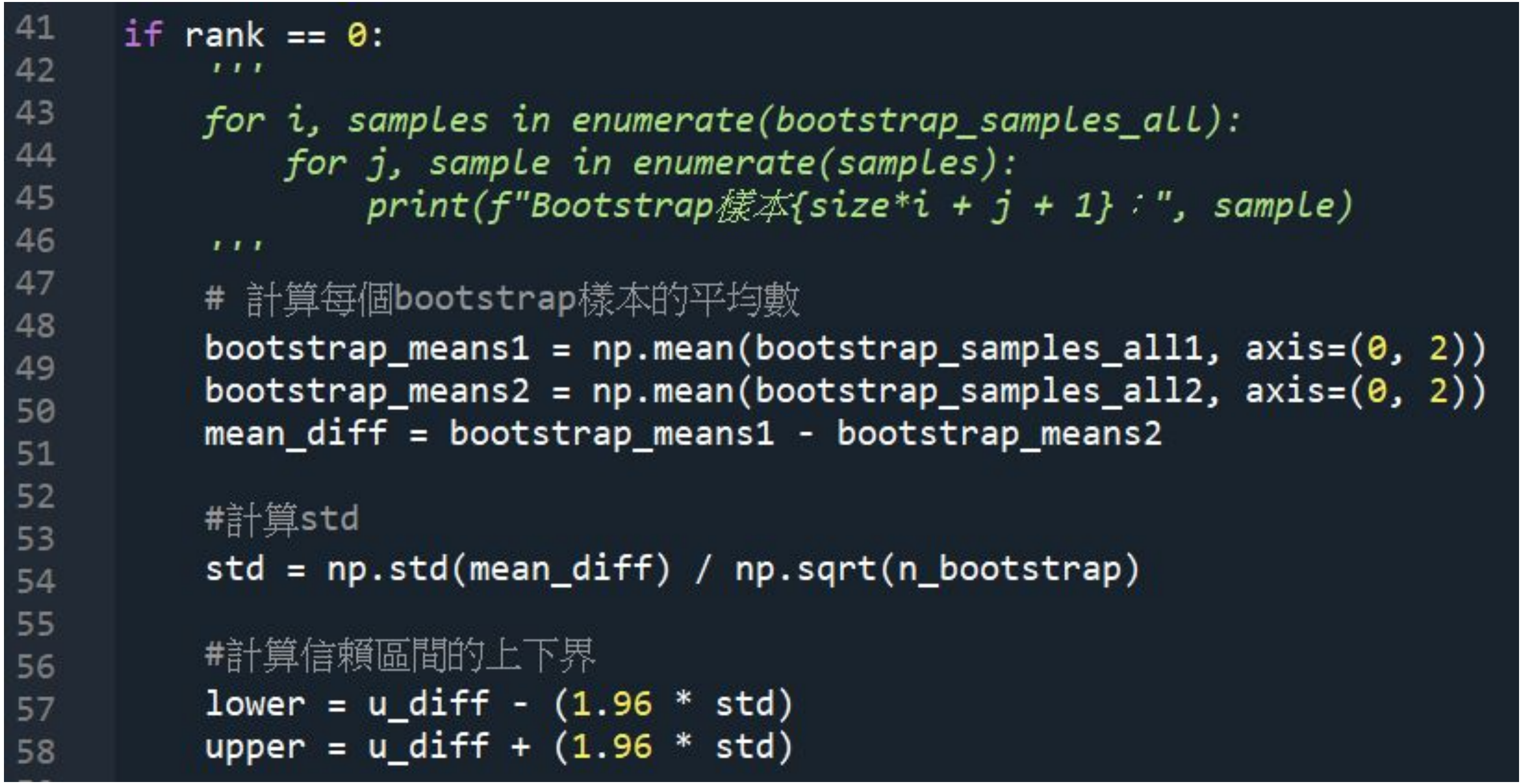
```python
import numpy as np
from mpi4py import MPI
import time

# 初始時間
a = time.time()

# 定義Bootstrap
def bootstrap(data, n_bootstrap):
    n = len(data)
    bootstrap_samples = np.zeros((n_bootstrap, n))
    for i in range(n_bootstrap):
        bootstrap_sample = np.random.choice(data, size=n, replace=True)
        bootstrap_samples[i] = bootstrap_sample
    return bootstrap_samples

# 初始化MPI環境
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()
```

MPI Application



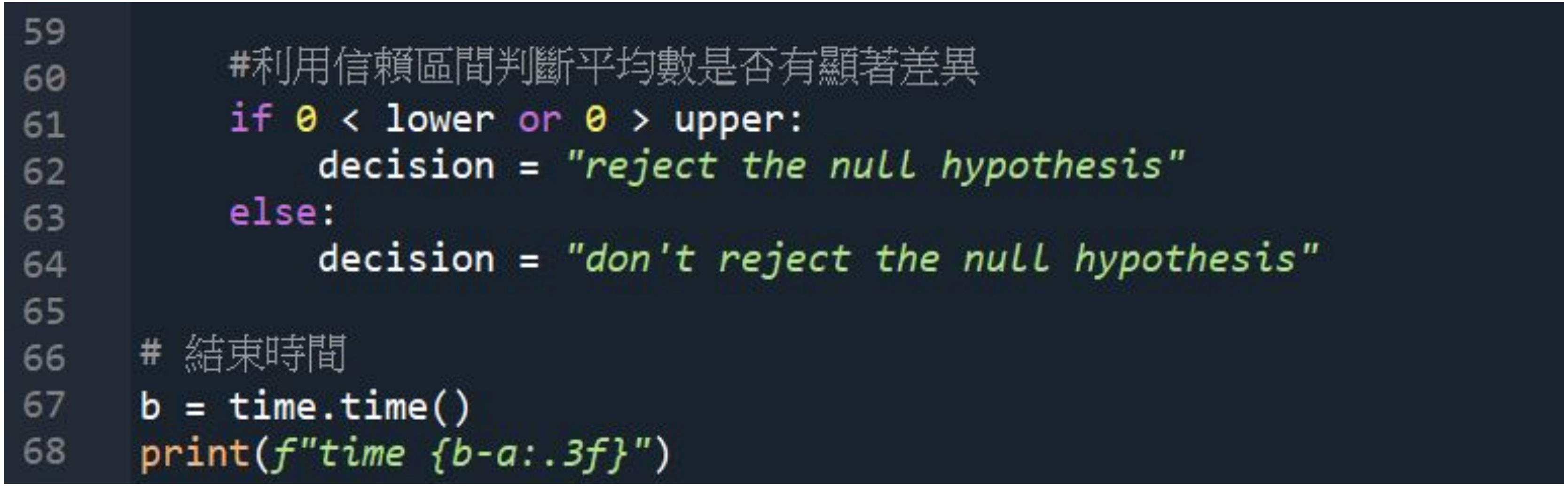
```
# 原始數據
strong_hand = np.array([77.3, 95.1, 97.2, 94.5, 83.6, 90.2, 84.3])
weak_hand = np.array([74.6, 83.4, 80.6, 90.2, 78.7, 83.6, 76.2])

#計算原始的平均數差異
u_diff = np.mean(strong_hand) - np.mean(weak_hand)

# 將10個bootstrap樣本平均分配給每個CPU
n_bootstrap = 1000000
n_bootstrap_per_cpu = n_bootstrap // size

# 每個CPU執行自己分配到的bootstrap樣本
bootstrap_samples_local1 = bootstrap(strong_hand, n_bootstrap_per_cpu)
bootstrap_samples_local2 = bootstrap(weak_hand, n_bootstrap_per_cpu)

# 將每個CPU的結果收集到根節點
bootstrap_samples_all1= comm.gather(bootstrap_samples_local1, root=0)
bootstrap_samples_all2 = comm.gather(bootstrap_samples_local2, root=0)
```

MPI Application



```
if rank == 0:
    '''
    for i, samples in enumerate(bootstrap_samples_all):
        for j, sample in enumerate(samples):
            print(f"Bootstrap樣本{size*i + j + 1}：", sample)
    '''
    # 計算每個bootstrap樣本的平均數
    bootstrap_means1 = np.mean(bootstrap_samples_all1, axis=(0, 2))
    bootstrap_means2 = np.mean(bootstrap_samples_all2, axis=(0, 2))
    mean_diff = bootstrap_means1 - bootstrap_means2

    #計算std
    std = np.std(mean_diff) / np.sqrt(n_bootstrap)

    #計算信賴區間的上下界
    lower = u_diff - (1.96 * std)
    upper = u_diff + (1.96 * std)
```

MPI Application

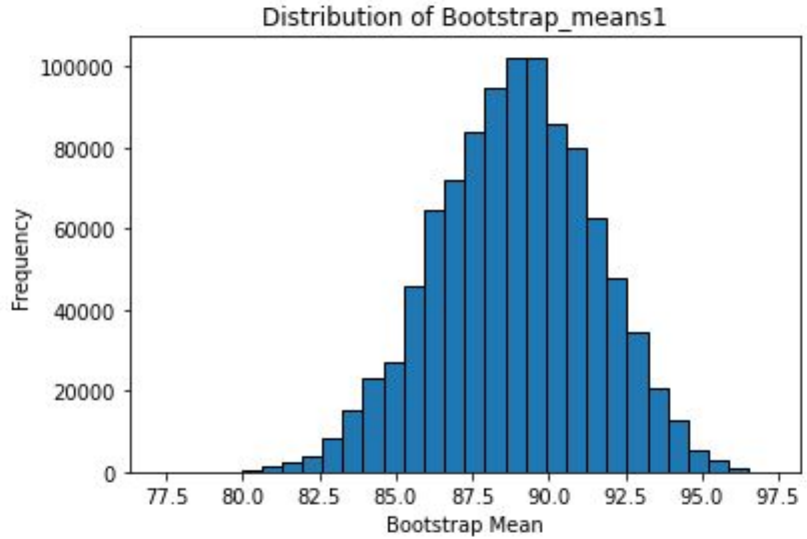


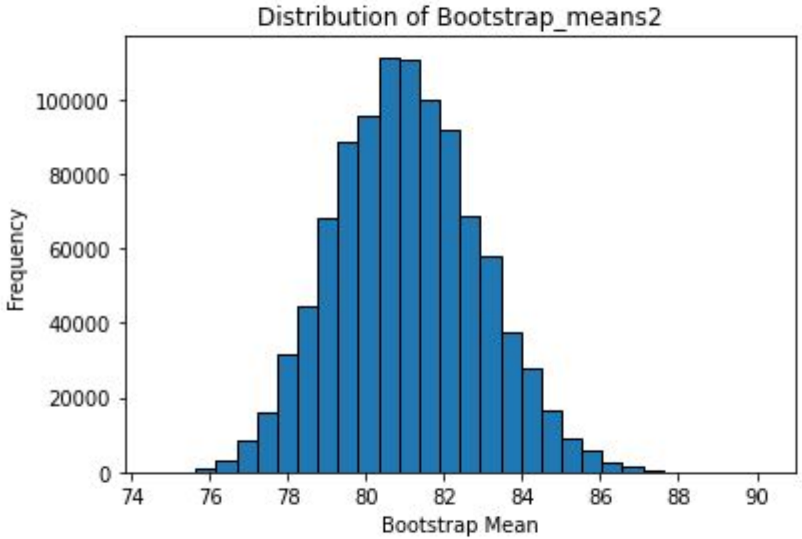
```python

    #利用信賴區間判斷平均數是否有顯著差異
    if 0 < lower or 0 > upper:
        decision = "reject the null hypothesis"
    else:
        decision = "don't reject the null hypothesis"

# 結束時間
b = time.time()
print(f"time {b-a:.3f}")
```

04

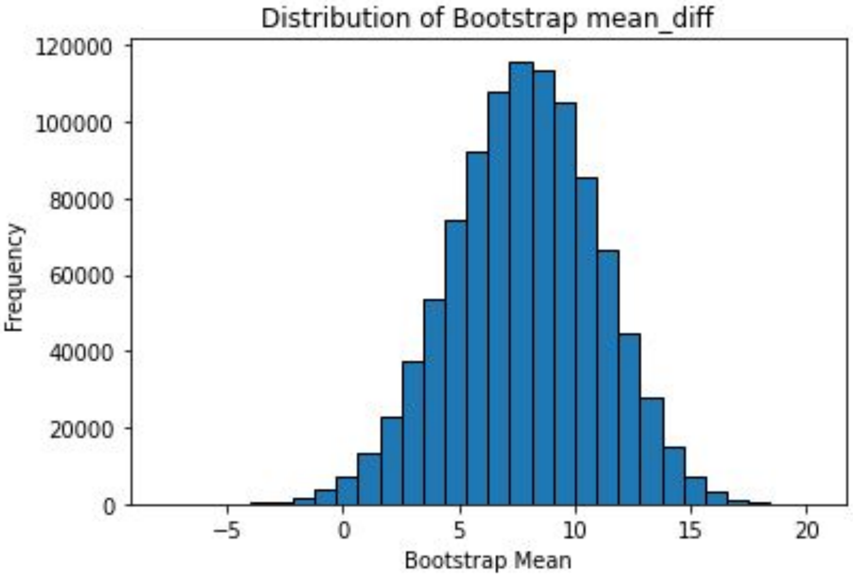
Conclusion

Conclusion


Distribution of Bootstrap_means1
Frequency
100000
80000
60000
40000
20000
0
77.5
80.0
82.5
85.0
87.5
90.0
92.5
95.0
97.5
Bootstrap Mean

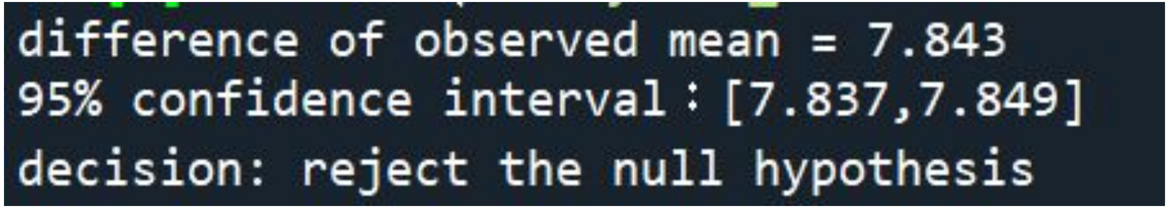

Distribution of Bootstrap_means2
Frequency
100000
80000
60000
40000
20000
0
74
76
78
80
82
84
86
88
90
Bootstrap Mean

Conclusion



According to the Central Limit Theorem, it can be observed that the distribution approaches a normal distribution.

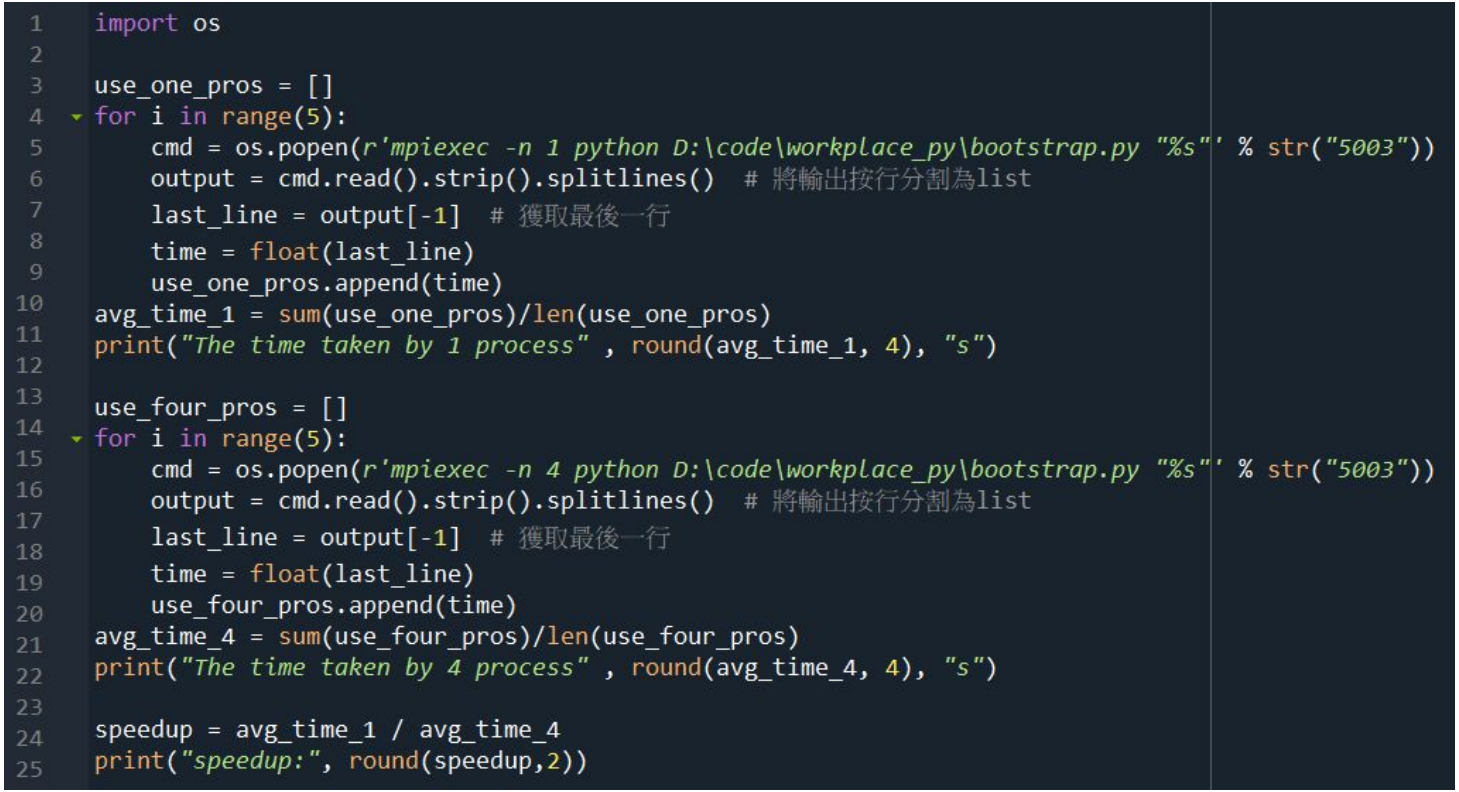
Conclusion



```
difference of observed mean = 7.843
95% confidence interval : [7.837,7.849]
decision: reject the null hypothesis
```

The confidence interval does not include zero, therefore the null hypothesis is rejected.

Conclusion

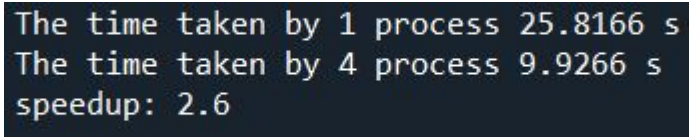


```python
import os

use_one_pros = []
for i in range(5):
    cmd = os.popen(r'mpiexec -n 1 python D:\code\workplace_py\bootstrap.py "%s"' % str("5003"))
    output = cmd.read().strip().splitlines()  # 將輸出按行分割為list
    last_line = output[-1]  # 獲取最後一行
    time = float(last_line)
    use_one_pros.append(time)
avg_time_1 = sum(use_one_pros)/len(use_one_pros)
print("The time taken by 1 process" , round(avg_time_1, 4), "s")

use_four_pros = []
for i in range(5):
    cmd = os.popen(r'mpiexec -n 4 python D:\code\workplace_py\bootstrap.py "%s"' % str("5003"))
    output = cmd.read().strip().splitlines()  # 將輸出按行分割為list
    last_line = output[-1]  # 獲取最後一行
    time = float(last_line)
    use_four_pros.append(time)
avg_time_4 = sum(use_four_pros)/len(use_four_pros)
print("The time taken by 4 process" , round(avg_time_4, 4), "s")

speedup = avg_time_1 / avg_time_4
print("speedup:", round(speedup,2))
```

Conclusion

Speed up : The time taken by 1 process / The time taken by 4 process

The time taken by 1 process 25.8166 s
The time taken by 4 process 9.9266 s
speedup: 2.6

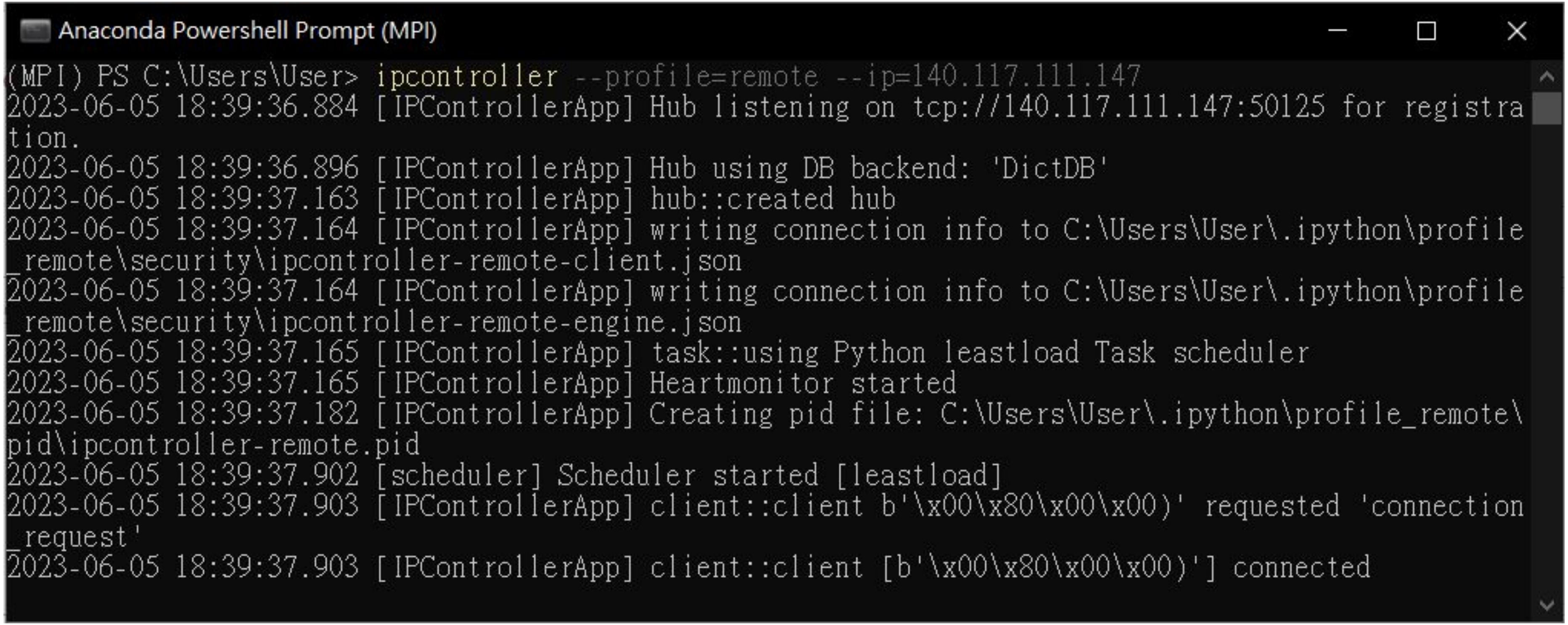


```
The time taken by 1 process 25.8166 s
The time taken by 4 process 9.9266 s
speedup: 2.6
```

05

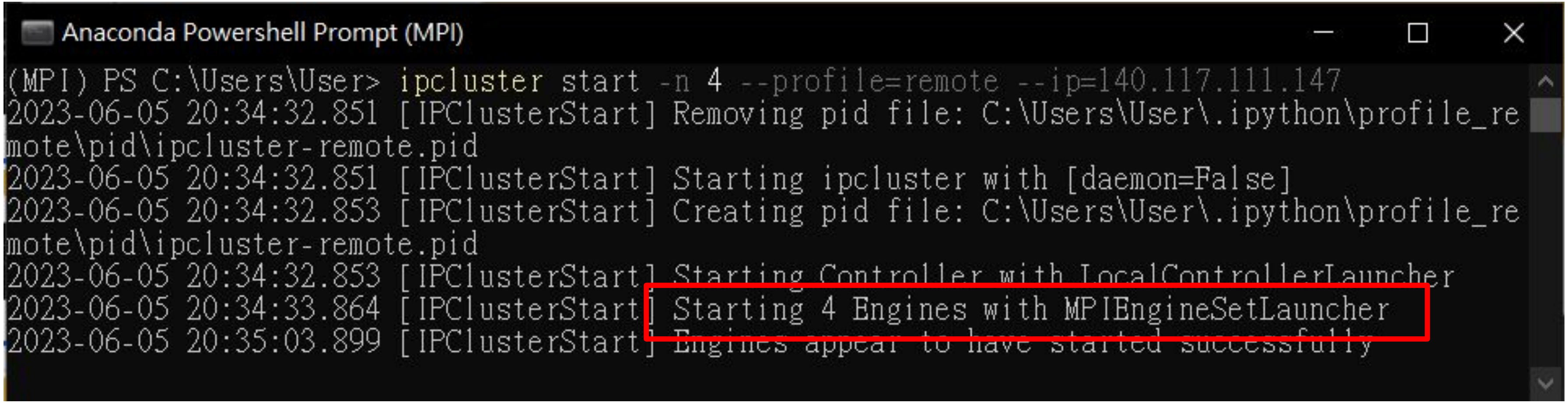
Extra try — Remote

Extra try — Remote



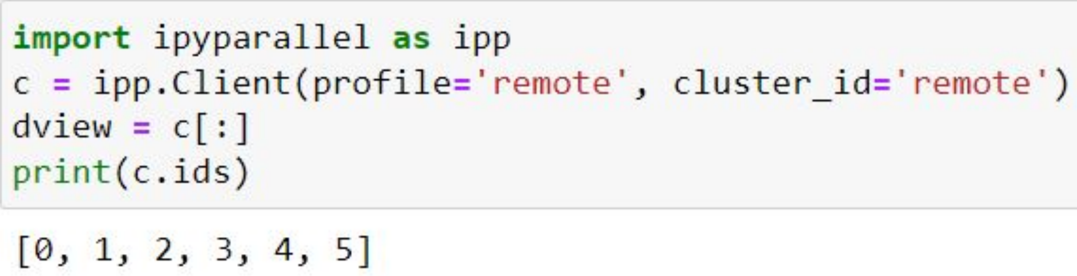
```
(MPI) PS C:\Users\User> ipcontroller --profile=remote --ip=140.117.111.147
2023-06-05 18:39:36.884 [IPControllerApp] Hub listening on tcp://140.117.111.147:50125 for registra
tion.
2023-06-05 18:39:36.896 [IPControllerApp] Hub using DB backend: 'DictDB'
2023-06-05 18:39:37.163 [IPControllerApp] hub::created hub
2023-06-05 18:39:37.164 [IPControllerApp] writing connection info to C:\Users\User\.ipython\profile
_remote\security\ipcontroller-remote-client.json
2023-06-05 18:39:37.164 [IPControllerApp] writing connection info to C:\Users\User\.ipython\profile
_remote\security\ipcontroller-remote-engine.json
2023-06-05 18:39:37.165 [IPControllerApp] task::using Python leastload Task scheduler
2023-06-05 18:39:37.165 [IPControllerApp] Heartmonitor started
2023-06-05 18:39:37.182 [IPControllerApp] Creating pid file: C:\Users\User\.ipython\profile_remote\
pid\ipcontroller-remote.pid
2023-06-05 18:39:37.902 [scheduler] Scheduler started [leastload]
2023-06-05 18:39:37.903 [IPControllerApp] client::client b'\x00\x80\x00\x00)' requested 'connection
_request'
2023-06-05 18:39:37.903 [IPControllerApp] client::client [b'\x00\x80\x00\x00)'] connected
```

Extra try — Remote



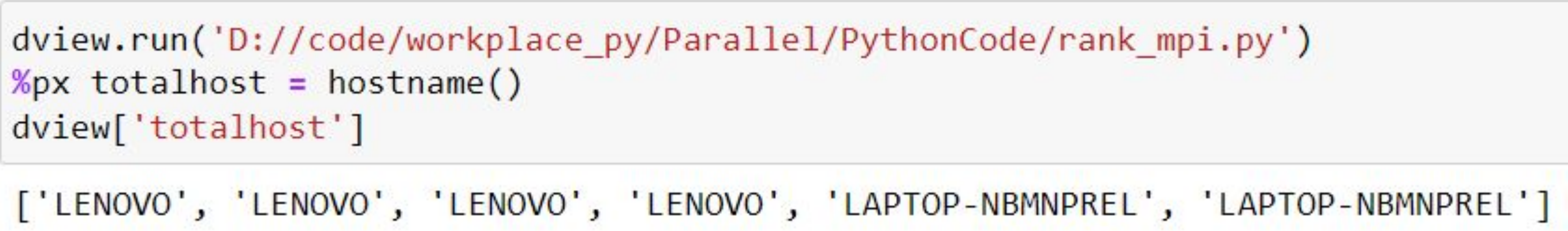
```
(MPI) PS C:\Users\User> ipcluster start -n 4 --profile=remote --ip=140.117.111.147
2023-06-05 20:34:32.851 [IPClusterStart] Removing pid file: C:\Users\User\.ipython\profile_re
mote\pid\ipcluster-remote.pid
2023-06-05 20:34:32.851 [IPClusterStart] Starting ipcluster with [daemon=False]
2023-06-05 20:34:32.853 [IPClusterStart] Creating pid file: C:\Users\User\.ipython\profile_re
mote\pid\ipcluster-remote.pid
2023-06-05 20:34:32.853 [IPClusterStart] Starting Controller with LocalControllerLauncher
2023-06-05 20:34:33.864 [IPClusterStart] Starting 4 Engines with MPIEngineSetLauncher
2023-06-05 20:35:03.899 [IPClusterStart] Engines appear to have started successfully
```

Extra try — Remote



```
import ipyparallel as ipp
c = ipp.Client(profile='remote', cluster_id='remote')
dview = c[:]
print(c.ids)
```

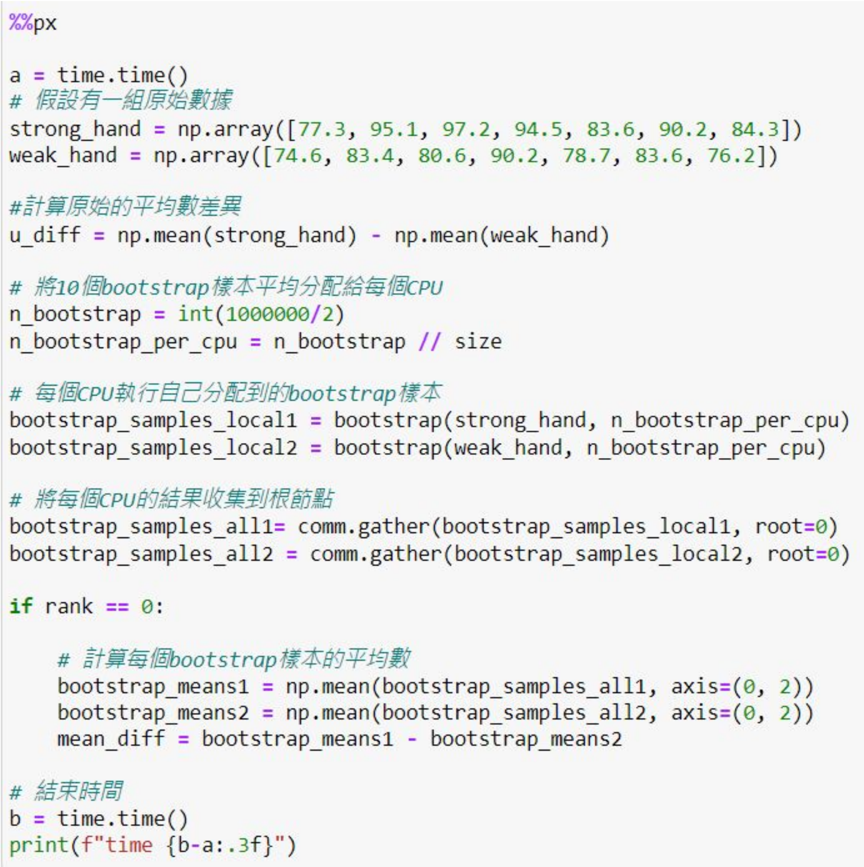
```
[0, 1, 2, 3, 4, 5]
```



```
dview.run('D://code/workplace_py/Parallel/PythonCode/rank_mpi.py')
%px totalhost = hostname()
dview['totalhost']
```

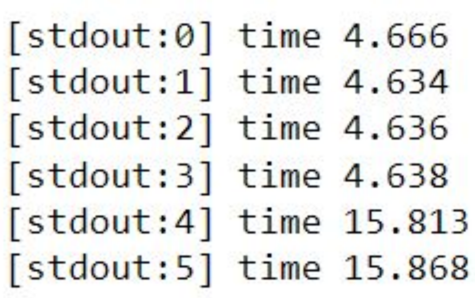
```
['LENOVO', 'LENOVO', 'LENOVO', 'LENOVO', 'LAPTOP-NBMNPREL', 'LAPTOP-NBMNPREL']
```

Extra try — Remote



```
%%px

a = time.time()
# 假設有一組原始數據
strong_hand = np.array([77.3, 95.1, 97.2, 94.5, 83.6, 90.2, 84.3])
weak_hand = np.array([74.6, 83.4, 80.6, 90.2, 78.7, 83.6, 76.2])

#計算原始的平均數差異
u_diff = np.mean(strong_hand) - np.mean(weak_hand)

# 將10個bootstrap樣本平均分配給每個CPU
n_bootstrap = int(1000000/2)
n_bootstrap_per_cpu = n_bootstrap // size

# 每個CPU執行自己分配到的bootstrap樣本
bootstrap_samples_local1 = bootstrap(strong_hand, n_bootstrap_per_cpu)
bootstrap_samples_local2 = bootstrap(weak_hand, n_bootstrap_per_cpu)

# 將每個CPU的結果收集到根節點
bootstrap_samples_all1= comm.gather(bootstrap_samples_local1, root=0)
bootstrap_samples_all2 = comm.gather(bootstrap_samples_local2, root=0)

if rank == 0:

    # 計算每個bootstrap樣本的平均數
    bootstrap_means1 = np.mean(bootstrap_samples_all1, axis=(0, 2))
    bootstrap_means2 = np.mean(bootstrap_samples_all2, axis=(0, 2))
    mean_diff = bootstrap_means1 - bootstrap_means2

# 結束時間
b = time.time()
print(f"time {b-a:.3f}")
```

Extra try — Remote



```
[stdout:0] time 4.666
[stdout:1] time 4.634
[stdout:2] time 4.636
[stdout:3] time 4.638
[stdout:4] time 15.813
[stdout:5] time 15.868
```

THE END

THANKS