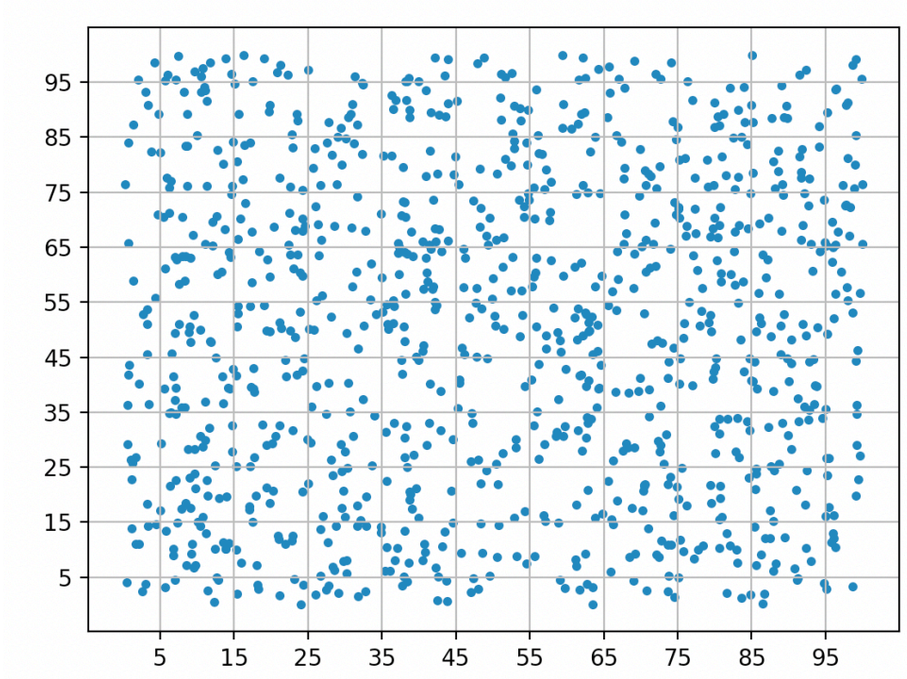
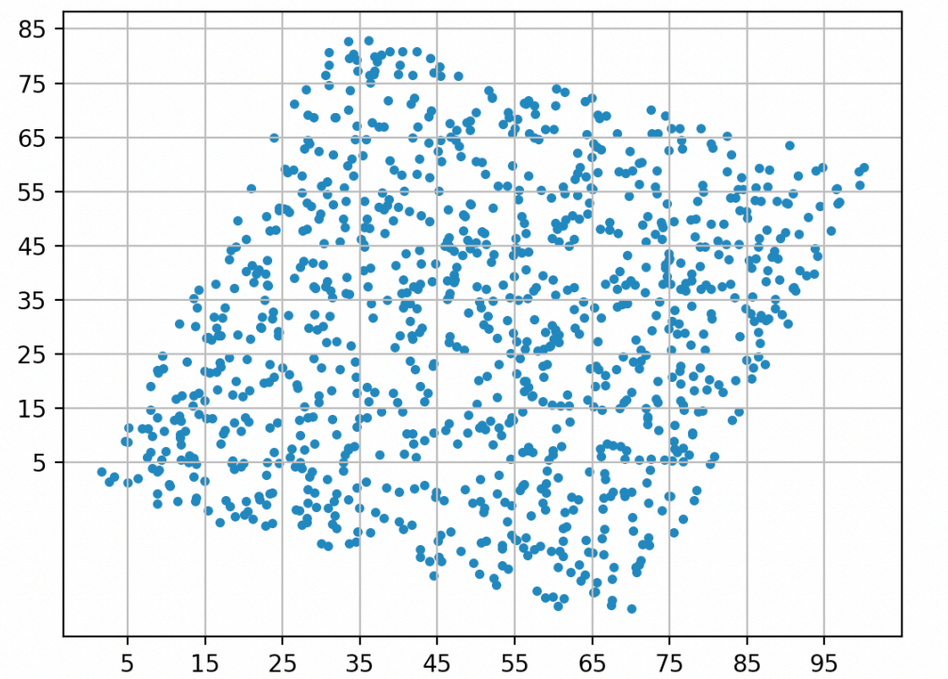
那一個比較正？
交錯亂度（Cross Entropy）量化正不正





步驟一、匯入套件

```
import matplotlib.pyplot as plt
import random
import numpy as np
```

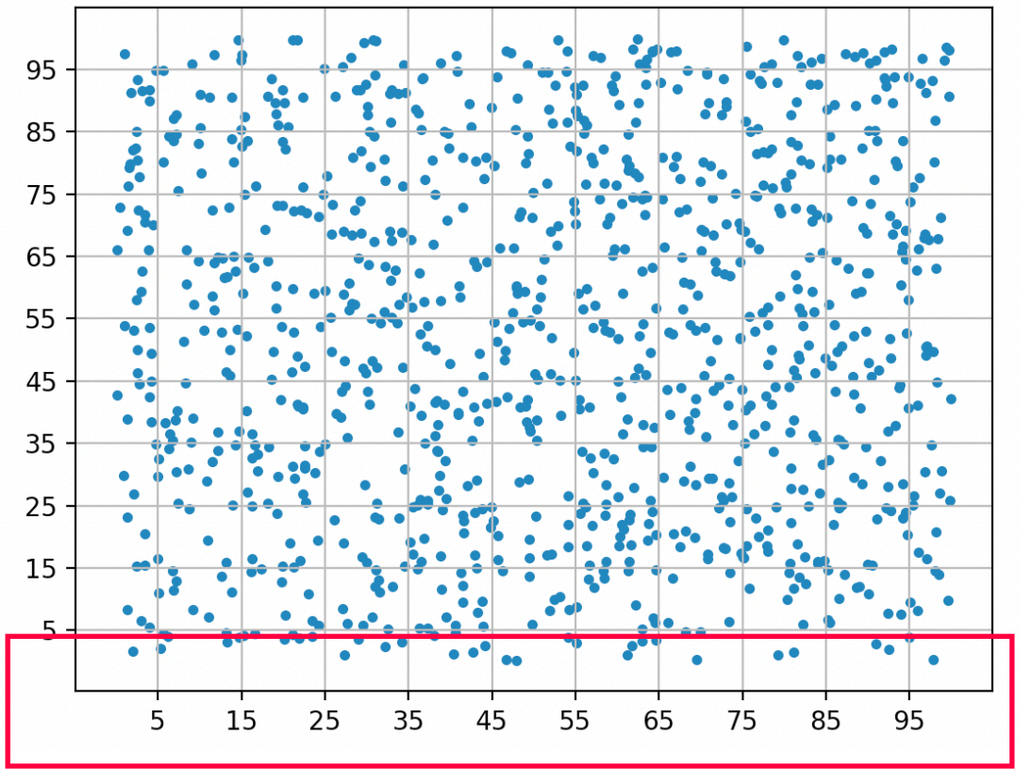
步驟二

在[0,100]×[0,100]產生

1000二維資料點

```
def produce_sig(N): # 1A
    x_list = []
    y_list = []
    for i in range(N):
        x_list.append(random.uniform(0,1)*100)
        y_list.append(random.uniform(0, 1)*100)
    return x_list, y_list
```

步驟三、在串列最小值與最大值間產生n個格點




```
def bin_points_list(data_list,n):
    max_v = int(np.floor(max(data_list)))
    min_v = int(np.ceil(int(min(data_list))))
    interval = int((max_v-min_v)/n)
    bin = np.linspace(min_v,max_v,n)
    print(len(bin))
    return bin
```

步驟四、繪製有格點的二維資料點圖

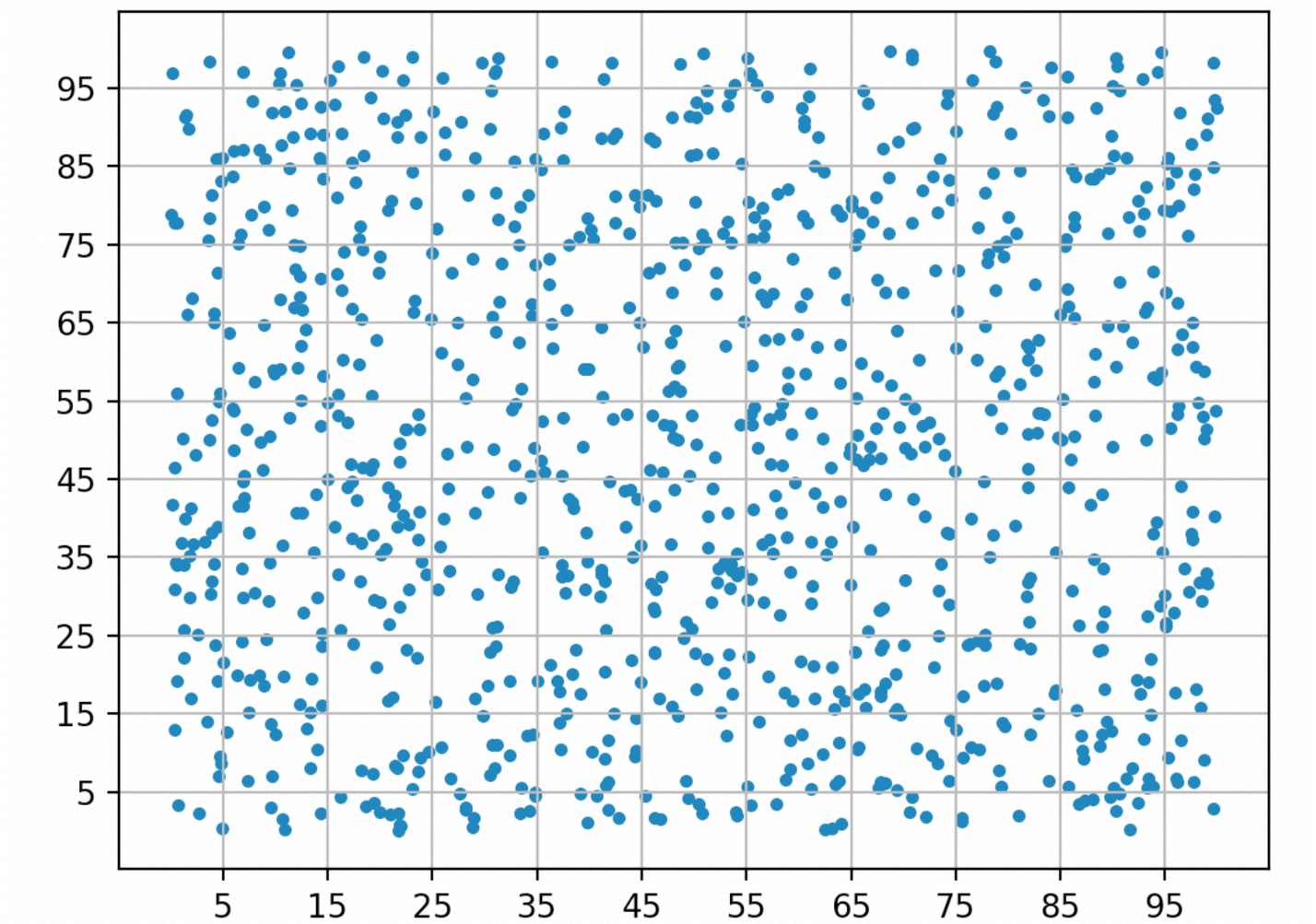
```
def plot_points(x_list,y_list,xticks, yticks): # 1D
    fig = plt.figure()
    ax = fig.gca()
    ax.set_xticks(xticks)
    ax.set_yticks(yticks)
    plt.scatter(x_list, y_list, marker='.')
    plt.grid()
```

步驟五A、測試1A

Test1

shape2D2.py中的函數
produce_sig
bin_points_list
plot_points

```
from shape2D2 import *
N = 1000
bin_size = 10
x_list, y_list = produce_sig(N)
x_bin = bin_points_list(x_list,bin_size)
y_bin = bin_points_list(y_list,bin_size)
plot_points(x_list,y_list,x_bin, y_bin)

plt.show()
```


95
85
75
65
55
45
35
25
15
5
5
15
25
35
45
55
65
75
85
95

步驟五B、測試1B

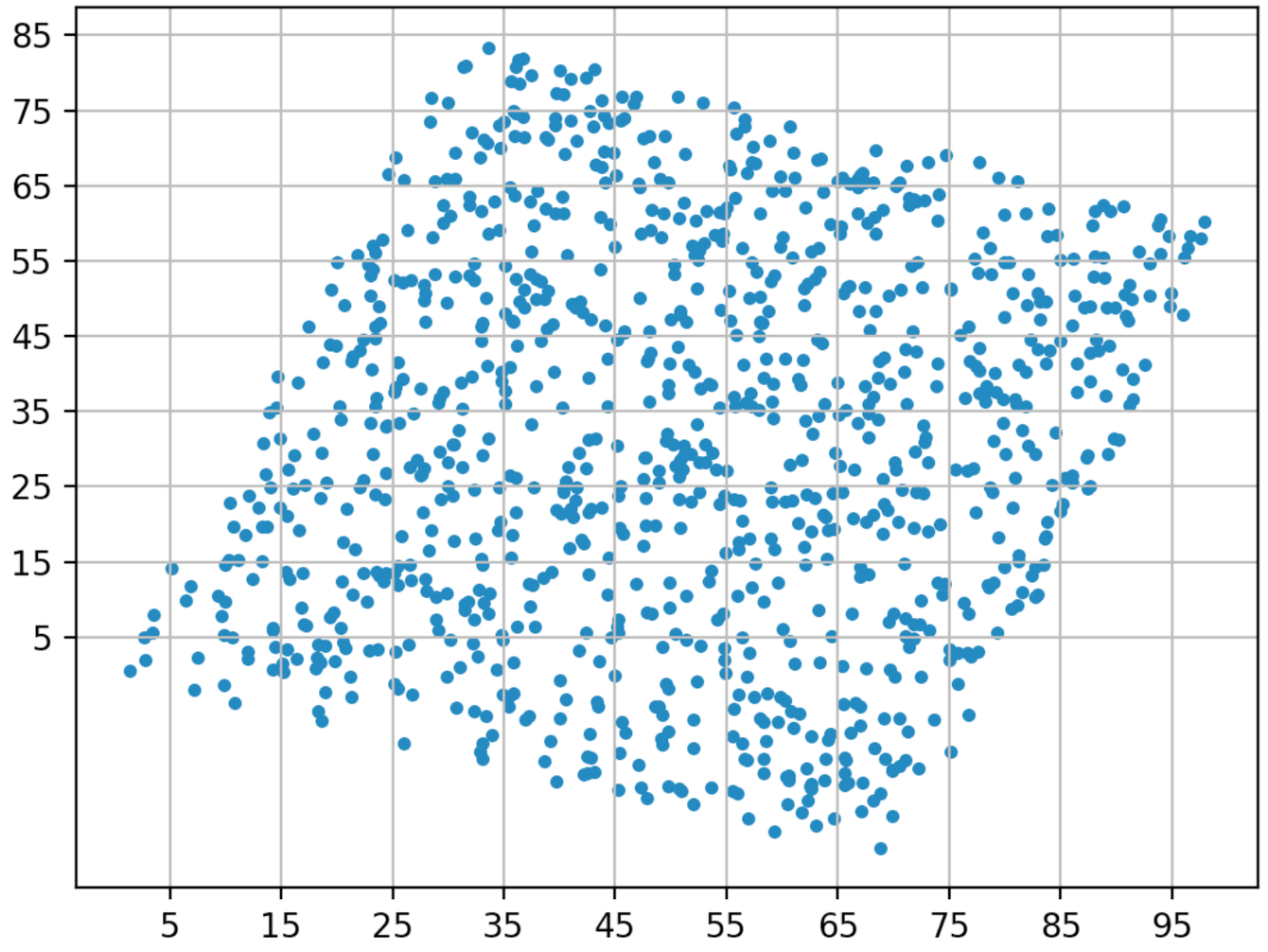
```
from shape2D2 import *
N = 1000
bin_size = 10
x_list, y_list = produce_sig(N)
A = np.matrix([[0.7,0.31],[-0.25,0.85]])

DA = A_sig(A,x_list,y_list)
x_list = list(DA[:,0])
y_list = list(DA[:,1])

x_bin = bin_points_list(x_list,bin_size)
y_bin = bin_points_list(y_list,bin_size)
plot_points(x_list,y_list,x_bin, y_bin)
plt.show()
```

shape2D.py中的函數

produce_sig

bin_points_list

plot_points


85
75
65
55
45
35
25
15
5
5
15
25
35
45
55
65
75
85
95

步驟六、my_histogram 橫座標與縱座標的直方累計圖

```
def my_hitogram(x,n): #3B
    # max_v = np.max(x)
    bin = bin_points_list(x,n)
    x2bin = np.zeros(n)
    for i in range(len(x)):
        ind = np.argmin(np.abs(float(x[i])- bi
        x2bin[ind] += 1
plt.figure()
plt.bar(range(n),x2bin
return x2bin, bin
```

產生n個格點

ind代表與x[i]最近的格點

求x[i]與n個資料格點間的絕對距離

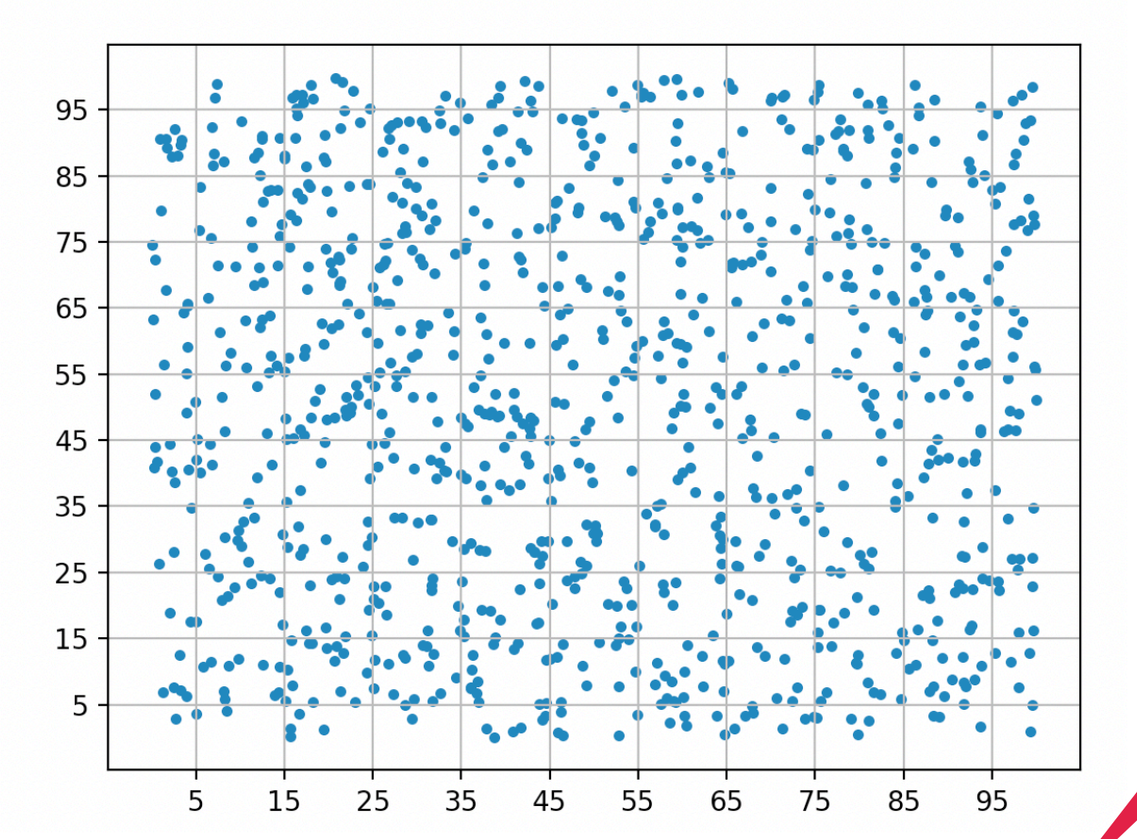
bar繪製直方圖

與各個格點最近的資料點個數

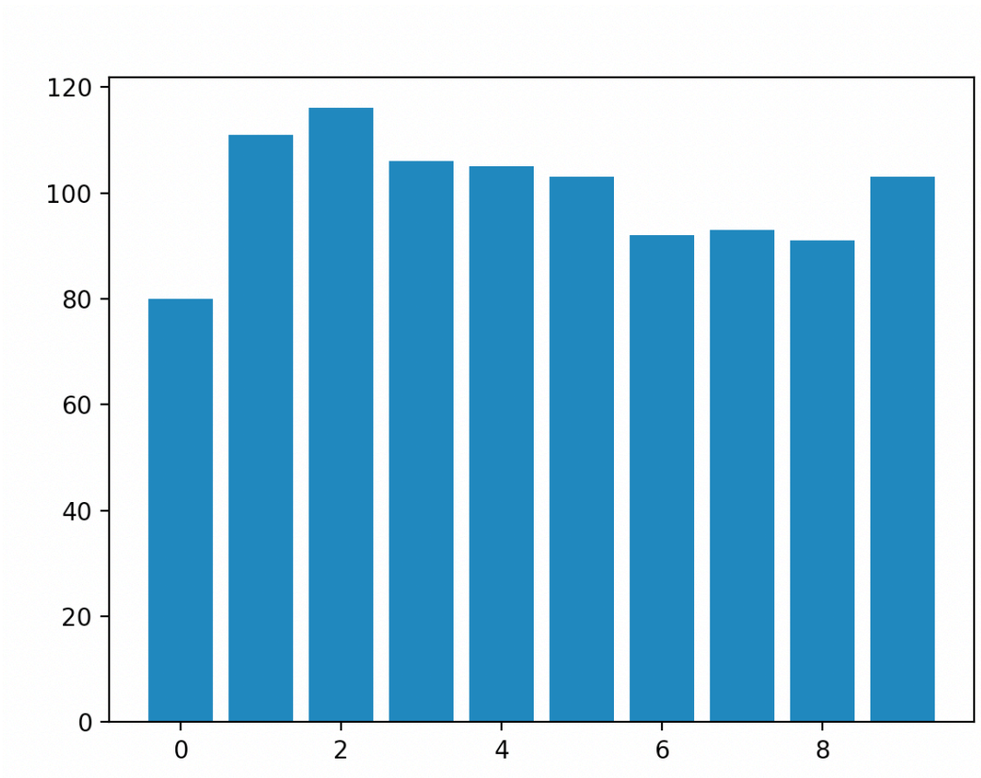
步驟七、測試2

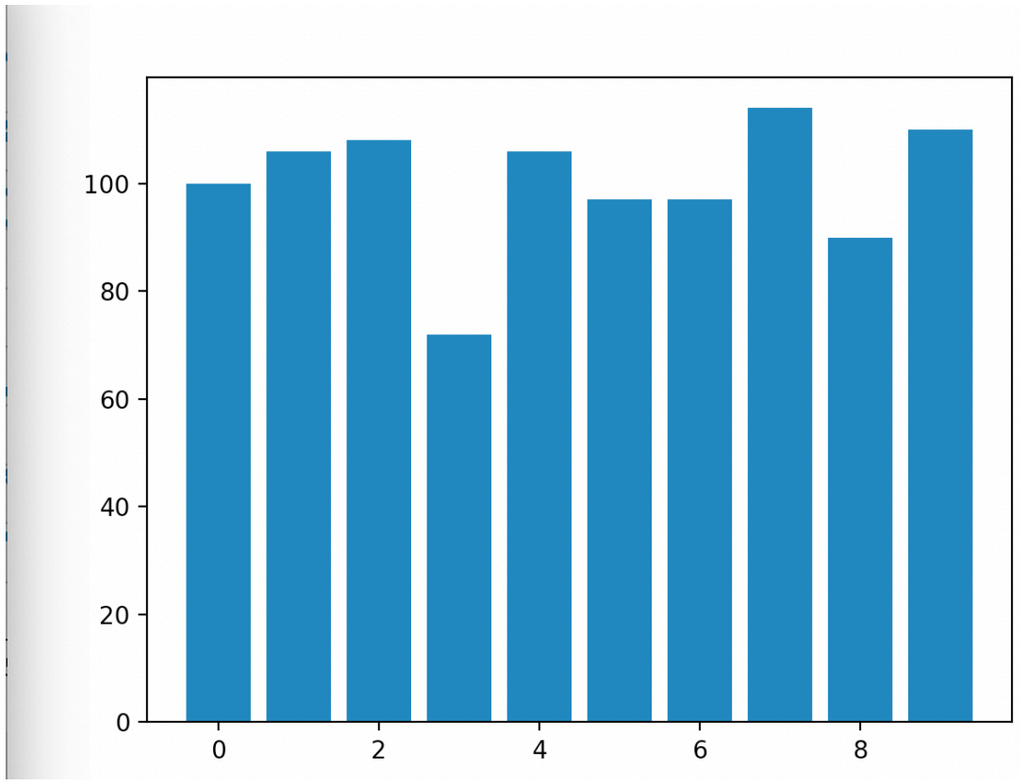
test2

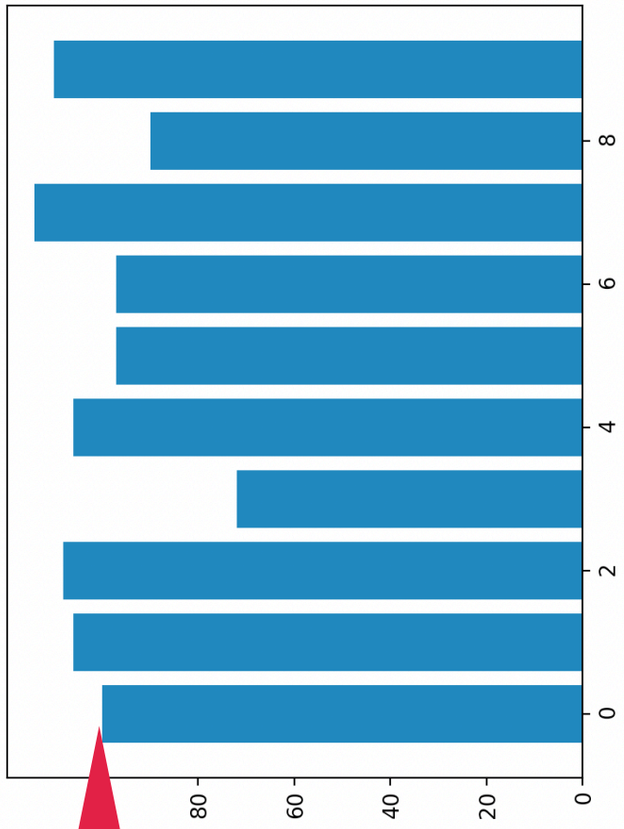
```
from shape2D2 import *
N = 1000
bin_size = 10
x_list, y_list = produce_sig(N)
bin = bin_points(bin_size)
plot_points(x_list,y_list,bin, bin)
my_hitogram(x_list,bin_size)
my_hitogram(y_list,bin_size)
plt.show()
```



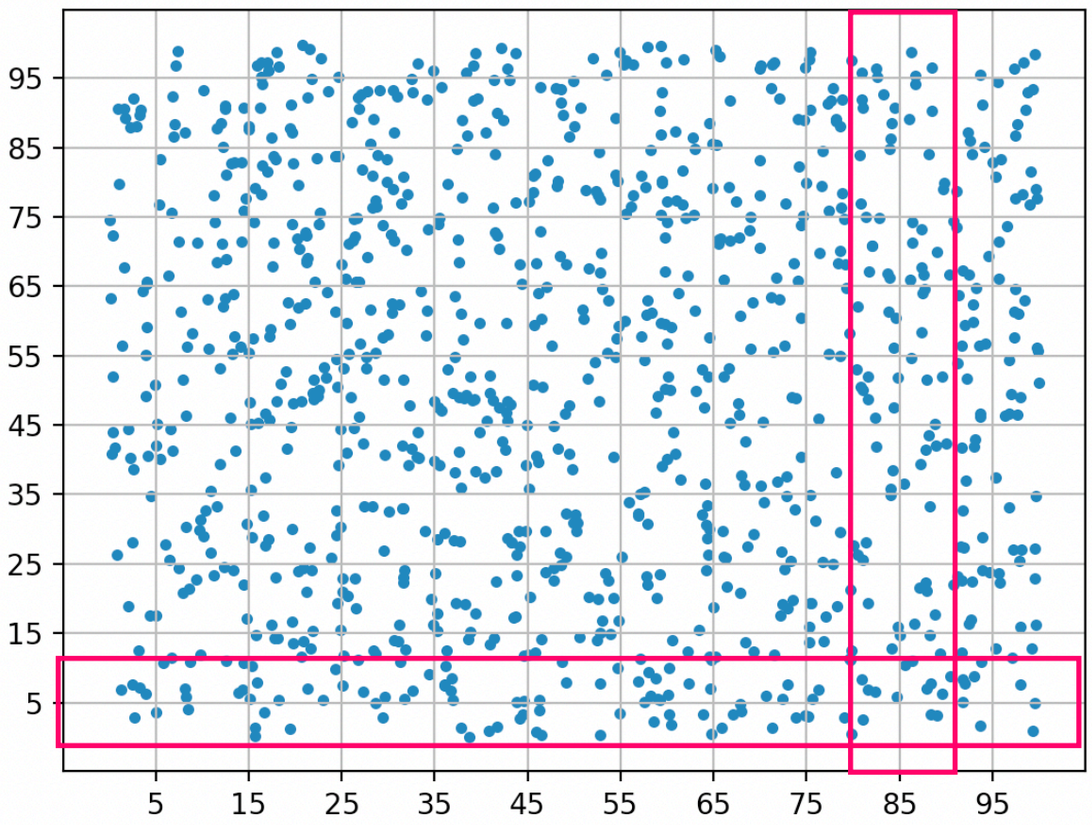
橫坐標與縱座標的直方累計圖

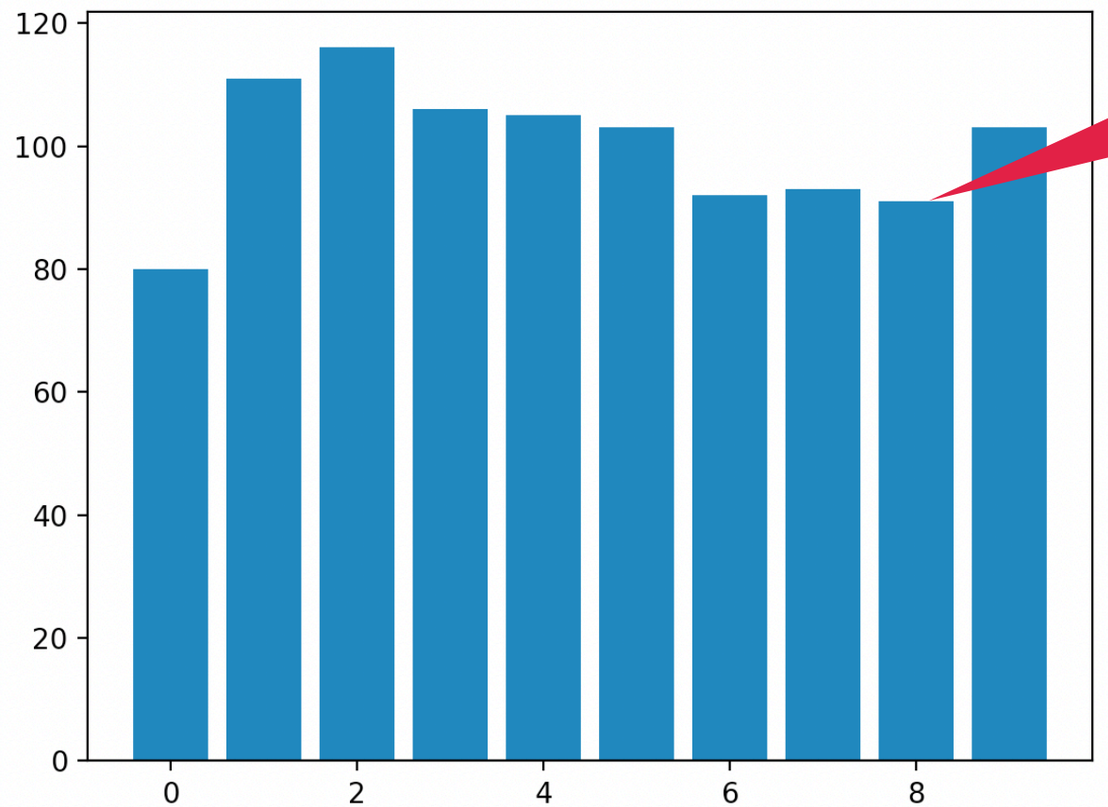






縱坐標與格點5最近的資料點總數





橫坐標與格點85最近的資料點總數

將個別直方圖的累計點數除以N=1000就可以近似個別機率

步驟八、
my_joint_histogram
縱橫聯合累計矩陣

xind代表與x[i]最近的格點
Yind代表與y[i]最近的格點

```
def my_joint_histogram(x,y,n,x_bin,y_bin):
    xybin = np.zeros((n,n))
    for i in range(len(x)):
        xind = np.argmin(np.abs(float(x[i])- x_bin))
        yind = np.argmin(np.abs(float(y[i])- y_bin))
        xybin[xind,yind] += 1
return xybin
```

xybin是10x10的陣列，它的內容代表落在相對應方格的資料點總數

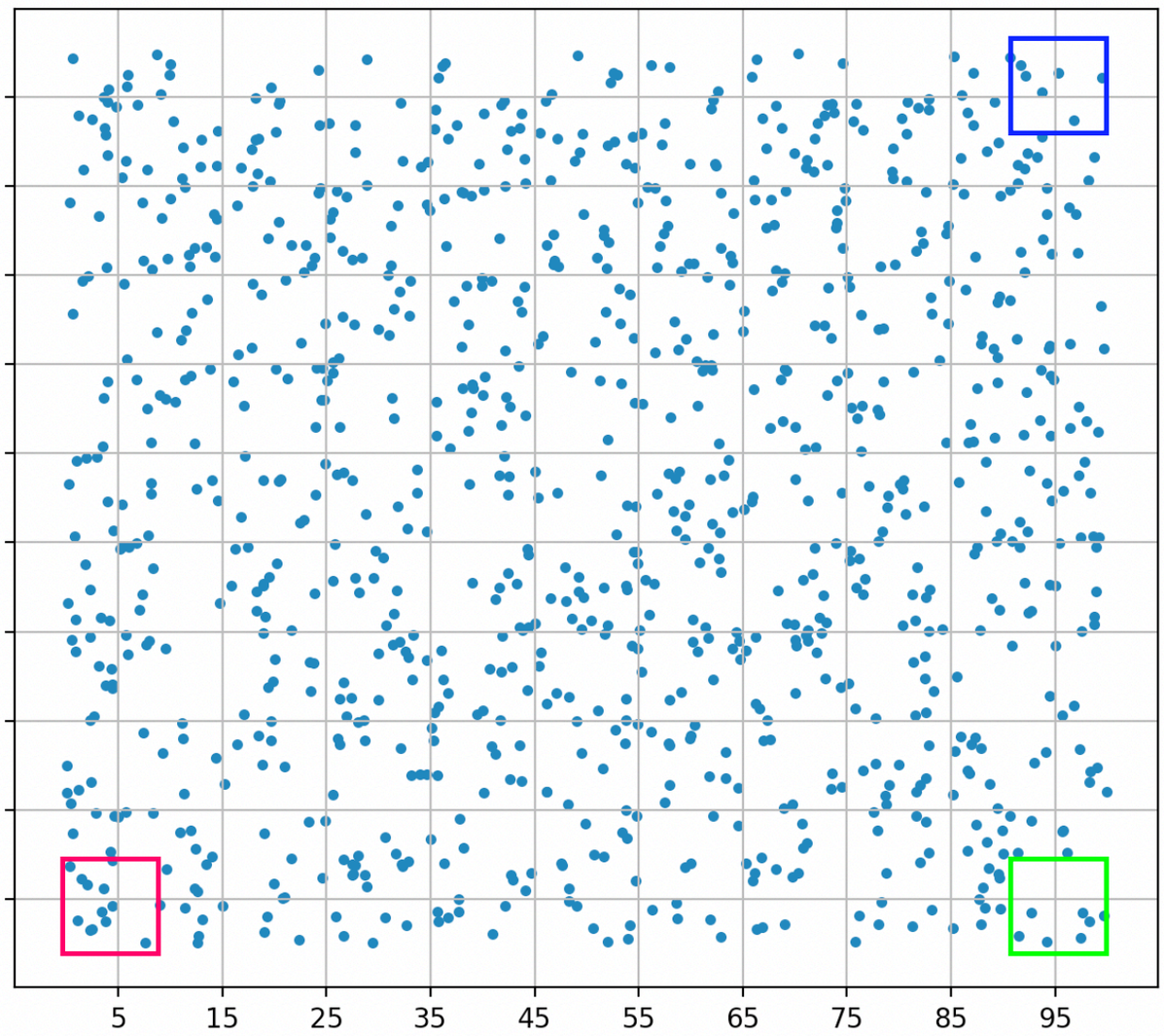
步驟九、測試3

test3

```
from shape2D2 import *
N = 1000
bin_size = 10
x_list, y_list = produce_sig(N)
bin = bin_points(bin_size)
plot_points(x_list,y_list,bin, bin)
x2bin, x_bin = my_hitogram(x_list,bin_size)
y2bin, y_bin = my_hitogram(y_list,bin_size)
joint =
my_joint_histogram(x_list,y_list,bin_size,bin,bin)
print(joint)
plt.show()
```

```
[[14. 11.  8. 17. 11.  8.  8.  8. 10. 16.]
 [12.  6. 11.  5.  8.  5. 10. 10. 13.  7.]
 [16.  4. 11.  7.  9.  9. 13. 11.  9.  8.]
 [11.  9.  9. 10.  5.  8. 10. 11. 12.  8.]
 [11.  7. 11. 18.  6.  9.  8. 10.  9. 12.]
 [11.  8. 12. 10. 14.  7. 11. 12. 12.  9.]
 [11.  7.  9. 14.  9.  7. 12. 11. 12.  8.]
 [ 6. 12.  7. 12. 12. 11. 11.  7. 12. 11.]
 [12. 16. 12. 12.  8. 10.  9. 11.  9. 12.]
 [ 7. 10.  6.  8. 13. 12. 10.  7. 12.  8.]]
```

joint_histogram
聯合累計圖

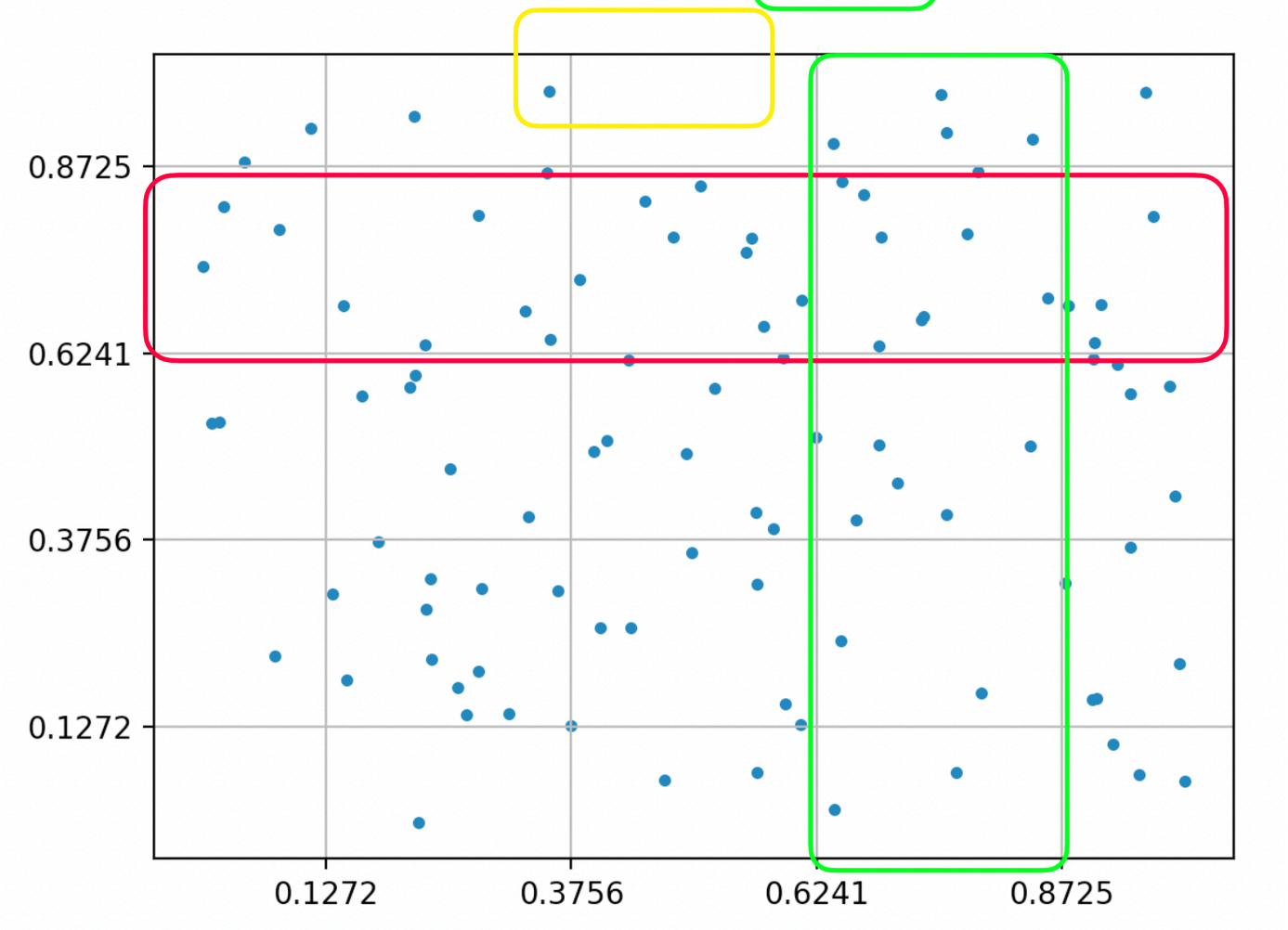


將聯合累計除以N就是聯合機率

$$[8 \quad 27 \quad 25 \quad 22 \quad 18]$$

$$\begin{bmatrix} 2 & 3 & 0 & 6 & 1 \\ 3 & 5 & 10 & 7 & 5 \\ 2 & 6 & 7 & 5 & 4 \\ 1 & 12 & 6 & 2 & 5 \\ 0 & 1 & 2 & 2 & 3 \end{bmatrix} \begin{bmatrix} 12 \\ 30 \\ 24 \\ 25 \\ 8 \end{bmatrix}$$

將個別累計除以N就是個別機率

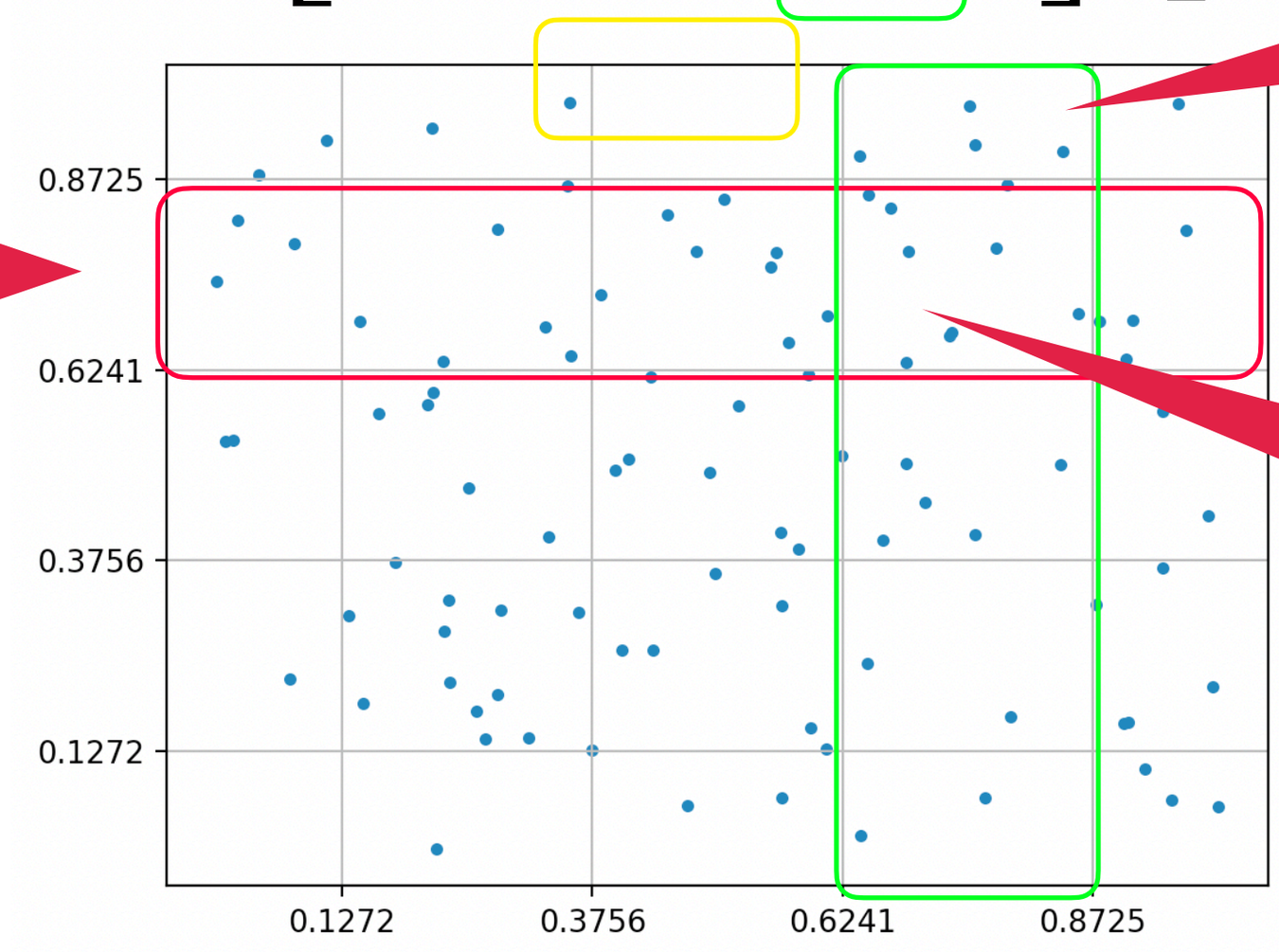


$$[8 \quad 27 \quad 25 \quad 22 \quad 18]$$

$$\begin{bmatrix} 2 & 3 & 0 & 6 & 1 \\ 3 & 5 & 10 & 7 & 5 \\ 2 & 6 & 7 & 5 & 4 \\ 1 & 12 & 6 & 2 & 5 \\ 0 & 1 & 2 & 2 & 3 \end{bmatrix} \begin{bmatrix} 12 \\ 31 \\ 24 \\ 25 \\ 8 \end{bmatrix}$$

個別機率
22/100

個別機率
31/100



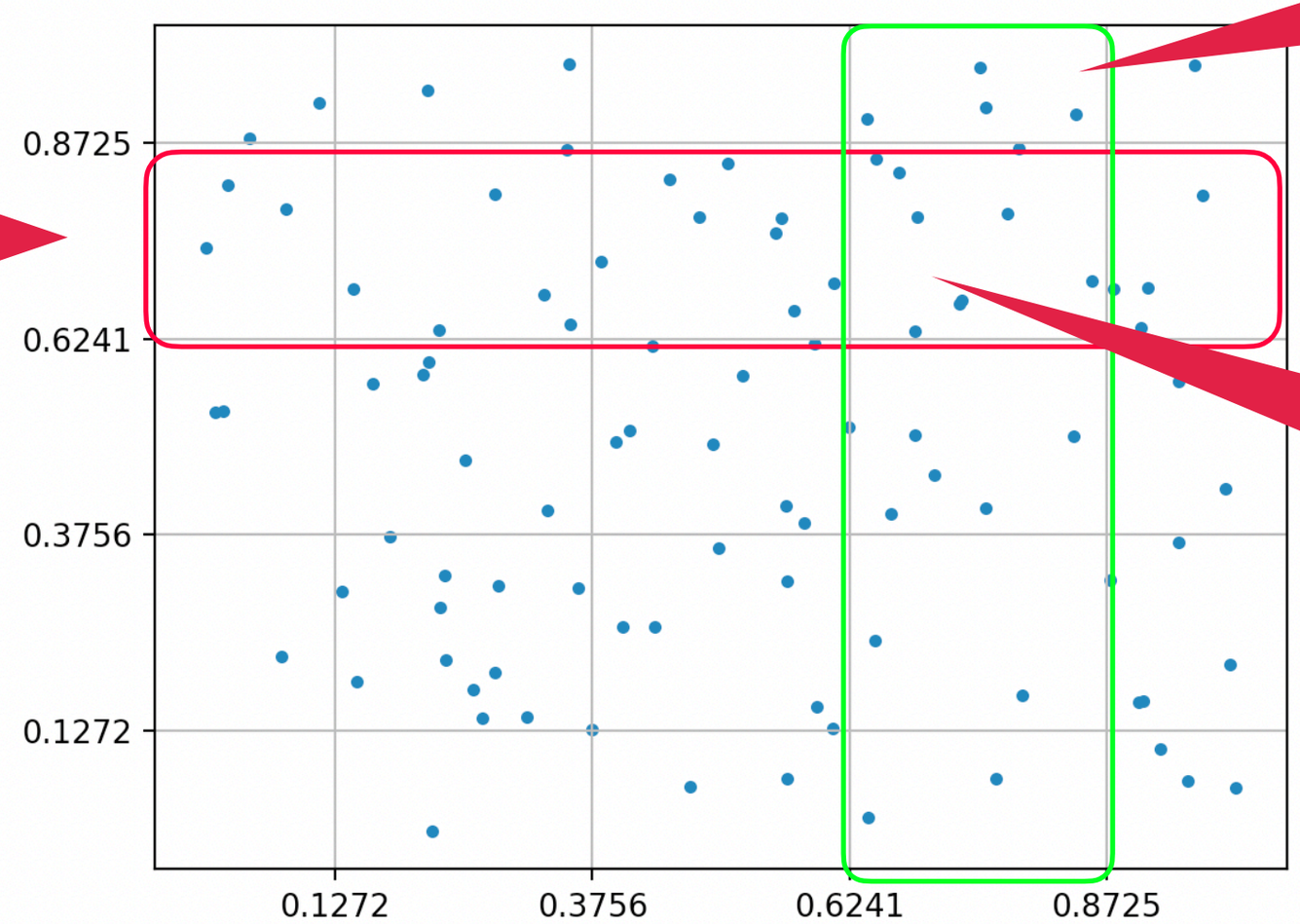
聯合機率
7/100

聯合機率與個別機率乘積的差距為：

$$D_{ij} = \frac{7}{100} log \frac{0.07}{0.31 * 0.22} = 0.0018$$

個別機率
22/100

個別機率
31/100



聯合
7/100

步驟十、計算所有D_{ii}與D

$$\mathbf{D} = \sum_{i,j} D_{ij}$$

圖形正不正的判斷：聯合機率與個別機率密乘積的差距越小越正

計算所有D_{ij}與D

```
def calculate_D(joint,x2bin,y2bin): # 4
    m = len(x2bin)
    n = len(y2bin)
    N = np.sum(x2bin)
    D = 0
    for i in range(m):
        for j in range(n):
            p = x2bin[i]/N*y2bin[j]/N
            q = joint[i,j]/N
            if abs(q) > 10**-6:
                D += q * np.log(q/p)
return D
```

步驟十一、圖形正不正分析

圖形正不正分析

```
def my_analysis(x_list, y_list, bin_size): # 5
    x2bin, x_bin = my_hitogram(x_list, bin_size)
    y2bin, y_bin = my_hitogram(y_list, bin_size)
    plot_points(x_list, y_list, x_bin, y_bin)
    joint = my_joint_histogram(x_list, y_list, bin_size,x_bin,y_bin)
    return  calculate_D(joint, x2bin, y2bin)
```

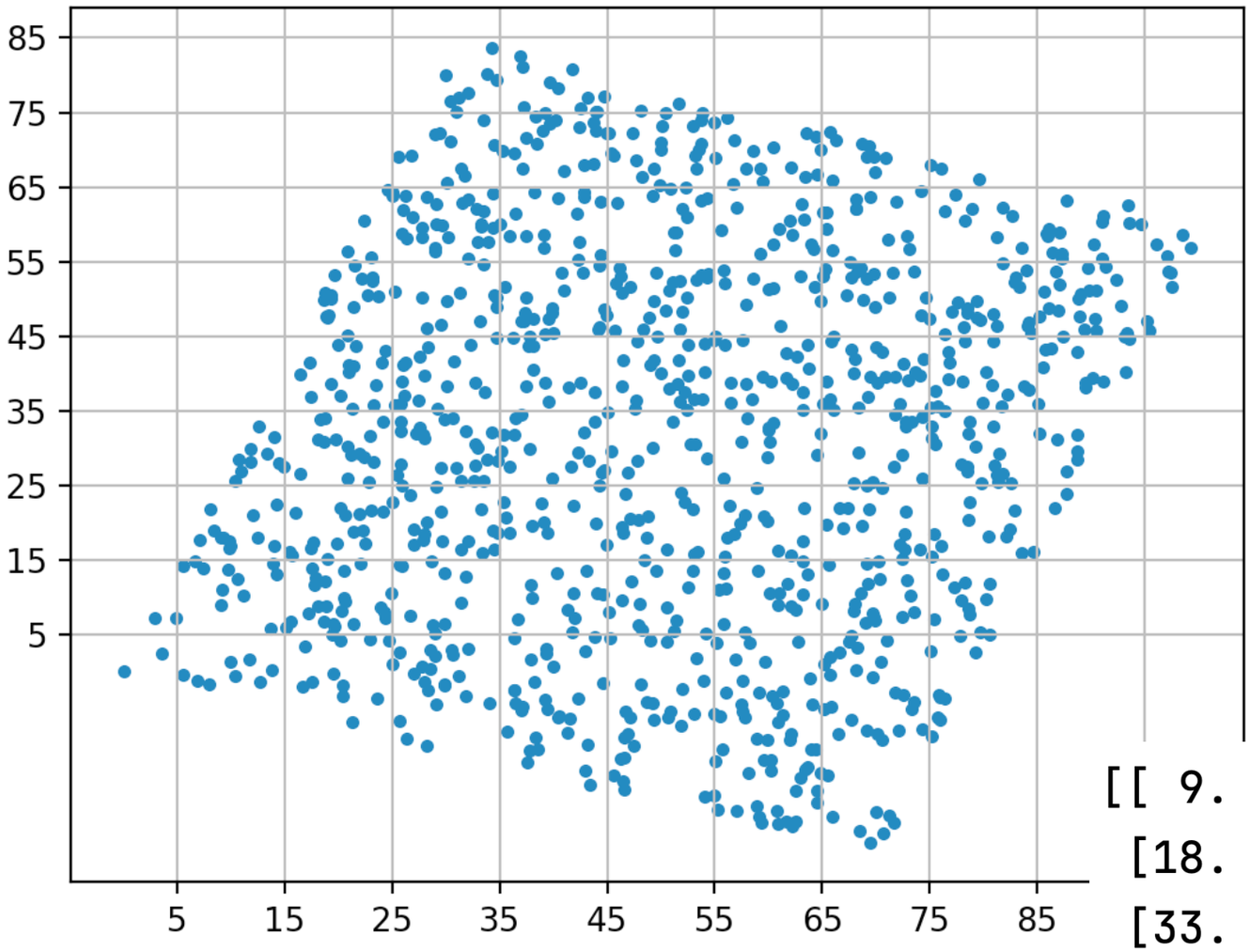
步驟十二、比較旋轉前後的D值，判斷獨立條件

```
def A_sig(A,x_list,y_list): # 2
    N = len(x_list)
    D = np.zeros((N,2))
    for i in range(N):
        D[i,0] = x_list[i]
        D[i,1] = y_list[i]
    DA = A @ np.transpose(D)
    return np.transpose(DA)
```

Test4

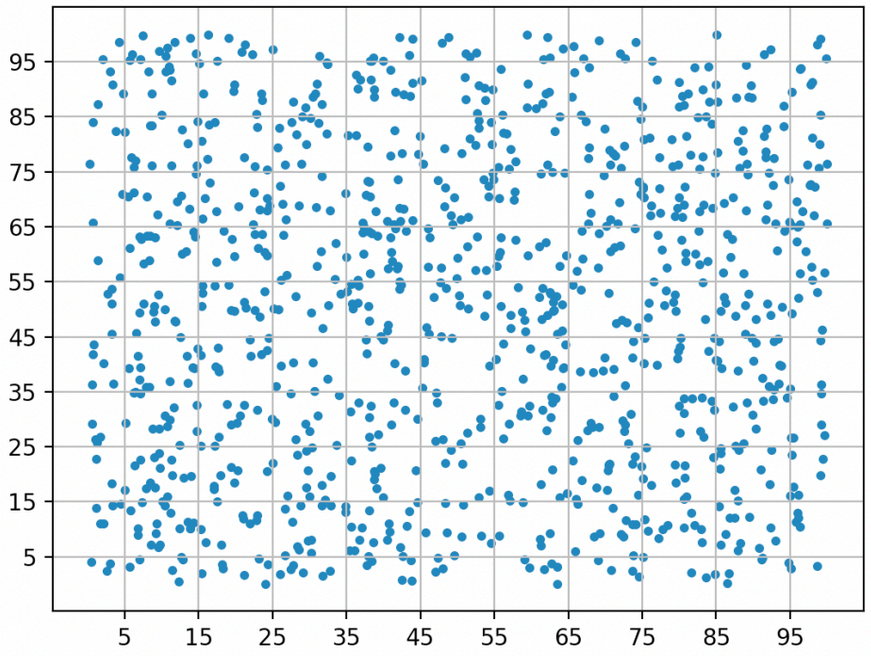
```
from shape2D2 import *
N = 1000
bin_size = 10
x_list, y_list = produce_sig(N)
A = np.matrix([[0.7,0.31],[-0.25,0.85]])
DA = A_sig(A,x_list,y_list)
x_list = list(DA[:,0])
y_list = list(DA[:,1])
```

```
bin = bin_points(bin_size)
plot_points(x_list,y_list,bin, bin)
x2bin, x_bin = my_hitogram(x_list,bin_size)
y2bin, y_bin = my_hitogram(y_list,bin_size)
joint =
my_joint_histogram(x_list,y_list,bin_size,bin,bin)
print(joint)
plt.show()
```



資料旋轉與聯合累計矩陣

```
[[ 9. 12.  1.  0.  0.  0.  0.  0.  0.  0.]
 [18. 16. 12. 10.  6.  4.  0.  0.  0.  0.]
 [33. 15. 20. 21. 14. 17. 12.  3.  0.  0.]
 [27. 11. 16. 15. 19. 17. 14. 16.  4.  0.]
 [35. 13. 14. 10. 14. 13. 17. 15.  1.  0.]
 [39. 13. 11. 18. 13. 17. 15. 11.  0.  0.]
 [54. 16.  9. 17. 13. 19. 17.  7.  0.  0.]
 [30. 15. 11. 24. 19.  7. 10.  0.  0.  0.]
 [ 2.  6. 12. 12. 17. 21.  3.  0.  0.  0.]
 [ 0.  0.  0.  2.  9. 12.  5.  0.  0.  0.]]
```

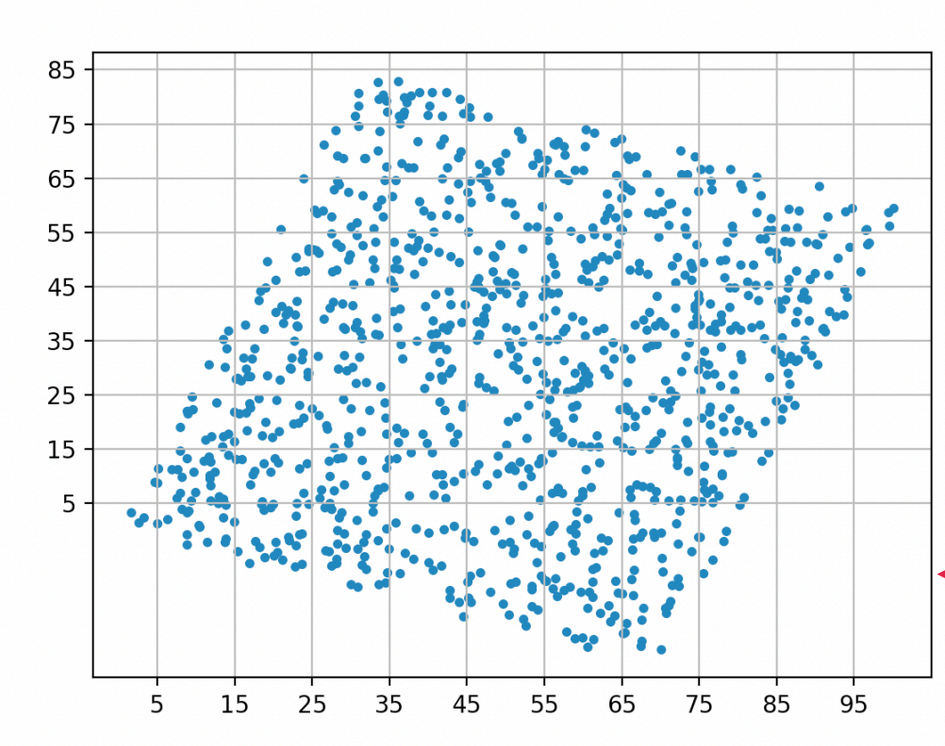


旋轉前

```
ffrom shape2D2 import *
N = 1000
bin_size = 10
x_list, y_list = produce_sig(N)
D1 = my_analysis(x_list, y_list, bin_size)
print("旋轉前 D值：", D1)
```

旋轉

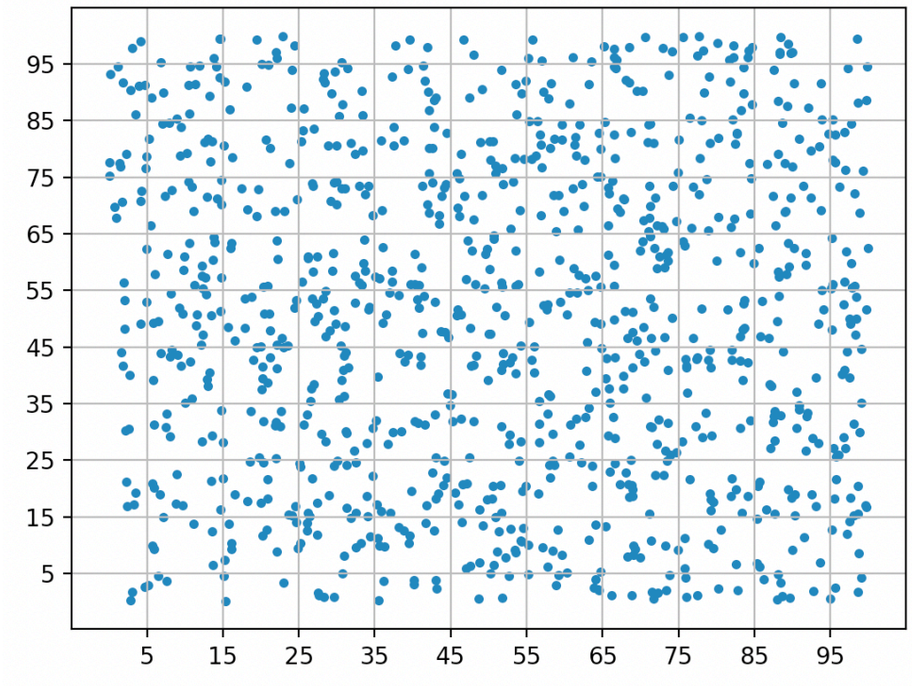
```
A = np.matrix([[0.7,0.31],[-0.25,0.85]])
DA = A_sig(A,x_list,y_list)
```



旋轉後

```
x_list = list(DA[:,0])
y_list = list(DA[:,1])
D2 = my_analysis(x_list, y_list, bin_size)
print("旋轉後 D值：", D2)

plt.show()
```

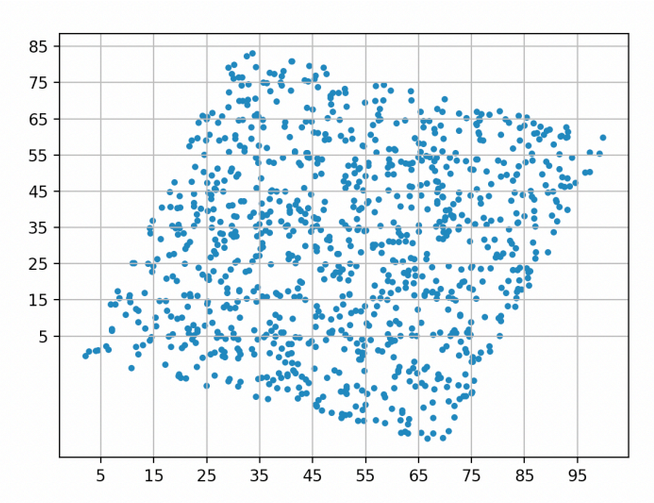
```
旋轉前 D值： 0.06137947421429748 6
旋轉後 D值： 0.17963488353293164
```



旋轉前

旋轉後的D值明顯上升，代表選轉後的兩個投影維度間的統計獨立性下降

旋轉



旋轉前 D值： 0.06137947421429748 6

旋轉後 D值： 0.17963488353293164

旋轉後