Colab Link: https://colab.research.google.com/drive/1GZwXkfEniAvxfUrbaM-HB6ciiY0IXuqV?usp=sharing

In [2]:
```
import random
import numpy as np
np.random.seed(10)
#a

N = 100
samples = []
def count_sample(data, sample):
    count = 0
    for d in data:
        if sample == d:
            count += 1
    return count
choices = [1, 2, 3, 4, 5, 6]

# Define the corresponding probabilities
probabilities = [0.25, 0.25, 0.125, 0.125, 0.125, 0.125]
for i in range(N):
    x = np.random.choice(choices, p=probabilities)
    y = np.random.choice(choices, p=probabilities)
    samples.append((x,y))
print(len(samples))

PMF = []
for i in range(1,7):
    PMF_x = []
    for j in range(1,7):
        PMF_x.append(count_sample(samples,(i,j))/ N)
    PMF.append(PMF_x)
PMF = np.array(PMF)
print(PMF)
print("==========================================")
#b
#Let X be the rows, Y be the columns
P_x = PMF.sum(axis=1)
F_x = np.cumsum(P_x)
#print(P_x)
#print(f"Fx = {F_x}")

P_y = PMF.sum(axis=0)
F_y = np.cumsum(P_y)
#print(f"Fy = {F_y}")
#Prove that F(x,y) = F(x)*F(y)

F_x_y = np.cumsum(np.cumsum(PMF, axis=1), axis=0)
# print(f"F(x,y) = {F_x_y}")

#change matrix to matrix of distance F(x,y) - F(x)*F(y)
F_x_rs = F_x.reshape(6,1)
F_y_rs = F_y.reshape(1,6)
F_x_mult_y = np.dot(F_x_rs, F_y_rs)
# print(f"F(x)*F(y) = {F_x_mult_y}")

distance_matrix = F_x_y - F_x_mult_y
distance_matrix = np.abs(distance_matrix)
# print(distance_matrix)

#Compute the mean
D = np.mean(distance_matrix)
print(f"mean difference of F(x,y)-F(x)*F(y) = {D}")
#X & Y are independent as the mean of F(x,y)-F(x)*F(y) approaches 0

print("==========================================")
#c
Z = []
for s in samples:
    Z1 = s[0]+s[1]
    Z2 = s[0]-s[1]
    Z.append((Z1,Z2))
#possible value of Z1: 2,3,4,5,6,7,8,9,10,11,12
#possible value of Z2: -5,-4,-3,-2,-1,0,1,2,3,4,5
#Create joint empirical PMF table of Z1,Z2
#Index 0,1,2,... represents Z1 = 2,3,4,.../ Z2=-5,-4,-3,...
Z_PMF = []
for i in range(2,13):
    Z_PMF_x = []
    for j in range(-5,6):
        Z_PMF_x.append(count_sample(Z,(i,j))/ N)
    Z_PMF.append(Z_PMF_x)
Z_PMF = np.array(Z_PMF)
# print(Z_PMF)

P_z1 = Z_PMF.sum(axis=1)
```

```
F_z1 = np.cumsum(P_z1)


P_z2 = Z_PMF.sum(axis=0)
F_z2 = np.cumsum(P_z2)


F_z1_z2 = np.cumsum(np.cumsum(Z_PMF, axis=1), axis=0)
# print(f"F(z1,z2) = {F_z1_z2}")

F_z1_rs = F_z1.reshape(11,1)
F_z2_rs = F_z2.reshape(1,11)
F_z1_mult_z2 = np.dot(F_z1_rs, F_z2_rs)
# print(f"F(z1)*F(z2) = {F_z1_mult_z2}")

distance_matrix = F_z1_z2 - F_z1_mult_z2
distance_matrix = np.abs(distance_matrix)
# print(distance_matrix)

#Compute the mean
D = np.mean(distance_matrix)
print(f"Mean difference of F(z1,z2)-F(z1)*F(z2) = {D}")
#It can be evaluated that F(x,y)-F(x)*F(y) < F(z1,z2)-F(z1)*F(z2), which is expected as x,y are independent, and z1,z2 is not
```

```
100
[[0.05 0.11 0.02 0.03 0.05 0.02]
 [0.1  0.04 0.05 0.02 0.03 0.02]
 [0.03 0.02 0.06 0.01 0.05 0.02]
 [0.03 0.01 0.02 0.01 0.01 0.01]
 [0.03 0.01 0.03 0.02 0.01 0.  ]
 [0.   0.05 0.   0.01 0.01 0.01]]
================================================
mean difference of F(x,y)-F(x)*F(y) = 0.006594444444444483
================================================
Mean difference of F(z1,z2)-F(z1)*F(z2) = 0.01211487603305789
```

In [ ]:
```
%%shell
 jupyter nbconvert --to html /content/assignment1_Q1_2.ipynb
```