STATS210 Computer Lab Report #2

Shuyi Guan sg466 1447

1. Estimate joint and marginal PMFs

a. Mathematical approach:

	j=0	j=1
i=0	1/8	1/8
i=1	1/4	1/2

Table 1: Joint PMF values

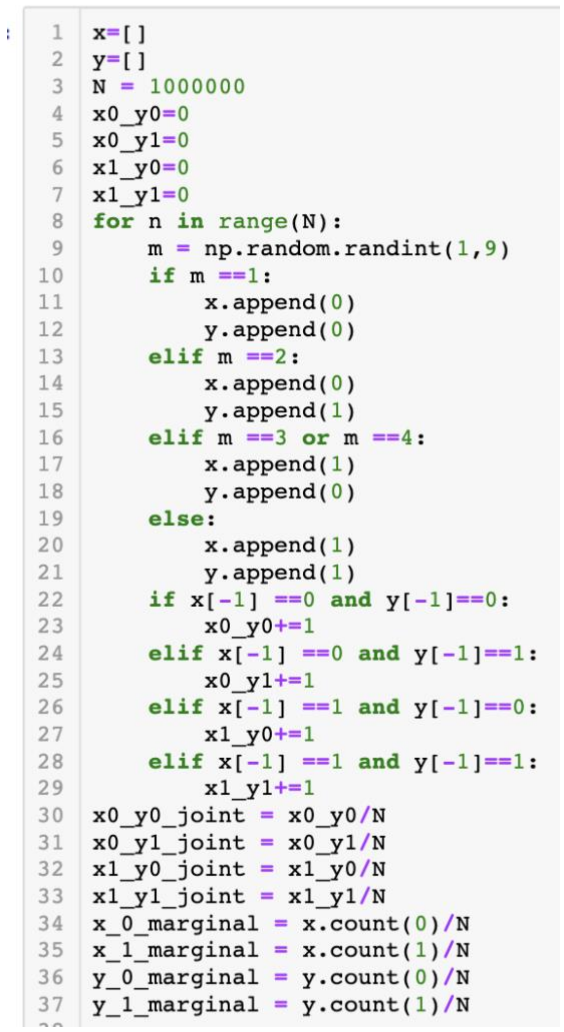
From the given table, we can get that the marginal PMF for x and y is

$$P_X(x) = \Sigma_y P_{XY}(x,y) = \begin{cases} \frac{1}{4}, x = 0 \\ \frac{3}{4}, x = 1 \end{cases} \quad \text{and} \quad P_Y(y) = \Sigma_x P_{XY}(x,y) = \begin{cases} \frac{3}{8}, y = 0 \\ \frac{5}{8}, y = 1 \end{cases}$$

The joint PMF is

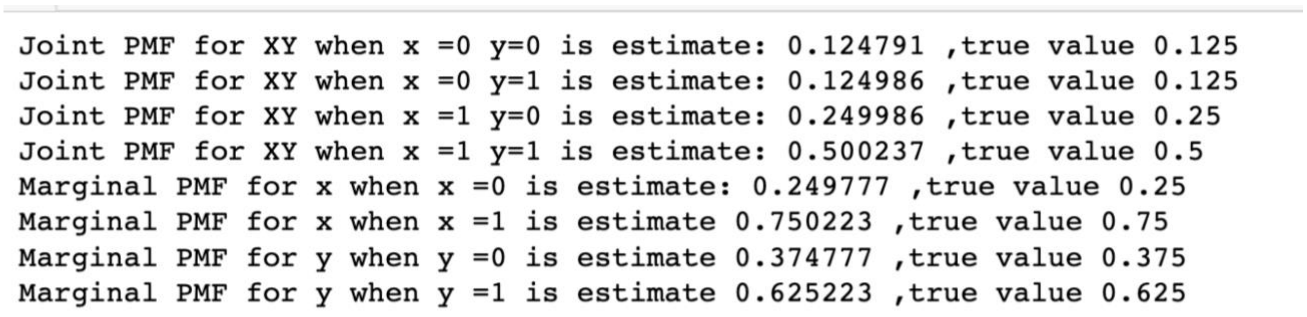
$$P_{XY}(x,y) = \begin{cases} \frac{1}{8}, x = 0, y = 0 \\ \frac{1}{8}, x = 0, y = 1 \\ \frac{1}{4}, x = 1, y = 0 \\ \frac{1}{2}, x = 1, y = 1 \end{cases}$$

b. Simulation explanation: We can easily find that x and y are not independent variables, so we will start the simulation by setting the joint PMFs. Let m be a random number from 1 to 8. According to the joint PMF, we can say that, if m =1, the event is (x=0,y=0), if m = 2, the event is (x=0,y=1) , if m = 3 or 4, the event is (x=1,y=0) , if m = 5 to 8, the event is (x=1,y=1). Then, we will calculate the respective PMFs through counting how many 0 or 1 are in lists x and y, and then divide it by the number of trials N.



```
x=[]
y=[]
N = 1000000
x0_y0=0
x0_y1=0
x1_y0=0
x1_y1=0
for n in range(N):
    m = np.random.randint(1,9)
    if m ==1:
        x.append(0)
        y.append(0)
    elif m ==2:
        x.append(0)
        y.append(1)
    elif m ==3 or m ==4:
        x.append(1)
        y.append(0)
    else:
        x.append(1)
        y.append(1)
    if x[-1] ==0 and y[-1]==0:
        x0_y0+=1
    elif x[-1] ==0 and y[-1]==1:
        x0_y1+=1
    elif x[-1] ==1 and y[-1]==0:
        x1_y0+=1
    elif x[-1] ==1 and y[-1]==1:
        x1_y1+=1
x0_y0_joint = x0_y0/N
x0_y1_joint = x0_y1/N
x1_y0_joint = x1_y0/N
x1_y1_joint = x1_y1/N
x_0_marginal = x.count(0)/N
x_1_marginal = x.count(1)/N
y_0_marginal = y.count(0)/N
y_1_marginal = y.count(1)/N
```

c. Results and comparison: We can see that the estimated marginal PMF and joint PMF are very similar to the true values.



```
Joint PMF for XY when x =0 y=0 is estimate: 0.124791 ,true value 0.125
Joint PMF for XY when x =0 y=1 is estimate: 0.124986 ,true value 0.125
Joint PMF for XY when x =1 y=0 is estimate: 0.249986 ,true value 0.25
Joint PMF for XY when x =1 y=1 is estimate: 0.500237 ,true value 0.5
Marginal PMF for x when x =0 is estimate: 0.249777 ,true value 0.25
Marginal PMF for x when x =1 is estimate 0.750223 ,true value 0.75
Marginal PMF for y when y =0 is estimate 0.374777 ,true value 0.375
Marginal PMF for y when y =1 is estimate 0.625223 ,true value 0.625
```

2. Determine correlation coefficient

a. Mathematical approach: Using the definition of correlation coefficient, we can calculate that the correlation coefficient as follows

$$E(XY) = 1 \times 1 \times \frac{1}{2} = \frac{1}{2}$$

$$\mu_x = 0 \times \frac{1}{4} + 1 \times \frac{3}{4} = \frac{3}{4} \; and \; \mu_y = 0 \times \frac{3}{8} + 1 \times \frac{5}{8} = \frac{5}{8}$$

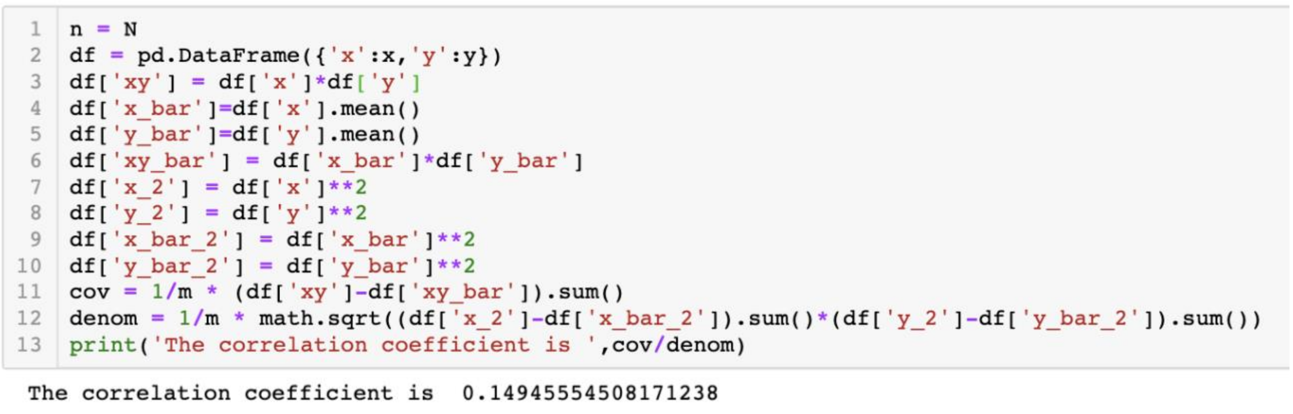
$$Var(X) = E(X^2) - E(X)^2 = \frac{3}{4} \cdot 1 - \left(\frac{3}{4} \cdot 1\right)^2 = \frac{3}{16}$$

$$Var(Y) = E(Y^2) - E(Y)^2 = \frac{5}{8} \cdot 1 - \left(\frac{5}{8} \cdot 1\right)^2 = \frac{15}{64}$$

$$\sigma_x \sigma_y = \sqrt{\frac{3}{16} \times \frac{15}{64}} = \frac{3\sqrt{5}}{32}$$

$$\hat{\rho}_{X,Y} = \frac{Cov(X,Y)}{\sqrt{Var(X)Var(Y)}} = \frac{E(XY) - \mu_x \mu_y}{\sigma_x \sigma_y} = \frac{\frac{1}{2} - \frac{3}{4} \times \frac{5}{8}}{\frac{3\sqrt{5}}{32}} = \frac{1}{3\sqrt{5}} = 0.14925$$

b. <u>Simulation explanation:</u> We will use the simulation process in problem 1. Next we can just calculate the correlation coefficient. For the simplicity of calculation, we will use a DataFrame to store the calculated data. By using the pre-installed calculation commands in Pandas DataFrame, we can quickly calculate the correlation coefficient. Explanation on labels: ‘x_bar’ and ‘y_bar’ means the mean values of x and y; ‘x_bar_2’ and ‘y_bar_2’ means the square of x_bar and y_bar.



```
n = N
df = pd.DataFrame({'x':x,'y':y})
df['xy'] = df['x']*df['y']
df['x_bar']=df['x'].mean()
df['y_bar']=df['y'].mean()
df['xy_bar'] = df['x_bar']*df['y_bar']
df['x_2'] = df['x']**2
df['y_2'] = df['y']**2
df['x_bar_2'] = df['x_bar']**2
df['y_bar_2'] = df['y_bar']**2
cov = 1/m * (df['xy']-df['xy_bar']).sum()
denom = 1/m * math.sqrt((df['x_2']-df['x_bar_2']).sum()*(df['y_2']-df['y_bar_2']).sum())
print('The correlation coefficient is ',cov/denom)
```

```
The correlation coefficient is  0.14945554508171238
```

c. <u>Results and comparison:</u> The simulated result is 0.14945, which is very similar to the mathematically calculated one 0.14925.

3. Sum of two geometric RVs

a. Mathematical approach: Given RVs X~geom(p), Y~geom(p), Prove that the PMF of Z=X+Y is given by $P_z(k) = p^2(k-1)(1-p)^{k-2}, k = 2,3, \dots$

Consider the meaning of geometric distribution, which is the number of Bernoulli trials until the first success occurs, thus, the meaning of Z is the count of trials until the second success, which means there are (k-1) ways to arrange the first success. An intuitive way to get the PMF is thus, $P_z(k) = (k-1)p^2(1-p)^{k-2}, k = 2,3, \dots$

Detailed proof:

We know that $P_X(x) = p(1-p)^{x-2}\ , x = 1, 2, 3 \dots\ , P_Y(y) = p(1-p)^{y-2}\ , y = 1, 2, 3 \dots$

We have,

$$\begin{aligned} P_z(k) &= P[X + Y = k] \\ &= \Sigma_{\mathrm{m=1}}^{k-1} P[X = m]P[Y = k - m] \\ &= \Sigma_{\mathrm{m=1}}^{k-1}\ p(1-p)^{m-1} \cdot p(1-p)^{k-m-1} \\ &= \ \Sigma_{\mathrm{m=1}}^{k-1} p^2(1-p)^{k-2} \\ &= (\Sigma_{\mathrm{m=1}}^{k-1} 1) \cdot p^2(1-p)^{k-2} \\ &= (k-1)p^2(1-p)^{k-2}\ , k = 2,3 \dots \end{aligned}$$

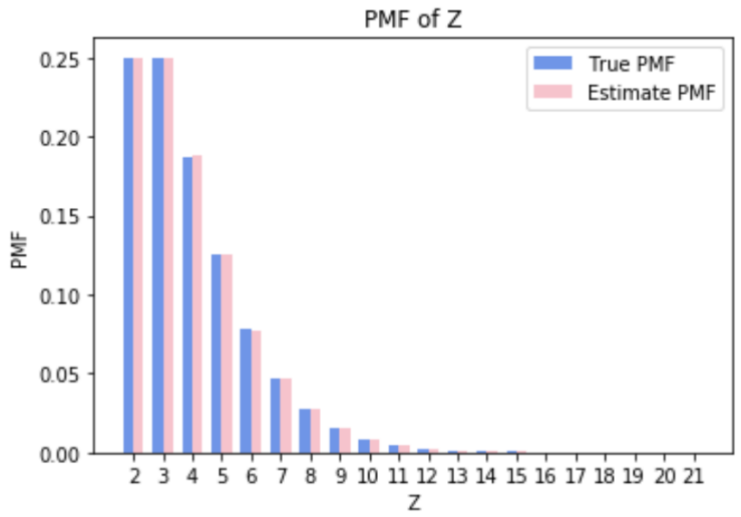
b. Simulation explanation: First, we will use the random.geometric() function to generate realizations of X and Y for N times. Then, we add them together and get Z = X+Y. Calculate PMF (for k = 2,3… 101): for every number in [2,101], count how many times the number appeared in Z and then divided by N. Append that number to Z_pmf, which represents the pmf of Z for a specific input number. Then, plot Z_pmf and Z_in_formula. Z_in_formula is the mathematically deduced formula for Z=X+Y, which is proved in part a.

```python
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
p=1/2
N=500000
X = np.random.geometric(p,N)
Y = np.random.geometric(p,N)
Z = X+Y
n=20
Z_pmf=[]
for i in range(2,n+2):
    Z_pmf.append((Z == i).sum() / N)
k = np.arange(2,n+2,1)
Z_in_formula = (p**2)*(k-1)*((1-p)**(k-2))
labels =k

# plot
x = np.arange(2,n+2,1)
width = 0.35  # the width of the bars
fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, Z_in_formula, width, label='True PMF',color='cornflowerblue')
rects2 = ax.bar(x + width/2, Z_pmf, width, label='Estimate PMF',color='pink')

plt.xlabel('x')
plt.ylabel('P(x)')
plt.title('PMF of Z')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend()
plt.show()
```

c. Results and comparison: we can see that these two graphs are very much similar, which proves that this is how we calculate the PMF of Z=X+Y, X and Y are both geometric RVs.

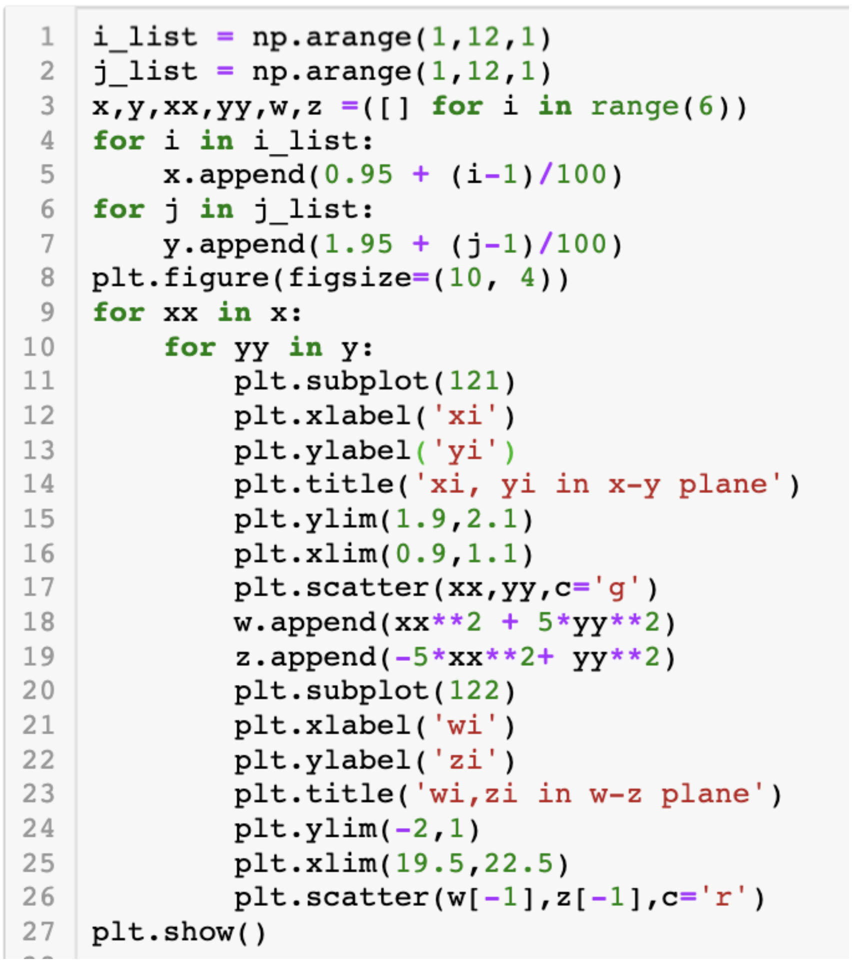


4. Nonlinear transformation

a. Mathematical approach: Calculate the Jacobian factor $\left|\det\left(\frac{\partial(w,z)}{\partial(x,y)}\right)\right|$ at x =1, y=2:

$$\left|\det\left(\frac{\partial(w,z)}{\partial(x,y)}\right)\right| = \begin{vmatrix} 2x & 10y \\ -10x & 2y \end{vmatrix} = \begin{vmatrix} 2 & 20 \\ -10 & 4 \end{vmatrix} = 208$$

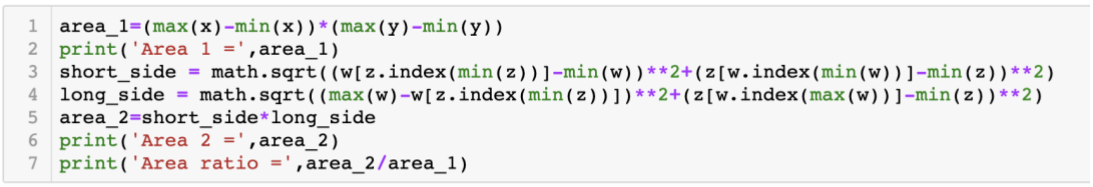
b. <u>Simulation explanation</u>: First, we will plot (x_i, y_j) points on the xy-plane, where $x_i = 0.95 + \frac{i-1}{100}, y_j = 1.95 + \frac{j-1}{100}$ for $i, j = 1,2,\dots,11$. Use *x* and *y* to store these x values and y values. Then use a loop to plot all x,y combinations to the graph, these points are in green. Then, we will transform these points into wz-plane, where $w = x^2 + 5y^2, z = -5x^2 + y^2$, storing all w values in *w* and all z values in *z.* By setting the xlim and ylim we can make sure the unit length on both axes are the same, thus provide a more accurate visual representation of what these two figures look like. We can roughly estimate that both these figures are rectangles. Then we can calculate the area by multiplying the sides.



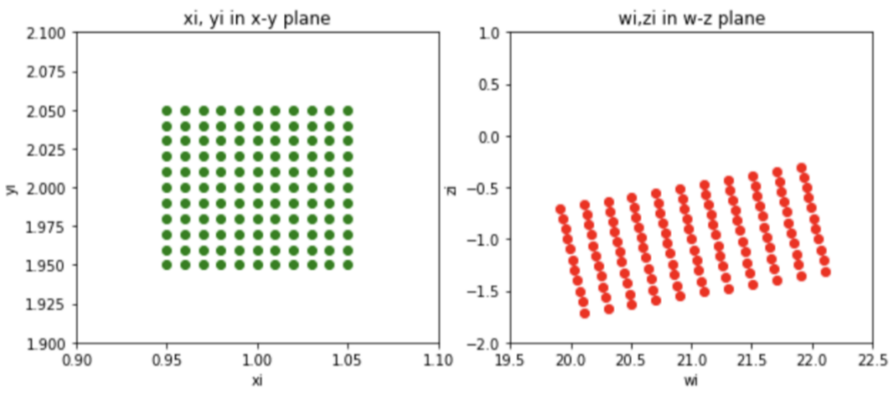
```python
i_list = np.arange(1,12,1)
j_list = np.arange(1,12,1)
x,y,xx,yy,w,z =([] for i in range(6))
for i in i_list:
    x.append(0.95 + (i-1)/100)
for j in j_list:
    y.append(1.95 + (j-1)/100)
plt.figure(figsize=(10, 4))
for xx in x:
    for yy in y:
        plt.subplot(121)
        plt.xlabel('xi')
        plt.ylabel('yi')
        plt.title('xi, yi in x-y plane')
        plt.ylim(1.9,2.1)
        plt.xlim(0.9,1.1)
        plt.scatter(xx,yy,c='g')
        w.append(xx**2 + 5*yy**2)
        z.append(-5*xx**2+ yy**2)
        plt.subplot(122)
        plt.xlabel('wi')
        plt.ylabel('zi')
        plt.title('wi,zi in w-z plane')
        plt.ylim(-2,1)
        plt.xlim(19.5,22.5)
        plt.scatter(w[-1],z[-1],c='r')
plt.show()
```

Calculate Areas:

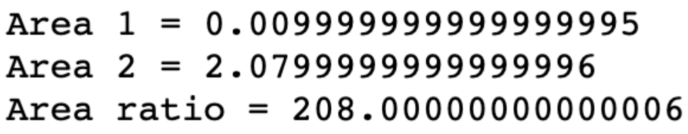
Area_1 is easy to get, since we stored all x, y values in list, we can easily get the length of each side, and then multiply them. Similarly, we can get area_2 by multiplying its long side and short side. But for area_2 the lines are not parallel to the x,y axes, so we need to use math.sqrt() to calculate the sides according to the Pythagorean theorem.



```python
area_1=(max(x)-min(x))*(max(y)-min(y))
print('Area 1 =',area_1)
short_side = math.sqrt((w[z.index(min(z))]-min(w))**2+(z[w.index(min(w))]-min(z))**2)
long_side = math.sqrt((max(w)-w[z.index(min(z))])**2+(z[w.index(max(w))]-min(z))**2)
area_2=short_side*long_side
print('Area 2 =',area_2)
print('Area ratio =',area_2/area_1)
```



c. Results and comparison: The estimated area ratio is 208.00000000000006, which is very similar to the calculated Jacobian factor of 208.

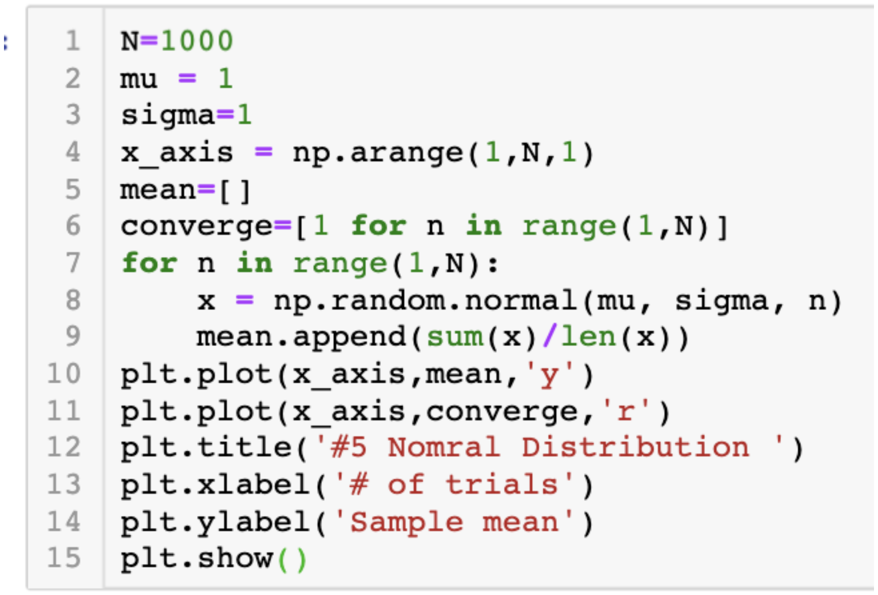


```
Area 1 = 0.009999999999999995
Area 2 = 2.0799999999999996
Area ratio = 208.00000000000006
```

5. Normal Distribution I.I.D. convergence

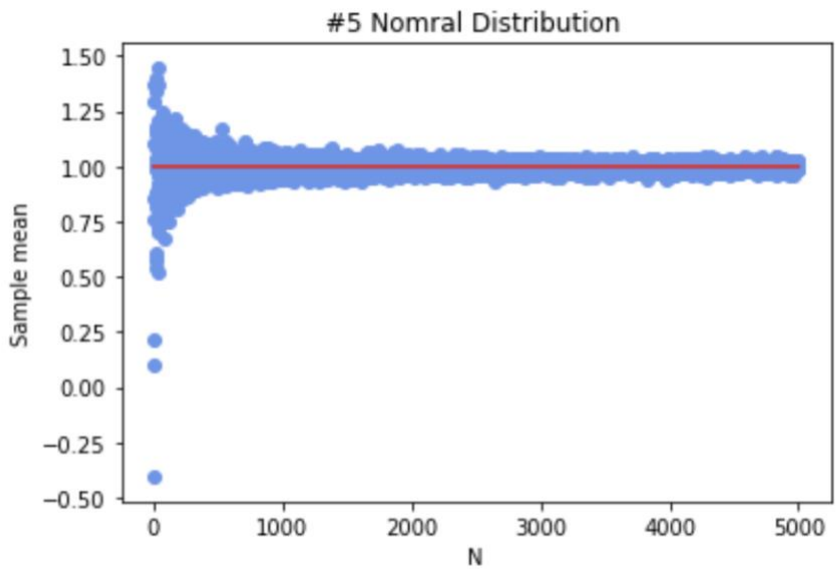
a. Mathematical approach: According to the Law of large numbers, as the sample size approaches infinity, the center of the distribution of the sample means becomes very close to the population mean. Since the mean of normal distribution (mu =1, sigma=1) is 1, as the sample size increases, the sample mean converges to 1.

b. Simulation explanation: Use *np.random.normal(mu,sigma,n)* to generate n normal distributed random variables, calculate the mean using sum/len, and we can easily plot the sample mean versus N.



```
N=1000
mu = 1
sigma=1
x_axis = np.arange(1,N,1)
mean=[]
converge=[1 for n in range(1,N)]
for n in range(1,N):
    x = np.random.normal(mu, sigma, n)
    mean.append(sum(x)/len(x))
plt.plot(x_axis,mean,'y')
plt.plot(x_axis,converge,'r')
plt.title('#5 Nomral Distribution ')
plt.xlabel('# of trials')
plt.ylabel('Sample mean')
plt.show()
```

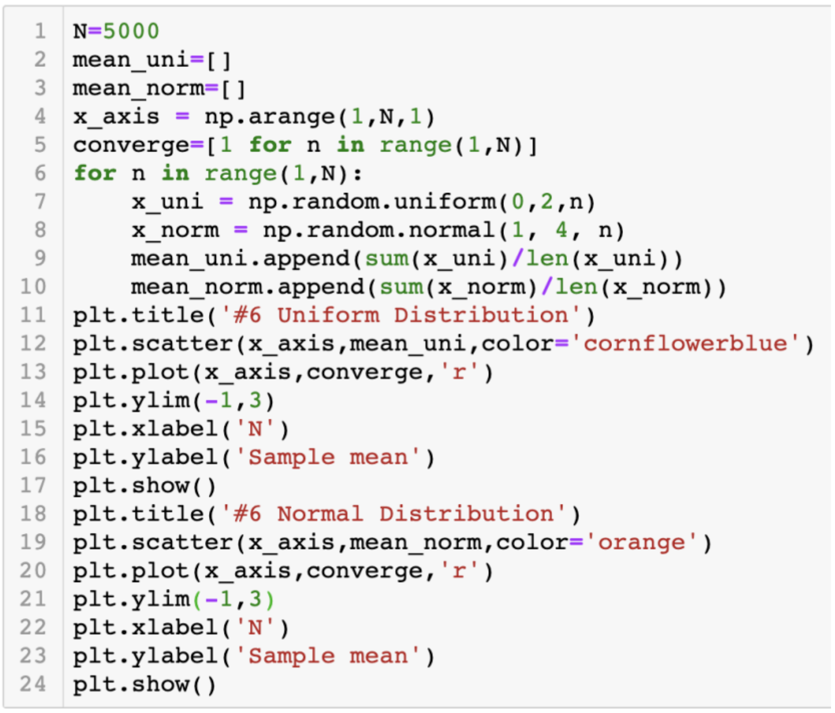
c. Results and comparison: We can see from the plot that it converges to 1.



6. Normal and Uniform Distribution I.I.D. convergence

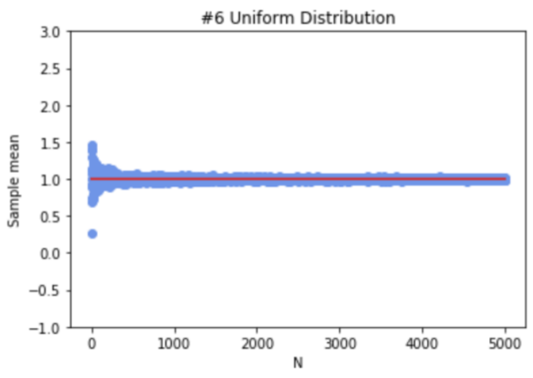
a. Mathematical approach: According to the Law of large numbers, as the sample size approaches infinity, the center of the distribution of the sample means becomes very close to the population mean. The mean of $U(0,2) = \frac{0+2}{2} = 1$, and $N(1,4) = 1$. The variance for $U(0,2) = \frac{2^2-0^2}{12} = \frac{1}{3}$, and $N(1,4) = 4$. Since the variance of the Uniform distribution (0,2) is smaller than the variance of $N(1,4)$, the Uniform distribution converges faster.

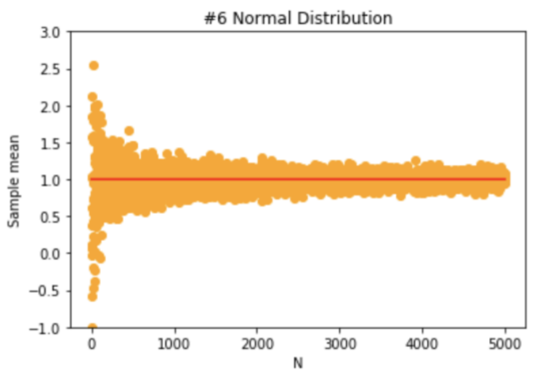
b. <u>Simulation explanation:</u> Use *np.random.normal(1,4,n)* to generate n normal distributed random variables, and use *np.random.uniform(0,2,n)* to generate n uniform distributed random variables, we can calculate the respective means using sum/len. Then, plot the sample mean versus N.



```python
N=5000
mean_uni=[]
mean_norm=[]
x_axis = np.arange(1,N,1)
converge=[1 for n in range(1,N)]
for n in range(1,N):
    x_uni = np.random.uniform(0,2,n)
    x_norm = np.random.normal(1, 4, n)
    mean_uni.append(sum(x_uni)/len(x_uni))
    mean_norm.append(sum(x_norm)/len(x_norm))
plt.title('#6 Uniform Distribution')
plt.scatter(x_axis,mean_uni,color='cornflowerblue')
plt.plot(x_axis,converge,'r')
plt.ylim(-1,3)
plt.xlabel('N')
plt.ylabel('Sample mean')
plt.show()
plt.title('#6 Normal Distribution')
plt.scatter(x_axis,mean_norm,color='orange')
plt.plot(x_axis,converge,'r')
plt.ylim(-1,3)
plt.xlabel('N')
plt.ylabel('Sample mean')
plt.show()
```

c. <u>Results and Comparison:</u> Set the y-limit from -1 to 3, and we can see from the plot that the uniform distribution converges faster.


#6 Uniform Distribution
Sample mean
N


#6 Normal Distribution
Sample mean
N