

▼ Laboratorio No. 3

- Paola De León 20361
- Gabriela Contreras 20213

```

1 #!pip install numpy-financial

1 import numpy as np
2 import random as rand
3 import scipy.stats as stats
4 import matplotlib.pyplot as plt
5
6 import math
7 import random
8 import numpy as np
9 import sympy as sp
10 from scipy.stats import gamma
11 import numpy_financial as npf
12

```

▼ Parte No.2 (Ejercicios sobre Números Aleatorios)

```

1 def ej2_1():
2     # Ejercicio 2: task 1
3     # Distribución gamma
4
5     x = np.linspace(0, 20, 1000)
6     lambdas = [2, 1, 0.5]
7     k = 3
8
9     plt.figure(figsize=(10, 6))
10    for lambdaVal in lambdas:
11        alpha = k
12        beta = 1 / lambdaVal
13
14        y = gamma.pdf(x, alpha, scale=1/beta)
15        labelVal = ("Valor de lambda: ", lambdaVal)
16        plt.plot(x, y, label=labelVal)
17
18    plt.xlabel("Tiempo de Espera")
19    plt.ylabel("Densidad de Probabilidad")
20    plt.title("Distribuciones Gamma para diferentes lambdas")
21    plt.legend()
22    plt.grid(True)
23    plt.show()
24
25    print(
26        '''
27        ¿Qué conclusiones puede obtener de las gráficas obtenidas en términos de los tiempos de espera
28        y el número de ocurrencias del evento? ¿Qué relación existe entre el tiempo de espera y el número
29        de ocurrencias de un evento?
30
31        En tanto al tiempo de espera se puede decir que existe una relación inversa en donde cuando el
32        valor de lambda aumente, el tiempo de espera promedio disminuye y la dispersión de estos tiempos
33        será menor. Por lo tanto, el valor lambda es un factor determinante para tiempo de espera.
34
35        ''')
36
37 # Ejercicio 2: task 2
38 def generador1(n:int):
39     x = 1
40     m = 2**35 - 1
41     numbers = []
42
43     for i in range(n):
44         x = 5**5 * x % m
45         numbers.append(x/m)
46
47     return numbers
48
49 def generador2(n:int):
50     x = 1
51     m = 2**35 - 1
52     numbers = []
53
54     for i in range(n):
55         x = 5**5 * x % m
56         numbers.append(x/m)
57
58     return numbers
59

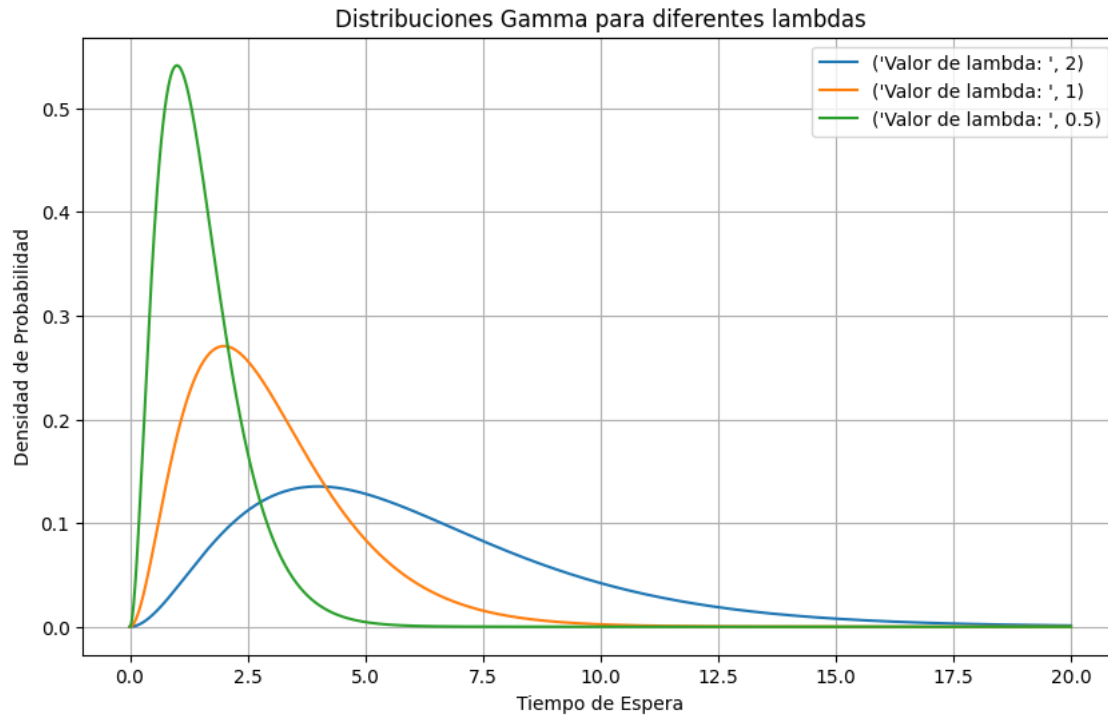
```

```

50     x = 1
51     m = 2**31 - 1
52     numbers = []
53
54     for i in range(n):
55         x = 7**5 * x % m
56         numbers.append(x/m)
57
58     return numbers
59
60 def generador3(n:int):
61     numbers = []
62
63     for i in range(n):
64         numbers.append(rand.random())
65
66     return numbers
67
68 def ej2_2():
69     repetitions = [100, 5000, 100000]
70
71     for i, rep in enumerate(repetitions):
72         dataGenerador1 = generador1(n=rep)
73         dataGenerador2 = generador2(n=rep)
74         dataGenerador3 = generador3(n=rep)
75
76         # Create the histogram in the corresponding subplot
77         plt.subplot(3, 3, i+1)
78         num_bins = 10
79         plt.hist(dataGenerador1, bins=num_bins, range=(0, 1), edgecolor='black')
80         plt.title(f'Generador 1 con {rep} repeticiones')
81         plt.xlabel('Valor')
82         plt.ylabel('Frecuencia')
83
84         plt.subplot(3, 3, i+4)
85         plt.hist(dataGenerador2, bins=num_bins, range=(0, 1), edgecolor='black')
86         plt.title(f'Generador 2 con {rep} repeticiones')
87         plt.xlabel('Valor')
88         plt.ylabel('Frecuencia')
89
90         plt.subplot(3, 3, i+7)
91         plt.hist(dataGenerador3, bins=num_bins, range=(0, 1), edgecolor='black')
92         plt.title(f'Generador 3 con {rep} repeticiones')
93         plt.xlabel('Valor')
94         plt.ylabel('Frecuencia')
95
96     plt.tight_layout()
97     plt.show()
98
99     print (
100         '''
101         ¿Qué generador le parece mejor? ¿Por qué?
102         El primer generador nos parece mejor pues cuenta con una distribución bastante similar por lo que
103         no existen valores que se repitan o mantengan en el mismo rango. Permitiendo así, obtener realmente
104         un valor aleatorio.
105         '''
106     )
107
108
109 print('EJERCICIO 2.1')
110 ej2_1()
111 print('EJERCICIO 2.2')
112 ej2_2()

```

EJERCICIO 2.1

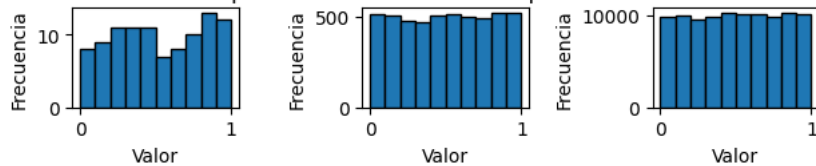


¿Qué conclusiones puede obtener de las gráficas obtenidas en términos de los tiempos de espera y el número de ocurrencias del evento? ¿Qué relación existe entre el tiempo de espera y el número de ocurrencias de un evento?

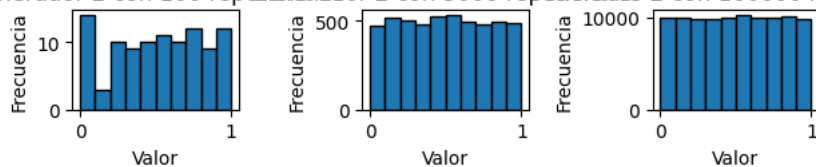
En tanto al tiempo de espera se puede decir que existe una relación inversa en donde cuando el valor de lambda aumenta, el tiempo de espera promedio disminuye y la dispersión de estos tiempos será menor. Por lo tanto, el valor lambda es un factor determinante para tiempo de espera.

EJERCICIO 2.2

Generador 1 con 100 repeticiones Generador 1 con 5000 repeticiones Generador 1 con 100000 repeticiones



Generador 2 con 100 repeticiones Generador 2 con 5000 repeticiones Generador 2 con 100000 repeticiones



Generador 3 con 100 repeticiones Generador 3 con 5000 repeticiones Generador 3 con 100000 repeticiones



▼ Parte No.3 (Ejercicios sobre Generación de V.A P1)



```
1 #TASK 1
2 p = 0.3 # parametro de forma
3 q = 1 - p
4 x = 1000
5
6 #Funcion para imprimir muestra cada 50 elementos
7 def imprimir_cada_x_elementos(arr):
8     for i in range(0, len(arr), 50):
9         sub_array = arr[i:i+50]
10        print(" ".join(map(str, sub_array)))
11
12 # Transformada Inversa distribución geométrica
13 def inversa_distGeom(q,x):
14     muestra = []
15     for i in range(x):
```

```

16         u = rand.uniform(0, 1)
17         X = int(np.ceil(np.log(u) / np.log(q)))
18         muestra.append(X)
19     return muestra
20
21 # PMF teórico de la distribución geométrica
22 def PMF_teorico(p,x,muestra):
23     k = range(1, max(muestra) + 1)
24     px = stats.geom.pmf(k,p)
25     return px , k
26
27
28
29 Muestra_geometrica = inversa_distGeom(q,x)
30 pmf , k= PMF_teorico(p,x,Muestra_geometrica)
31 # Imprimir el array en notación decimal
32 np.set_printoptions(suppress=True)
33
34 print("-- Muestra aleatoria generada por medio de una distribución geométrica --")
35 imprimir_cada_x_elementos(Muestra_geometrica)
36
37
38 print("\n-- PMF teórico de una distribución geométrica --")
39 print(pmf)
40
41 print("\n-- Historgrama --")
42 bns = max(Muestra_geometrica)-min(Muestra_geometrica)+1
43 plt.hist(Muestra_geometrica, bins=bns, density=True, edgecolor='black', label='Muestra')
44 plt.plot(k,pmf,'ro-', label='PMF teórico')
45 plt.title('Histograma de la muestra vs PMF teórico de la distribución geométrica')
46 plt.legend()
47 plt.grid(True)
48 plt.show()
49 #geometrica = stats.geom(p)

```

```

-- Muestra aleatoria generada por medio de una distribución geométrica --
2 4 1 1 3 1 3 4 8 2 3 1 7 7 2 1 7 3 1 1 1 5 10 6 1 10 2 5 1 2 1 2 1 1 5 4 3 1 1 5 16 2 1 8 1 3 1 3 1 8
1 2 2 2 5 2 6 2 1 1 8 1 8 1 1 3 1 2 2 15 2 5 5 3 3 1 9 9 1 1 8 3 6 1 2 5 2 4 2 5 2 3 4 1 1 2 4 1 1 2
2 2 2 1 2 5 9 2 11 2 2 3 2 8 1 3 1 3 1 5 4 2 3 1 1 2 1 6 1 2 11 3 2 1 3 7 1 1 1 1 1 2 4 3 2 10 2 1 1 14
2 3 5 1 4 6 1 2 4 2 1 1 3 12 1 5 3 1 9 3 1 1 5 1 4 1 6 9 2 1 3 1 2 9 2 7 5 2 7 1 2 5 2 1 1 1 4 3 8 1
7 2 3 2 2 2 2 5 3 1 1 1 2 3 4 2 6 2 2 2 1 2 5 2 2 2 4 1 7 3 1 5 1 1 3 1 5 2 2 3 3 1 1 2 2 2 3 7 1 5
1 5 2 6 9 2 9 2 2 2 3 3 1 14 2 1 3 10 2 4 5 2 2 1 2 1 4 5 5 4 3 1 1 1 5 1 1 9 4 2 9 4 1 7 1 2 4 3 6 1
6 1 3 18 3 1 1 2 1 1 3 1 3 2 8 1 1 1 1 4 3 1 7 2 2 1 2 3 11 2 1 1 3 1 2 1 2 2 1 1 1 3 1 1 1 3 2 2 1 4
3 2 2 1 2 9 4 6 2 4 2 1 4 2 2 3 6 1 1 8 1 7 8 5 1 3 5 2 5 2 1 1 5 1 1 4 1 1 6 2 1 1 1 3 6 1 6 5 1 4
3 4 3 2 9 5 1 4 4 1 4 1 8 1 5 4 4 5 5 2 2 6 2 4 2 4 5 1 2 2 3 5 4 2 1 8 1 2 4 1 6 4 2 2 2 5 13 9 1 2

1 # TASK 2
2 lambd = 3
3 x = 1000
4
5 #Funcion para imprimit muestra cada 50 elementos
6 def imprimir_cada_x_elementos(arr):
7     for i in range(0, len(arr), 50):
8         sub_array = arr[i:i+50]
9         print(" ".join(map(str, sub_array)))
10
11 def negative_poisson(x):
12     muestra = []
13     C = stats.poisson.pmf(lambd,3) / (3 * np.exp(-3) / 2)
14     while len(muestra) < x:
15         y = np.random.exponential(scale=1)
16         u = np.random.uniform(0, 1)
17
18         res = stats.poisson.pmf(lambd,y) / (C * stats.expon.pdf(y))
19
20         if u <= res * C:
21             muestra.append(int(y))
22     return muestra
23
24 Muestra_negativa = negative_poisson(x)
25
26 print("-- Muestra aleatoria generada por medio de una distribución poisson --")
27 imprimir_cada_x_elementos(Muestra_negativa)
28
29
30 k = np.arange(0, max(Muestra_negativa) + 1)
31 pmf = [stats.poisson.pmf(lambd,k) for k in k]
32
33
34 print("\n-- Historgrama --")
35 bns = np.arange(-0.5, max(Muestra_negativa) + 1.5, 1)
36 plt.hist(Muestra_negativa, bins=bns, density=True, edgecolor='black', label='Muestra')
37 plt.plot(k,pmf,'ro-', label='PMF teórico')
38 plt.title('Historgrama de la muestra vs PMF teórico de la distribución geométrica')
39 plt.legend()
40 plt.grid(True)
41 plt.show()
42

```

```

-- Muestra aleatoria generada por medio de una distribución poisson --
3 3 1 1 12 1 2 3 1 2 2 1 1 1 3 2 1 2 3 4 1 1 2 3 1 2 2 2 3 1 1 1 1 3 2 1 3 1 2 0 1 1 2 2 1 1 5 1 2 0
1 1 3 3 3 2 2 1 0 2 1 1 2 2 1 0 5 1 2 1 1 2 0 1 1 1 1 1 2 1 1 1 2 1 2 1 0 0 1 0 3 2 3 1 2 1 1 3 2 2
2 1 5 1 2 2 0 3 1 2 3 1 2 1 2 1 2 1 1 0 1 1 2 2 1 0 1 2 0 1 1 2 2 2 1 1 1 1 0 1 3 2 2 1 1 2 1 1 1
1 1 2 2 2 0 1 1 1 1 4 2 3 0 1 1 1 2 2 2 2 2 2 1 2 1 3 1 1 2 3 2 4 2 2 1 2 0 2 3 2 1 2 1 1 2 2 0 1 3
2 2 1 2 1 1 0 0 1 2 1 1 2 1 1 2 3 1 2 2 2 2 1 4 1 2 2 1 2 3 1 1 1 1 2 2 1 3 3 0 1 1 1 1 2 1 3 1 3 2
0 0 3 1 0 3 2 1 1 1 5 2 2 1 2 4 7 2 2 1 2 3 3 1 2 1 1 4 2 1 3 1 1 2 2 1 2 3 1 3 1 1 3 3 2 1 1 1 2 1
2 1 1 0 2 1 3 2 1 5 2 1 2 3 1 3 1 1 2 5 0 1 1 1 5 2 1 1 1 2 3 3 1 1 2 3 3 1 1 1 2 1 1 5 1 1 1 2 2 4
3 2 2 1 1 2 3 1 2 1 1 3 2 0 2 2 1 1 2 3 5 1 4 5 0 2 1 2 1 1 4 1 0 3 1 2 2 1 6 1 1 2 2 5 3 3 1 1 2 1
1 1 0 1 1 2 2 1 1 2 1 2 1 1 2 1 2 1 1 0 0 2 4 1 2 2 1 5 1 1 1 1 1 2 3 1 1 1 3 3 2 4 1 2 4 0 2 2 1 2
4 1 2 1 2 1 1 2 0 1 2 1 1 5 1 1 1 3 2 1 2 1 2 1 2 1 1 1 1 1 2 2 2 0 1 2 3 1 2 3 3 3 1 2 0 1 1 1 3 2
2 1 2 4 2 2 1 2 1 5 1 1 1 0 1 1 1 2 0 1 1 2 1 2 1 3 2 2 0 2 2 3 1 1 5 1 1 1 0 2 1 3 7 2 2 2 2 4 1 2 1 6
6 2 1 1 1 1 2 1 1 1 2 2 2 1 3 1 1 1 4 2 1 3 3 1 3 1 1 5 1 2 1 1 1 0 2 2 2 3 1 5 4 1 2 1 1 1 0 1 0 1 3
2 0 1 4 2 0 1 1 1 1 4 2 1 3 1 2 1 1 1 1 0 1 2 0 2 1 2 1 1 2 2 1 1 3 4 1 1 1 1 1 2 1 2 5 3 2 1 1 2
1 1 1 1 4 4 5 1 1 4 1 3 1 1 0 4 1 2 4 3 1 1 2 3 2 3 1 2 2 2 1 3 2 8 1 4 1 2 1 2 1 4 1 1 2 2 1 3 2 3
1 1 2 0 2 1 1 1 1 1 1 3 3 2 1 5 0 1 2 3 1 2 1 1 1 1 1 1 2 0 3 2 1 2 2 1 4 2 3 3 2 2 1 1 1 1 1 1
2 1 1 1 1 2 1 1 2 3 2 0 1 1 2 1 3 3 4 3 2 1 3 1 2 1 2 1 2 2 4 5 3 1 3 1 2 1 1 2 2 1 1 1 5 2 1 3 1 2
1 4 1 1 6 2 2 2 2 1 1 2 1 1 1 4 1 3 0 1 1 1 2 2 1 2 1 0 2 1 4 2 2 2 7 1 2 1 3 2 2 3 2 1 3 2 2 1 0 2
0 1 0 1 1 2 5 2 3 1 2 2 0 3 2 2 3 1 2 0 0 2 1 2 1 1 1 1 3 1 1 1 1 3 2 2 1 2 2 3 2 1 0 2 0 1 1 2 1
2 2 2 1 2 2 3 2 0 1 1 1 1 1 2 2 1 2 1 1 2 2 1 2 5 1 1 1 4 4 1 1 1 2 3 3 1 2 2 2 4 1 1 2 1 1
2 2 0 2 0 1 1 1 1 2 1 1 3 1 1 2 0 1 1 2 5 2 1 1 3 0 2 1 1 2 1 2 1 3 3 2 2 3 1 1 1 2 1 4 1 2 1 1 1 1

-- Histograma --

1 #TASK 3
2 x = 1000
3 val = np.arange(0, 11)
4
5 def imprimir_cada_x_elementos(arr):
6     for i in range(0, len(arr), 50):
7         sub_array = arr[i:i+50]
8         print(" ".join(map(str, sub_array)))
9
10 def aceptacion_rechazo(x,val,prop,C):
11     muestra = []
12     while len(muestra) < x:
13         y = np.random.choice(val)
14         u = np.random.uniform(0, 1)
15
16         res = (np.power(y,2)) / 55/ (C * prop[y - 1])
17
18         if u <= res :
19             muestra.append(int(y))
20     return muestra
21
22 x = 1000
23 val = np.arange(0, 11)
24 prob = [(np.power(i,2)) / 55 for i in val]
25 propuesta = np.ones_like(val) / len(val)
26 C = max(prob) / max(propuesta)
27 muestra_def = aceptacion_rechazo(x,val,propuesta,C)
28
29
30 print("-- Muestra aleatoria generada por medio de una distribución poisson --")
31 imprimir_cada_x_elementos(muestra_def)
32
33
34 print("\n-- Histograma --")
35 #OPCION 1
36 plt.hist(muestra_def, bins=np.arange(0.5, 10.6, 1), density=True, edgecolor='black', label='Muestra')
37 plt.plot(val, prob,'ro-', label='PMF objetivo', alpha=0.7)
38 plt.title('Histograma de la muestra')
39 plt.legend()
40 plt.grid(True)
41 plt.show()
42
43
44 #OPCION 2
45
46 fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))
47 ax1.hist(muestra_def, bins=np.arange(0.5, 10.6, 1), density=True, edgecolor='black', label='Muestra')
48 ax1.set_title('Histograma de la muestra')
49 ax1.legend()
50 ax1.grid(True)
51
52 ax2.scatter(val, prob, label='PMF objetivo', alpha=0.7)
53 ax2.set_title('Gráfico del PMF objetivo')
54 ax2.grid(True)
55

```

```
56
57 plt.tight_layout()
58 plt.show()
59
60 #OPCION 3
61 plt.bar(val, prob, label='PMF objetivo', alpha=0.7, width=0.4)
62 plt.hist(muestra_def, bins=np.arange(0.5, 10.6, 1), density=True, edgecolor='black', label='Muestra', rwidth=0.4)
63 plt.title('Histograma de la muestra y PMF objetivo')
64 plt.xlabel('Valor')
65 plt.ylabel('Probabilidad')
66 plt.legend()
67 plt.grid(True)
68 plt.show()
```

```
-- Muestra aleatoria generada por medio de una distribución poisson --
10 10 7 8 9 10 5 9 8 10 5 7 10 9 10 9 10 8 6 10 10 9 9 6 8 9 6 7 9 8 9 8 6 9 7 4 6 6 6 10 9 10 9 8 8 9 8 7 10 7
5 6 9 9 10 9 8 8 2 10 7 6 9 7 9 6 6 7 9 6 7 10 7 9 10 10 7 2 9 8 6 7 7 8 8 10 4 6 9 5 10 10 9 10 3 7 8 8 9 10
8 10 10 9 7 9 9 10 10 9 8 6 9 8 8 10 7 9 4 7 9 9 7 9 10 9 9 10 9 9 6 6 8 10 9 9 5 9 9 8 9 8 8 10 10 10 5 10 8
10 8 3 10 7 7 9 9 10 5 7 10 4 10 7 5 5 8 3 6 5 6 9 4 9 10 9 8 8 10 8 10 7 6 10 10 10 8 10 7 5 6 10 7 10 8 10 6 2 10
8 7 7 9 8 10 9 9 10 10 8 5 7 10 3 6 8 7 10 9 8 5 9 5 4 7 6 6 5 8 5 10 8 6 10 4 6 9 8 10 7 10 9 9 5 5 7 5 8 9
8 10 6 8 9 4 9 6 8 9 7 9 10 8 9 9 7 10 10 10 10 9 3 9 9 8 10 5 9 8 8 9 8 7 6 9 6 6 3 6 10 6 10 8 10 7 9 9 8 2
10 8 9 9 9 8 7 10 5 6 6 8 6 10 8 6 8 9 9 9 10 9 6 8 8 5 5 10 8 10 10 9 9 5 6 9 10 10 9 8 9 10 10 7 9 4 5 9 10 10
10 9 8 10 9 9 8 7 10 8 10 6 10 7 10 10 9 9 9 10 10 5 3 4 6 10 4 9 10 5 8 9 7 9 6 8 5 5 10 10 8 7 10 10 7 2 9 10 7 9
3 8 9 10 8 7 6 10 7 9 6 8 6 8 4 9 4 7 7 10 5 7 9 8 6 10 7 8 5 9 9 3 9 10 8 8 8 10 5 9 10 8 7 7 10 7 6 8 10 9
8 9 10 7 2 2 9 8 8 9 9 9 5 10 10 10 7 7 9 9 10 9 7 10 8 9 7 7 7 8 8 9 3 10 6 9 9 8 7 4 7 10 10 7 10 8 10 10 5
10 10 6 7 10 10 10 9 5 10 5 6 9 7 9 6 8 6 9 5 9 8 8 10 10 9 8 5 8 8 10 10 9 9 7 10 10 3 4 9 9 9 10 9 6 10 10 9 10 4
3 6 10 7 8 7 8 7 10 6 6 10 10 8 7 7 10 9 4 7 9 10 10 6 10 9 6 10 6 8 6 9 8 5 6 6 8 10 8 9 10 7 1 3 2 5 8 7 10 5
7 9 6 8 2 7 5 9 7 9 9 10 10 10 7 10 9 10 7 6 8 9 2 7 9 9 7 8 10 9 9 7 10 10 8 4 7 8 7 8 6 10 8 10 7 8 10 9 8 9
3 3 9 7 10 8 7 10 5 10 9 9 8 7 3 6 3 10 7 5 7 9 9 9 9 8 6 8 8 10 8 8 5 6 9 10 6 10 9 10 10 10 7 10 8 9 8 10 5 9
6 5 3 6 7 10 9 7 9 10 4 10 2 10 9 3 8 7 9 3 10 9 8 8 7 8 7 9 10 10 5 6 9 5 9 10 8 9 10 7 7 7 6 7 10 8 9 8 3 8
9 10 10 5 7 10 7 10 1 8 9 7 9 2 10 8 10 8 10 5 9 8 10 7 9 10 9 7 9 9 10 10 10 6 10 9 6 7 4 10 1 8 3 6 2 6 8 2 6 10
9 9 10 9 4 6 8 9 5 10 3 9 10 7 9 10 10 9 4 6 5 7 10 9 7 10 6 7 6 8 10 8 8 8 7 9 7 10 10 4 8 5 10 9 9 9 8 10 7 9
10 10 9 8 9 7 10 7 10 5 7 7 8 10 5 9 10 5 10 8 9 7 8 7 9 8 8 6 7 10 9 10 8 6 7 6 10 6 9 10 5 10 9 9 10 9 8 10 9 9
8 7 10 9 3 2 5 1 9 7 9 8 8 9 8 9 6 8 6 8 8 8 5 10 10 10 9 8 5 10 10 9 10 8 8 9 10 4 5 8 10 10 2 9 10 4 4 6 8 10
10 8 4 5 5 2 9 7 9 9 10 7 8 8 10 8 8 5 9 3 4 9 9 10 7 10 9 10 6 9 7 10 9 9 10 5 2 9 8 8 10 10 9 7 5 10 5 10 8 10
```

```
-- Histograma --
```



▼ Parte No.4 (Ejercicios sobre Generación de V.A P2)



```
1 def ej4_1():
2     print(
3         '''
4         ** Método de descomposición: **
5         Este método permite obtener variables aleatorias con una distribución dada a partir de variables
6         uniformes entre los rangos 0 y 1 en donde estas pueden llegar a ser complejas; es por esto que
7         utilizando este método se reduce la generación de variables aleatorias para lograr simplificar los
8         pasos y que cada una de las secciones sea más manejable.
9
10        La base del mismo consiste en descomponer la función de distribución acumulada (CDF)
11        de la V.A deaseada en subintervalos para simplificarlas y así realizar las transformaciones para
12        ajustar esas distribuciones en cada subintervalo al intervalo de interés.
13
14        -
15
16        ** Pasos: **
17        1. Determinar la CDF de la variable que se busca generar.
18        2. Descomponer la función en subintervalos para calcular la inversa.
19        3. Generar una V.A uniforme U en el intervalo de 0 a 1.
20        4. Utilizar la inversa de la función para transformar U en una V.A en el subintervalo.
21        5. Repetir los pasos 3 y 4 para obtener la muestra.
22
23        '''
24    )
25
26 def ej4_2():
27     # Función de probabilidades
28     p = [0.1, 0.2, 0.3, 0.4]
29     x = [1, 2, 3, 4]
30
31     # Variable aleatoria
32     random_var = np.random.choice(x, p=p, size=1000)
33
34     # Histo
35     plt.hist(random_var, bins=[0.5, 1.5, 2.5, 3.5, 4.5], rwidth=0.8, density=True)
36     plt.show()
37
```



```

37
38 def ej4_3():
39
40     vpn_comparar = 0.1
41
42     # Flujos
43     flujoInitHotel = -800
44     flujoInitCC = -900
45     flujosHotel = [(-800, 50), (-800, 100), (-700, 150), (300, 200), (400, 200), (500, 200), (200, 8440)]
46     flujosCC = [(-600, 50), (-200, 50), (-600, 100), (250, 150), (350, 150), (400, 150), (1600, 6000)]
47
48     # Simulaciones
49     for num_simulaciones in [100, 1000, 10000]:
50
51         npvHotel = []
52         npvCC = []
53
54         for i in range(num_simulaciones):
55             simuHotel = []
56             simuCC = []
57
58             for flujo in flujosHotel:
59                 simulado = np.random.normal(flujo[0], flujo[1])
60                 simuHotel.append(simulado)
61
62             for flujo in flujosCC:
63                 simulado = np.random.normal(flujo[0], flujo[1])
64                 simuCC.append(simulado)
65
66             # Calcular el NPV para hotel y centro comercial
67             npvHotel.append(npf.npv(vpn_comparar, np.append(flujoInitHotel, simuHotel)))
68             npvCC.append(npf.npv(vpn_comparar, np.append(flujoInitCC, simuCC)))
69
70             # Calcular promedio del NPV
71             npv_promedio_hotel = np.mean(npvHotel)
72             npv_promedio_centro_comercial = np.mean(npvCC)
73
74             print(f'''
75 Cant simulaciones: {num_simulaciones}
76 - Hotel: {npv_promedio_hotel}
77 - Centro Comercial: {npv_promedio_centro_comercial}
78 ''')
79
80             print(f"      Conclusión Hotel ({num_simulaciones} simulaciones)")
81             if npv_promedio_hotel < 0 :
82                 print(f'''
83 No se recomienda invertir en el proyecto del hotel pues se estima que habrá una pérdida de
84 aproximadamente {npv_promedio_hotel*-1}. Por lo tanto se puede decir que el proyecto NO es
85 rentable.
86 ''')
87             else:
88                 print(f'''
89 Se recomienda invertir en el proyecto del hotel pues se estima que habrá una ganancia de
90 aproximadamente {npv_promedio_hotel}. Por lo tanto se puede decir que el proyecto SI es
91 rentable.
92 ''')
93
94             print(f"      Conclusión Centro Comercial ({num_simulaciones} simulaciones)")
95             if npv_promedio_centro_comercial < 0 :
96                 print(f'''
97 No se recomienda invertir en el proyecto del centro comercial pues se estima que habrá una
98 pérdida de aproximadamente {npv_promedio_centro_comercial*-1}. Por lo tanto se puede decir que el pro-
99 yecto NO es rentable.
100 ''')
101             else:
102                 print(f'''
103 Se recomienda invertir en el proyecto del centro comercial pues se estima que habrá una
104 ganancia de aproximadamente {npv_promedio_centro_comercial}. Por lo tanto se puede decir que el pro-
105 yecto SI es rentable.
106 ''')
107
108 print('EJERCICIO 4.1')
109 ej4_1()
110 print('EJERCICIO 4.2')
111 ej4_2()

```

EJERCICIO 4.1

**** Método de descomposición: ****

Este método permite obtener variables aleatorias con una distribución dada a partir de variables uniformes entre los rangos 0 y 1 en donde estas pueden llegar a ser complejas; es por esto que utilizando este método se reduce la generación de variables aleatorias para lograr simplificar los pasos y que cada una de las secciones sea más manejable.

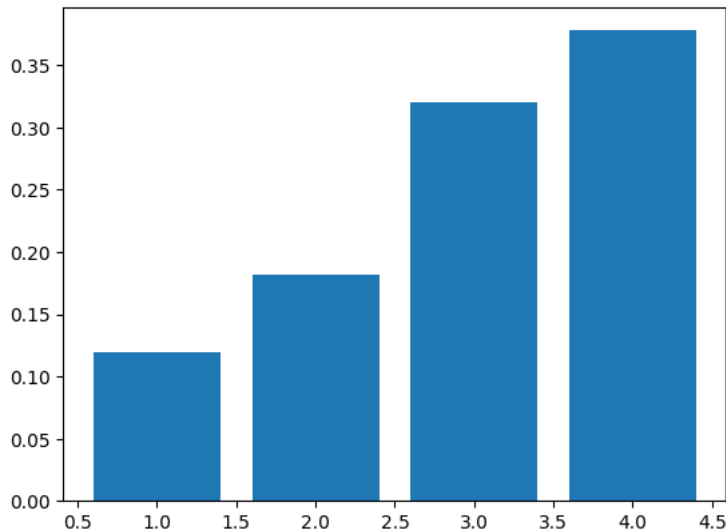
La base del mismo consiste en descomponer la función de distribución acumulada (CDF) de la V.A deaseada en subintervalos para simplificarlas y así realizar las transformaciones para ajustar esas distribuciones en cada subintervalo al intervalo de interés.

-

**** Pasos: ****

1. Determinar la CDF de la variable que se busca generar.
2. Descomponer la función en subintervalos para calcular la inversa.
3. Generar una V.A uniforme U en el intervalo de 0 a 1.
4. Utilizar la inversa de la función para transformar U en una V.A en el subintervalo.
5. Repetir los pasos 3 y 4 para obtener la muestra.

EJERCICIO 4.2



```
1 print('EJERCICIO 4.3')
2 ej4_3()
```

EJERCICIO 4.3

Cant simulaciones: 100

- Hotel: -1937.9630823617324

- Centro Comercial: -80.18189742360177

Conclusión Hotel (100 simulaciones)

No se recomienda invertir en el proyecto del hotel pues se estima que habrá una pérdida de aproximadamente 1937.9630823617324. Por lo tanto se puede decir que el proyecto NO es rentable.

Conclusión Centro Comercial (100 simulaciones)

No se recomienda invertir en el proyecto del centro comercial pues se estima que habrá una pérdida de aproximadamente 80.18189742360177. Por lo tanto se puede decir que el proyecto NO es rentable.

Cant simulaciones: 1000

- Hotel: -1874.6730207209355

- Centro Comercial: -584.4003957237794

Conclusión Hotel (1000 simulaciones)

No se recomienda invertir en el proyecto del hotel pues se estima que habrá una pérdida de aproximadamente 1874.6730207209355. Por lo tanto se puede decir que el proyecto NO es rentable.

Conclusión Centro Comercial (1000 simulaciones)

No se recomienda invertir en el proyecto del centro comercial pues se estima que habrá una pérdida de aproximadamente 584.4003957237794. Por lo tanto se puede decir que el proyecto NO es rentable.

```
Cant simulaciones: 10000  
- Hotel: -1870.5778082358863  
- Centro Comercial: -593.3505422913836
```

Conclusión Hotel (10000 simulaciones)

No se recomienda invertir en el proyecto del hotel pues se estima que habrá una pérdida de aproximadamente 1870.5778082358863. Por lo tanto se puede decir que el proyecto NO es rentable.

Conclusión Centro Comercial (10000 simulaciones)

No se recomienda invertir en el proyecto del centro comercial pues se estima que habrá una pérdida de aproximadamente 593.3505422913836. Por lo tanto se puede decir que el proyecto NO es rentable.

[Colab hotel results](#) [Colab center results](#)

✓ 1s completed at 11:33 PM

