

# Maestria en Ing. de Sistemas UNAM- Procesos estocásticos . Dra: Patricia Aguilar

Alumno: Sergio Ibarra R (414025796)

## Problemas sobre la aplicación de conceptos de variable aleatoria. Marzo 2023

### Problema 1

En cada uno de los siguientes enunciados, indique con una D si la variable aleatoria es discreta o con una C si es continua.

- a) El tiempo de espera para un corte de cabello. ....( Continuo)
- b) El número de automóviles que rebasa un corredor cada mañana. ....(Discreto)
- c) El número de hits de un equipo femenino de softbol de preparatoria. ....(Discreto)
- d) El número de pacientes atendidos en el South Strand Medical Center entre las seis y diez de la noche, cada noche. ....(Discreto)
- e) La distancia que recorrió en su automóvil con el último tanque de gasolina. ....(Continuo)
- f) El número de clientes del Wendy's de Oak Street que utilizaron las instalaciones. (Discreto)
- g) La distancia entre Gainesville, Florida, y todas las ciudades de Florida con una población de por lo menos 50 000 habitantes. ....(Continuo)

### Problema 2

En un estudio realizado en Estados Unidos en 2001 (USA Today, 6 de septiembre, 2001), 38% de los alumnos de cuarto grado de primaria no podía leer un libro apropiado para su edad. Los datos siguientes muestran el número de sujetos, por edad, que se identificaron como niños con problemas de aprendizaje que requieren educación especial. La mayoría tiene problemas de lectura que debieron identificarse y corregirse antes del tercer grado. La ley federal estadounidense actual prohíbe que la mayoría de los niños reciba ayuda adicional de programas de educación especial hasta que el retraso sea de aproximadamente dos años de aprendizaje, y por lo general eso significa hasta tercer grado o grados superiores.

Edad	Número de niños
6	37,369
7	87,436
8	160,840
9	239,719
10	286,719
11	306,533

12	310,787
13	302,604
14	289,168

Suponga que se desea seleccionar una muestra de menores con problemas de aprendizaje y que deben tomar educación especial a efecto de incluirlos en un programa diseñado para mejorar su capacidad de lectura. Sea  $X$  una variable aleatoria que indica la edad de un niño seleccionado al azar,

- Use los datos para construir la función de probabilidad para  $X$  y trace su gráfica.
- Calcule la edad promedio de los niños con problemas de aprendizaje..
- Genere una muestra de las edades de 30 niños con problemas de aprendizaje y calcule la edad promedio. Muestre paso a paso su procedimiento realizado ya sea manualmente o con algún software, en cuyo caso es necesario mostrar la codificación.

### a) Use los datos para construir la función de probabilidad para $X$ y trace su gráfica.

Vamos a importar el documento donde se guardó la tabla con la información sobre la cantidad de niños en problemas de aprovechamiento en USA

```
In [191]: import pandas as pd

# Read the Excel file into a dictionary of data frames
df_importado = pd.read_excel(R'C:\Users\sergi\OneDrive\Documentos\MIS_UNAM\Segundo_semes
print(df_importado)
type(df_importado)
```

```
{'Problema2':      Edad  Número de niños
0         6         37369
1         7         87436
2         8        160840
3         9        239719
4        10        286719
5        11        306533
6        12        310787
7        13        302604
8        14        289168}
```

Out[191]: dict

```
In [192]: df_USA_niños_escolar = df_importado['Problema2']
print(df_USA_niños_escolar)
type(df_USA_niños_escolar)
```

```
      Edad  Número de niños
0         6         37369
1         7         87436
2         8        160840
3         9        239719
4        10        286719
5        11        306533
6        12        310787
7        13        302604
8        14        289168
```

Out[192]: pandas.core.frame.DataFrame

Vamos a "separar" la información de la edad y guardarla en una variable llamada `df_USA_niños_escolar_edad`

```
In [193]: df_USA_niños_escolar_edad = df_USA_niños_escolar['Edad']
df_USA_niños_escolar_edad

Out[193]:
0      6
1      7
2      8
3      9
4     10
5     11
6     12
7     13
8     14
Name: Edad, dtype: int64
```

```
In [194]: type(df_USA_niños_escolar_edad)

Out[194]: pandas.core.series.Series
```

```
In [195]: df_USA_niños_escolar_edad[0]

Out[195]: 6
```

Vamos a "separar" la información del número de niños por edad y guardarla en una variable llamada `df_USA_niños`

```
In [196]: df_USA_niños = df_USA_niños_escolar['Número de niños']
df_USA_niños

Out[196]:
0      37369
1      87436
2     160840
3     239719
4     286719
5     306533
6     310787
7     302604
8     289168
Name: Número de niños, dtype: int64
```

Transformando la data de numero de niños en una lista para poder sumar sus elementos usarlo para calcular el promedio de cada edad

```
In [197]: # Transform the series to a list
df_USA_niños_list = df_USA_niños.tolist()

df_USA_niños_list

Out[197]: [37369, 87436, 160840, 239719, 286719, 306533, 310787, 302604, 289168]
```

```
In [198]: # Get the sum of all elements in the list
total_niños_USA = sum(df_USA_niños_list)
total_niños_USA

Out[198]: 2021175
```

Calculando las probabilidades por edad como:

$$P(x = edad) = \frac{\text{Numero de ninios en edad } X = x}{\text{Numero total de ninios}}$$

y las añadimos a la tabla llamada `df_USA_niños_escolar`

```
In [248... df_USA_niños_escolar['Probabilidad_por_edad'] = (df_USA_niños / total_niños_USA)
print(df_USA_niños_escolar['Probabilidad_por_edad'])

0    0.018489
1    0.043260
2    0.079577
3    0.118604
4    0.141858
5    0.151661
6    0.153766
7    0.149717
8    0.143069
Name: Probabilidad_por_edad, dtype: float64
```

```
In [250... print(df_USA_niños_escolar)
```

	Edad	Número de niños	Probabilidad_por_edad
0	6	37369	0.018489
1	7	87436	0.043260
2	8	160840	0.079577
3	9	239719	0.118604
4	10	286719	0.141858
5	11	306533	0.151661
6	12	310787	0.153766
7	13	302604	0.149717
8	14	289168	0.143069

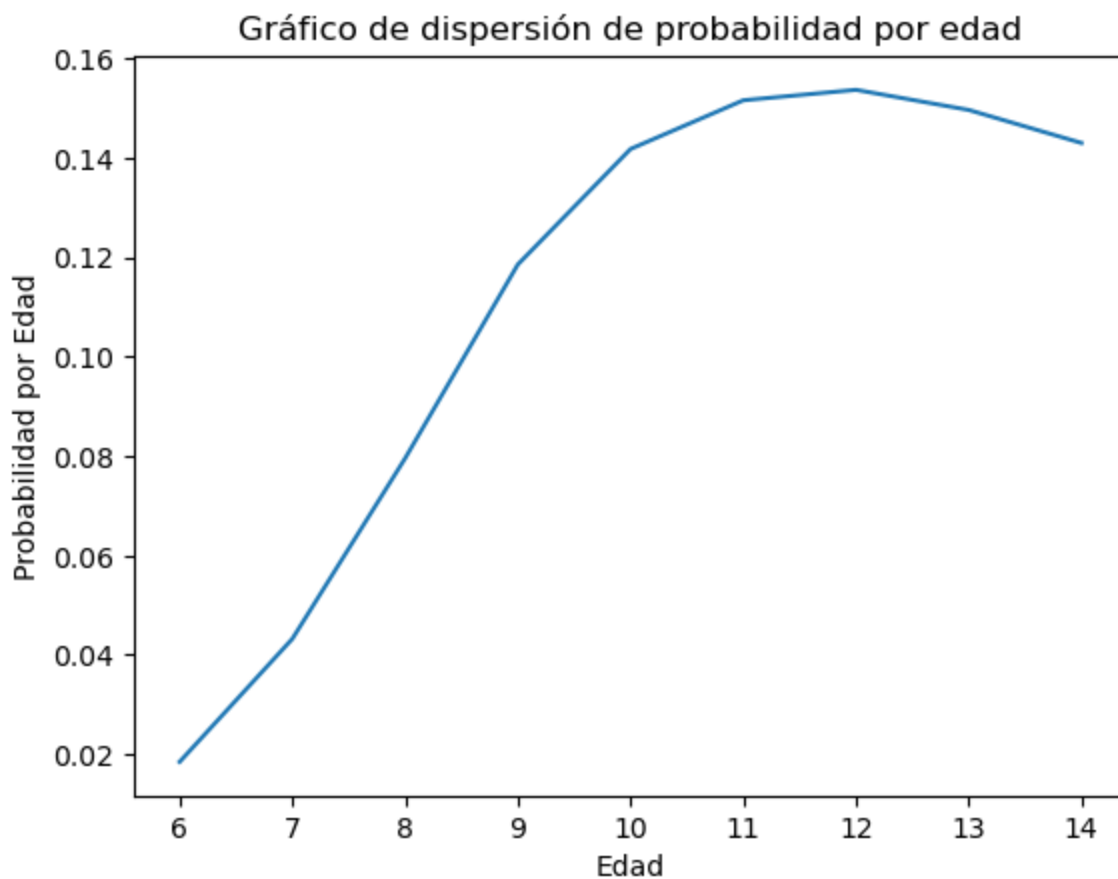
**\_\_Por lo tanto la función de probabilidad de X sería\_\_**

Edad	Valor de probabilidad
6	0.0185
7	0.0432
8	0.7996
9	.1187
10	.1419
11	.1517
12	.1538
13	.1497
14	.1431

**\_\_Y la gráfica de la función de probabilidad de X sería\_\_**

```
In [251... import matplotlib.pyplot as plt

# Assuming that df_USA_niños_escolar is a pandas dataframe
plt.plot(df_USA_niños_escolar['Edad'], df_USA_niños_escolar['Probabilidad_por_edad'])
plt.xlabel('Edad')
plt.ylabel('Probabilidad por Edad')
plt.title('Gráfico de dispersión de probabilidad por edad')
plt.show()
```



**b) Calcule la edad promedio de los niños con problemas de aprendizaje.**

La edad promedio se calculará como la esperanza de los valores de la función de probabilidad

```
In [202...] esperanza_edad = round(sum(df_USA_niños_escolar['Edad']*(df_USA_niños_escolar['Probabili
esperanza_edad
```

Out[202]: 11.0

```
In [203...] from IPython.core.display import HTML

texto = f'<span style="color:red">_La edad promedio de los niños con problemas de aprend
HTML(texto)
```

Out[203]: \_La edad promedio de los niños con problemas de aprendizaje\_ es: 11.0 años

**c) Genere una muestra de las edades de 30 niños con problemas de aprendizaje y calcule la edad promedio. Muestre paso a paso su procedimiento realizado ya sea manualmente o con algún software, en cuyo caso es necesario mostrar la codificación.**

Para poder generar números aleatorios en Python a partir de una x y una f(x) ambos deben estar en un formato de dato tipo 'lista'

```
In [204...] #Transformando los datos de la columna probabilidad en una tipo lista

df_USA_niños_probabilidad_list = (df_USA_niños_escolar['Probabilidad_por_edad']/100).tol
df_USA_niños_probabilidad_list
```

Out[204]: [0.01848875035560998,  
0.04325998490976783,  
0.07957747349932588,

```
0.11860378245327594,  
0.1418575828416639,  
0.15166079137135577,  
0.15376550768736008,  
0.14971687261122862,  
0.143069254270412]
```

```
In [205... #Transformando los datos de la columna edad en una tipo lista  
df_USA_niños_escolar_edad_list = df_USA_niños_escolar_edad.tolist()  
  
df_USA_niños_escolar_edad_list
```

```
Out[205]: [6, 7, 8, 9, 10, 11, 12, 13, 14]
```

Ahora procederemos a generar los 30 numeros random, con la función de python que se llama random.choice y que toma 3 parametros:

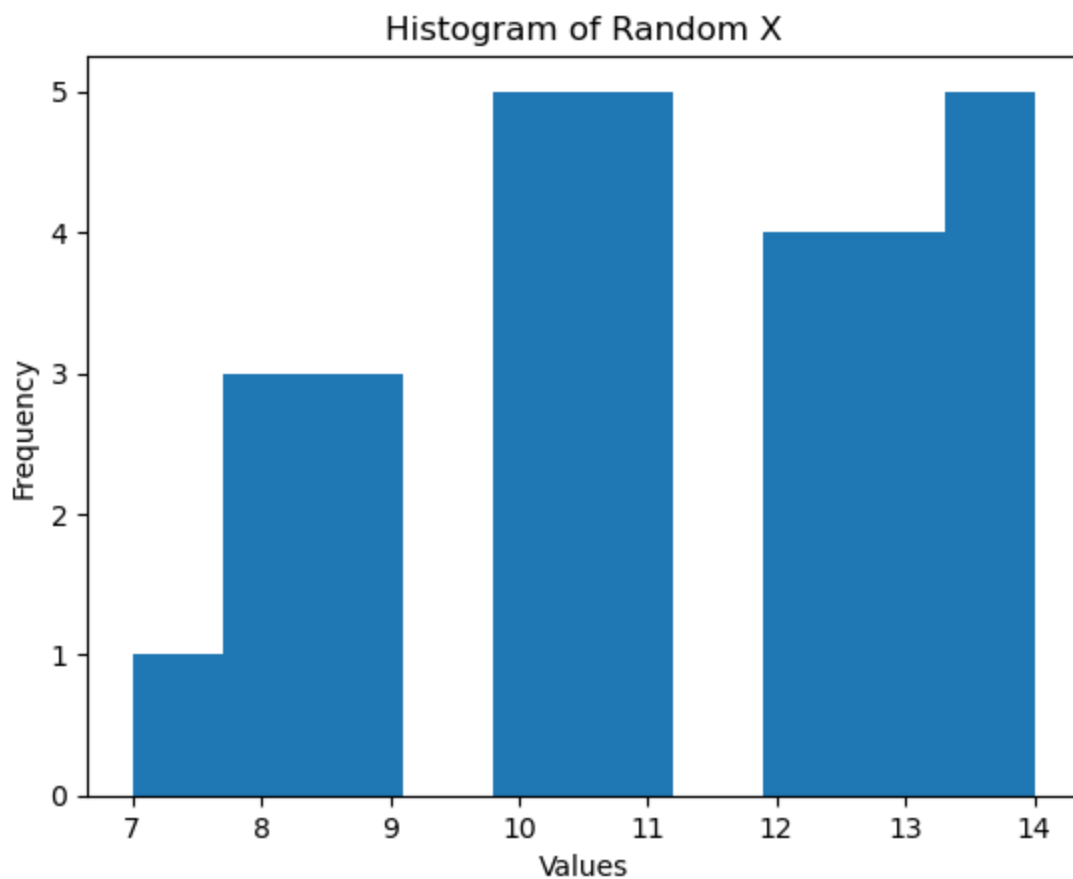
- a) Los valores de la variable x en la función de distribución
- b) Los valores de f(x) en la función de distribución
- c) El número de valores random que se quiere generar

```
In [206... import numpy as np  
  
random_30_edades = np.random.choice(  
df_USA_niños_escolar_edad_list, size=30, p=df_USA_niños_probabilidad_list )  
random_30_edades
```

```
Out[206]: array([ 8,  9, 13, 13, 10, 12, 14, 10, 14, 10, 10, 11,  8, 12, 11,  8, 10,  
        7, 14,  9, 12,  9, 14, 12, 11, 11, 14, 13, 13, 11])
```

Vamos a graficar los valores generados

```
In [207... import matplotlib.pyplot as plt  
  
plt.hist(random_30_edades, bins=10)  
plt.xlabel('Values')  
plt.ylabel('Frequency')  
plt.title('Histogram of Random X')  
plt.show()
```



Ahora calculemos la edad promedio de los valores simulados

```
In [208... import numpy as np

edad_promedio_simulada = round(np.mean(random_30_edades), 2)
```

```
In [209... from IPython.core.display import HTML

texto = f'<span style="color:red">_La edad promedio de los valores simulados es__ es: {e
HTML(texto)
```

Out[209]: \_La edad promedio de los valores simulados es\_\_ es: 11.1 años

ES MUY CERCANA A LA ESPERANZA\_EDAD QUE NO ES SIMULADA!

In [ ]:

### Problema 3

Considere la variable aleatoria  $Y$  cuya función de distribución (acumulativa) es la que se indica a continuación.

- Decida si la variable aleatoria  $Y$  es discreta o continua. Justifique su respuesta.
- Construya la función  $f_Y(y)$ .
- Calcule la media y la varianza de  $Y$

In [ ]:

$$F_Y(y) = \begin{cases} 0, & y \leq 0 \\ y/8, & 0 < y < 2 \\ y^2/16, & 2 \leq y < 4 \\ 1, & y \geq 4 \end{cases}$$

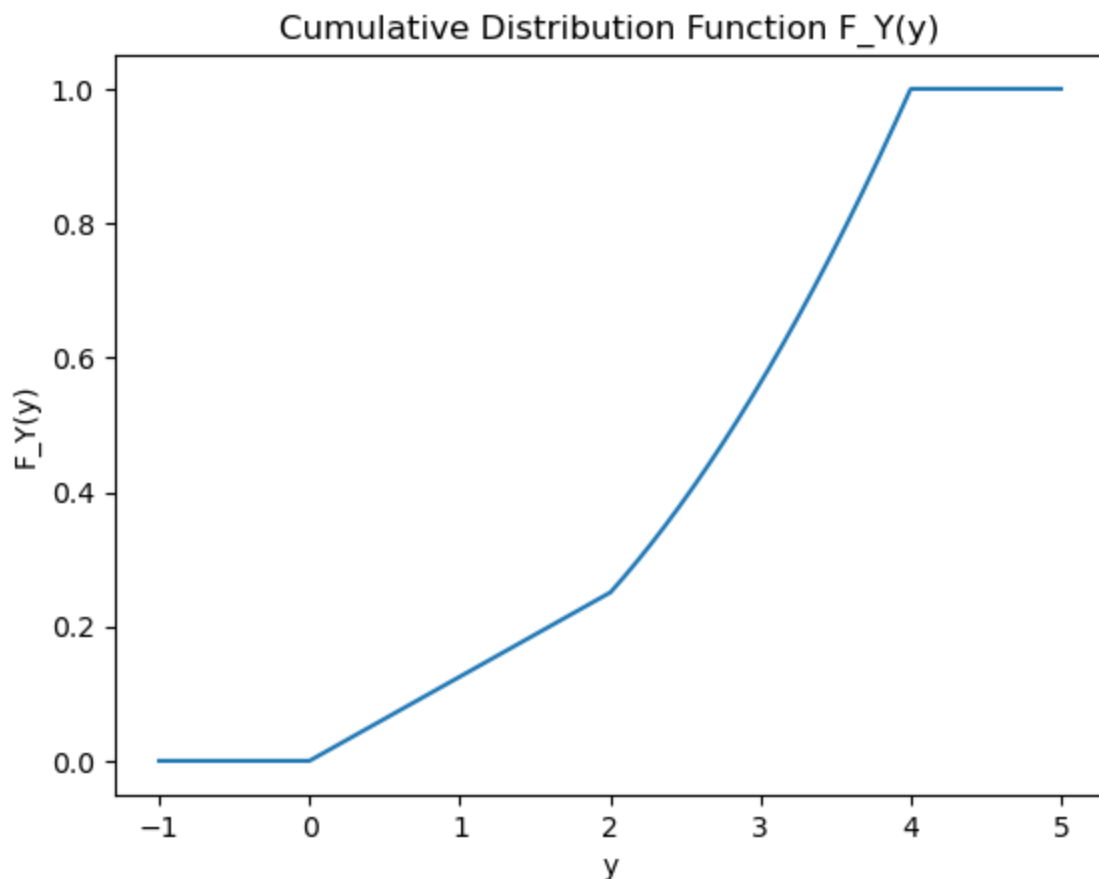
Primero que todo vamos a graficar la funcion de distribución acumulada para la variable Y y observar que forma tiene

```
In [210... import numpy as np
import matplotlib.pyplot as plt

def F_Y(y):
    return np.piecewise(y, [y <= 0, (y > 0) & (y < 2), (y >= 2) & (y < 4), y >= 4],
                        [0, lambda y: y/8, lambda y: y**2/16, 1])

y = np.linspace(-1, 5, 1000)
F = F_Y(y)

plt.plot(y, F)
plt.title('Cumulative Distribution Function F_Y(y)')
plt.xlabel('y')
plt.ylabel('F_Y(y)')
plt.show()
```



Se observa claramente una función continua y creciente hasta 1, por lo que podemos estar seguros que F(y) es una función de distribución acumulativa

**a) Decida si la variable aleatoria Y es discreta o continua. Justifique su respuesta.**

Debido a que la función de distribución  $F_Y(y)$  SI ES CONTINUA. Se puede decir entonces que la Variable aleatoria X de la que proviene TAMBIÉN ES CONTINUA



## b) Construya la función $f_Y(y)$ .

Nosotros sabemos que para cualquier variable aleatoria  $Y$  su función de distribución acumulativa se define como:

$$F_Y(y) = P(Y \leq y)$$

Y sabemos que la relación entre la función de distribución acumulada  $F(Y)$  y la función de distribución de densidad de probabilidad  $f(y)$  es:

$$F_Y(Y) = \int_{-\infty}^y f_Y(t) dt$$

Y por lo tanto se podría decir que es posible calcular la función de densidad de probabilidad  $f(y)$  a partir de su función de densidad acumulada de probabilidad  $F_Y(Y)$  de la siguiente manera:

$$f_Y(y) = F'_Y(t) = \frac{dF_Y}{dy}$$

Por lo tanto en nuestro caso habría que derivar cada una de las secciones de la función acumulativa  $F(y)$  con respecto a  $y$  para obtener  $f(y)$

Por lo tanto la función de densidad probabilidad de  $Y$  sería -  $f(Y)$  -

$$f(y) = \begin{cases} 0, & y \leq 0 \\ 1/8, & 0 < y < 2 \\ y/8, & 2 \leq y < 4 \\ 0, & y \geq 4 \end{cases}$$

Vamos a graficar la función de densidad de probabilidad  $f(y)$  para ver como luce

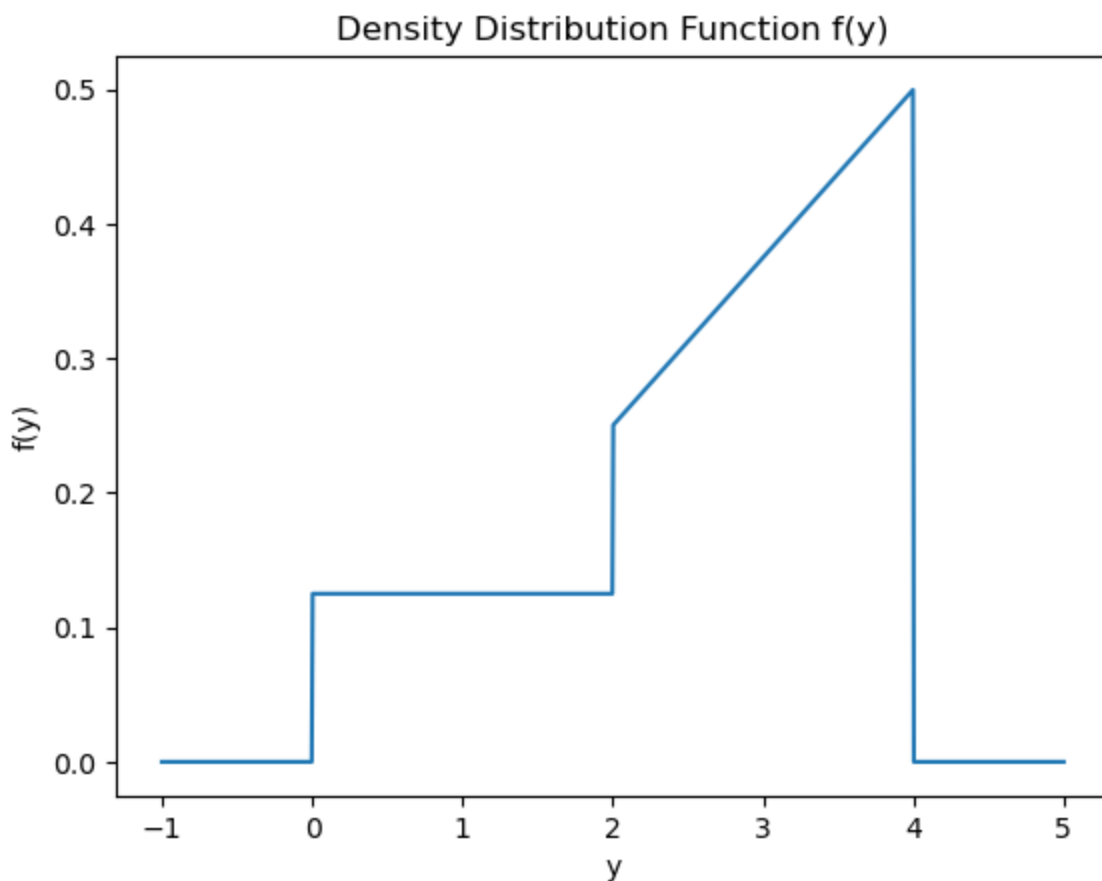
In [211...

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

def f_y(y):
    return np.piecewise(y, [y<=0, (y>0)&(y<2), (y>=2)&(y<4), y>=4], [0, 1/8, lambda y: y/8, 0])

y = np.linspace(-1, 5, 1000)
f = f_y(y)

plt.plot(y, f)
plt.title('Density Distribution Function f(y)')
plt.xlabel('y')
plt.ylabel('f(y)')
plt.show()
```



Se observa una función con valores mayores a cero y cuya área parece justamente tender al valor de la unidad

### c) Calcule la media y la varianza de Y

Recordando que la esperanza de una variable aleatoria Y, dada su función de distribución  $f(y)$  se definía para el caso de las variables discretas como:

$$E(Y) = \sum_{-\infty}^{\infty} y * p(y)$$

Para el caso de las variables continuas dicha esperanza o media se define como:

$$E(Y) = \int_{-\infty}^{\infty} y * p(y)$$

Que para nuestro caso sería:

$$E(y) = \int_0^4 f(y) * y = \int_0^4 \begin{cases} 0, & y \leq 0 \\ 1/8, & 0 < y < 2 \\ y/8, & 2 \leq y < 4 \\ 0, & y \geq 4 \end{cases} * y dy$$

Entonces resolviendo la integral en Python tenemos:

```
In [212... import numpy as np
from scipy.integrate import quad

def f_y(y):
    return np.piecewise(y, [y<=0, (y>0)&(y<2), (y>=2)&(y<4), y>=4], [0, 1/8, lambda y: y

##Se define la función que integrará y por lo tanto calculará la media
def mean_f(f, a, b):
    integrand = lambda y: y*f(y)
    ## FUNCIÓN QUE REALIZA LA INTEGRACIÓN!
    return quad(integrand, a, b)[0]

##Aqui se usa/llama la función que integra y calcula media
mean_Y = round(mean_f(f_y, 0, 4),2)
```

```
In [213... from IPython.core.display import HTML

texto = f'<span style="color:red">_La media de la variable Y sería: {mean_Y} </span>'

HTML(texto)
```

Out[213]: **\_La media de la variable Y sería: 2.58**

Para el caso de la varianza de la variable aleatoria Y, se define como, la esperanza del cuadrado de la desviación con respecto a la media, es decir:

$$Var(Y) = E[(Y - \mu)^2]$$

Ahora en el caso de conocer la funcion de probabilidad de densidad dicha varianza se calcula como:

$$Var(Y) = \sigma^2(y) = \int_{-\infty}^{\infty} (Y - \mu)^2 * f(y)$$

Y en nuestro caso:

$$Var(y) = \int_0^4 f(y) * (Y - \mu)^2 = \int_0^4 \begin{cases} 0, & y \leq 0 \\ 1/8, & 0 < y < 2 \\ y/8, & 2 \leq y < 4 \\ 0, & y \geq 4 \end{cases} * (Y - \mu)^2 dy$$

Y en Python el código para integrar y genera la varianza es:

```
In [214... import numpy as np
from scipy.integrate import quad

def f_y(y):
    return np.piecewise(y, [y<=0, (y>0)&(y<2), (y>=2)&(y<4), y>=4], [0, 1/8, lambda y: y

##Se define la función que integrará y por lo tanto calculará la varianza
def variance_f(f, mean_Y, a, b):
    integrand = lambda y: (y-mean_Y)**2*f(y)
    return quad(integrand, a, b)[0]

##Aqui se usa/llama la función que integra y calcula varianza
variance_Y = round(variance_f(f_y, mean_Y, 0, 4),2)
```

```
In [215]: from IPython.core.display import HTML

texto = f'<span style="color:red">_La varianza de la variable Y sería: {variance_Y} </sp

HTML(texto)
```

Out[215]: \_La varianza de la variable Y sería: 1.16

## Problema 4

In [ ]:

La proporción de tiempo por día en la que todas las cajas de un supermercado están ocupadas, es una variable aleatoria  $Y$  con función de densidad

$$f_Y(y) = \begin{cases} c * y^2(1 - y)^4, & 0 \leq y \leq 1 \\ 0, & \text{en otro caso} \end{cases}$$

En donde  $c$  es una constante.

- Encuentre el valor de  $c$  que hace de  $f_Y(y)$  una función de densidad
- Obtenga la media y la varianza de  $Y$ .
- Genere 30 valores aleatorios de  $Y$  y calcule el promedio de dichos valores. Muestre paso a paso su procedimiento realizado ya sea manualmente o con algún software, en cuyo caso es necesario mostrar la codificación

### a) Encuentre el valor de $c$ que hace de $f_Y(y)$ una función de densidad

Entonces se debe encontrar los valores de  $c$  para los cuales la integral de  $f(x)$  desde 0 hasta 1 sea exactamente igual a 1. que es una de las propiedades de la función de distribución de una v.a.  $X$

La propiedad de la función de distribución  $f(x)$  se expresa entonces como:

$$\int_0^1 f_x(x) = 1$$

Que en nuestro caso es:

$$\int_0^1 c * y^2 * (1 - y)^4 = 1$$

Entonces resolvamos la integral usando Python

```
In [216]: import sympy as sym

# define the symbol y and the constant c
y = sym.symbols('y')
c = sym.symbols('c')

# define the integrand
```

```
f = c * y**2 * (1-y)**4

# integrate the function with respect to y
F = sym.integrate(f, y)

# print the result
print("The antiderivative of f(y) = c*y^2*(1-y)^4 with respect to y is:")
print(F)
```

The antiderivative of  $f(y) = c*y^2*(1-y)^4$  with respect to  $y$  is:  
 $c*y^{7/7} - 2*c*y^{6/3} + 6*c*y^{5/5} - c*y^{4/4} + c*y^{3/3}$

Entonces el resultado de la integral es:

$$\int_0^1 c * y^2 * (1 - y)^4 dy = \frac{c * y^7}{7} - \frac{2 * c * y^6}{3} + \frac{6 * c * y^5}{5} - c * y^4 + \frac{c * y^3}{3}$$

Y evaluando los limites de integración tenemos:

$$\left[ \frac{c * y^7}{7} - \frac{2 * c * y^6}{3} + \frac{6 * c * y^5}{5} - c * y^4 + \frac{c * y^3}{3} \right]_0^1 = \left[ \frac{c}{7} - \frac{2 * c}{3} + \frac{6 * c}{5} - c * + \frac{c}{3} \right] - (0)$$

Sumando los terminos, tenemos:

$$1 = \frac{1c}{105}$$

**\_\_Por lo tanto el valor de C que hace que la función f(y) sea una función de densidad de probabilidad es C=105- \_\_**

## b) Obtenga la media y la varianza de Y.

La media se calcula como:

$$E(Y) = \int_{-\infty}^{\infty} y * p(y)$$

Que en nuestro caso es:

$$E(X) = \int_{-\infty}^{\infty} y * p(y) = \int_{-\infty}^{\infty} \begin{cases} c * y^2(1 - y)^4, & 0 \leq y \leq 1 \\ 0, & \text{en otro caso} \end{cases} * y dy$$

Entonces resolviendo la integral en Python tenemos:

```
In [217... import numpy as np
from scipy.integrate import quad

def f_y(y):
    return np.piecewise(y, [y<=0, (y>0)&(y<1), y>1], [0, lambda y: 105*(y**2)*(1-y)**4,

## Se define la función que integrará y por lo tanto calculará la media
```

```
def mean_f(f, a, b):
    integrand = lambda y: y*f(y)
    ## FUNCIÓN QUE REALIZA LA INTEGRACIÓN!
    return quad(integrand, a, b)[0]

##Aquí se usa/llama la función que integra y calcula media
mean_Y2 = round(mean_f(f_y, 0, 4),2)
```

```
In [218... from IPython.core.display import HTML

texto = f'<span style="color:red">_La MEDIA de la variable Y sería: {mean_Y2} </span>'

HTML(texto)
```

Out[218]: \_La MEDIA de la variable Y sería: 0.37

Y para el cálculo de la varianza tenemos:

$$E(X) = \int_{-\infty}^{\infty} (Y - \mu)^2 * p(y) = \int_{-\infty}^{\infty} \begin{cases} c * y^2(1 - y)^4, & 0 \leq y \leq 1 \\ 0, & \text{en otro caso} \end{cases} * (Y - \mu)^2 dy$$

```
In [219... import numpy as np
from scipy.integrate import quad

def f_y(y):
    return np.piecewise(y, [y<=0, (y>0)&(y<1), y>1], [0, lambda y: 105*(y**2)*(1-y)**4,

##Se define la función que integrará y por lo tanto calculará la varianza
def variance_f(f, mean, a, b):
    integrand = lambda y: (y-mean_Y2)**2*f(y)
    return quad(integrand, a, b)[0]

##Aquí se usa/llama la función que integra y calcula varianza
variance_Y2 = round(variance_f(f_y, mean_Y2, 0, 4),2)
```

```
In [220... from IPython.core.display import HTML

texto = f'<span style="color:red">_La VARIANZA de la variable Y sería: {variance_Y2} </s

HTML(texto)
```

Out[220]: \_La VARIANZA de la variable Y sería: 0.03

c) Genere 30 valores aleatorios de Y y calcule el promedio de dichos valores. Muestre paso a paso su procedimiento realizado ya sea manualmente o con algún software, en cuyo caso es necesario mostrar la codificación

**Paso 1. Se generan los 30 numeros aleatorios ente [0,1] con una probabilidad de distribución uniforme**

```
In [221... import random

# generate 30 random numbers between 0 and 1
random_numbers_30 = [random.random() for _ in range(30)]

print(random_numbers_30)

[0.2418072199090664, 0.8756324540668448, 0.05311413255923336, 0.5194537042228833, 0.9862
244244520539, 0.6909687448481003, 0.2223031945701036, 0.6658146095458647, 0.851890263522
```

4037, 0.11874554679164107, 0.2578849162390534, 0.21362979249809466, 0.3432715363953813, 0.1032094775749145, 0.7436542053368869, 0.6261066908323387, 0.4378862156350616, 0.7284075041949227, 0.00037576682906215186, 0.10958114472596725, 0.7533607660447219, 0.1643865069310475, 0.4299471063033371, 0.2234789904229525, 0.07866138516534105, 0.2466890802750723, 0.21202778815668144, 0.7787760907206084, 0.17508535984579665, 0.41172288459055806]

## Paso 2. Igualar el valor aleatorio con la funcion acumulada de probabilidad de la variable x

Se procede a igualar el valor aleatorio generado  $r_i$  con la funcion de distribucion acumulada:

$$r_i = F(y_i)$$

Que en nuestro caso recordar que

$$F(y_i) = \int_0^1 105 * y^2 * (1 - y)^4 dy$$

Y resolviendo en Python tenemos:

```
In [222... import sympy as sym

# define the symbol y and the constant c
y = sym.symbols('y')

# define the integrand
f_ = 105 * y**2 * (1-y)**4

# integrate the function with respect to y
F_ = sym.integrate(f_, y)

# print the result
print("The antiderivative of f(y) = 105*y^2*(1-y)^4 with respect to y is:")
print(F_)
```

The antiderivative of f(y) = 105\*y^2\*(1-y)^4 with respect to y is:  
15\*y\*\*7 - 70\*y\*\*6 + 126\*y\*\*5 - 105\*y\*\*4 + 35\*y\*\*3

Es decir:

$$valor_{r,andom_i} = 15y^7 - 70y^6 + 125y^5 - 105y^4 + 35y^3$$

Entonces para nuestro primer valor simulado que es 0.105, lo igualamos a nuestra  $F(y_i)$ :

## Paso 3. Calcular el valor de la variable x para cada elemento previamente simulado

Te calcula las 7 soluciones por cada valor de  $r_i$

```
In [223... import numpy as np

# Define the coefficients of the polynomial
coeffs = [15, -70, 125, -105, 35, 0, -0.53]

# Find the roots of the polynomial
roots = np.roots(coeffs)

# Print the roots
print(roots)
```

[ 1.79584535+0.j 0.96850278+0.63368849j 0.96850278-0.63368849j  
0.88154565+0.j 0.157833 +0.j -0.1055629 +0.j ]

Te calcula solo soluciones reales para todos los valres en random\_numbers\_30

In [225.. `from numpy import roots`

`random_numbers_30`

```
for i in range(len(random_numbers_30)):
    roots_of_equation = roots([15, -70, 125, -105, 35, 0, 0, -random_numbers_30[i]])
    real_roots = [root.real for root in roots_of_equation if root.imag == 0]
    print(f"For {random_numbers_30[i]}: {real_roots}")
```

```
For 0.2418072199090664: [1.7870509445124372, 0.9461389771044434, 0.2496072070006195]
For 0.8756324540668448: [1.794933320132203]
For 0.05311413255923336: [1.7845952698235719, 0.9891440129051062, 0.13184876621641745]
For 0.5194537042228833: [1.7905705076883516, 0.8633860467138114, 0.37602979502211914]
For 0.9862244244520539: [1.796255250238559]
For 0.6909687448481003: [1.792691942057852, 0.7874963402913643, 0.4653687834507372]
For 0.2223031945701036: [1.7867995697627168, 0.9509582508624214, 0.23999222049371935]
For 0.6658146095458647: [1.7923832470353556, 0.8009420398814632, 0.4504067971457991]
For 0.8518902635224037: [1.7946475586480353]
For 0.11874554679164107: [1.78545546533382, 0.9750350902512301, 0.18228019518380992]
For 0.2578849162390534: [1.787257742053146, 0.9420908847241949, 0.25735745506169205]
For 0.21362979249809466: [1.7866876048466378, 0.9530702707121939, 0.2356293010105158]
For 0.3432715363953813: [1.7883497628468228, 0.9193246557760796, 0.296752028285372]
For 0.1032094775749145: [1.7852524338166222, 0.97844899804414, 0.17195860225333942]
For 0.7436542053368869: [1.793335847428453, 0.7544750918389745, 0.5011085402237775]
For 0.6261066908323387: [1.791894260332239, 0.8201298255352014, 0.4285236895305332]
For 0.4378862156350616: [1.789547717088384, 0.891094426913375, 0.33897807648760186]
For 0.7284075041949227: [1.7931498750010662, 0.7648946250439753, 0.4899644326657852]
For 0.00037576682906215186: [1.7838992682539152, 0.9999248353355833, 0.02256767161208129]
For 0.10958114472596725: [1.7853357454302934, 0.9770546524295374, 0.1762744703742382]
For 0.7533607660447219: [1.7934540883477665, 0.7473475169249615, 0.5086716231768842]
For 0.1643865069310475: [1.7860498193733279, 0.9647193208562508, 0.2095193886364268]
For 0.4299471063033371: [1.7894476778575479, 0.8936074613718288, 0.3354328489417036]
For 0.2234789904229525: [1.786814739608707, 0.9506704858409156, 0.2405792839078753]
For 0.07866138516534105: [1.7849308837008735, 0.9837479775656228, 0.15397252674713868]
For 0.2466890802750723: [1.7871137766854854, 0.9449171573835677, 0.25197591306530776]
For 0.21202778815668144: [1.7866669124859242, 0.9534583211934369, 0.23481696887491751]
For 0.7787760907206084: [1.793763116125916, 0.7261613988048168, 0.530904647103034]
For 0.17508535984579665: [1.786188693498851, 0.9622360927901992, 0.21542057620615002]
For 0.41172288459055806: [1.789217706543183, 0.899268529304757, 0.3273112787660425]
```

In [226.. `type(real_roots)`

`list`

Out[226]:

Te calcula solo soluciones reales para todos los valores en random\_numbers\_30 y las guarda en all\_real\_roots

In [227.. `from numpy import roots`

`random_numbers_30 # your list of 30 random numbers`

`all_real_roots = [] # create an empty list to store real roots for each iteration`

```
for i in range(len(random_numbers_30)):
    roots_of_equation = roots([15, -70, 125, -105, 35, 0, 0, -random_numbers_30[i]])
    real_roots = [root.real for root in roots_of_equation if root.imag == 0]
    all_real_roots.append(real_roots) # append the real roots to the list of all real r

print(all_real_roots)
```



```
[[1.7870509445124372, 0.9461389771044434, 0.2496072070006195], [1.794933320132203], [1.7845952698235719, 0.9891440129051062, 0.13184876621641745], [1.7905705076883516, 0.8633860467138114, 0.37602979502211914], [1.796255250238559], [1.792691942057852, 0.7874963402913643, 0.4653687834507372], [1.7867995697627168, 0.9509582508624214, 0.23999222049371935], [1.7923832470353556, 0.8009420398814632, 0.4504067971457991], [1.7946475586480353], [1.78545546533382, 0.9750350902512301, 0.18228019518380992], [1.787257742053146, 0.9420908847241949, 0.25735745506169205], [1.7866876048466378, 0.9530702707121939, 0.2356293010105158], [1.7883497628468228, 0.9193246557760796, 0.296752028285372], [1.7852524338166222, 0.97844899804414, 0.17195860225333942], [1.793335847428453, 0.7544750918389745, 0.5011085402237775], [1.791894260332239, 0.8201298255352014, 0.4285236895305332], [1.789547717088384, 0.891094426913375, 0.33897807648760186], [1.7931498750010662, 0.7648946250439753, 0.4899644326657852], [1.7838992682539152, 0.9999248353355833, 0.02256767161208129], [1.7853357454302934, 0.9770546524295374, 0.1762744703742382], [1.7934540883477665, 0.7473475169249615, 0.5086716231768842], [1.7860498193733279, 0.9647193208562508, 0.2095193886364268], [1.7894476778575479, 0.8936074613718288, 0.3354328489417036], [1.786814739608707, 0.9506704858409156, 0.2405792839078753], [1.7849308837008735, 0.9837479775656228, 0.15397252674713868], [1.7871137766854854, 0.9449171573835677, 0.25197591306530776], [1.7866669124859242, 0.9534583211934369, 0.23481696887491751], [1.793763116125916, 0.7261613988048168, 0.530904647103034], [1.786188693498851, 0.9622360927901992, 0.21542057620615002], [1.789217706543183, 0.899268529304757, 0.3273112787660425]]
```

```
In [228... all_real_roots[0]
```

```
Out[228]: [1.7870509445124372, 0.9461389771044434, 0.2496072070006195]
```

Te Guarda el valor mínimo entre 0 y 1 como resultado de resolver la ecuación de  $F(x_i)=r_i$  para cada  $i$  en los 30 elementos simulados anteriormente

```
In [229... final_real_roots = []

for roots_list in all_real_roots:
    smallest_root = min(roots_list)
    final_real_roots.append(smallest_root)
```

```
In [230... from IPython.core.display import HTML

texto = f'<span style="color:red">La lista de los 30 números simulados finales es:</span>'

HTML(texto)
```

```
Out[230]: La lista de los 30 números simulados finales es: [0.2496072070006195, 1.794933320132203,
0.13184876621641745, 0.37602979502211914, 1.796255250238559, 0.4653687834507372,
0.23999222049371935, 0.4504067971457991, 1.7946475586480353, 0.18228019518380992,
0.25735745506169205, 0.2356293010105158, 0.296752028285372, 0.17195860225333942,
0.5011085402237775, 0.4285236895305332, 0.33897807648760186, 0.4899644326657852,
0.02256767161208129, 0.1762744703742382, 0.5086716231768842, 0.2095193886364268,
0.3354328489417036, 0.2405792839078753, 0.15397252674713868, 0.25197591306530776,
0.23481696887491751, 0.530904647103034, 0.21542057620615002, 0.3273112787660425]
```

El promedio de los valores simulados es:

```
In [232... import numpy as np

average_final_real_roots = round(np.mean(final_real_roots),2)
```

```
In [233... from IPython.core.display import HTML
```

```
texto = f'<span style="color:red">El valor promedio de los valores simulados es : {avera
HTML(texto)
```

Out[233]: El valor promedio de los valores simulados es : 0.45:

In [ ]:

## Problema 5

Considere la v.a.  $X$  con función de probabilidad

x	-5	-1	1	1.5	3
fx(X)	0.2	0.01	0.3	0.29	0.2

In [ ]:

Construya muestras de la v.a.  $X$ , de los tamaños indicados en clase, y en cada caso sobreponga la gráfica de la distribución de  $X$  (histograma de probabilidad) al histograma de frecuencias de la muestra correspondiente. Adicionalmente, calcule la media y la varianza de cada muestra y compárelas con  $E(X)$  y  $Var(X)$

Creamos los objetos "tipo lista" de los valores de  $x$  y  $f(x)$

```
In [234...] x1 = [-5, -1, 1, 1.5, 3]
x1
fx1 = [0.2, 0.01, 0.3, 0.29, 0.2]
fx1
```

Out[234]: [0.2, 0.01, 0.3, 0.29, 0.2]

Importamos las librerías necesarias para realizar las muestras de tamaño  $n$

```
In [235...] from random import random
import numpy as np
import matplotlib.pyplot as plt
```

Usamos la función `np.random.choice` (previamente usada y explicada en el problema 2 inciso c)

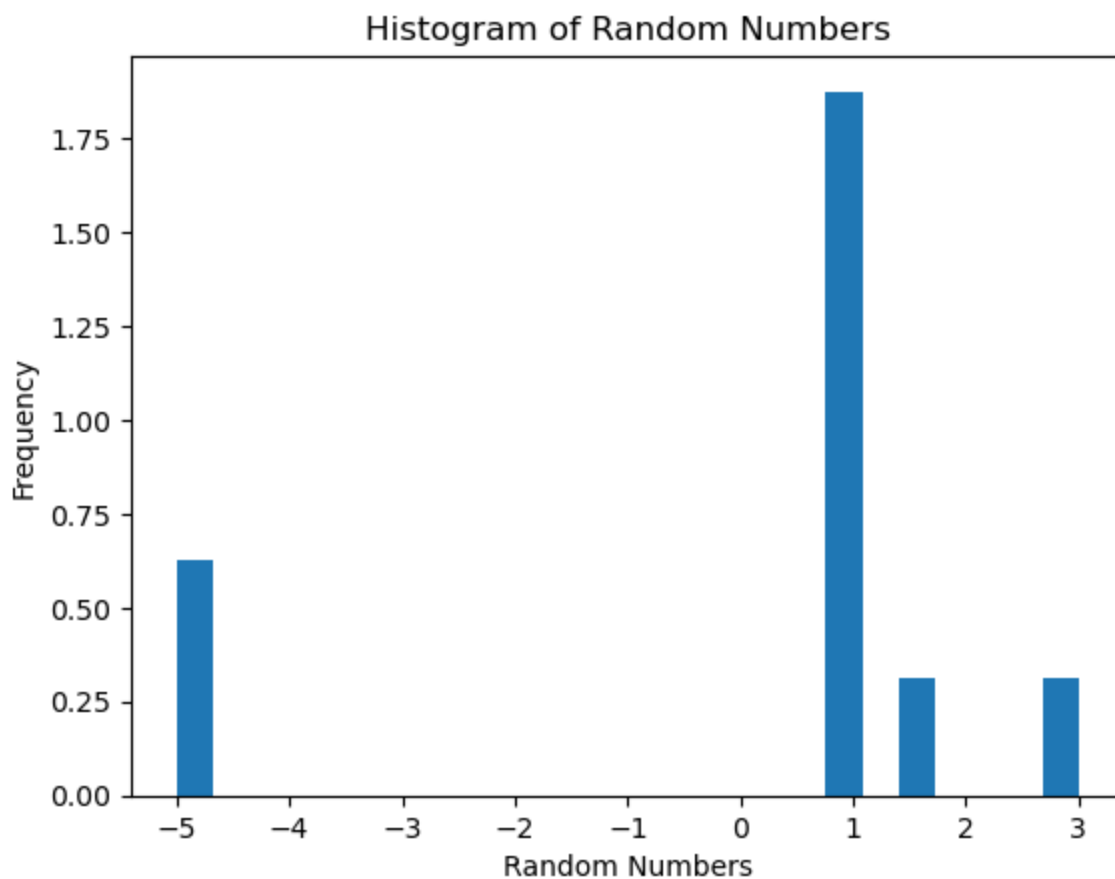
**Se generarán entonces los números para tamaño de muestra  $n=10$**

```
In [236...] random_numbers_fx1_10 = np.random.choice(x1, size=10, p=fx1)
print(random_numbers_fx1_10)

[-5.  1. -5.  1.  3.  1.5  1.  1.  1.  1.]
```

Graficamos el resultado para tamaño de muestra  $n=10$

```
In [237...] plt.hist(random_numbers_fx1_10, bins=25, density=True)
plt.xlabel('Random Numbers')
plt.ylabel('Frequency')
plt.title('Histogram of Random Numbers')
plt.show()
```



Calculamos el promedio para tamaño de muestra n=10

```
In [238.. import numpy as np

average_random_numbers_fx1_10= round(np.mean(random_numbers_fx1_10),2)
average_random_numbers_fx1_10
```

Out[238]: 0.05

Calculamos la varianza para tamaño de muestra n=10

```
In [239.. import numpy as np

variance_random_numbers_fx1_10= round(np.var(random_numbers_fx1_10),2)
variance_random_numbers_fx1_10
```

Out[239]: 6.72

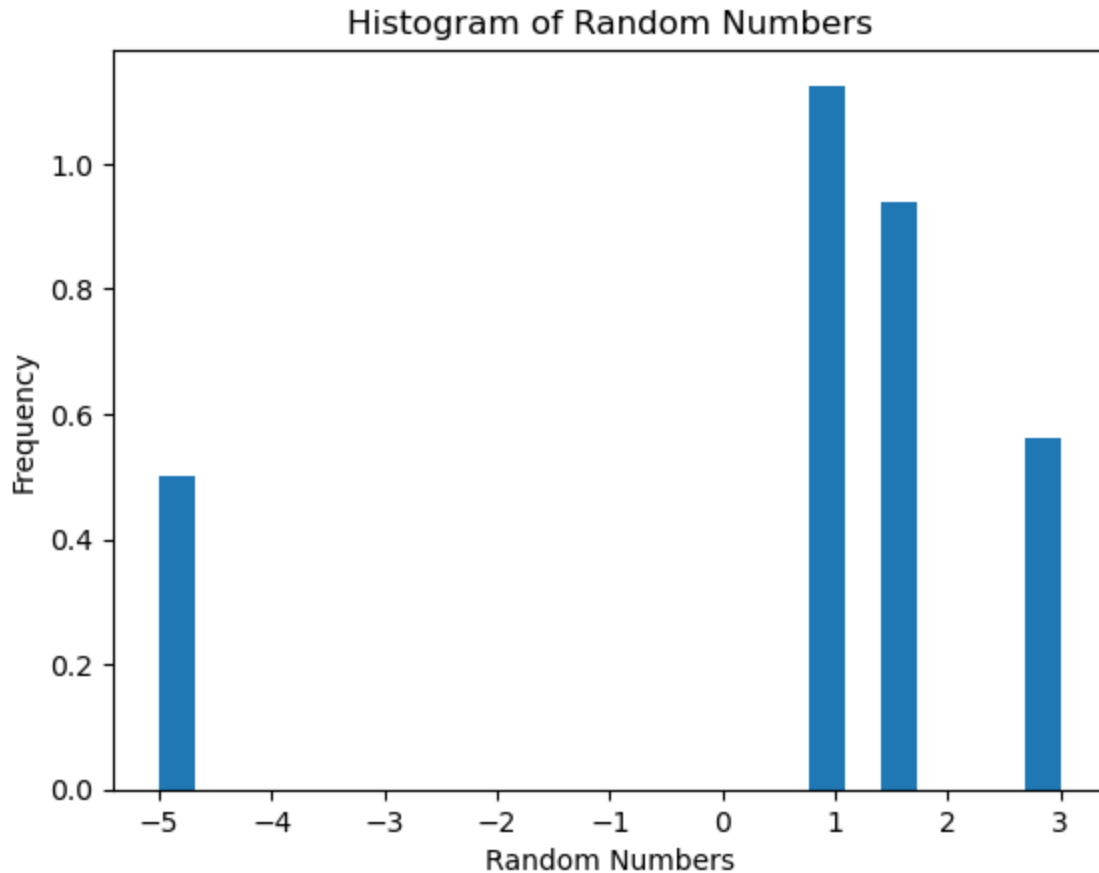
**Se generarán entonces los números para tamaño de muestra n=100**

```
In [240.. random_numbers_fx1_100 = np.random.choice(x1, size=100, p=fx1)
print(random_numbers_fx1_100)
```

```
[ 3.  -5.   3.  -5.   1.5  3.   1.   1.5  1.  -5.  -5.  -5.   1.5  1.5
  1.5  1.   1.5  1.  -5.   1.   3.   1.5  1.5  1.   3.   1.   1.5  1.
  1.   1.5  1.   1.   1.5  3.   3.   3.   1.5 -5.  -5.   1.5  1.   1.
  1.   3.   1.   1.   1.   1.   1.  -5.   1.   3.   1.5  1.  -5.  -5.
  1.  -5.   1.   3.   1.5  1.   3.   1.   1.5  1.5  1.   1.5 -5.   3.
  1.   3.   1.   1.   1.5  1.5  1.   3.   1.  -5.  -5.   1.5  1.   1.5
  1.5  1.5  1.5 -5.   1.5  1.   3.   1.5  1.5  1.   3.   1.   1.5  3.
  1.   1.5]
```

Graficamos el resultado para tamaño de muestra n=100

```
In [241... plt.hist(random_numbers_fx1_100, bins=25, density=True)
plt.xlabel('Random Numbers')
plt.ylabel('Frequency')
plt.title('Histogram of Random Numbers')
plt.show()
```



Calculamos el promedio para tamaño de muestra n=100

```
In [242... import numpy as np

average_random_numbers_fx1_100= round(np.mean(random_numbers_fx1_100),2)
average_random_numbers_fx1_100
```

Out[242]: 0.55

Calculamos la varianza para tamaño de muestra n=100

```
In [243... import numpy as np

variance_random_numbers_fx1_100= round(np.var(random_numbers_fx1_100),2)
variance_random_numbers_fx1_100
```

Out[243]: 6.35

**Se generarán entonces los números para tamaño de muestra n=1000**

```
In [244... random_numbers_fx1_1000 = np.random.choice(x1, size=1000, p=fx1)
print(random_numbers_fx1_1000)
```

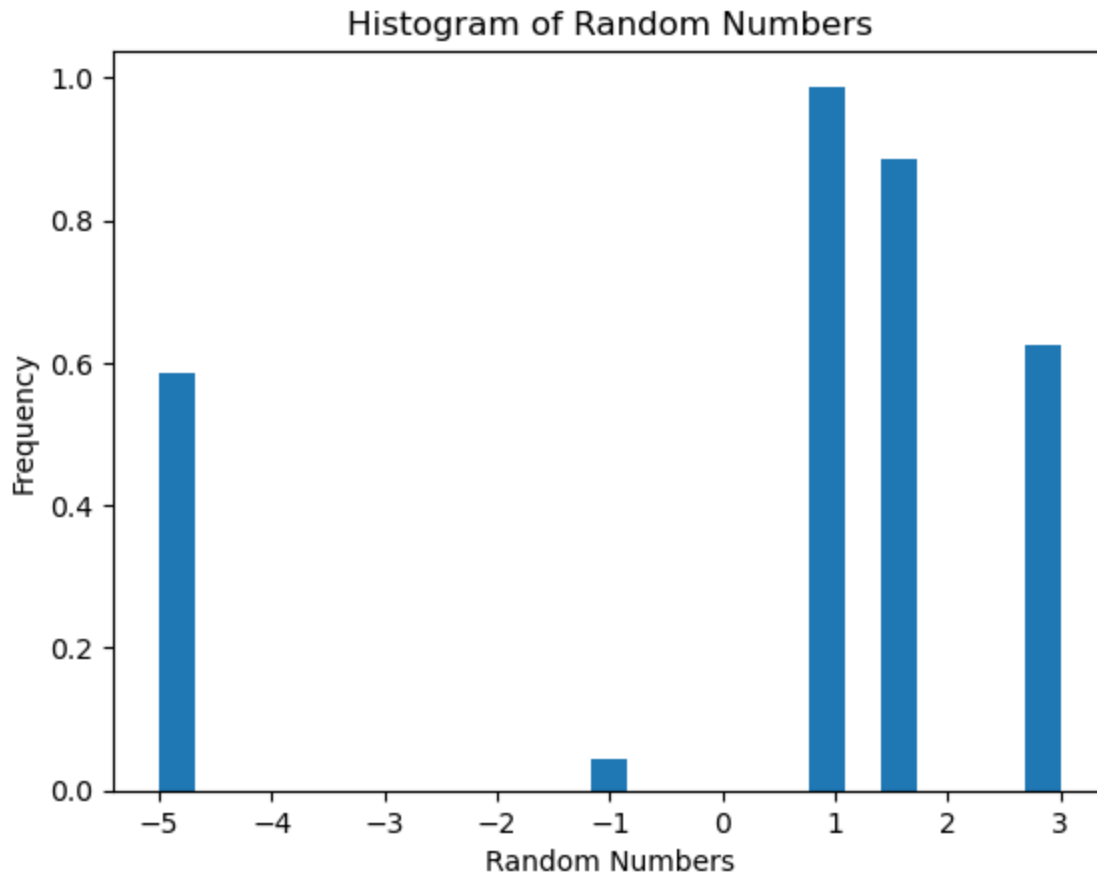
```
[ 1.  1. -5.  3. -5. -5.  1.5  3.  3.  1.5  3. -5.  1.5 -5.
 1.  1. -5.  1.  1.  1.5  1.5  1.5 -5.  1.5  3.  1.5  1.  1.
 1.5  1. -5.  1.  3. -5.  1.5 -5.  3.  3.  1.  1.  1.5  1.5
 1.  3.  1.  1.5  1.  1. -5.  3.  1.  1.  1. -5.  1.  1.5
 1.  1.  1. -5. -5.  1.  1.5  1.5  3. -5.  1.  3.  3.  1.]
```

-5.	-5.	1.	-5.	1.	1.	-5.	1.	3.	-5.	3.	-5.	1.5	-5.
-5.	3.	-5.	1.5	3.	1.	1.	1.5	1.5	1.5	1.	-5.	1.5	3.
1.5	-5.	3.	1.	1.5	3.	-5.	-5.	-5.	1.5	1.5	1.	1.	1.
3.	1.	-5.	1.5	1.5	1.5	1.	3.	1.5	1.	-5.	1.5	1.	1.
-5.	-5.	1.5	1.	-5.	1.	1.	1.5	1.	-5.	1.	3.	1.5	1.
1.	1.5	1.5	3.	3.	-5.	1.	-5.	1.5	1.	1.	-5.	3.	1.
-5.	1.5	-5.	1.	1.5	-5.	-5.	1.	1.5	1.	1.5	1.5	-5.	1.
1.5	1.	1.	3.	-5.	1.5	1.5	-5.	1.	1.5	3.	1.5	1.5	1.5
1.5	1.	1.	-5.	-5.	1.	1.	3.	1.	1.	1.5	1.5	1.5	1.5
1.	3.	-5.	1.5	3.	-5.	1.	1.	1.	3.	1.5	-5.	-5.	1.
3.	1.	1.5	1.	1.	1.	1.5	1.5	3.	-5.	1.	1.5	1.	3.
-5.	1.5	1.5	-5.	1.5	-5.	3.	1.5	1.5	3.	-5.	1.5	1.	-1.
3.	1.5	1.	1.5	1.	1.	1.5	1.5	-5.	1.	1.	1.5	-5.	1.5
1.5	1.	1.5	1.5	3.	1.	1.	1.	-5.	1.	3.	3.	1.5	1.5
-5.	1.	3.	1.	1.	3.	1.	3.	-5.	3.	3.	3.	1.	1.5
1.5	-1.	3.	1.	1.	3.	-5.	1.5	3.	3.	3.	-5.	3.	-5.
1.	1.5	1.	-5.	1.	1.	3.	-5.	-5.	3.	1.	3.	3.	1.5
1.5	1.5	1.5	1.5	1.	1.	1.	1.5	-1.	1.	1.	1.	1.5	1.
3.	1.	1.	1.5	3.	1.	-5.	1.	1.	1.5	-5.	3.	1.	1.5
1.5	1.5	1.	1.	-5.	1.	1.	1.	1.5	1.5	-5.	1.	1.5	1.5
-5.	1.	3.	1.	1.	1.5	1.	1.5	1.5	1.5	1.	1.	1.	-5.
1.	-5.	1.	1.	-5.	1.5	1.	1.	3.	1.	-1.	1.5	1.5	-5.
3.	1.	3.	1.5	-5.	1.	-5.	1.5	1.	3.	-5.	3.	1.5	1.
1.	1.	3.	-5.	1.	-5.	-5.	-5.	-5.	3.	1.5	1.5	3.	1.5
3.	1.	3.	1.5	1.5	1.5	1.	-5.	1.	-5.	-1.	1.5	1.5	3.
1.5	1.	1.5	3.	1.	1.	3.	1.5	1.	1.5	-5.	-5.	-5.	1.
1.	1.5	1.5	1.5	1.	3.	1.5	-5.	3.	-5.	1.5	1.5	1.	1.
1.	1.5	1.5	-5.	3.	1.5	1.5	3.	1.	-1.	1.5	1.	3.	-5.
-5.	1.	3.	1.5	1.	1.5	1.	-1.	1.	3.	1.	3.	1.	1.
1.5	1.5	1.	1.	-1.	1.	1.5	3.	3.	1.	1.5	1.5	1.	-5.
1.5	1.5	1.	3.	1.5	-5.	1.5	3.	3.	1.	-5.	3.	1.5	3.
3.	-5.	-5.	1.5	-5.	-5.	3.	3.	1.5	1.	1.	1.	-5.	-5.
-5.	3.	-5.	3.	-5.	1.	1.	-5.	-5.	1.5	1.5	1.5	1.	-5.
1.5	1.	-1.	-5.	3.	1.5	-5.	1.5	1.5	3.	-5.	1.	1.	1.
3.	-5.	1.5	-5.	1.5	1.5	1.	-5.	1.	1.5	-5.	1.	1.5	3.
1.	1.	3.	1.	1.5	1.5	1.5	1.5	1.	1.	3.	1.5	-5.	1.
1.	1.5	1.5	1.5	1.5	1.	3.	1.5	3.	-5.	1.5	1.	1.5	-5.
1.	-1.	1.	1.5	3.	1.5	1.	1.5	3.	1.5	1.	-5.	3.	-5.
3.	1.5	1.	-5.	3.	1.5	-1.	3.	1.5	1.	-5.	1.5	1.5	1.
1.	-5.	3.	1.5	3.	1.5	1.5	-5.	3.	1.5	1.	1.5	-5.	1.5
3.	3.	3.	3.	1.5	1.5	3.	-5.	1.5	3.	3.	1.	3.	1.5
3.	-5.	-5.	1.	-5.	1.	3.	3.	1.5	3.	1.5	3.	1.5	3.
-5.	1.5	1.	1.5	1.	3.	-5.	-5.	3.	3.	3.	1.5	-5.	1.5
1.5	3.	1.5	3.	1.	1.	1.	3.	3.	3.	3.	1.5	3.	1.5
-5.	1.5	1.5	-5.	1.5	1.5	3.	1.5	1.5	3.	3.	1.5	1.	1.5
3.	1.	3.	1.5	1.5	-5.	1.5	3.	3.	3.	-5.	3.	1.	1.
1.	1.	-5.	-5.	-1.	1.5	1.5	1.	1.	1.5	3.	3.	1.	3.
1.	1.5	3.	3.	1.5	1.5	3.	-5.	3.	3.	1.	1.	3.	1.5
3.	1.5	1.5	1.5	-5.	1.5	1.	3.	1.5	1.5	1.	1.5	1.	3.
1.	1.	1.	1.	1.	1.	1.	1.	1.	-5.	1.5	1.	1.	1.
-5.	3.	-5.	-5.	3.	3.	1.	1.5	1.	1.	3.	1.	3.	1.5
1.	3.	1.	1.	1.	3.	1.	1.	1.	3.	3.	-5.	-5.	1.
-5.	1.	-5.	1.5	-5.	3.	3.	-5.	3.	1.	1.	-5.	1.	-5.
1.5	3.	-5.	1.	1.	-5.	1.	1.	3.	1.	-5.	1.5	-5.	1.5
1.	3.	-5.	-5.	1.	1.	1.5	-5.	1.	1.	1.5	1.5	1.	-5.
1.5	1.	-5.	-5.	-5.	1.	3.	1.	1.5	1.5	1.5	1.	1.	1.5
1.	1.	1.5	1.5	1.5	3.	1.5	1.	-5.	-5.	1.5	3.	1.5	1.5
3.	3.	-5.	1.5	1.	1.	1.	1.5	3.	1.	1.5	1.	1.5	1.
1.	1.	1.	-5.	1.	1.	-5.	1.5	1.	1.	-5.	-5.	1.	1.
1.5	-1.	1.	3.	1.	1.5	1.	1.	1.5	3.	-5.	1.	3.	1.5
3.	1.	1.	1.	3.	1.5	3.	1.5	1.	1.5	1.5	1.	1.	1.
1.5	-1.	1.	1.5	1.	1.	-5.	1.5	3.	1.5	1.5	3.	1.	1.5
1.	1.	1.5	1.	1.5	1.5	-5.	3.	1.	1.5	-5.	1.	-5.	3.
1.	1.5	1.	1.5	3.	1.	1.5	-5.	1.	1.5	1.	-5.	3.	3.
3.	1.5	1.	-5.	-5.	1.5	1.5	3.	-5.	-5.	1.	3.	1.5	1.

-5. -5. 1.5 -5. 1.5 3. 1. 1. 1.5 3. 1.5 1.5 -5. 1.5  
3. 3. 1. 3. 3. 1.5]

Graficamos el resultado para tamaño de muestra n=1000

```
In [245... plt.hist(random_numbers_fx1_1000, bins=25, density=True)  
plt.xlabel('Random Numbers')  
plt.ylabel('Frequency')  
plt.title('Histogram of Random Numbers')  
plt.show()
```



Calculamos el promedio para tamaño de muestra n=1000

```
In [246... import numpy as np  
  
average_random_numbers_fx1_1000= round(np.mean(random_numbers_fx1_1000),2)  
average_random_numbers_fx1_1000
```

Out[246]: 0.39

Calculamos la varianza para tamaño de muestra n=1000

```
In [247... import numpy as np  
  
variance_random_numbers_fx1_1000= round(np.var(random_numbers_fx1_1000),2)  
variance_random_numbers_fx1_1000
```

Out[247]: 7.29

In [ ]: