Estocasticos_Tarea2_Variables_aleatorias_Ibarra_Sergio

March 1, 2023

[ ]:

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

0.2 Alumno: Sergio Ibarra R (414025796)

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

0.3.1 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 femenil 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)

[ ]:

0.3.2 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

Edad	Número de niños
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 una variable aleatoria que indica la edad de un niño seleccionado al azar,

a) Use los datos para construir la función de probabilidad para y trace su gráfica.
b) Calcule la edad promedio de los niños con problemas de aprendizaje..
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.

a) Use los datos para construir la función de probabilidad para 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

[5]:
```
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_semestre\Estocasticos_UNAM\UNAM_Estocasti
↪xlsx', sheet_name=['Problema2'])
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}
```

[5]: dict

[6]:
```
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
```

[6]:
```
pandas.core.frame.DataFrame
```

[7]:
```
df_USA_niños_escolar_edad= df_USA_niños_escolar['Edad']
df_USA_niños_escolar_edad
```

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

[8]:
```
type(df_USA_niños_escolar_edad)
```

[8]:
```
pandas.core.series.Series
```

[9]:
```
df_USA_niños_escolar_edad[0]
```

[9]:
```
6
```

[10]:
```
df_USA_niños= df_USA_niños_escolar['Número de niños']
df_USA_niños
```

[10]:
```
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

[11]:
```
# Transform the series to a list
df_USA_niños_list = df_USA_niños.tolist()

df_USA_niños_list
```

[11]:
```
[37369, 87436, 160840, 239719, 286719, 306533, 310787, 302604, 289168]
```

[12]:
```
# Get the sum of all elements in the list
total_niños_USA = sum(df_USA_niños_list)
total_niños_USA
```

[12]:
```
2021175
```

Calculando las probabilidades y añadiéndolas al df_USA_niños_escolar

[13]:
```
df_USA_niños_escolar['Probabilidad_por_edad'] = (df_USA_niños /
 ↪total_niños_USA)*100
print(df_USA_niños_escolar['Probabilidad_por_edad'])
```

```
0     1.848875
1     4.325998
2     7.957747
3    11.860378
4    14.185758
5    15.166079
6    15.376551
7    14.971687
8    14.306925
Name: Probabilidad_por_edad, dtype: float64
```

[14]:
```
print(df_USA_niños_escolar)
```

```
   Edad  Número de niños  Probabilidad_por_edad
0     6            37369               1.848875
1     7            87436               4.325998
2     8           160840               7.957747
3     9           239719              11.860378
4    10           286719              14.185758
5    11           306533              15.166079
6    12           310787              15.376551
7    13           302604              14.971687
8    14           289168              14.306925
```

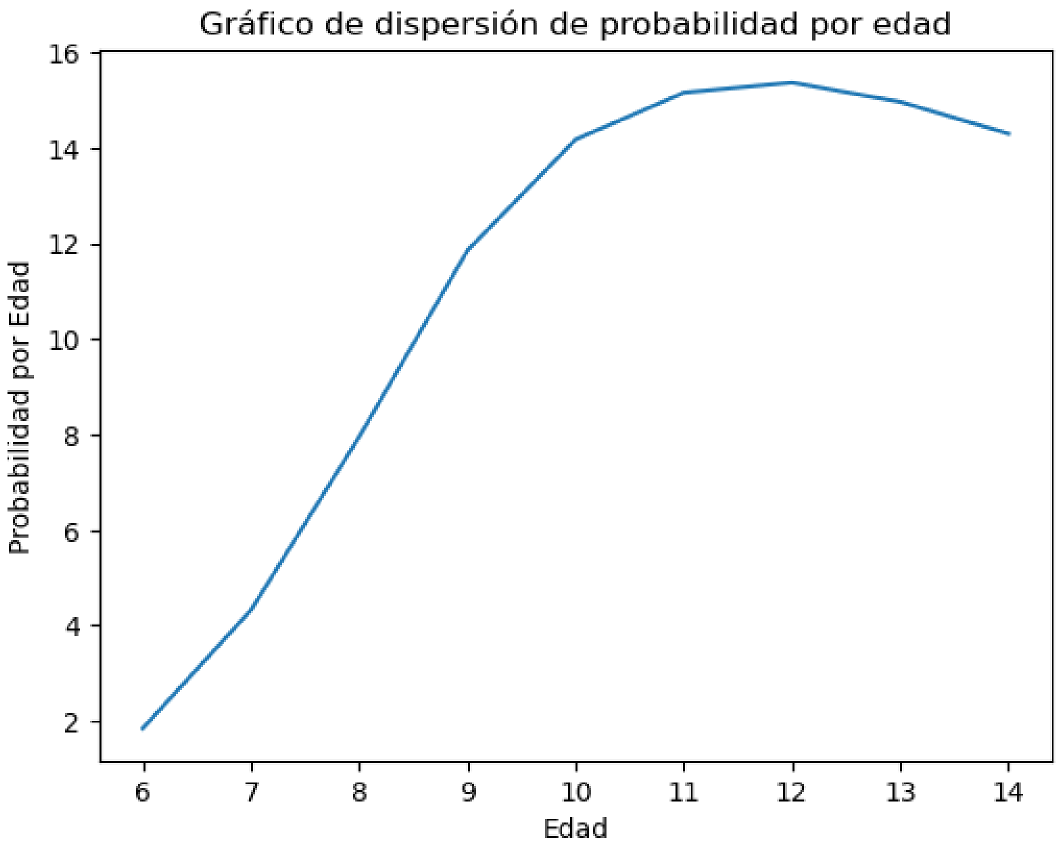
Por lo tanto la función de probabilidad de X sería

Edad	Valor de probabilidad
6	1.85
7	4.32
8	7.96
9	11.87
10	14.19
11	15.17
12	15.38
13	14.97
14	14.31

Y la gráfica de la función de probabilidad de X sería

[15]:
```
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

[17]:
```
print(df_USA_niños_escolar['Probabilidad_por_edad']/100)
```

```
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
```

[18]:
```
esperanza =␣
↪sum(df_USA_niños_escolar['Edad']*(df_USA_niños_escolar['Probabilidad_por_edad']/
↪100))
esperanza
```

[18]: 10.99912575605774

[20]:
```
print(f"_La edad promedio de los niños con problemas de aprendizaje__ es:
↪{esperanza}" + "años")
```

_La edad promedio de los niños con problemas de aprendizaje__
es:10.99912575605774años

[ ]:
```

```

[ ]:
```

```

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'

[ ]:
```

```

[39]:
```
#Transformando los datos de la columna probabilidad en una tipo lista

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

[39]:
```
[0.01848875035560998,
 0.04325998490976783,
 0.07957747349932588,
 0.11860378245327594,
 0.1418575828416639,
 0.15166079137135577,
 0.15376550768736008,
 0.14971687261122862,
 0.143069254270412]
```

[40]:
```
#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
```

[40]: [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

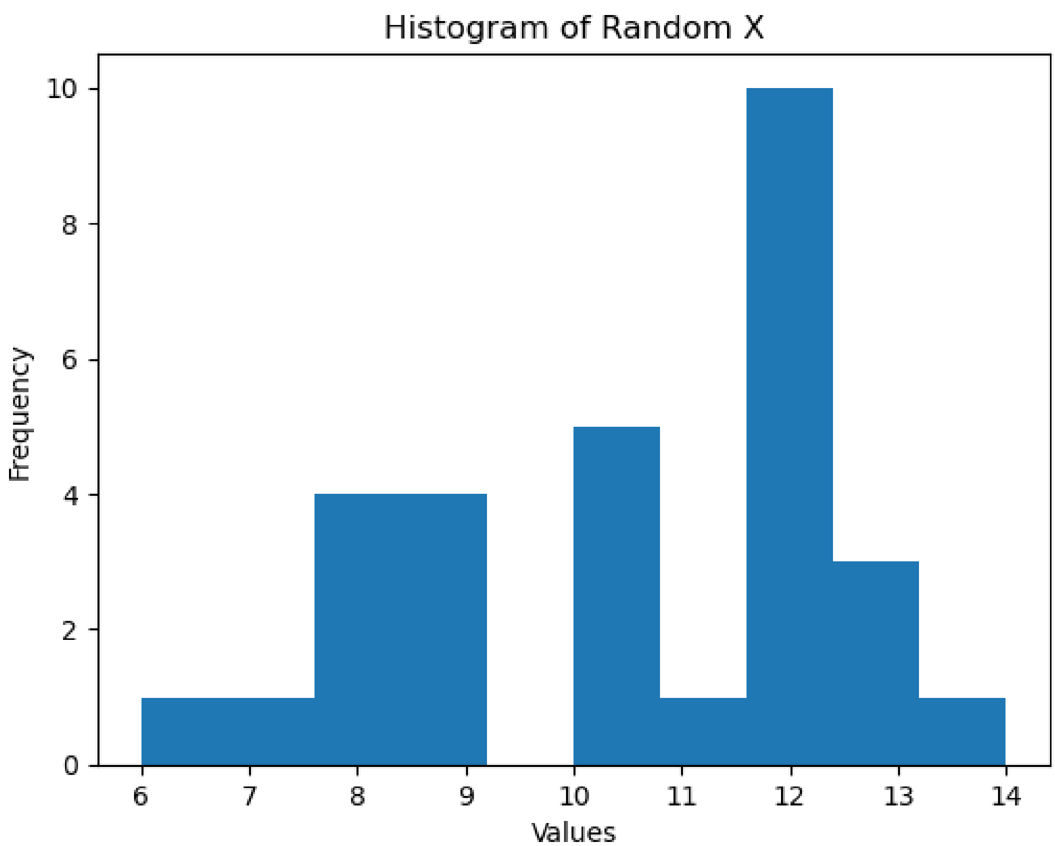
[41]:
```
random_30_edades = np.random.choice(
df_USA_niños_escolar_edad_list, size=30, p=df_USA_niños_probabilidad_list )
random_30_edades
```

[41]:
```
array([ 8,  9, 12, 10, 12, 12, 11,  8, 14, 12, 12,  7, 13,  9, 12, 12, 10,
       10, 13,  9,  8, 13, 10,  8,  9, 10, 12, 12,  6, 12])
```

Vamos a graficar los valores generados

[43]:
```
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

[44]:
```
import numpy as np
```

```
average = np.mean(random_30_edades)

print("La edad promedio de los valores simulados es:", average)
```

```
La edad promedio de los valores simulados es: 10.5
```

ES MUY CERCANA A LA ESPERANZA ANTERIORMENTE CALCULADA DE 10.9!!!

[ ]:

```

```

0.3.3 Problema 3

Considere la variable aleatoria cuya función de distribución (acumulativa) es la que se indica a continuación. a) Decida si la variable aleatoria es discreta o continua. Justifique su respuesta. b) Construya la función ( ). c) Calcule la media y la varianza de Y

[ ]:

```

```

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

Vamos a graficar la funcion de distribución acumulada para la variable Y

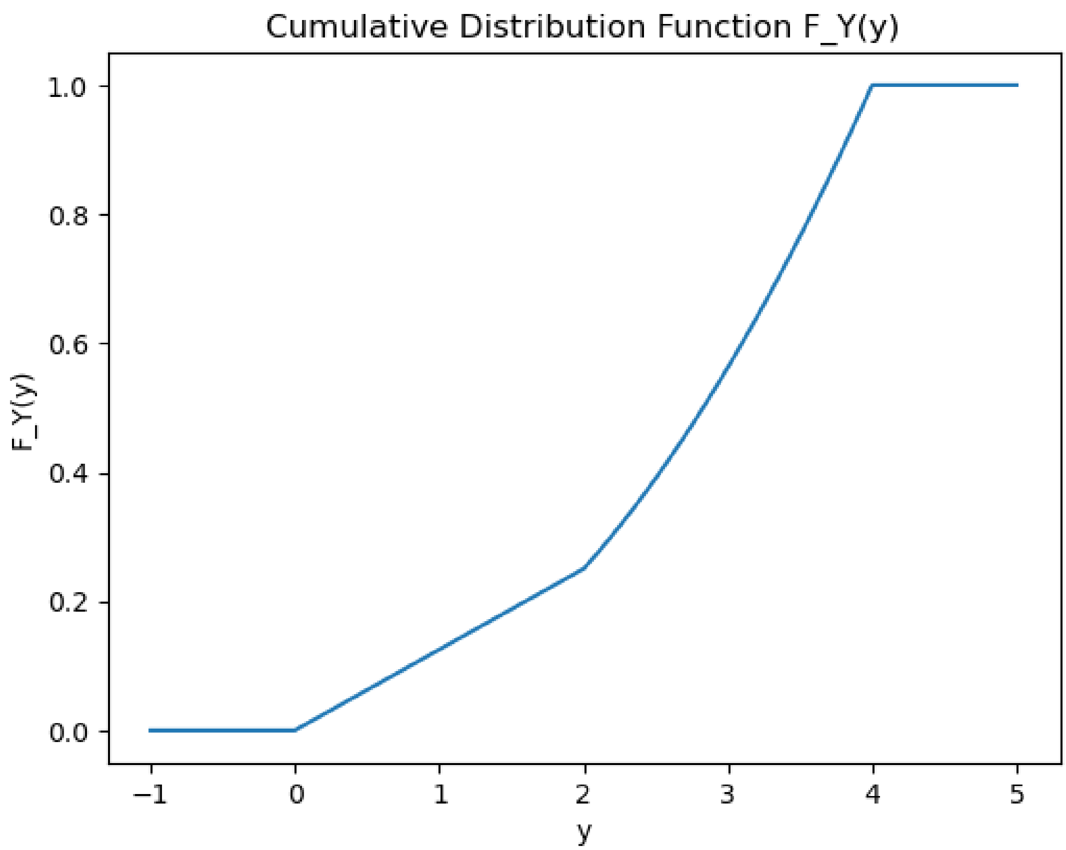
[33]:

```
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()
```



[ ]:

```

```

a) Decida si la variable aleatoria 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

[ ]:

```

```

[ ]:

```
# Transformando los datos de
df_USA_niños_list = df_USA_niños.tolist()

df_USA_niños_list
```

[ ]:

```

```

b) Construya la función ( ). Nosotros sabemos que para cualquier variable aleatoria Y su función de distribución acumulativa se define como:

$$F_y(y) = P(Y <= y)$$

Y sabemos que la relación entre la función de distribución acumulada F(Y) y la función de distribcuión de masa 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 acumuladad de probabilidad Fy(Y) de la siguiente manera:

$$f_y(y) = F_y'(t) = \frac{dFy}{dy}$$

Por lo tanto en nuestro caso habría que deribar cada una de las seccionde de F(y) con respecto a y para obtener f(y)

[ ]:

```

```

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

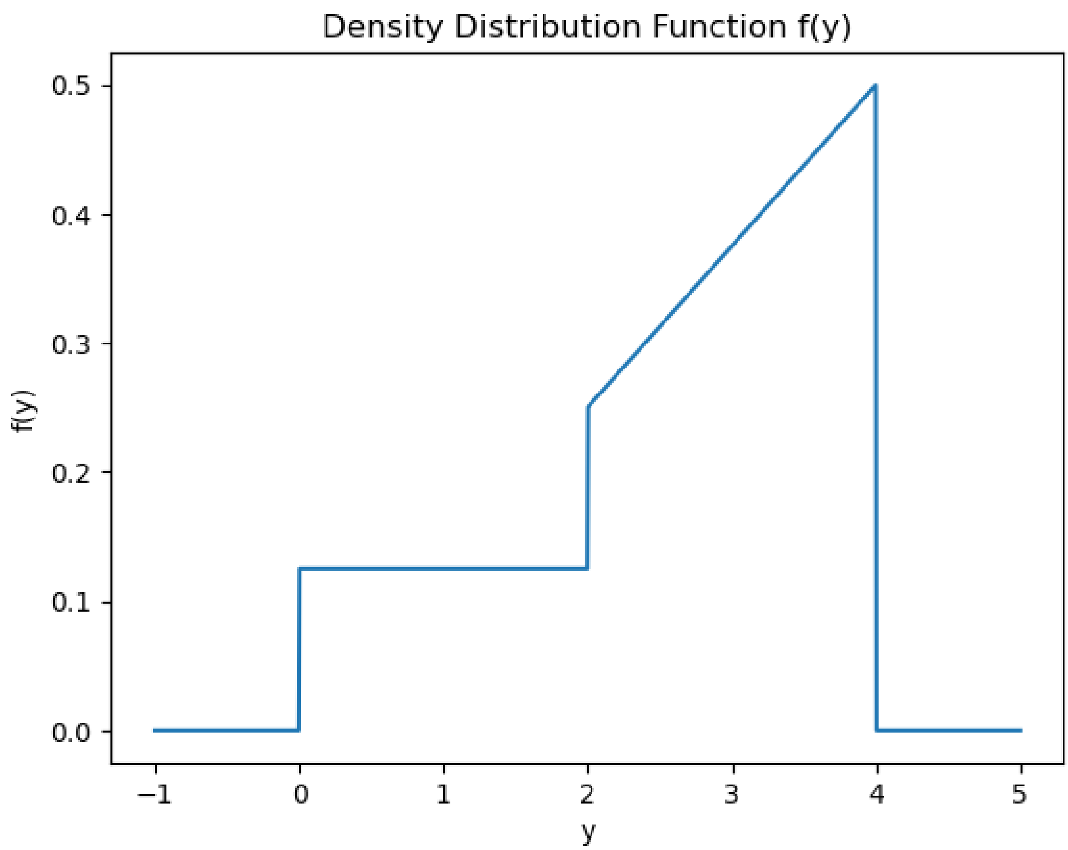
[36]:

```
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()
```



[ ]:

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 definia para el caso de las variables discretas como:

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

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

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

Que para nuestro caso sería:

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

Entonces resolviendo la integral en Python tenemos:

[47]:
```
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/8, 0])

##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 = mean_f(f_y, 0, 4)

print('La media de f(y) es: ', mean)
```

```
La media de f(y) es:  2.5833333333333335
```

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:

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

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

[46]:
```
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/8, 0])
```

```
##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)**2*f(y)
    return quad(integrand, a, b)[0]

##Aqui se usa/llama la función que integra y calcula varianza
variance = variance_f(f_y, mean, 0, 4)

print('La varianza de f(y) es: ', variance)
```

```
La varianza de f(y) es:  1.1597222222222223
```

[ ]:

0.3.4 Problema 4

[ ]:

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

[ ]:

$$f_Y(y) = \begin{cases} c * y^2(1-y)^4, & 0 <= y <= 1 \\ 0, & enotrocaso \end{cases}$$

En donde es una constante.

a) Encuentre el valor de c que hace de ( ) una función de densidad
b) Obtenga la media y la varianza de .
c) Genere 30 valores aleatorios de 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 ( ) 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

[ ]:

[49]:
```python
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 + 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 intetración tenemos:

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

[ ]:
```
Sumando los terminos, tenemos:
```

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

Entonces C= 105

[61]:
```python
import sympy as sp

# Define the constant
c = sp.symbols('c')

# Define the expression
expr = c/7 - 2*c/3 + 6*c/5 - c + c/3
```

```
# Simplify the expression
expr_simplified = sp.simplify(expr)

# Display the result
print("The sum of the expression is:", expr_simplified)
```

```
The sum of the expression is: c/105
```

b) Obtenga la media y la varianza de . La media se calcula como:

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

Que en nuestro caso es:

$$E(X) = \int_{-\infty}^{\infty} y * p(y) = \int_{-\infty}^{\infty} \begin{cases} c * y^2(1-y)^4, & 0 <= y <= 1 \\ 0, & enotrocaso \end{cases} * ydy$$

Entonces resolviendo la integral en Python tenemos:

```
[65]: 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, 0])

##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 = mean_f(f_y, 0, 4)

print('La media de f(y) es: ', mean)
```

```
La media de f(y) es:  0.37499999999999994
```

```
[ ]:
```

Y para el cálculo de la varianza tenemos:

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

[67]:
```
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, 0])

##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)**2*f(y)
    return quad(integrand, a, b)[0]

##Aqui se usa/llama la función que integra y calcula varianza
variance = variance_f(f_y, mean, 0, 4)

print('La varianza de f(y) es: ', variance)
```

```
La varianza de f(y) es:  0.026041666666666664
```

c) Genere 30 valores aleatorios de 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 probabilida de distribución uniforme

[68]:
```
import random

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

print(random_numbers_30)
```

```
[0.10535636880346122, 0.06115679458405121, 0.7387282435229764,
0.16585004123353186, 0.8566979986632247, 0.9124804066617263, 0.8674437240760611,
0.3323966533719981, 0.7166840467592036, 0.10173805187349438, 0.3142553909663095,
0.17247085469518275, 0.81254996705136, 0.6564333368429263, 0.39493052645315807,
0.30699659347409514, 0.6309756685512766, 0.6873721757614752,
0.20859319232062634, 0.9160417821523986, 0.9823623017468084, 0.9890967523027694,
0.9837966092776028, 0.13777661451012835, 0.502271915318776, 0.9939338799946478,
0.9942655497708811, 0.7036178598388086, 0.3733131875464971, 0.1995927597310354]
```

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

Se procede a igular el valor aleatorio generado ri con la funcion de distribucion acumulada:

$$r_i = F(y_i)$$

Que en nuestro caso recodar que

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

Y resolviendo en Python tenemos:

[71]:
```
import sympy as sym

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

# 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:
c*y**7/7 - 2*c*y**6/3 + 6*c*y**5/5 - c*y**4 + c*y**3/3
```

Es decir:

$$F(x_i) = \left[\frac{c*y^7}{7} - \frac{2*c*y^6}{3} + \frac{6*c*y^5}{5} - c*y^4 + \frac{c*y^3}{3}\right] * 105$$

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

$$0.105 = \left[\frac{c*y^7}{7} - \frac{2*c*y^6}{3} + \frac{6*c*y^5}{5} - c*y^4 + \frac{c*y^3}{3}\right] * 105$$

[ ]:
```
```

[74]:
```
import sympy

c, y = sympy.symbols('c y')

# define the equation
eq = sympy.Eq(0.105, (c*y**7/7 - 2*c*y**6/3 + 6*c*y**5/5 - c*y**4 + c*y**3/3) *
 ↪105)

# solve for the roots
roots = sympy.roots(eq.lhs - eq.rhs, y)
```

```
print(roots)
```

```
{}
```

[ ]:

[78]:
```
import sympy

c, y = sympy.symbols('c y')

# define the equation
eq = sympy.Eq(0.105, (c*y**7/7 - 2*c*y**6/3 + 6*c*y**5/5 - c*y**4 + c*y**3/3) *␣
↪105)

# find numerical approximations to the roots
solutions = []
for i in range(7):
    try:
        root = sympy.nsolve(eq.subs(c, 1), i/10)
        solutions.append(root)
    except:
        pass

print(solutions)
```

```
[0.173078028215428, 0.173078028215428, 0.173078028215428, 0.173078028215428,
0.173078028215428, 0.173078028215428]
```

[ ]:

[80]:
```
import sympy
random_numbers_30

c, y = sympy.symbols('c y')

# define the equation
eq = sympy.Eq(0.105, (c*y**7/7 - 2*c*y**6/3 + 6*c*y**5/5 - c*y**4 + c*y**3/3) *␣
↪105)

# iterative function to calculate the roots
def calculate_roots(solutions):
    roots = []
    for random in random_numbers_30:
        try:
            root = sympy.nsolve(eq.subs(c, 1), random)
            roots.append(root)
        except:
```

```
            pass
    return roots

# example usage
roots = calculate_roots(solutions)
print(roots)
```

```
[0.173078028215428, 0.173078028215428, 0.173078028215428, 0.173078028215428,
0.173078028215428, 0.173078028215428, 0.173078028215428, 0.173078028215428,
0.173078028215428, 0.173078028215428, 0.173078028215428, 0.173078028215428,
0.173078028215428, 0.173078028215428]
```

[ ]:

```

```

[83]:

```
import sympy

c, y = sympy.symbols('c y')

# define the equation
eq = sympy.Eq(0.105, (c*y**7/7 - 2*c*y**6/3 + 6*c*y**5/5 - c*y**4 + c*y**3/3) *␣
 ↪105)

def solve_equation(c_val):
    # define a new equation with a numerical value for c
    eq_c = eq.subs(c, c_val)
    roots = []
    for i in range(7):
        try:
            # find a numerical approximation to the root using nsolve
            root = sympy.nsolve(eq_c, i/10, verify=False)
            roots.append(root)
        except:
            pass
    return roots

# example usage
solutions = [0.10535636880346122, 0.06115679458405121, 0.7387282435229764, 0.
 ↪16585004123353186, 0.8566979986632247, 0.9124804066617263, 0.
 ↪8674437240760611, 0.3323966533719981, 0.7166840467592036, 0.
 ↪10173805187349438, 0.3142553909663095, 0.17247085469518275, 0.
 ↪81254996705136, 0.6564333368429263, 0.39493052645315807, 0.
 ↪30699659347409514, 0.6309756685512766, 0.6873721757614752, 0.
 ↪20859319232062634, 0.9160417821523986, 0.9823623017468084, 0.
 ↪9890967523027694, 0.9837966092776028, 0.13777661451012835, 0.
 ↪502271915318776, 0.9939338799946478, 0.9942655497708811, 0.7036178598388086,␣
 ↪0.3733131875464971, 0.1995927597310354]
for sol in solutions:
```

```python
    roots = solve_equation(sol)
    print(f"For c={sol}, roots={roots}")
```

```
For c=0.10535636880346122, roots=[1.01938158277286, 0.813507373418118,
0.813507373418118, 0.813507373418118, 0.813507373418118, 0.813507373418118,
0.813507373418118]
For c=0.06115679458405121, roots=[0.988834158403220, 1.07474938690848,
-14.7884895961030, 0.990588571488919, 1.18017747105169, -0.872199386986636,
0.851340178125456]
For c=0.7387282435229764, roots=[0.196542731368755, 0.196542731368755,
0.196542731368755, 0.196542731368755, 0.196542731368755, 0.196542731368755,
0.847954802599348]
For c=0.16585004123353186, roots=[0.424913750470530, 0.424913750470530,
0.424913750470530, 0.424913750470530, 0.424913750470530, 0.424913750470530,
0.424913750470530]
For c=0.8566979986632247, roots=[0.184582821574864, 0.184582821574864,
0.184582821574864, 0.184582821574864, 0.184582821574864, 0.184582821574864,
0.848222679342362]
For c=0.9124804066617263, roots=[0.179776343576541, 0.179776343576541,
0.179776343576541, 0.179776343576541, 0.179776343576541, 0.179776343576541,
0.848313788374165]
For c=0.8674437240760611, roots=[0.183620083602509, 0.183620083602509,
0.183620083602509, 0.183620083602509, 0.183620083602509, 0.183620083602509,
0.848241616855458]
For c=0.3323966533719981, roots=[0.283522608013146, 0.283522608013146,
0.283522608013146, 0.283522608013146, 0.283522608013146, 0.283522608013146,
0.840962362254430]
For c=0.7166840467592036, roots=[0.199112187674740, 0.199112187674740,
0.199112187674740, 0.199112187674740, 0.199112187674740, 0.199112187674740,
0.847888389308260]
For c=0.10173805187349438, roots=[4.41548923881829, 1.04251685814976,
0.938779708878904, 1.07612891827494, 0.996932889978287, 3.46655624041812,
0.937571396968142]
For c=0.3142553909663095, roots=[0.291675126291277, 0.291675126291277,
0.291675126291277, 0.291675126291277, 0.291675126291277, 0.291675126291277,
0.839080162710113]
For c=0.17247085469518275, roots=[0.413339070876920, 0.413339070876920,
0.413339070876920, 0.413339070876920, 0.413339070876920, 0.413339070876920,
0.413339070876920]
For c=0.81254996705136, roots=[0.188742361408836, 0.188742361408836,
0.188742361408836, 0.188742361408836, 0.188742361408836, 0.188742361408836,
0.848136625539100]
For c=0.6564333368429263, roots=[0.206816381876709, 0.206816381876709,
0.206816381876709, 0.206816381876709, 0.206816381876709, 0.206816381876709,
0.847666957022388]
For c=0.39493052645315807, roots=[0.260586111786944, 0.260586111786944,
0.260586111786944, 0.260586111786944, 0.260586111786944, 0.260586111786944,
0.844411714183162]
```

```
For c=0.30699659347409514, roots=[0.295181241221152, 0.295181241221152,
0.295181241221152, 0.295181241221152, 0.295181241221152, 0.295181241221152,
0.838116611725983]
For c=0.6309756685512766, roots=[0.210417001058699, 0.210417001058699,
0.210417001058699, 0.210417001058699, 0.210417001058699, 0.210417001058699,
0.847550655887729]
For c=0.6873721757614752, roots=[0.202727321732011, 0.202727321732011,
0.202727321732011, 0.202727321732011, 0.202727321732011, 0.202727321732011,
0.847788854765500]
For c=0.20859319232062634, roots=[0.365598553172102, 0.365598553172102,
0.365598553172102, 0.365598553172102, 0.365598553172102, 0.365598553172102,
0.386647644705445]
For c=0.9160417821523986, roots=[0.179484815368601, 0.179484815368601,
0.179484815368601, 0.179484815368601, 0.179484815368601, 0.179484815368601,
0.848319046143871]
For c=0.9823623017468084, roots=[0.174354388892777, 0.174354388892777,
0.174354388892777, 0.174354388892777, 0.174354388892777, 0.174354388892777,
0.848406909793763]
For c=0.9890967523027694, roots=[0.173862961470695, 0.173862961470695,
0.173862961470695, 0.173862961470695, 0.173862961470695, 0.173862961470695,
0.848414881800120]
For c=0.9837966092776028, roots=[0.174249295343861, 0.174249295343861,
0.174249295343861, 0.174249295343861, 0.174249295343861, 0.174249295343861,
0.848408620924169]
For c=0.13777661451012835, roots=[0.493183956935771, 0.493183956935771,
0.493183956935771, 0.493183956935771, 0.493183956935771, 0.493183956935771,
0.493183956935771]
For c=0.502271915318776, roots=[0.232951245609747, 0.232951245609747,
0.232951245609747, 0.232951245609747, 0.232951245609747, 0.232951245609747,
0.846575038086005]
For c=0.9939338799946478, roots=[0.173513116720054, 0.173513116720054,
0.173513116720054, 0.173513116720054, 0.173513116720054, 0.173513116720054,
0.848420511688047]
For c=0.9942655497708811, roots=[0.173489223635329, 0.173489223635329,
0.173489223635329, 0.173489223635329, 0.173489223635329, 0.173489223635329,
0.848420894820859]
For c=0.7036178598388086, roots=[0.200694440268430, 0.200694440268430,
0.200694440268430, 0.200694440268430, 0.200694440268430, 0.200694440268430,
0.847845730980428]
For c=0.3733131875464971, roots=[0.267753496166624, 0.267753496166624,
0.267753496166624, 0.267753496166624, 0.267753496166624, 0.267753496166624,
0.843555998072613]
For c=0.1995927597310354, roots=[0.375629681570648, 0.375629681570648,
0.375629681570648, 0.375629681570648, 0.375629681570648, 0.375629681570648,
0.375629681570648]
```

[ ]:

```
```

[98]:
```
aleatorios_ajustados = []
for sol in solutions:
    roots = solve_equation(sol)
    for root in roots:
        if 0 <= root <= 1:
            aleatorios_ajustados.append(root)
            break
print(aleatorios_ajustados)
```

```
[0.813507373418118, 0.988834158403220, 0.196542731368755, 0.424913750470530,
0.184582821574864, 0.179776343576541, 0.183620083602509, 0.283522608013146,
0.199112187674740, 0.938779708878904, 0.291675126291277, 0.413339070876920,
0.188742361408836, 0.206816381876709, 0.260586111786944, 0.295181241221152,
0.210417001058699, 0.202727321732011, 0.365598553172102, 0.179484815368601,
0.174354388892777, 0.173862961470695, 0.174249295343861, 0.493183956935771,
0.232951245609747, 0.173513116720054, 0.173489223635329, 0.200694440268430,
0.267753496166624, 0.375629681570648]
```

Por lo tanto los valores aleatorios serían:

[99]:
```
aleatorios_ajustados
```

[99]:
```
[0.813507373418118,
 0.988834158403220,
 0.196542731368755,
 0.424913750470530,
 0.184582821574864,
 0.179776343576541,
 0.183620083602509,
 0.283522608013146,
 0.199112187674740,
 0.938779708878904,
 0.291675126291277,
 0.413339070876920,
 0.188742361408836,
 0.206816381876709,
 0.260586111786944,
 0.295181241221152,
 0.210417001058699,
 0.202727321732011,
 0.365598553172102,
 0.179484815368601,
 0.174354388892777,
 0.173862961470695,
 0.174249295343861,
 0.493183956935771,
 0.232951245609747,
 0.173513116720054,
```

```
 0.173489223635329,
 0.200694440268430,
 0.267753496166624,
 0.375629681570648]
```

[ ]:
```
El promedio de los valores simulados es:
```

[100]:
```python
import numpy as np

average_aleatorios_ajustados = np.mean(aleatorios_ajustados)

print("El valor promedio de los valores simulados es :",␣
 ↪average_aleatorios_ajustados)
```

```
El valor promedio de los valores simulados es : 0.314914718612951
```

El valor promedio de los valores simulados es 0.314 que es cercano a la esperanza calculada de la función de 0.37

[ ]:
```
```

[ ]:
```
```

0.3.5 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

[ ]:
```
```

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 ( ) y ( )

[ ]:
```
```

[ ]:
```
```

[ ]:
```
```