Ejercicio18

August 30, 2024

Ejercicio 18:

(a) Tenemos 2 dados con 6 resultados posibles, los cuales sumaremos. Por lo tanto, nuestra variable aleatoria x = dado1 + dado2 vive en un espacio muestral [2,12] de los números naturales, es decir, es discreta.

(b) Todos los resultados en cada dado son equiprobables, luego, veamos individualmente cada valor considerando que el valor que obtenemos de cada dado individualmente es un evento independiente (es decir, un dado no condiciona al otro):

P(x=2) = P_1(1)*P_2(1) = (1/6)(1/6) = 1/36 ; Siendo P_1(1) y P_2(1) las probabilidades de los dados 1 y 2 de obtener como resultado 1.

P(x=3) = P_1(1)*P_2(2) + P_1(2)*P_2(1) = 2/36

P(x=4) = P_1(2)*P_2(2) + P_1(1)*P_2(3) + P_2(1)*P_1(3) = 3/36

P(x=5) = 4/36

P(x=6) = 5/36

P(x=7) = 6/36

P(x=8) = 5/36

P(x=9) = 4/36

P(x=10) = 3/36

P(x=11) = 2/36

P(x=12) = 1/36

Por lo tanto, la distrubución será discreta, simétrica y centrada en x=7.

(c) Ahora la idea es, como en el ejercicio 17, definir la suma acumulada de una distribución aleatoria uniforme e ir separando los valores por categorías según la distribución teórica.

[ ]:
```
#Primero, usamos el generador del ejercicio 16 e importamos numpy y matplotlib para tenerlo a mano:
import numpy as np
import matplotlib.pyplot as plt
def ran(a=1664525, c=1013904223, M=2**32): #Defino el generador con los parámetros solicitados
    ran.current=((a*ran.current+c)%M) #Busca el atributo seed, y lo cambia
```

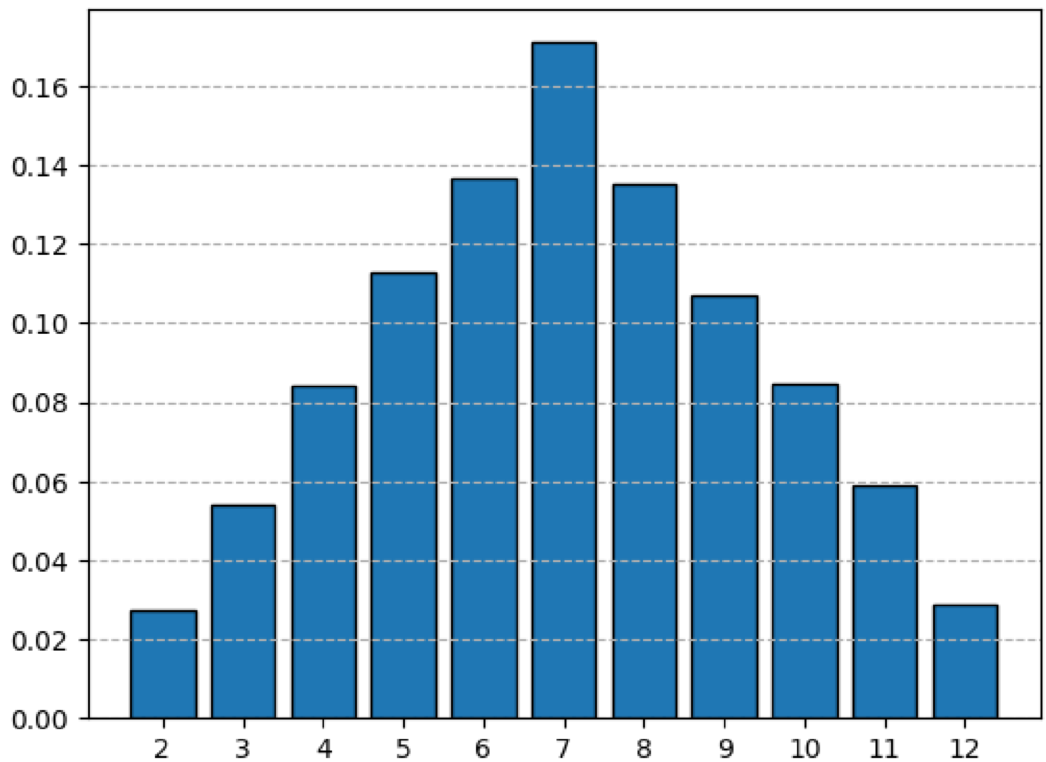
```
    return ran.current/M

ran.current = 45
```

[ ]:
```
#Planteo la muestra:
probabilidades = [1/36 , 2/36 , 3/36 , 4/36 , 5/36 , 6/36 , 5/36 , 4/36 , 3/36␣
↪, 2/36 , 1/36] #Defino las probabilidades
Sumas_posibles = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12] #Defino las sumas posibles
prob_acumuladas = np.cumsum(probabilidades) #Calculo las probabilidades␣
↪acumuladas.
```

[ ]:
```
# Genero una lista aleatoria y las meto en cada tipo en función de si es menor␣
↪a la probabilidad acumulada.
n = 10000  # Cantidad de galaxias a generar
tiradas_sum = []

for i in range(n):
    r = ran() #Genero un número aleatorio
    if r < prob_acumuladas[0]:
        tiradas_sum.append(2)
    elif r < prob_acumuladas[1]:
        tiradas_sum.append(3)
    elif r < prob_acumuladas[2]:
        tiradas_sum.append(4)
    elif r < prob_acumuladas[3]:
        tiradas_sum.append(5)
    elif r < prob_acumuladas[4]:
        tiradas_sum.append(6)
    elif r < prob_acumuladas[5]:
        tiradas_sum.append(7)
    elif r < prob_acumuladas[6]:
        tiradas_sum.append(8)
    elif r < prob_acumuladas[7]:
        tiradas_sum.append(9)
    elif r < prob_acumuladas[8]:
        tiradas_sum.append(10)
    elif r < prob_acumuladas[9]:
        tiradas_sum.append(11)
    else:
        tiradas_sum.append(12)
```

[ ]:
```
categ , frecuencia = np.unique(tiradas_sum, return_counts=True) #Calculo la␣
↪frecuencia de cada suma

prob = frecuencia/n #Calculo la probabilidad de cada suma
```

```
[ ]: plt.bar(categ,prob,edgecolor='black') #Grafico las probabilidades empíricas.
     plt.xticks(Sumas_posibles)
     plt.grid(True, axis='y', linestyle='--') #Agregamos grilla
     plt.show()
```

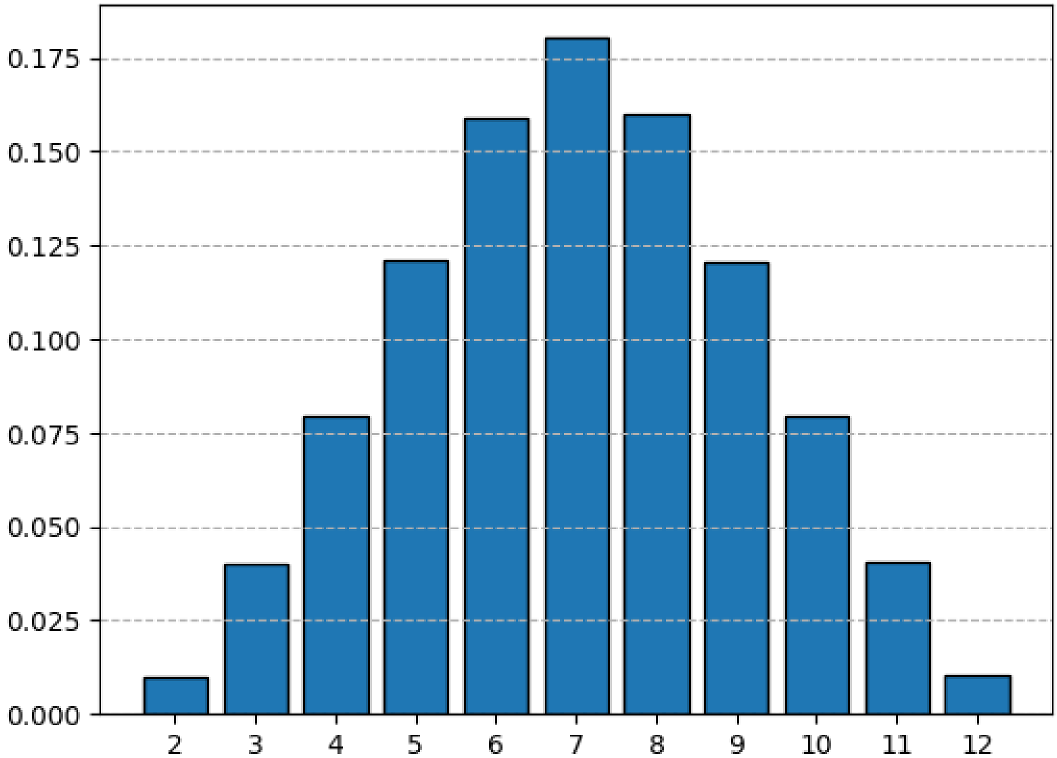


(d) Seguimos usando el generador ran() para ahora simular el experimento:

```
[ ]: #Creemos nuestros dados aleatorios:

     #Generamos una lista de números entre 0 y 1:
     n = 100000
     dado1 = []
     for i in range(n): #Bucle que genera números y los mete en la lista
         dado1.append(round(ran()*5+1)) #Los transformo al intervalo y convierto en␣
      ↪enteros, luego los meto a la lista


     dado1 = np.array(dado1) #La convertimos en array para operar más cómodos
     print(dado1[:5])

     #Repetimos para un 2° dado:
```

```python
n = 100000
dado2 = []
for i in range(n): #Bucle que genera números y los mete en la lista
    dado2.append(round(ran()*5+1)) #Los transformo al intervalo y convierto en␣
 ↪enteros, luego los meto a la lista


dado2 = np.array(dado2) #La convertimos en array para operar más cómodos
print(dado2[:5])

experimento = dado1 + dado2

print(experimento[:5])

categ , frecuencia = np.unique(experimento, return_counts=True)
prob = frecuencia/n #Calculo la probabilidad.

#Lo graficamos

plt.xticks(np.arange(2,13)) #Los ticks del eje x, para escalearlo bien
plt.bar(categ,prob,edgecolor='black') #Grafico las probabilidades empíricas.
plt.grid(True, axis='y', linestyle='--') #Agregamos grilla

plt.show()
```

```
[5 5 5 4 4]
[2 6 3 4 4]
[ 7 11  8  8  8]
```



Podemos observar que la distribución empírica cumple con lo predicho por la teórica.

Conclusión:

En el trabajo hemos creado un generador de números pseudo-aleatorios de congruencia lineal con el cual hemos simulado experimentos y distribuciones teóricas. Observamos entonces como al aumentar la cantidad de números aleatorios las distribuciones y los experimentos se aproximan a los que nos brinda la teoría.