MM3014 - Variables aleatorias

February 18, 2025

1 Adrian Arimany Zamora - 211063

2 Variables aleatorias

Sea $(S, \mathcal{S})$ un espacio muestral.

Un mapeo $X : S \to \mathbb{R}$ se llama **variable aleatoria** si el conjunto $\{\omega \in S : X(\omega) \leq x\} \in \mathcal{S},\ \forall x \in \mathbb{R}$.

[3]:
```
import itertools as it
import pandas as pd
import math
```

2.1 Ejemplo 3.

Supongamos que se lanzan 2 dados. Sea $X : S \to \mathbb{R}$ con $X((m, n)) = m + n$ una variable aleatoria (X es la suma de los dados). Los siguientes son ejemplos de eventos:

[5]:
```
A = set({1,2,3,4,5,6})
S = set(it.product(A,repeat=2))

X1 = set([k for k in S if k[0]+k[1]==1])
X2 = set([k for k in S if k[0]+k[1]<=3])
X3 = set([k for k in S if k[0]+k[1]>=10])
X4 = set([k for k in S if 2<=k[0]+k[1]<=12])

print(f"X=1 es {X1}")
print(f"X 3 es {X2}")
print(f"X 10 es {X3}")
print(f"2 X 12 es {set(list(X4)[0:5])}")
```

```
X=1 es set()
X 3 es {(1, 1), (1, 2), (2, 1)}
X 10 es {(5, 5), (6, 5), (4, 6), (6, 4), (5, 6), (6, 6)}
2 X 12 es {(3, 4), (4, 3), (3, 1), (5, 4), (4, 6)}
```

2.2 Ejercicio 4.

Supongamos que se lanza una moneda cuatro veces. Sea X el número de caras en la secuencia observada una variable aleatoria.

a. Elabore una tabla de frecuencias de los valores de X.

b. Liste los resultados del evento $\{X = 3\}$.

[ ]:
```
### problem a:
def coin(n):
    #Assuming each observed random vaariable is independent, by the multiplication rule we can form a product of independent random variables
    #Here we define p = 1 (as success or the observaed random variable is a head); q = 0 (failure or the observed random variable is a tail)
    return it.product([0, 1], repeat=n)

def freq_table(iterable):
    #Define the list
    freq = {}
    for x in iterable:
        heads_count = sum(x) #We sum all the observed heads in the 4-tuple of (0,1)
        freq[heads_count] = freq.get(heads_count, 0) + 1 #Counts the frequency that there is a success p in the tuple.
    return freq #returns the event {X=0:frequency, X=1: frequency,..., X=4: frequency}

if sum(freq_table(coin(4)).values()) == len(list(coin(4))): #We check if the sum of the 16 possible outcomes is equal to the number of possible outcomes
    print(f"There is a total of {len(list(coin(4)))} possible outcomes")
    print(f"The sum of the frequencies is {sum(freq_table(coin(4)).values())} and it is equal to the number of possible outcomes")
    print(f"The frequencies are: {freq_table(coin(4))}")
else:
    print("Error: The sum of the frequencies is not equal to the number of possible outcomes")

### probelm b:

#Notice the following like saying {X = n}, where n is the observaed value
def exact_event(n):
    outcomes = [x for x in coin(4) if sum(x) == n] #So the list is stored in a single line
    if outcomes:
        print(f"The number of events where (X={n}) is:", len(outcomes))
        print(f"The events where (X={n}) are:", outcomes)
    else:
        print(f"Notice: There are no events where (X={n})")
exact_event(3)
```

```
There is a total of 16 possible outcomes
The sum of the frequencies is 16 and it is equal to the number of possible
```

```
outcomes
The frequencies are: {0: 1, 1: 4, 2: 6, 3: 4, 4: 1}
The number of events where (X=3) is: 4
The events where (X=3) are: [(0, 1, 1, 1), (1, 0, 1, 1), (1, 1, 0, 1), (1, 1, 1,
0)]
```

2.3 Ejercicio 5.

Supongamos que se lanzan tres dados. Sean X la suma de los dados y Y el producto de los dados, variables aleatorias.

a. Elabore una tabla de frecuencias de los valores de X. ¿Cuál es el más frecuente? ¿Cuántas veces ocurrió?

b. Elabore una tabla de frecuencias de los valores de Y. ¿Cuál es el más frecuente? ¿Cuántas veces ocurrió?

c. Liste los resultados del evento $\{X \leq 5\}$.

d. Liste los resultados del evento $\{8 \leq Y \leq 12\}$.

[ ]:
```python
### Inciso a:
def die_sumX(n):
    #Throws the die n times, then sums them up, storing a int; remember that␣
 ↪the die only has 6 sides
    return (sum(x) for x in it.product(range(1, 7), repeat=n))

def freq_table_sumX(iterable):
    #Returns the frequency of each thrown die
    freq = {}
    for x in iterable:
        freq[x] = freq.get(x, 0) + 1
    return freq

#Notice that the highest frequency is not necessary one single observation.
def highest_frequencies(freq):
    highest_freq = max(freq.values())
    return [k for k,v in freq.items() if v == highest_freq]


#The parameter is asking for the number that the 6-sided die will be thrown
def frequency_table_sumX_result(n):
    freq = freq_table_sumX(die_sumX(n))
    if sum(freq.values()) == len(list(die_sumX(n))):
        print(f"The sum of the frequencies is {sum(freq.values())} and it is␣
 ↪equal to the number of possible outcomes")
        print(f"The frequencies are: {freq}")
        print("Also: ")
        highest_freqs = highest_frequencies(freq)
        if len(highest_freqs) > 1:
```

```
            print(f"There are {len(highest_freqs)} highest frequencies of␣
 ↪{max(freq.values())} at X={', '.join(map(str, highest_freqs))}")
        else:
            print(f"The highest frequency is {max(freq.values())} at␣
 ↪X={max(freq, key=freq.get)}")
    else:
        print("Error: The sum of the frequencies is not equal to the number of␣
 ↪possible outcomes")

frequency_table_sumX_result(3)
```

```
The sum of the frequencies is 216 and it is equal to the number of possible
outcomes
The frequencies are: {3: 1, 4: 3, 5: 6, 6: 10, 7: 15, 8: 21, 9: 25, 10: 27, 11:
27, 12: 25, 13: 21, 14: 15, 15: 10, 16: 6, 17: 3, 18: 1}
There are 2 highest frequencies with 27 at X=10, 11
```

[ ]:
```
### Inciso b:

#The coding logic is the same as before, only that this time it stores the␣
 ↪product of the 6-sided die

def die_productY(n):
    return (math.prod(x) for x in it.product(range(1, 7), repeat=n))

def freq_table_productY(iterable):
    freq = {}
    for y in iterable:
        freq[y] = freq.get(y, 0) + 1
    return freq

def frequency_table_productY_result(n):
    freq = freq_table_productY(die_productY(n))
    if sum(freq.values()) == len(list(die_productY(n))):
        print(f"The sum of the frequencies is {sum(freq.values())} and it is␣
 ↪equal to the number of possible outcomes")
        print(f"The frequencies are: {freq}")
        print("Also: ")
        highest_freqs = highest_frequencies(freq)
        if len(highest_freqs) > 1:
            print(f"There are {len(highest_freqs)} highest frequencies of␣
 ↪{max(freq.values())} at Y={', '.join(map(str, highest_freqs))}")
        else:
            print(f"The highest frequency is {max(freq.values())} at␣
 ↪Y={max(freq, key=freq.get)}")
    else:
```

```
        print("Error: The sum of the frequencies is not equal to the number of possible outcomes")

frequency_table_productY_result(3)
```

```
The sum of the frequencies is 216 and it is equal to the number of possible
outcomes
The frequencies are: {1: 1, 2: 3, 3: 3, 4: 6, 5: 3, 6: 9, 8: 7, 10: 6, 12: 15,
9: 3, 15: 6, 18: 9, 16: 6, 20: 9, 24: 15, 25: 3, 30: 12, 36: 12, 32: 3, 40: 6,
48: 9, 50: 3, 60: 12, 72: 9, 27: 1, 45: 3, 54: 3, 75: 3, 90: 6, 108: 3, 64: 1,
80: 3, 96: 3, 100: 3, 120: 6, 144: 3, 125: 1, 150: 3, 180: 3, 216: 1}
Also:
There are 2 highest frequencies of 15 at Y=12, 24
```

[ ]:
```
### inciso c:

# Liste los resultados del evento $\{X\leq 5\}$.


#n = upper limit and die_count = the number of dice thrown
def leq_sum(n, die_count):
    outcomes = [x for x in die_sumX(die_count) if x <= n]
    if outcomes:
        print(f"The number of events where (X<={n}) is:", len(outcomes))
        print(f"The events where (X<={n}) are:", outcomes)
    else:
        print(f"Notice: There are no events where (X<={n})")
leq_sum(n = 5, die_count=3)
```

```
The number of events where (X<=5) is: 10
The events where (X<=5) are: [3, 4, 5, 4, 5, 5, 4, 5, 5, 5]
```

The listed events don't make much sense, as list in this example, so to make sense lets redo some of the code so that the values are stored as a data frame.

[ ]:
```
def die_sumX_df(n):
    return ((x, sum(x)) for x in it.product(range(1, 7), repeat=n))


def leq_sum_df(n, dice_count=3):
    outcomes = [(roll, total) for roll, total in die_sumX_df(dice_count) if total <= n]

    if outcomes:
        df = pd.DataFrame([x[0] for x in outcomes], columns=[f'Die {i+1}' for i in range(dice_count)])
        print(f"The number of events where (X<={n}) is: {len(outcomes)}")
        print(df.to_string(index=False))
```

```
    else:
        print(f"Notice: There are no events where (X<={n})")


leq_sum_df(5)
```

```
The number of events where (X<=5) is: 10
 Die 1  Die 2  Die 3
     1      1      1
     1      1      2
     1      1      3
     1      2      1
     1      2      2
     1      3      1
     2      1      1
     2      1      2
     2      2      1
     3      1      1
```

The idea being that the sum of each row equates to be less or equal to 5. Which if you sum them all and put it in a vector row, you get what I obtained before.

[ ]:
```
### inciso d:

# Liste los resultados del evento $\{8\leq Y\leq 12\}$.

#n = the lower limig, m = the upper limit, and die_count = the number of dice␣
 ↪thrown
def leq_product(n,m, die_count):
    outcomes = [y for y in die_productY(die_count) if n <= y <= m]
    if outcomes:
        print(f"The number of events where ({n}<=Y<={m}) is:", len(outcomes))
        print(f"The events where ({n}<=Y<={m}) are:", outcomes)
    else:
        print(f"Notice: There are no events where ({n}<=Y<={m})")
leq_product(n = 8, m = 12, die_count=3)
#Again what you are seeing as events are just the row vector for each die␣
 ↪multiplied that falls within the defined interval
```

```
The number of events where (8<=Y<=12) is: 31
The events where (8<=Y<=12) are: [8, 10, 12, 9, 12, 8, 12, 10, 12, 8, 10, 12, 8,
12, 12, 8, 10, 12, 9, 12, 12, 9, 12, 8, 12, 8, 12, 10, 10, 12, 12]
```