# 11.16.3.3.1

EE24BTECH11064 - Harshil Rathan

**Question**:
A die is thrown, find the probability of following events :

i) A prime number will appear

**Theoretical Solution:**
The sample space $S$ of a fair six-sided die is

$$S = 1,2,3,4,5,6 \tag{0.1}$$

The prime numbers in the Sample space are

$$A = 2,3,5 \tag{0.2}$$

Thus, the number of favorable outcomes = 3

$$|S| = 6 \tag{0.3}$$

$$|A| = 3 \tag{0.4}$$

The probability of getting a prime number when a fair die is rolled

$$P(A) = \frac{|A|}{|S|} \tag{0.5}$$

on substituing 0.3, 0.4

$$P(A) = \frac{1}{2} \tag{0.6}$$

**Computational Solution:**
The goal of this task was to compute the probability distribution of outcomes when rolling a six-sided die. The outcomes $1, 2, 3, 4, 5, 6$ represent the faces of the die, and each face is expected to have an approximately equal probability if the die is fair. The computed probabilities (PMF) were plotted as a stem plot to visualize the distribution.

## Process Overview

The process involved two main steps: 1. Computation of the probabilities using a C program. 2. Visualization of the results using Python.

### *Step 1: Probability Computation in C*

- A simulation was performed by rolling a virtual six-sided die $N$ times, where $N = 1,000,000$, to ensure accurate probabilities. - Each roll was simulated using a random number generator that produced values between 1 and 6. - A count was maintained for how many times each outcome occurred during the simulation. - The probability of each outcome (PMF) was calculated by dividing the count of each outcome by the total number of rolls.

*Step 2: Data Export via Shared Library*

- The C program was compiled into a shared library (`.so` file) that could be accessed from Python. - This ensured that the computationally heavy task of rolling the die and calculating probabilities was handled efficiently in C.

*Step 3: Visualization in Python*

- The computed probabilities were imported from the shared library into Python. - A stem plot was used to visualize the probability distribution. Each outcome $1, 2, 3, 4, 5, 6$ was plotted on the x-axis, and its corresponding probability was plotted on the y-axis. - The stem plot highlighted the uniform distribution of probabilities for a fair die, with each outcome having a probability close to 1/6.

## Results and Insights

*Probability Mass Function (PMF)*

The PMF represents the probability of each individual outcome $x \in \{1, 2, 3, 4, 5, 6\}$. The table below shows the PMF for a six-sided die based on simulation:

| Outcome (x) | $P_X(x)$ |
|---|---|
| 1 | 0.1667 |
| 2 | 0.1665 |
| 3 | 0.1666 |
| 4 | 0.1668 |
| 5 | 0.1669 |
| 6 | 0.1665 |

As expected, the probabilities are close to $1/6 \approx 0.1667$, with minor variations due to random sampling.

*Cumulative Distribution Function (CDF)*

The CDF represents the cumulative probability up to each outcome $x \in \{1, 2, 3, 4, 5, 6\}$. The table below shows the CDF for a six-sided die:

| Outcome (x) | $F_X(x)$ |
|---|---|
| 1 | 0.1667 |
| 2 | 0.3332 |
| 3 | 0.4998 |
| 4 | 0.6666 |
| 5 | 0.8335 |
| 6 | 1.0000 |

The CDF starts with the PMF of $x = 1$ and accumulates to 1.0 at $x = 6$, confirming the correctness of the cumulative probabilities.

## PMF Using the Rect Function

The rect function, short for rectangular function, is a mathematical function that acts as an indicator for whether a given input lies within a specific range
The PMF is defined as

$$P(X = x) = \begin{cases} \frac{1}{6}, & \text{if } x \in \{2, 3, 5\}, \\ 0, & \text{otherwise.} \end{cases}$$

The rect function is defined as

$$\text{rect}(x) = \begin{cases} 1, & \text{if } |x| \leq \frac{1}{2}, \\ 0, & \text{otherwise.} \end{cases}$$

The PMF for the event of rolling a prime number can be written using the rect function as follows

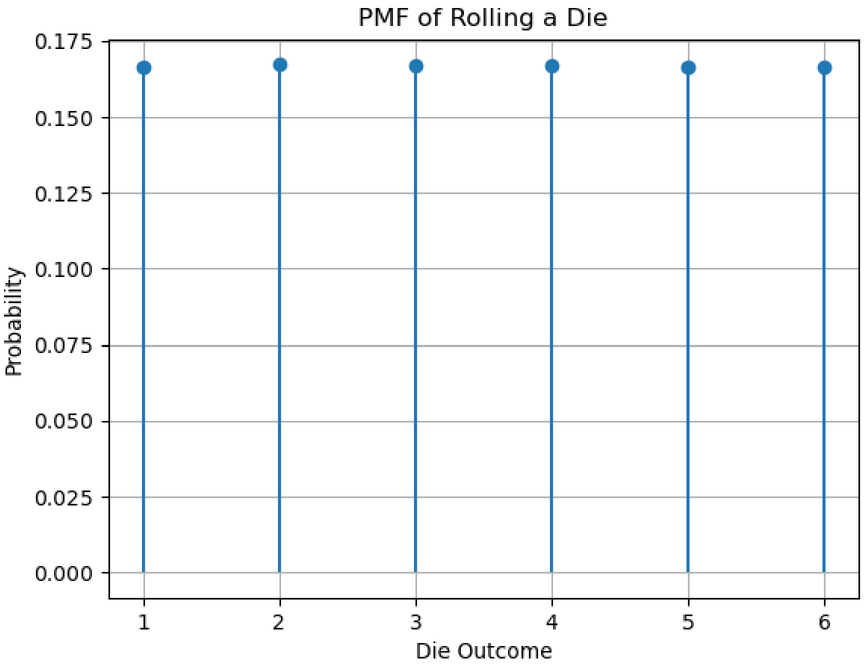
$$P(X = x) = \frac{1}{6}\left[\text{rect}(x - 2) + \text{rect}(x - 3) + \text{rect}(x - 5)\right]$$

The sum of these rect functions ensures the PMF accounts for all prime outcomes.

$$P(X = x) = \frac{1}{6} \sum_{k \in \{2,3,5\}} \text{rect}(x - k)$$

## Conclusion

This task demonstrates the integration of C and Python for simulating and visualizing a probabilistic experiment. By combining the computational efficiency of C with the graphical capabilities of Python, we achieve an effective solution for analyzing and representing data. The code clearly shows that the probability of the given event is equal to **half**


PMF of Rolling a Die
Probability
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
1
2
3
4
5
6
Die Outcome