Probabilistic Dice

M.R. Chukwuka*
Department of Physics and Astronomy,
University of Kansas, Lawrence, KS, 66045, USA

(Dated: February 17, 2023)

I. INTRODUCTION

In probabilistic programming languages, computing the likelihood that an event will occur based on the distribution specified by the program is known as probabilistic inference. As reachability and other well-known tasks of program analysis are generalized by probabilistic inference, inference for a sufficiently expressive language is a very challenging task in program analysis. Scaling inference's secret is to carefully make assumptions about program structure and impose limitations on the types of programs that can be produced while still using a clear and functional language [1]. Dice is a probabilistic programming language focused on fast exact inference for discrete probabilistic programs. This Probabilistic Dice is a program aimed at generating and analyzing data from dice rolls using histograms. The program provides a probabilistic perspective on dice rolling, allowing users to explore the statistical properties of different roll outcomes. A 9 faced dice is used for this analogy. By generating histograms of the face numbers and sums of the rolls, users can gain insights into the probability distributions of the rolls and make informed decisions about betting or strategy. With Probabilistic Dice, you can discover the fascinating world of probabilities and statistics behind the seemingly simple game of rolling dice.

II. HYPOTHESIS EXPLAINING DICE ROLLS

On a set of impartial, fair dice, there is an equal chance of rolling each face number. That is, each face number has a 1/9 chance of appearing on any given roll, which is equal for all face numbers. Furthermore, according to the hypothesis, the total of the face numbers on each roll should have an approximately normal distribution, with the most often sums falling inside the range of probable values (9 and 10) and the least frequent sums falling closer to the edges (2 and 18). This theory is predicated on the notion that the dice are impartial and that each roll is a separate occurrence. However, due to elements like uneven die weighting, flaws in the dice, or non-randomness in the rolling procedure, actual dice rolls may differ from this idealized theory.

* Email: mikemors@ku.edu

III. CODE AND EXPERIMENTAL SIMULATION

A Python program that generates data from dice rolls and plots histograms of the resulting face numbers and sums of face numbers is included in the source code. Data is first loaded from a file by the software (in this case, a file called "Generated.txt" containing the results of a simulated dice roll experiment). The sum of the face numbers for each experiment is then calculated after flattening the data into a one-dimensional array. The Matplotlib package is then used by the program to produce two histograms. The distribution of face numbers across all dice rolls is shown in the first histogram, while the distribution of face number sums across all experiments is shown in the second histogram.Using the "show()" function from the Matplotlib library, which opens a new window displaying the plot, the histograms are presented. One dice was rolled repeatedly while keeping track of the outcomes in the experimental simulation that produced the data used in the software. The "Rgen.py" file contains the data generation program. The purpose of the experiment is to examine the claims that each face number has an equal chance of rolling, and that the sums of face numbers have a normal distribution. The theoretical predictions based on the hypothesis might then be compared to the experimental data using statistical tests and measurements like chi-squared tests, p-values, and standard deviation.

IV. ALGORITHM ANALYSIS

The algorithm for the code is as follows; The code Checks for the "-input" flag in the command-line arguments. If found, extract the filename of the input file containing the data (Generated.txt). It Loads the data from the input file using the NumPy library's "loadtxt" function. This produces a two-dimensional NumPy array containing the face numbers of the dice rolls for each experiment. It Flattens the two-dimensional array into a one-dimensional array using the NumPy library's "flatten" function. This produces a single array containing all of the face numbers for all the experiments. It Calculates the sum of the face numbers for each experiment using the NumPy library's "sum" function with axis 0. It Creates a histogram of the face numbers using the Matplotlib library's "hist" function. This plots the distribution of the face numbers across all the experiments. It also Cre-

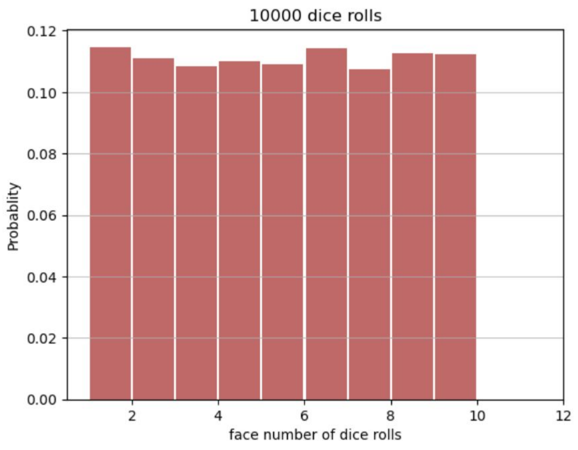

FIG. 1. A distribution of the face numbers rolled in each dice experiment. Each bar in the histogram represents the number of times a particular face number (1 to 9) was rolled across all the experiments. The x-axis shows the face numbers and the y-axis shows the probability of rolling that face number, which is normalized to be between 0 and 1.

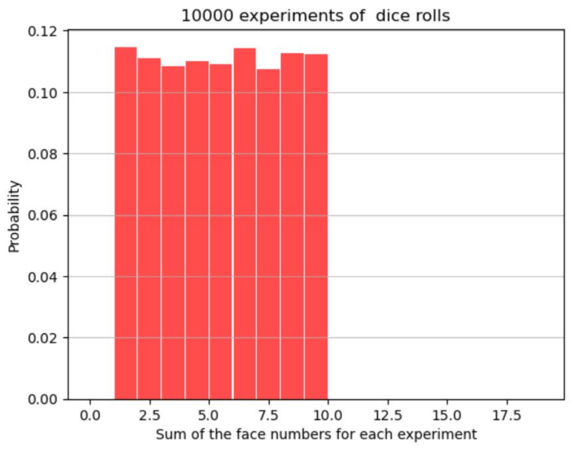

FIG. 2. A distribution of the sums of the face numbers rolled in each dice experiment. Each bar in the histogram represents the number of times a particular sum was obtained across all the experiments. The x-axis shows the possible sums (ranging from the minimum possible sum to the maximum possible sum) and the y-axis shows the probability of obtaining that sum, which is normalized to be between 0 and 1.

ates a histogram of the sums of the face numbers using the Matplotlib library's "hist" function. This plots the distribution of the sums of the face numbers across all the experiments. It Formats the histograms with axis labels, titles, and grid lines using the Matplotlib library's formatting functions. It Displays the histograms in separate windows using the Matplotlib library's "show" function. Overall, the algorithm reads in the data, processes it to extract the relevant information, and creates visualizations of the data using histograms. The code uses the NumPy and Matplotlib libraries to carry out these operations.

V. CONCLUSION

This is a Python script that loads data from a file, generates two histograms and displays them using Matplotlib.The output is essentially what is desired, 2 histograms analysing probabilistic data generated from another python script and stored in a text file. All the python scripts and file are stored in the same directory. These histograms can be used to analyze the statistics of the dice rolls and make predictions about the outcomes of future rolls. For example, we can see that the first histogram is roughly uniform, which suggests that each face number has an equal probability of being rolled. The second histogram shows that the most common sums are around the middle of the possible range (9 and 10), and less common sums are those closer to the edges (2, 18). This information can be useful in deciding which sum to bet on, for example.

VI. REFERENCE

[1] Steven Holtzen, Guy Van den Broeck, and Todd Millstein. 2020. Scaling Exact Inference for Discrete Probabilistic Programs. Proc. ACM Program. Lang. 4, OOPSLA, Article 140 (November 2020), 31 pages. https://doi.org/10. 1145/3428208.