Modeling Dice Rolls

Hannah Lorrimore

December 10, 2014

1 Introduction

A normal dice has 6 sides, each with equal probabilities of being rolled (1/6). Each time we roll the dice, it does not affect the next roll, so we can say that each roll is independent from eachother. We can use various methods of modelling this. We can consider either modeling it using just one roll, and having the outcome as success or failure of getting a desired number. We can also consider multiple rolls, and look at models concerning this. Lastly we can consider non fair die, where the probability of rolling each number is not the same and can be greater or less than 1/6.

2 Mathematical Models

2.1 Bernouilli Experiments

A simple singular dice roll can be modelled as a Bernouilli Experiment, where the outcome is either success or failure. In this case, I will consider success rolling a 1, and failure not rolling a 1. In this case we say p is the probability of success, and q is the probability of failure, such that $q=1-p$. Hence p = 1/6 and q = 5/6. As each roll is independent, the expected value $E(X)$ or the mean/average, is simply p. [1].

2.2 Binomial Trails

If we roll the dice n times, we can now model this Binomially, where p is still the probablity of success (rolling a 1). Each roll of the dice is independent, hence p remains the same. It can be shown that the expected value (or mean) is simply: $E(X)=np$. So if we were to roll a normal dice 60 times, and p = 1/6, we would expect E(X) to be 10, as we expect to get 10 successes per 60 rolls of the dice, i.e. that 1/6 of the rolls are a success. [2].

3 Code Based Model

3.1 Simple Dice Roll Function

We can use the random function in sage/python to model a dice roll. As following the models above, we're only interested in one outcome of the dice, so the coding reflects this.

```
from __future__ import division
"""
If in Python, the above code allows us to enter fractions as x/y
and have the output as a float instead of an interger.
"""
import random  # Allows us to use the random function


def diceroll(num, y):
    """
    Function to emulate dice rolling.

    Arguments:
        num: number of times dice is rolled
        y: allows you to adjust the probability of the dice roll, to emulate non-fair die.

    Outputs:
        Returns success or fail, depending on what number is randomly generated and your value of y.
    """
    while num > 0:
        x = random.random()  # Makes x a random number between 0 and 1
        if x >= 0 and x <y * 1/6:  # Sets the probability of achieving the wanted number
            print "Success"
        else:  # When unwanted number is rolled
            print "Fail"
        num -= 1
```

3.2 Normal Dice Roll Graph

We can then use sage to plot a graph with the x axis being the number of times the dice was rolled, and the y axis as the fraction of times the desired number appeared. The code for which can be seen here: `https://cloud.sagemath.com/projects/5410fc03-882f-4df0-a280-d37b01305eeb/files/grapha.sagews` and the outcome can be seen in Figure 1. From this graph we can see that as n is small, the spread of the fractions on the y axis is larger than as higher values of n. As n increases, the points appear to be converging on a single value. I would expect this convergence to increase if the maximum value of n was raised to a higher value.

3.3 Non-fair dice

We can repeat this process with non-fair dice, where the probability of rolling a specific number is not equal to 1/6. If we plot these different dice rolls on the same graph, we can compare any overlap. The code for which can be seen here: `https://cloud.sagemath.com/projects/5410fc03-882f-4df0-a280-d37b01305eeb/files/graphb.sagews` and the outcome is seen in Figure 2. The graph shows us 6 types of dice, ranging from when the probability of success is 0 to 1. For 0 and 1, the points form a singular line. This is because if the probability of success is 1, failure cannot occur, similiarly, if the probability of success is 0, only failure can occur. For the values inbetween 0 and 1, a similar pattern can be seen in Figure 1, where the spread is larger for smaller values of n and the spread decreases as n increases. It can also be seen that for smaller values of n, the spread of points can overlap, and if the points were all the same colour it would be difficult to determine which value of k the points came from. If you considered this in real life applications, if you wanted to roll a dice to determine if it was fair or not, you would have to roll it over 100 times (estimation) to have a fair idea if it was fake or not.

3.4 Average probability of success

We can use sage to take an average value of all these points to determine an average fraction across all rolls undertaken. Sage can then be used to put this as a single line on the graph, in order to compare it to the spread of the points. If we include the actual probabilites as similar lines, we can then compare the average value to it. The code for this can be seen here: `https://cloud.sagemath.com/projects/5410fc03-882f-4df0-a280-d37b01305eeb/files/graphc.sagews` and the outcomes can be seen as Figure 3 and Figure 4. For Figure 3 the maximum value for n is only 10, so only a small ammount of points were taken into account. This graph shows us that the average taken compared to the actual probabilites can vary, except for where the probability of success in the diceroll function was 0 or 1, which is to be expected. However when we compare this to Figure 4, where the maximum value of n is 500, the average value is close (nearly identical to) the actual value. It can also be seen that the line for the average value for each plot appears to be going through the center of all the points for its plots, and as n increases the points appear to be converging to that line. From this we can make an assumption that as we increase the maximum n, the average value converges to the actual probability.

4 Conclusions

While using maths gives us a good estimation of what sort of values we expect, modeling through computing gives us a better idea of how real life values will occur. The computer modeling also confirms the mathematical modeling, as the values gained from rolling the dice appear to have an average value of the actual probabilities.

References

[1] David Stirzaker Geoffrey Grimmett. *Probability and Random Processes.* Oxford University Press, 2001.

[2] Elliot Tanis Robert Hogg. *Probability and Statistical Interference.* Pearson, 2014.

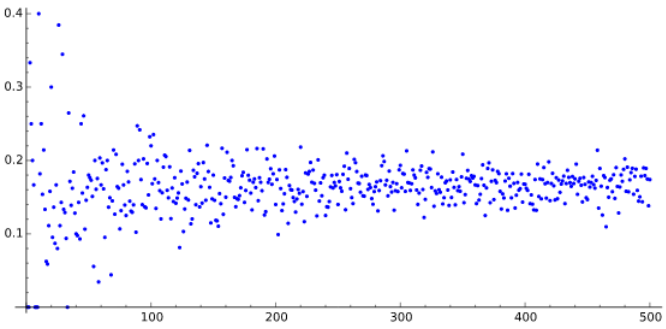

Figure 1: Graph showing fraction of times the desired number was rolled per trial, where p = 1/6 and $1 \leq n \leq 500$.

Figure 2: Graph showing fraction of times the desired number was rolled per trial, where $p = k/6$ and $1 \leq n \leq 500$. For p = 0: the points are red, p = 1/6: orange, p = 2/6: green, p = 3/6: blue, p = 4/6: purple, p = 5/6: pink, p = 1: black.

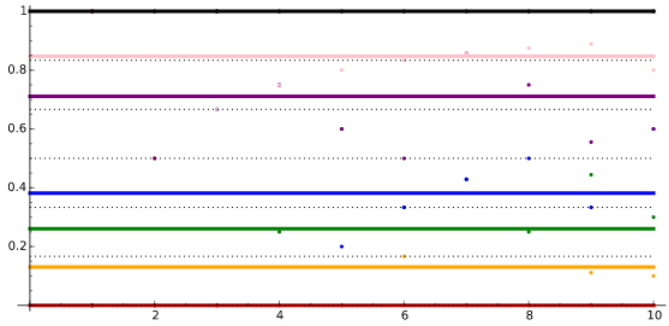

Figure 3: (for colouring system, see Figure 2) Graph showing fraction of times the desired number was rolled per trial, where $p = k/6$ and $1 \leq n \leq 10$. Graph also includes average value for each k as single line, and actual probabilities for each dice.

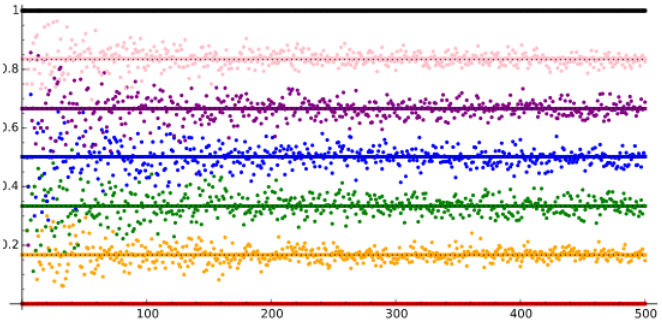

Figure 4: (for colouring system, see Figure 2) Graph showing fraction of times the desired number was rolled per trial, where $p = k/6$ and $1 \leq n \leq 500$. Graph also includes average value for each k as single line, and actual probabilities for each dice.