Playing with Dice

Tidewater Big Data Enthusiasts
Chuck Cartledge
Developer

June 16, 2021

Contents

List of Tables

List of Figures

1 Introduction

We will be exploring how the number of dice, and their pips interact by focusing on:

- how the "expected value" for a particular dice and pip combination changes when either the number of dice, or the number of pips change,
- how the distribution of pip summations is affected by the number of dice and pips, and
- how different implementations of dice and pip interaction tools affect our ability to explore this problem domain.

We'll do this by creating a tool to help us explore this domain, then look at the classic 2-dice 6-pip combination world. After developing a set of equations for this "world," we'll use them to help us explore larger worlds where the number of dice remains constant, but the pips change, then where the pips remain constant, but the dice change.

All source code, and an interactive 3D web page are included in this report.

2 Analysis

As with most dice analyzes, we will start with the classic 2 dice, 6 pips per die. Then we will expand the analysis to support different number of dice and different number of pips.

2.1 Algorithms

A few simple equations are useful when talking about the different values the dice can sum to are needed (see Table 1). Using these ideas, we can enumerate the many combinations possible with 2 dice and 6 pips (see Table 4). Tabular data works well for "small" numbers of values, but graphic visualizations can offer different insights (see Figure 1).

Two different implemenations of a recursive approach to constructing all possible dice and pip combinations were written, and are included in this report (see Section A). The recursive approach uses a function to recursively assigns a value to each die, except for the last die, where all values are assigne and the sum of all dice computed. At the end of the program run, the number of occurences for each possible combination is reported. These recursive approaches work, well enough for small numbers of dice and for small numbers of pips (see Table 2). Based on the measured program execution times, it was decided that if $NumberOfCombinations$ <4,096 then the R implementation was fast enough. For $NumberOfCombinations$ >4,096 the C++ implementation would be used. After repeated runs of the program, a better (more timely) solution was desired.

A complete binonial based solution was found in [2] for the probability of a specific dice sum (see Equation 1). Give the probability of a certain value from a set of possible values

Table 1: Useful equations. A collection of equations and concepts used when talking about dice and associated values.

Variable		Derivative
$Dice$	is	the number of die being considered
$MinPip$	is	the lowest number of pips on a die (usually 1)
$MaxPips$	is	the maximum number of pips on a die (usually 6)
Sum	is	the current sum of all $Dice$ pips
$MinValue$	=	$Dice * MinPips$
$MaxValue$	=	$Dice * MaxPips$
$NumberOfCombinations$	=	$(MaxPips - MinPips + 1)^{Dice}$
$EVDie$	=	$\sum_{i=MinPips}^{MaxPips} \frac{i}{MaxPips-MinPips+1}$
$EVDice$	=	$Dice * EVDie$

and the total number of value, we can compute the number of occurences of that value within the set.

$$\begin{aligned} limit &= \frac{Sum - Dice}{MaxPips} \\ p &= \frac{1}{MaxPips^{Dice}} \sum_{l=0}^{limit} (-1)^l \binom{Dice}{l} \binom{Sum - MaxPips * l - 1}{Dice - 1} \end{aligned} \tag{1}$$

By iterating between $MinValue$ and $MaxValue$, computing the probability of each value, we can compute the number of occurences for each possible value. Equation 1 is significantly faster than either the versions of the recursive approachces (see Table 2). Using the Uspensky equation (see Equation 1), we can now explore other dice and pip combinations in a timely manner.

Table 2: Measured performance of different approaches in seconds. Each combination of pips per die and number of dice can have 0, 2, or 3 values. The first value is a recursive R approach. The second value (in parentheses) is the execution time of a recursive C++ program mimicking the R implementation. The third value in square brackets in a R implementation of a binomial algorithm. In all cases, all possible combinations were computed and reported.

Num. Dice	**Pips per Die**				
	2	**3**	**4**	**5**	**6**
2	0.023 (0.028) [0.01]				
3		0.032 (0.031)			
4			0.032 (0.03)		
5				0.038 (0.03)	
6					0.106 (0.033)
7					0.82 (0.042)
8	0.027 (0.031)		0.164 (0.041)	1.083 (0.057)	3.098 (0.113)
9	0.03 (0.031)	0.172 (0.033)	0.847 (0.047)	3.846 (0.135)	16.899 (0.799)
10	0.034 (0.033)	0.172 (0.033)	2.364 (0.099)	17.806 (0.884)	107.479 (3.523) [0.1]

Table 3: Effect of adding one die to a set of dice.

Attribute	**Behavior**
$MinPossibleValue$	moves one $MinPips$ to the right
$MaxPossibleValue$	moves one $MaxPips$ to the right
$EVPossibleValue$	moves one $EVDie$ to the right

2.2 Classic 2-dice analysis

Table 4: All possible combinations of 2 die, each with 6 pips. Drawing of the LaTeX dice comes from[1].

Sum	**Combinations**	**Occurrences**	**Probability**
<2	–	0	0
2	⚀ ⚀	1	$\frac{1}{NumberOfCombinations}$ (0.0277)
3	⚀ ⚁ or ⚁ ⚀	2	$\frac{2}{NumberOfCombinations}$ (0.0555)
4	⚀ ⚂ or ⚁ ⚁ or ⚂ ⚀	3	$\frac{3}{NumberOfCombinations}$ (0.0833)
5	⚀ ⚃ or ⚁ ⚂ or ⚂ ⚁ or ⚃ ⚀	4	$\frac{4}{NumberOfCombinations}$ (0.1111)
6	⚀ ⚄ or ⚁ ⚃ or ⚂ ⚂ or ⚃ ⚁ or ⚄ ⚀	5	$\frac{5}{NumberOfCombinations}$ (0.1388)
7	⚀ ⚅ or ⚁ ⚄ or ⚂ ⚃ or ⚃ ⚂ or ⚄ ⚁ or ⚅ ⚀	6	$\frac{6}{NumberOfCombinations}$ (0.1666)
8	⚁ ⚅ or ⚂ ⚄ or	5	$\frac{5}{NumberOfCombinations}$ (0.1388)

(Continued on the next page.)

Table 4. (Continued from the previous page.)

Sum	Combinations	Occurrences	Probability
	⚃ ⚃ or ⚄ ⚂ or ⚅ ⚁		
9	⚂ ⚅ or ⚃ ⚄ ⚄ ⚃ or ⚅ ⚂	4	$\frac{4}{NumberOfCombinations}$ (0.1111)
10	⚃ ⚅ or ⚄ ⚄ or ⚅ ⚃	3	$\frac{3}{NumberOfCombinations}$ (0.0833)
11	⚄ ⚅ or ⚅ ⚄	2	$\frac{2}{NumberOfCombinations}$ (0.0555)
12	⚅ ⚅	1	$\frac{1}{NumberOfCombinations}$ (0.0277)
>12	–	0	0

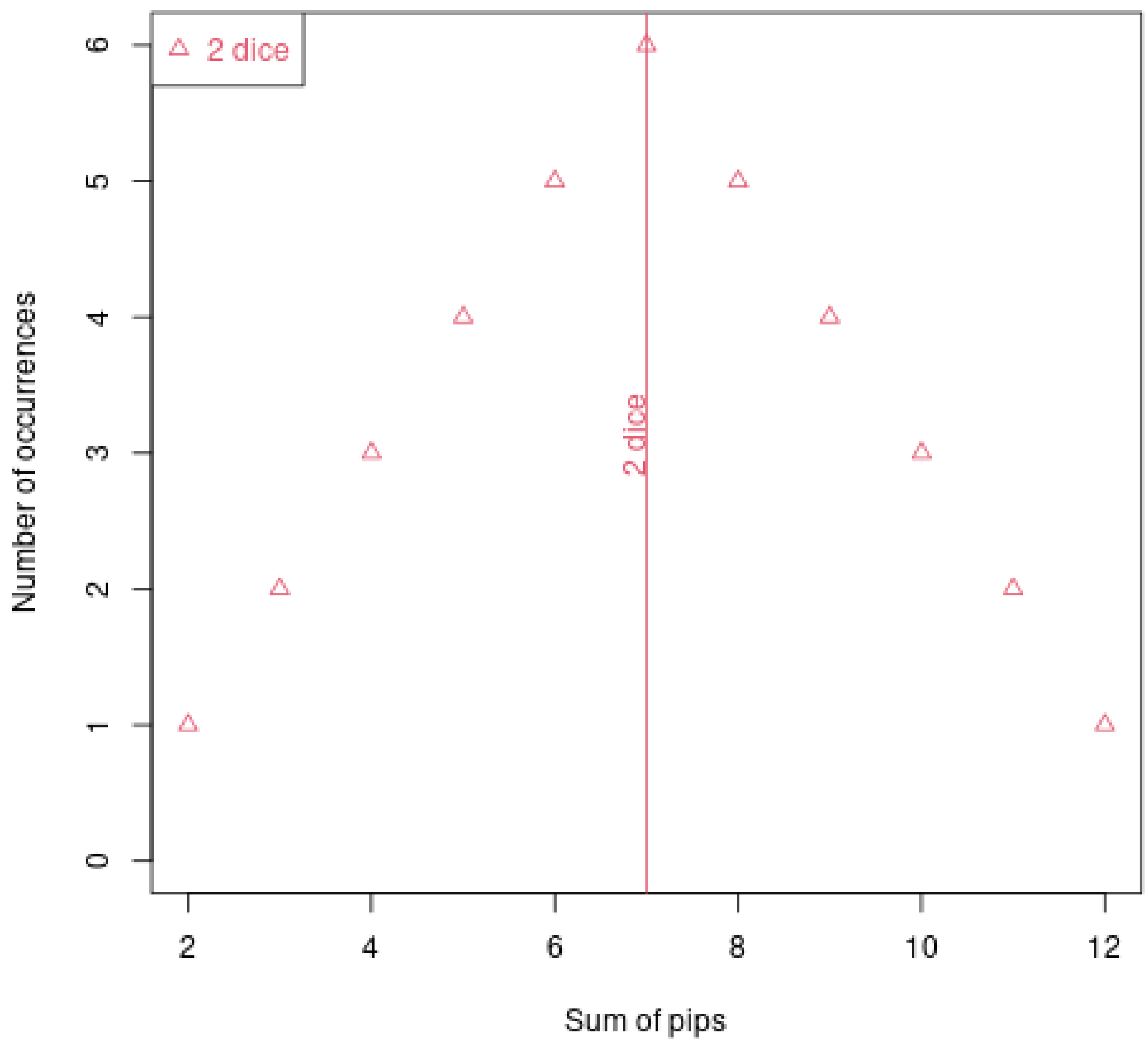

Figure 1: Histgram of summations of 2 dice each with 6 pips. All possible pip sums are computed and the number of occurrences of each is shown. For example, there are 4 dice combinations that sum to 4. The $EVDice$ for 2 dice with 6 pips is 7, and is shown with the vertical line.

2.3 Constant pips, increasing dice

Figure 1 showed the summuation distribution of 2 dice and 6 pips. If we increase the number of dice from 2 to 6, and keep the number of pips constant at 6, we can observe how the histogram changes shape and moves (see Figure 2). These movements can be summarized, and generalized (see Table 3). While curve characteristics can be determined and predicted, the higher number of dice examples over whelms the lower numbered examples, and make it hard to see possible patterns.

Raw histogram data for dice sets from 2 to 6, each with 6 pips, show how higher numbered sets have a more $EVDice$ values than lowered numbered sets (see Figure 3). The plotted curves hint that there may be other factors are work, but hidden from this type of view.

When multiple histograms of data are normalized so that the occurrences at $EVDice$ is one, other aspects of the computed data appear. First is the "flairing" of the occurrences as the number of dice becomes higher. This supports the intuition that it becomes harder and harder to get extreme values as the number of dice increase. Second, almost hand-in-hand with the first, is that low dice numbers can almost be approximated with a straight line. An Internet search will return simplifications of Uspensky's equation tailored for 2 dice and 6 pips. These simplifications work for that set of die and pips, but generally do not transfer to other combinations.

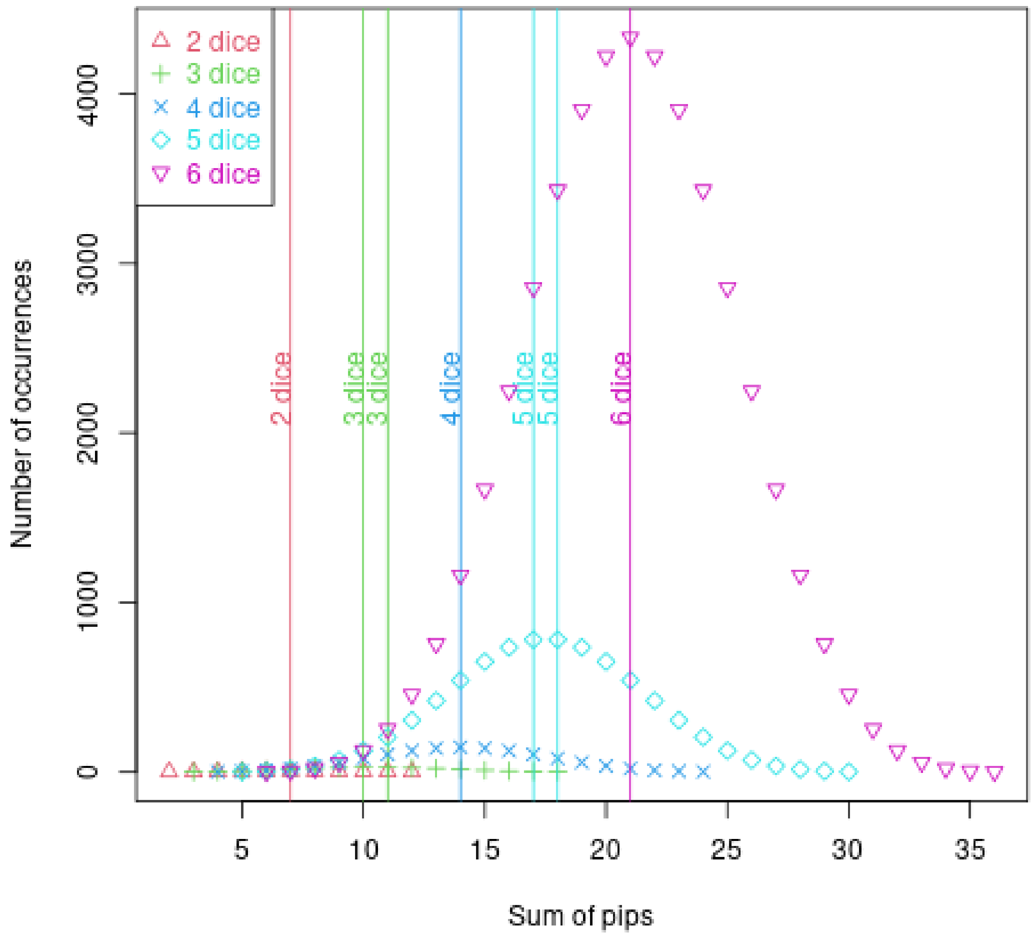

Figure 2: Histograms of summations of 2 to 6 dice, each with 6 pips. The *EV Dice* for each combination is shown with the vertical line. The number of occurences of the higher dice numbers over whelms the lower dice numbers.

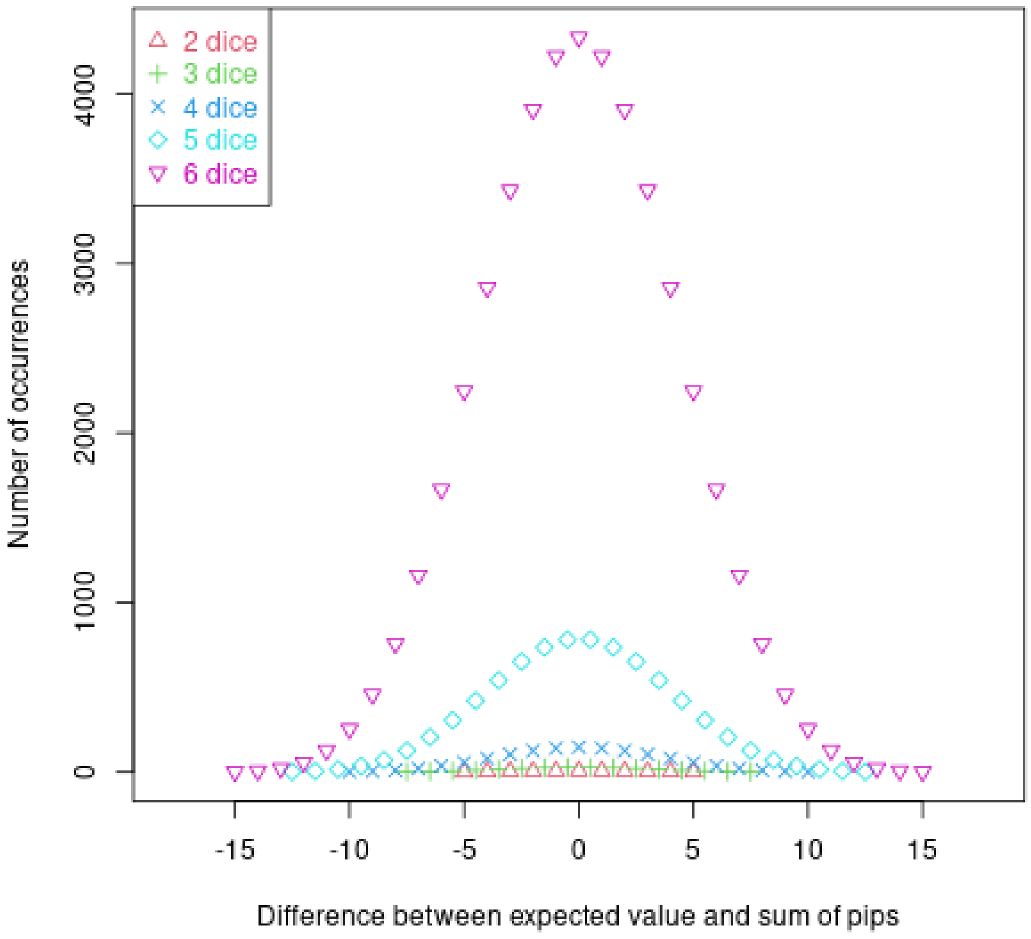

Figure 3: Histograms of summations of 2 to 6 dice, each with 6 pips centered on $EVDice$. Each histogram is centered at $EVDice$ for that set of dice, and differences between a particular $EVDice$ is reported as positive or negative.

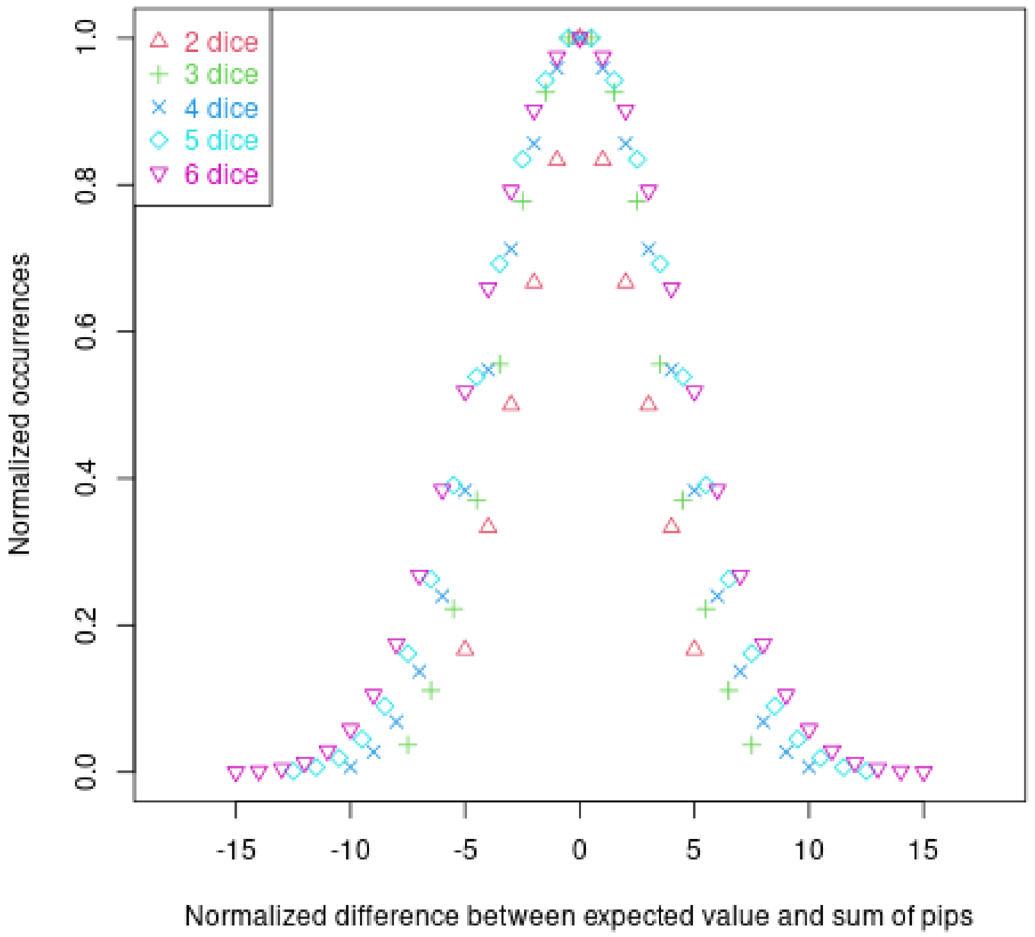

Figure 4: Normalized histograms of summations of 2 to 6 dice, each with 6 pips centered on $EVDice$. Each histogram is centered at $EVDice$ for that set of dice, and differences between a particular $EVDice$ is reported as positive or negative. The occurrences are normalized for each set of dice, so that the number of occerences at $EVDice$ is 1.

2.4 Constant dice, increasing pips

In previous sections, we looked at the effect on $EVDice$ when the number of pips was held constant, and the number of dice increased. Now we will look at the effect when the number of dice is held constant, and the number of pips varies. As the number of pips increases, the $EVDice$ moves 1 to the right (see Figure 5). Viewing the summation data, centered on the $EVDice$ reveals a very orderly pattern (see Figure 6). Normalizing the individual $EVDice$ summations to 1, maintains an orderly pattern (see Figure 7).

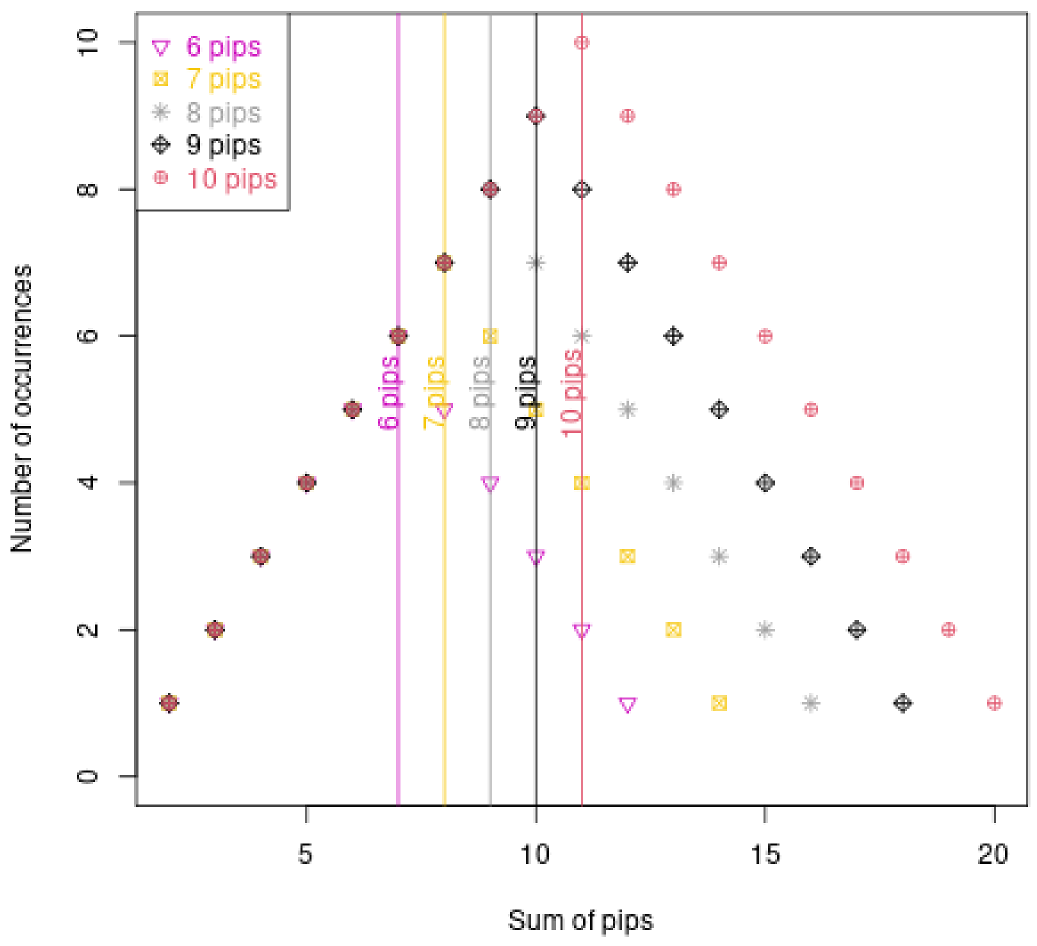

Figure 5: Expected value for 2 dice and 6 to 10 pips. As the number of pips increases, the $EV\,Dice$ moves 1 to the right.

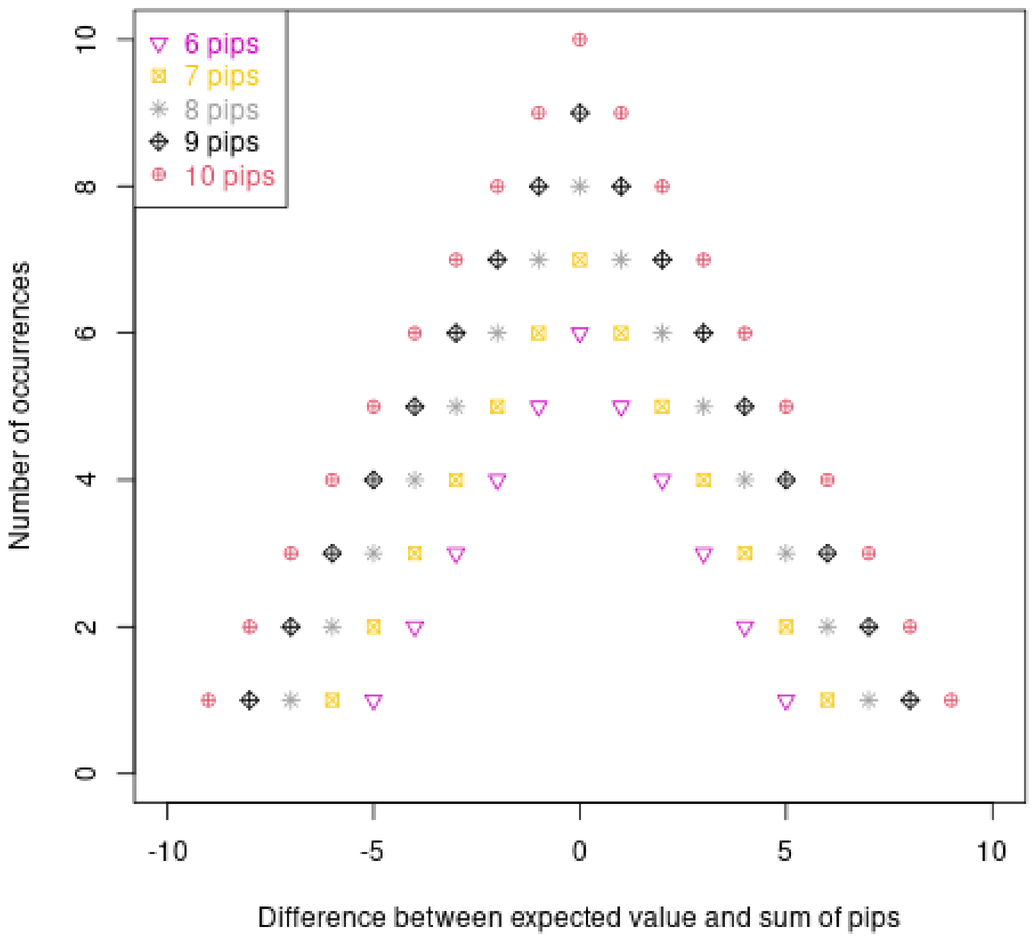

Figure 6: Occurrences of differences between sum and expected value for 2 dice and 6 to 10 pips.

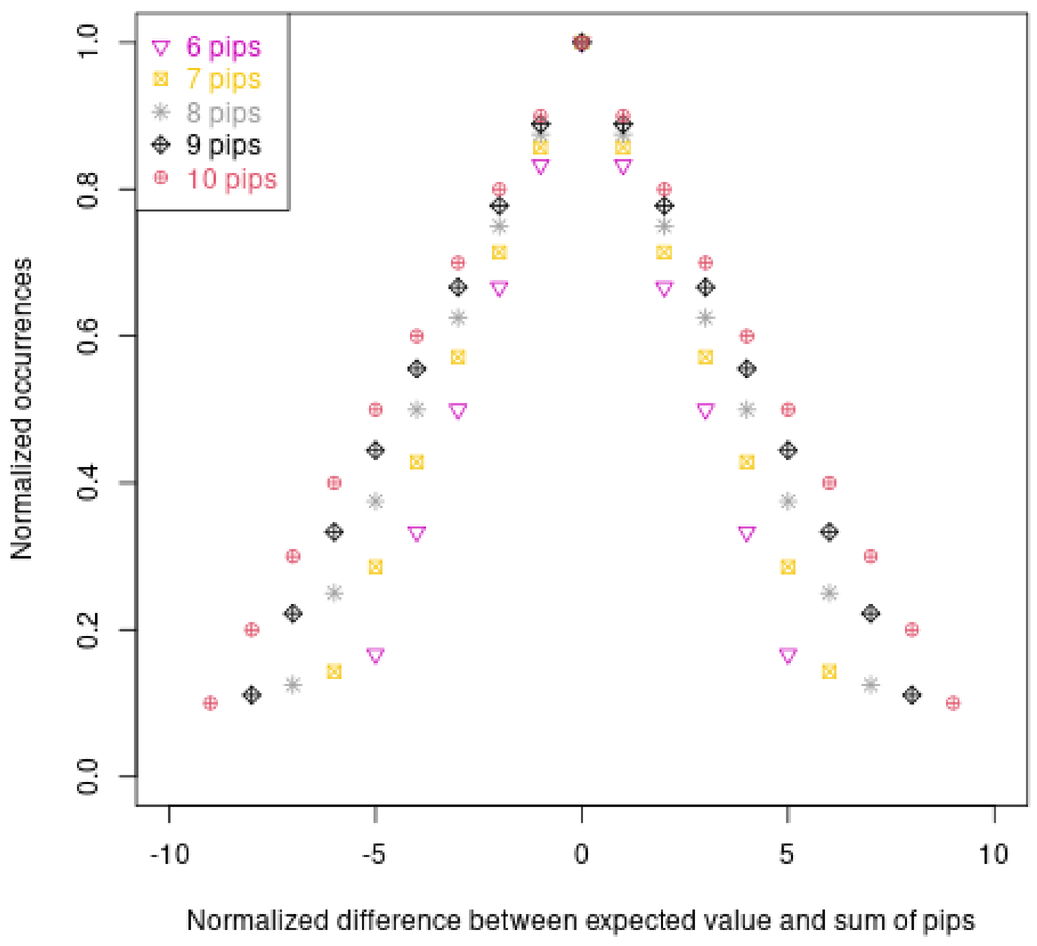

Figure 7: Normalized occurrences of differences between sum and expected value for 2 dice and 6 to 10 pips.

2.5 Increasing dice, increasing pips

The previous sections have shown the interactions between number of dice and number of pips on the range of possible values, while focusing on $EVDice$. Examination of equations in table 1 shows that the $EVDice$ is a linear function, therefore plotting the interaction of the number of dice and the number of pips in a 3D realm would yield a plane. Examination of the interactoin of a varying nunber of dice and a fixed numbers of pips yields a much more interesting shape (see Figures 8, and 9). Shape attributes like scalloping, and $EVDice$ curvature become apparent when interacting with the data.

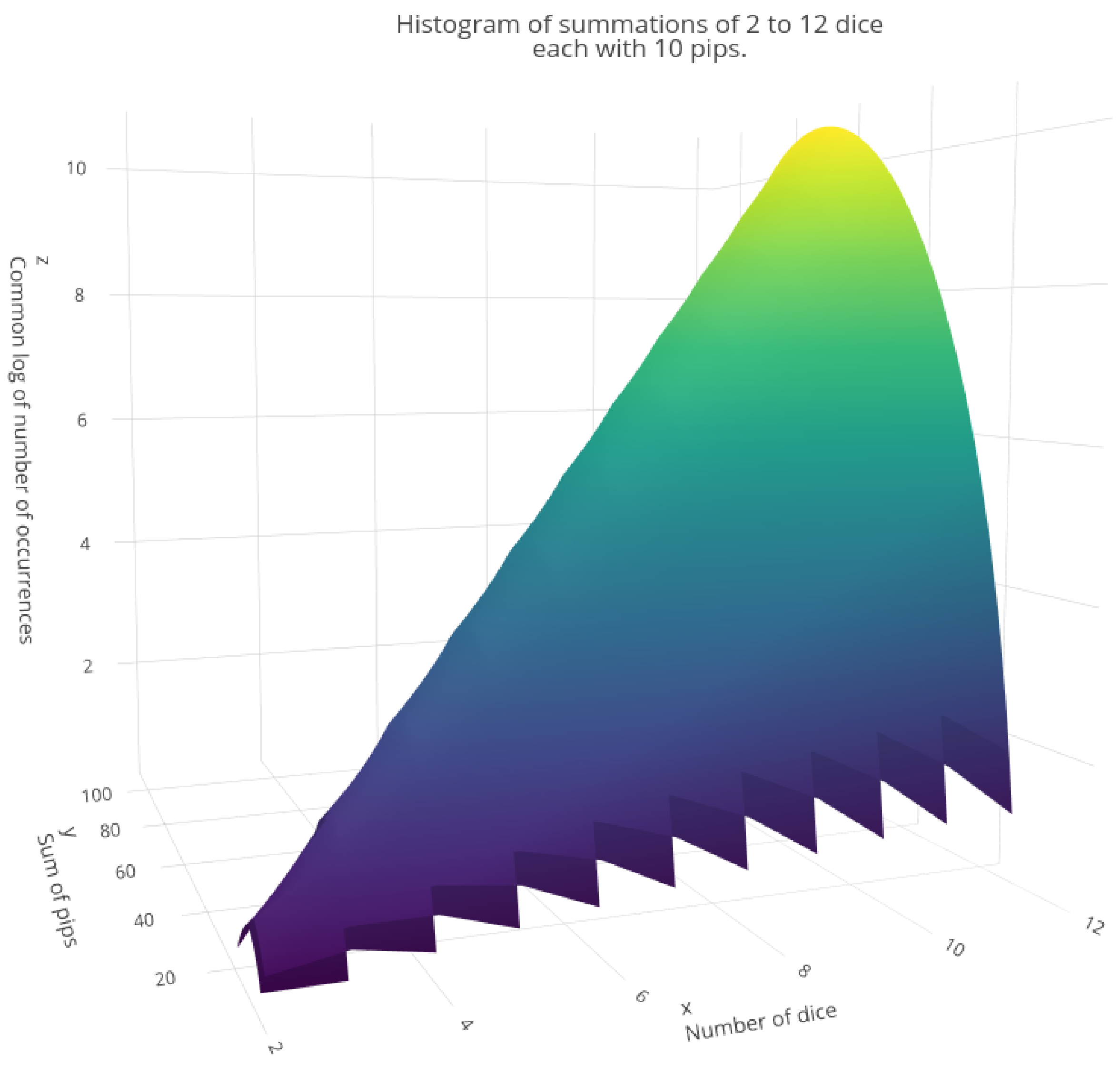

Figure 8: 3D plot showing summation of pips with 2 to 12 dice, each with 10 pips. The embedded ZIP file contains the index.html file and libraries to create an interactive plot with "hover text." The "scallopping" along the edge illustrate that some values are impossible to obtain with the given number of dice and pips. For instance, it is impossible to get any combination of pips using 4 dice that sums to 3.

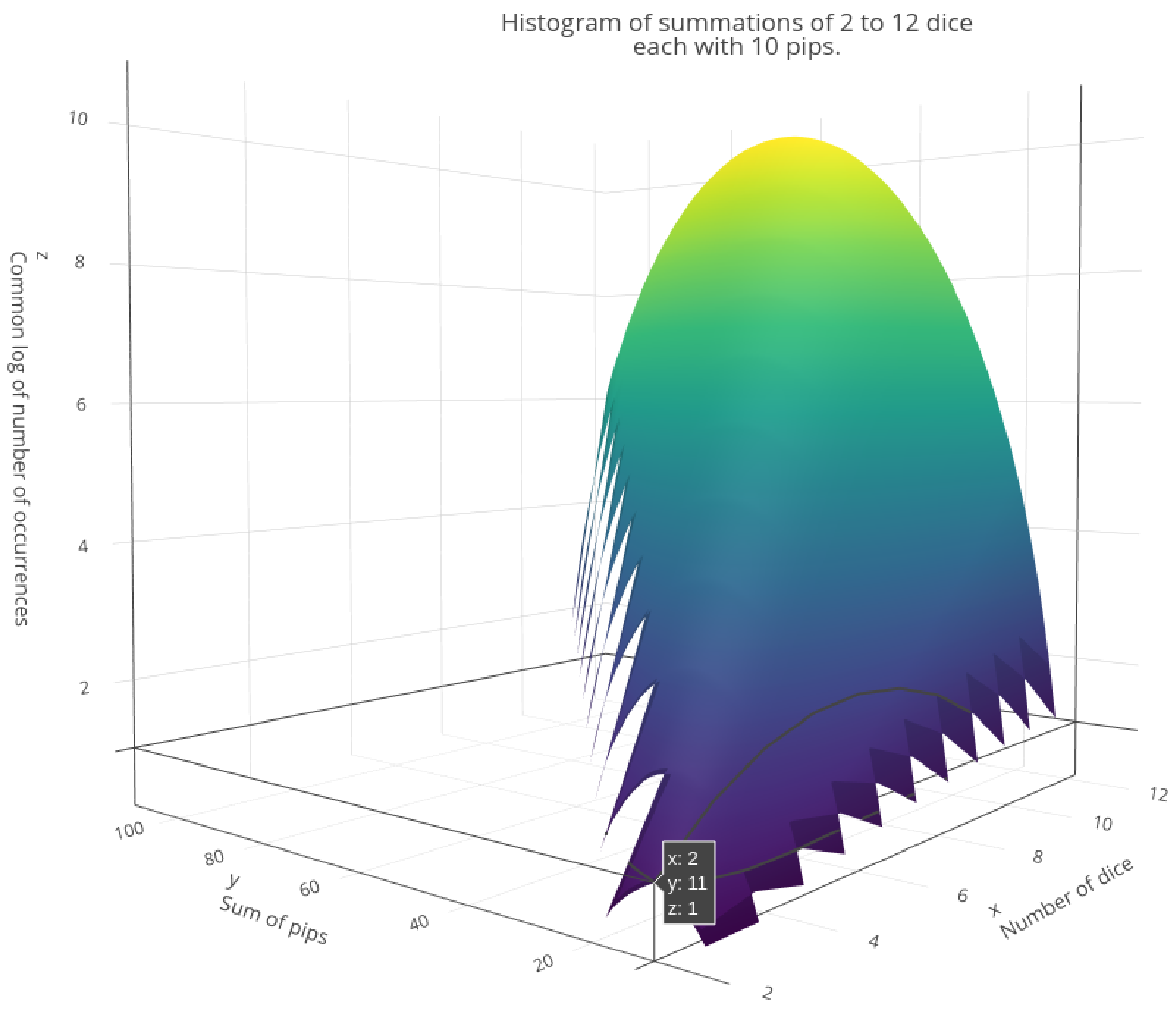

Figure 9: 3D plot showing summation of pips with 2 to 12 dice, each with 10 pips. The embedded ZIP file contains the index.html file and libraries to create an interactive plot with "hover text." The hover text shows the number of dice selected, the sum of the pips for those die, and the common log of the number of times that sum occurred.

3 Conclusion

During the course of exploring how the number of dice and the number of pips interact, we developed three approaches. The first two were recursive approaches in R and C++. These approaches were "good enough" (i.e. fast enough) to support out exploration with small numbers of dice and pips. They were too slow for exploring larger number of dice or pips. A binomial approach based on Uspensky[2] was used for more interesting combinations.

We formulated equations to predict:

- The total number of combinations possible based on the number of dice and the number of pips per die.
- The expected value for any dice and pip combination based on a large number of "rolls."
- The distribution of all values based on any dice and pip combination based on a large number of "rolls."
- How the expected value "moves" when the number of dice or the number of pips changes.

Dice are fun. This was a fun exploration.

A Miscellaneous files

A collection of files used in the creation of this report.

- An R based recursive implementation to find all possible dice combinations based on number of die and number of pips per die
- A C++ based recursive implementation based on the R implementation
- An R based binonial based implementation
- The web based rotatable 3D files used to create the 3D images

Default security settings for Adobe brand readers, prevent the extraction of ZIP files. A ZIP file is embedded in this document. To extract the ZIP file:

1. Extract the file from this document. The file's extension is: piz
2. Change the file's extension from "piz" to "zip"
3. "Unzip" the file.

After unzipping the file, the 3D interactive image can be shown in the default browswer by opening the "index.html" file.

References

[1] Jesse Hammer, *tikzdice*, `https://github.com/jessehamner/tikzdice`, 2021.

[2] James Victor Uspensky, *Introduction to mathematical probability*, McGraw-Hill Book Company, New York, 1937.