Riddler May 13, 2022: Nonconformist dice game

Mark Girard

May 19, 2022

From Ross O'Brien comes a game of nonconformist dice:

Question 1. You begin by rolling four fair tetrahedral dice whose four sides are numbered 1 through 4 and examining the result.

- Divide the resulting dice into two groups: those whose values are unique, and those which are duplicates.
- If all four dice are in the 'unique' group, you win.
- If all four are in the 'duplicate' group, you lose.
- Otherwise, take all dice in the duplicate group and re-roll them.

Repeat this process until lyou either win or lose.

For example, if you roll a 1, 2, 2 and 4, then the 1 and 4 will go into the 'unique' group, while the 2s will go into the 'duplicate' group. You reroll the 2s and the result happens to be 1 and 3. The 'unique' group now consists of 3 and 4, while the 'duplicate' group will have two 1s. You re-roll the two 1's and obtain a 1 and a 2. Now all dice are showing a different number, so you win!

What is your probability of winning the game?

Generalized game with n dice each having k-sides

We can generalize the game in the following manner. Let us supppose that we have n different k-sided dice. Then we can play the same game described in the problem statement. We can assume that $n \leq k$, because otherwise if $k < n$ there are more dice than distinct sides and we can never win.

Begin by rolling all n dice. Let d be the number of dice showing a distinct number.[1] Note that it is impossible to have exactly $d = n - 1$, because otherwise the n^{th} remaining

[1] For example, if five 6-sided dice are rolled yielding the outcome (⚀, ⚂, ⚀, ⚅, ⚂), then $d = 3$ as the sides ⚀, ⚅ and ⚂ are each displayed exactly once and there are two dice showing ⚀.

die must also be a distict number. There are a few different possible outcomes after rolling the dice:

- If $d = 0$, then you lose.
- If $d = n$, then you win!
- Otherwise, if $d \in \{1, 2, \ldots, n-2\}$, we pick up the $n - d$ dice that are not showing a distinct side and re-roll only those dice.

As others have pointed out, the distinct 'states' of this game can be completely described by the number d at the end of a roll.[2] Moreover, the 'starting' position of the game is equivalent to being in the state where $d = 1$. Indeed, when rolling all of the dice you must first roll one die, after which you are in exactly the same position as if you had rolled all the dice then picked up all but one.

The game can therefore be described by an absorbing Markov chain having the $n - 1$ nodes $0, 1, 2, \ldots, n-2$, and n. The absorbing nodes are 0 (the losing node) and n (the winning node), where the initial node is node 1. It remains to determine the transition probabilites between these nodes.

Original game with $n = k = 4$

In the case when $n = k = 4$ (as in the originally proposed game), straightforward counting yields the following matrix of transition probabilities:

$$P = \begin{array}{c} \\ 1 \\ 2 \\ 0 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 0 & 4 \end{array} \\ \begin{pmatrix} \frac{3}{16} & \frac{9}{16} & \frac{5}{32} & \frac{3}{32} \\ \frac{1}{8} & \frac{5}{8} & \frac{1}{8} & \frac{1}{8} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \tag{1}$$

where the (i, j)-entry of the matrix is the probability that you move to state j conditioned on being in state i. This matrix has the form:

$$P = \begin{pmatrix} Q & R \\ 0 & I \end{pmatrix}$$

and the absorbing probabilities (i.e., the probability of ending in state j conditioned on being in the state i) are given by

$$(I - Q)^{-1} R = \begin{array}{c} \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 4 \end{array} \\ \begin{pmatrix} \frac{11}{20} & \frac{9}{20} \\ \frac{31}{60} & \frac{29}{60} \end{pmatrix} \end{array}. \tag{2}$$

[2] For example, by simply relabeling everything, the state (⚀, ⚁, ⚀, ⚂, ⚀) (where one subsequently picks up and re-rolls the two ⚀s) is equivalent to the state (⚁, ⚂, ⚁, ⚀, ⚁) (where one picks up and re-rolls the two ⚁s). In both cases $d = 3$.

Namely, the probability of winning the game (getting to the state where all 4 are distinct while starting from the position that 1 is distinct) with four 4-sided dice is 9/20 or 45%.

Five 5-sided dice

We can perform the same analysis as above but with five fair 5-sided dice (assuming such objects exist). The calculations for computing the number of ways to reach each state were tedious, but the resulting matrix of transition probabilities for this game is:

$$P = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 0 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 0 & 5 \end{array} \\ \begin{pmatrix} \frac{8}{25} & \frac{24}{125} & \frac{48}{125} & \frac{41}{625} & \frac{24}{625} \\ \frac{38}{125} & \frac{21}{125} & \frac{54}{125} & \frac{6}{125} & \frac{6}{125} \\ \frac{6}{25} & \frac{3}{25} & \frac{14}{25} & 0 & \frac{2}{25} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

and computing the absorbing probabilities we find

$$(I - Q)^{-1}R = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 5 \end{array} \\ \begin{pmatrix} \frac{26251}{75595} & \frac{49344}{75595} \\ \frac{24916}{75595} & \frac{50679}{75595} \\ \frac{21114}{75595} & \frac{54481}{75595} \end{pmatrix} \end{array}.$$

The probability of winning the game at the start is equal to $49344/75595 \approx 0.6527$, or about 65.27%. Your odds of winning go up with more dice!

Winning the generalized game ()

In the end, I wasn't able to find a good way of doing this other than to simply enumerate all possible outcomes and count the number of dice showing a distinct side. Fortunately I was able to write some code to do this for me. The code can be found at the end of this document.

The complexity of the computation grows as $O(k^n)$. After games of about 9 dice it gets too big for me to do on my computer. Here is a table showing the exact odds of winning a game with n dices having k sides.

(Notes: If there is only one die then you trivially win on the first throw. If there are 2 dice, the odds that the second die is the same as the first is $1/k$, so the odds of winning are $1 - 1/k$. If $n > k$ then you can never win.)

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 1 & \frac{2}{3} & \frac{2}{3} & 0 & 0 & 0 & 0 \\ 1 & \frac{3}{4} & \frac{6}{7} & \frac{9}{20} & 0 & 0 & 0 \\ 1 & \frac{4}{5} & \frac{12}{13} & \frac{336}{473} & \frac{49344}{75595} & 0 & 0 \\ 1 & \frac{5}{6} & \frac{20}{21} & \frac{150}{181} & \frac{2120}{2391} & \frac{109150}{208811} & 0 \\ 1 & \frac{6}{7} & \frac{30}{31} & \frac{135}{152} & \frac{162735}{171506} & \frac{125767120}{150660819} & \frac{12734579130720}{22172908070131} \end{pmatrix}$$

		Number of dice (n)						
		1	2	3	4	5	6	7
Number of sides (k)	1	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	2	1.000000	0.500000	0.000000	0.000000	0.000000	0.000000	0.000000
	3	1.000000	0.666667	0.666667	0.000000	0.000000	0.000000	0.000000
	4	1.000000	0.750000	0.857143	0.450000	0.000000	0.000000	0.000000
	5	1.000000	0.800000	0.923077	0.710359	0.652742	0.000000	0.000000
	6	1.000000	0.833333	0.952381	0.828729	0.886658	0.522722	0.000000
	7	1.000000	0.857143	0.967742	0.888158	0.948859	0.834770	0.574331

Some code

```
from itertools import product
from collections import Counter
from sympy.matrices import Matrix, eye
from sympy import Rational, print_latex
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

def make_transition_probs(n, k):
    states = list(range(n-1)) + [n]
    P = {i:{j:0 for j in states} for i in states}

    # Loop through all possible outcomes when rolling (n-1) k-sided dice
    for roll in product(range(k), repeat=(n-1)):
        # Loop through the previous states 1, 2, ..., n-2
        for i in range(1, (n-1)):
            # Assume wlog that the i unique dice are labeled 1, 2, ..., i
            prev_tally = Counter(range(i))
            # Ignore the first i dice from the roll, and add the rest to the tallys
```

```
            curr_tally = Counter(roll[i-1:]) + prev_tally
            # Count up the number of distinct dice in the new tally
            j = sum(x == 1 for x in (curr_tally).values())
            # Add this to the number of counts
            P[i][j] += 1

    # Divide each count by the total number of possible outcomes (k**(n-1))
    for i, j in product(states, repeat=2):
        P[i][j] /= Rational((k**(n-1)))

    # Insert absorbing probabilities
    P[0][0] = 1
    P[n][n] = 1

    return P

def make_matrices(n,k):
    P = make_transition_probs(n, k)

    Q = np.array([[P[i][j] for j in range(1, n-1)] for i in range(1, n-1)])
    R = np.array([[P[i][j] for j in (0,n)] for i in range(1, n-1)])

    P = np.block([[Q,R],[np.zeros((2,n-2),dtype=int),eye(2)]])

    return P, Q, R

def get_win_prob(n,k):
    if n > k:
        return 0
    if n == 1:
        return 1
    if n == 2:
        return Rational((k-1),k)
    _, Q, R = make_matrices(n,k)
    N = Matrix(eye(n-2) - Q)
    T = N.inv()*R
    return T[0,-1]

def main():
    nmax = 6
    kmax = 6
    y = np.array([[get_win_prob(n,k) for n in range(1,nmax)] for k in range(1,kmax)])
```

```
    print("LaTeX code for matrix")
    print_latex(Matrix(y))

    df = pd.DataFrame(y.astype(float))
    df.columns = df.index = range(1,len(df)+1)
    green = plt.get_cmap("Greens")
    df = pd.concat([df], keys=['Number of sides (k)'])
    iterables = [["Number of dice (n)"], range(1,nmax)]
    df.columns = pd.MultiIndex.from_product(iterables)
    s = df.style.background_gradient(vmin=0, vmax=1, cmap=green)
    print("HTML code for table")
    print(s._repr_html_())

if __name__ == "__main__":
    main()
```