Vamshi Jandhyala

Four Numbers from Six Dice

✠

July 2026

A counting puzzle from the Monthly Mindbenders.[1]

Problem

If you roll six dice, you could get the same number on every die, or, in contrast, all different numbers. What is the probability of getting exactly four distinct numbers? The puzzle suggests making a guess first, then doing the calculation, on the grounds that the answer may surprise you.

Solution

Six dice, each showing one of six faces, give $6^6 = 46656$ equally likely rolls. Count the ones in which exactly four distinct faces appear.

Such a roll is built by two independent choices. First decide *which* four of the six faces show up, in $\binom{6}{4} = 15$ ways. Then decide which die shows which of those four faces, and here the condition bites: each of the four chosen faces must appear at least once, or the roll would have fewer than four distinct values. So the assignment of the six dice to the four faces must be a surjection.

The number of surjections from a set of six onto a set of four is counted by inclusion and exclusion on the faces that fail to appear:

$$\sum_{j=0}^{4} (-1)^j \binom{4}{j} (4-j)^6 = 4^6 - 4 \cdot 3^6 + 6 \cdot 2^6 - 4 \cdot 1^6 = 4096 - 2916 + 384 - 4 = 1560.$$

Equivalently $4!\,S(6,4) = 24 \times 65 = 1560$, writing S for the Stirling number of the second kind: partition the six dice into four unlabelled nonempty blocks in $S(6,4) = 65$ ways, then attach the four faces to the blocks in $4!$ ways.

Multiplying the two choices,

$$\#\{\text{rolls with exactly four distinct faces}\} = 15 \times 1560 = 23400,$$

and therefore

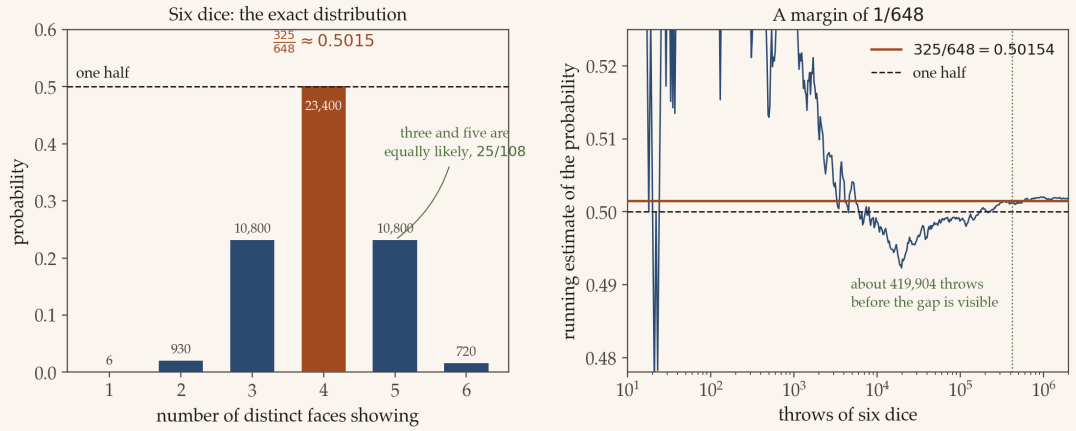
$$\Pr(\text{exactly four distinct}) = \frac{23400}{46656} = \boxed{\frac{325}{648} \approx 0.5015.}$$

[1] Peter Winkler, *Monthly Mindbenders*, National Museum of Mathematics, July 2026, momath.org/mindbenders.

That is the surprise the puzzle advertises. Exactly four distinct numbers is more likely than every other outcome put together, and yet only barely, because

$$\frac{325}{648} = \frac{1}{2} + \frac{1}{648}.$$

The count clears half by a single part in 648. Most guesses land well below a half, on the reasoning that four is one particular value out of six possibilities, which quietly assumes the six values are anywhere near equally likely. They are not.



Left: the exact distribution of the number of distinct faces on six dice. The four-bar clears one half, but only just. Right: a simulated running estimate against the two lines 1/2 *and* 325/648*, which sit* 1/648 *apart.*

The rest of the distribution

The same argument with k in place of four gives the whole picture, since $\binom{6}{k}$ choices of face set combine with $\sum_j (-1)^j \binom{k}{j} (k-j)^6$ surjections:

distinct faces k	1	2	3	4	5	6
rolls	6	930	10800	23400	10800	720
probability	$\frac{1}{7776}$	$\frac{155}{7776}$	$\frac{25}{108}$	$\frac{325}{648}$	$\frac{25}{108}$	$\frac{5}{324}$

The six counts sum to 46656, which is the arithmetic check worth doing.

Two features stand out. The distribution is sharply peaked at four, which is why the answer beats a half at all: the neighbouring values 3 and 5 together carry only about 46 per cent, and the extremes are negligible. All six dice matching happens once in 7776 rolls, and all six differing only once in 65.

The second is a coincidence that looks like a symmetry and is not. Three distinct faces and five distinct faces have exactly the same probability, 25/108, since

$$\binom{6}{3} \times 540 = 20 \times 540 = 10800 \quad \text{and} \quad \binom{6}{5} \times 1800 = 6 \times 1800 = 10800.$$

The two routes to 10800 are quite different, a small face set filled generously against a large face set filled sparsely, and the equality does not survive a change in the number

of dice. Roll seven dice and the counts for three and five distinct faces are 63210 and 352800, nothing alike. It is a numerical accident of the case where the number of dice equals the number of faces.

How fine the margin is

A margin of $1/648$ is small enough to be worth a second look, because it is the difference between the answer and a coin flip. Suppose you tried to establish the excess experimentally. Estimating a probability near $\frac{1}{2}$ from n throws carries a standard error of about $1/(2\sqrt{n})$, so to see a gap of $1/648$ at all you need

$$\frac{1}{2\sqrt{n}} \lesssim \frac{1}{648} \cdot \frac{1}{2}, \qquad \text{that is} \qquad n \gtrsim 648^2 = 419904.$$

The number of throws needed is, pleasingly, the square of the denominator. Rolling six dice four hundred thousand times, at a generous ten seconds a throw, is about seven weeks of continuous rolling. The exact count takes a line of arithmetic, and this is the ordinary situation in combinatorics rather than a curiosity: counting settles in a moment what sampling would need a lifetime to resolve.

Python code

The formula is short enough to be worth checking against something dumber, so the listing enumerates all 46656 rolls and counts distinct faces directly, with no combinatorics at all. It then rebuilds the same table from the closed form and compares, and finally throws real dice at it.

```
from fractions import Fraction
from itertools import product
from math import comb, factorial
import numpy as np

counts = [0]*7                  # brute force: every one of the 6^6 rolls
for roll in product(range(6), repeat=6):
    counts[len(set(roll))] += 1
print("counts :", counts[1:], " total", sum(counts), "= 6^6 =", 6**6)
print("probs :", [str(Fraction(c, 6**6)) for c in counts[1:]])

surj = lambda n, k: sum((-1)**j * comb(k, j) * (k-j)**n for j in range(k+1))
formula = [comb(6, k) * surj(6, k) for k in range(1, 7)]
print("closed form :", formula, " matches:", formula == counts[1:])

p4 = Fraction(counts[4], 6**6)
print(f"P(four distinct) = {p4} = {float(p4):.8f}")
print(f" = 1/2 + {p4 - Fraction(1,2)}; 4! S(6,4) = {factorial(4)*65}")
print(f" P(3) = P(5)? {counts[3]} vs {counts[5]}")
print(f" seven dice, three vs five: "
      f"{comb(7,3)*surj(7,3)} vs {comb(7,5)*surj(7,5)}")

rng = np.random.default_rng(20260716)    # and now actually roll them
d = np.sort(rng.integers(0, 6, (5_000_000, 6)), axis=1)
nd = (d[:, 1:] != d[:, :-1]).sum(1) + 1
print(f"Monte Carlo: P(4) = {(nd == 4).mean():.6f}")
# counts : [6, 930, 10800, 23400, 10800, 720] total 46656 = 6^6 = 46656
```

```
# probs : ['1/7776', '155/7776', '25/108', '325/648', '25/108', '5/324']
# closed form : [6, 930, 10800, 23400, 10800, 720] matches: True
# P(four distinct) = 325/648 = 0.50154321
#   = 1/2 + 1/648; 4! S(6,4) = 1560
#   P(3) = P(5)? 10800 vs 10800
#   seven dice, three vs five: 63210 vs 352800
# Monte Carlo: P(4) = 0.501894
```

Five million throws put the estimate at 0.501894 against the exact 0.501543. The discrepancy is 0.00035, and the standard error at that sample size is 0.00022, so the simulation is doing what it should. It also shows the trouble with settling this by experiment: even five million throws leave the estimate wobbling by more than half the quantity being measured.