Science Spectrum

Member-only story

Derangements and the Probability That Nothing Is in the Right Place

A party game, a beautiful formula, and a surprising connection to a famous constant.

Dr. Robert Kübler

9 min read · 3 hours ago

--

Photo by freestocks on Unsplash

Imagine that your friend group has a special tradition: every time someone has a birthday, each of the eight guests wraps a small present, writes their name on it, and puts it on a table. Then the birthday kid blindfolds each guest, spins them around, and lets them grab a random present from the pile. The only rule: **you are not allowed to keep your own gift**. If that happens, everyone puts their presents back, and the procedure starts all over again. Now, we can ask the obvious question:

> What is the probability that nobody grabs their own present?

You might think: "With eight people, surely at least *someone* accidentally picks their own." And your instinct is not terrible. In about 63% of cases, at least one person does end up with their own gift. But in the remaining **37% of cases**, every single person walks away with someone else's present. Round valid, everyone happy.

And here is the surprising part: this number barely changes. Whether there are 5, 50, or 5000 guests, the probability that nobody ends up with their own present is always about **36.8%**. Always. No matter the group size.

Why 36.8%? Where does this number come from? And what famous mathematical constant is hiding behind it? Let us find out.

What Is a Derangement?

Let us set up the problem properly. We have n objects sitting in positions 1 through n. A **permutation** is any rearrangement of these objects. If, after the rearrangement, **no object is still in its original position**, we call it a **derangement**.

Let us use a small example. With $n = 3$ guests and three presents, the arrangement (2, 3, 1) means: person 1 grabs present 2, person 2 grabs present 3, and person 3 grabs present 1. Nobody got their own. This is a derangement.

On the other hand, (1, 3, 2) is **not** a derangement because person 1 still ended up with their own present. We say that position 1 is a **fixed point**.

A derangement is a permutation with no fixed points.

We write D_n for the number of derangements of n elements. Our main goal: find a formula for D_n. If we have this number, we can compute the

probability of nobody getting their own present as D_n / n!, where n! is the number of permutations of the n presents.

Building Intuition: Small Cases

Let us start with small cases to build some intuition.

***n* = 1:** The only permutation is (1). Person 1 keeps their own present. Not a derangement. So $D_1 = 0$.

***n* = 2:** Two permutations: (1, 2) and (2, 1). Only (2, 1) has no fixed points: the two presents are simply swapped. $D_2 = 1$.

***n* = 3:** We have 3! = 6 permutations. Let us check them all:

- (1, 2, 3) — everyone keeps their own. Three fixed points. No.
- (1, 3, 2) — person 1 still has theirs. No.
- (2, 1, 3) — person 3 still has theirs. No.
- **(2, 3, 1)** — nobody has their own. **Yes!**
- **(3, 1, 2)** — nobody has their own. **Yes!**
- (3, 2, 1) — person 2 still has theirs. No.

So $D_3 = 2$. Out of 6 possible shuffles, only 2 are complete mix-ups.

***n* = 4:** I will spare you the tedium. There are 4! = 24 permutations, and only $D_4 = 9$ of them are derangements. *It is a good exercise to verify this yourself!*

Let us collect what we have (and a bonus row):

n	n!	D_n	$D_n/n!$
1	1	0	0
2	2	1	0.5
3	6	2	0.333...
4	24	9	0.375
5	120	44	0.3666...

The last column (the probability that a random permutation is a derangement) seems to **converge** to something. But to what? Let us simulate first, then do the math.

Simulation

Before we derive the formula (no worries, we will), let us take the lazy route and simulate it in Python.

```
import random


def is_derangement(perm: list[int]) -> bool:
    return all(perm[i] != i for i in range(len(perm)))

def estimate_derangement_probability(n: int, n_trials: int = 100_000) -> float:
    count = 0
    for _ in range(n_trials):
        perm = list(range(n))
        random.shuffle(perm)
```

Let us try it for a few values of *n*:

```
for n in [1, 2, 3, 4, 5, 10, 20, 100]:
    prob = estimate_derangement_probability(n)
    print(f"n = {n:>3}: P[derangement] ≈ {prob:.4f}")
```

The output looks something like this:

```
n =   1: P[derangement] ≈ 0.0000
n =   2: P[derangement] ≈ 0.5001
n =   3: P[derangement] ≈ 0.3338
n =   4: P[derangement] ≈ 0.3745
n =   5: P[derangement] ≈ 0.3671
n =  10: P[derangement] ≈ 0.3681
n =  20: P[derangement] ≈ 0.3676
n = 100: P[derangement] ≈ 0.3679
```

Interesting! The probability locks in at roughly **0.3679** and barely moves after $n = 5$. Does this number ring a bell? Well, perhaps not, but if we invert it, we get 1 / 0.3679 = 2.718… which looks awfully close to **Euler's number e**. So we guess that P[derangement] is about 1 / e. That is not a coincidence.

> The probability that a random permutation of n elements is a derangement converges to 1/e as n grows.

So, whether the party has 8 guests or 8000, about **36.8%** of all possible outcomes result in nobody getting their own present. Interesting! Now, let us prove it.

The Naive Attempt (and Why It Fails)

You might be tempted to reason like this: person 1 has a $(n - 1) / n$ chance of NOT grabbing their own present. Same for person 2, 3, …, n. So the probability that **nobody** gets their own is roughly $\boldsymbol{((n - 1) / n)^n}$, which converges to — you guessed it — $1 / e$. Right answer!

But here is the problem: this calculation assumes that the events "person 1 does not get their own" and "person 2 does not get their own" are **independent**. They are not. If person 1 grabs person 2's present, that changes person 2's options entirely. The events are **entangled.**

The naive multiplication gives the correct **limit** by a lucky coincidence, but the wrong exact answer for any finite n. For $n = 3$, the naive approach gives $(2 / 3)^3 \approx 0.296$, while the true probability is $2 / 6 \approx 0.333$.

> **Note:** It is a beautiful accident that both approaches (the naive independent one and the correct one we will cover in a second) converge to the same limit $1 / e$, but for entirely different reasons.

To get exact answers, we need a more powerful tool.

Deriving the Exact Formula

The right tool for the job is the **inclusion-exclusion formula.** No worries if this sounds intimidating. I will walk you through it step by step. *Note, however, that we will not prove this formula.*

Counting the Complement

Here is the trick. Instead of directly counting derangements (permutations with **no** fixed point), let us count the opposite: permutations that have **at least one** fixed point. Then

$$D_n = n! - (\#\text{permutations with at least one fixed point})$$

Let A_i be the set of permutations where element i is a fixed point. We want $|A_1 \cup A_2 \cup \dots \cup A_n|$, the number of permutations where at least one element is fixed. The inclusion-exclusion principle says:

$$\left|\bigcup_{i=1}^{n} A_i\right| = \sum_{i=1}^{n} |A_i| - \sum_{1\le i<j\le n} |A_i \cap A_j| + \sum_{1\le i<j<k\le n} |A_i \cap A_j \cap A_k| - \cdots + (-1)^{n+1} |A_1 \cap \cdots \cap A_n|$$

The inclusion-exclusion formula. Image by the author.

Looks daunting? Let us compute each term. It is easier than it looks.

Computing the Terms

One fixed point. If element i is fixed, the remaining $n - 1$ elements can be arranged however they want. So $|A_i| = (n - 1)!$. There are n choices for i, so

$$\Sigma|A_i| = n \cdot (n - 1)! = n!$$

Two fixed points. If elements i and j are both fixed, the remaining $n - 2$ can do whatever they like. So $|A_i \cap A_j| = (n - 2)!$. There are ($n$ choose 2) such pairs, giving

$$\Sigma|A_i \cap A_j| = (n \text{ choose } 2) \cdot (n - 2)! = n! / 2!$$

I think you see the pattern. For ***k* fixed points**, the contribution is

$$(n \text{ choose } k) \cdot (n - k)! = n! / k!$$

Easy, right?

Putting It All Together

Plugging everything into inclusion-exclusion:

$|A_1 \cup \ldots \cup A_n| = n!/1! - n!/2! + n!/3! - \ldots \pm n!/n!$

And since $D_n = n!$ minus the above, we can write our solution as

$$D_n = n! \sum_{i=0}^{n} \frac{(-1)^i}{i!}$$

The derangement formula. Image by the author.

Let us quickly **verify** this against our earlier results:

- $D_1 = 1! \cdot (1 - 1) = 0$ √
- $D_2 = 2! \cdot (1 - 1 + 1/2) = 1$ √
- $D_3 = 6 \cdot (1 - 1 + 1/2 - 1/6) = 2$ √
- $D_4 = 24 \cdot (1 - 1 + 1/2 - 1/6 + 1/24) = 9$ √

So, our beautiful formula also seems to be correct indeed!

Why Does e Show Up?

Now for the final act. Look at the derangement probability:

$$\frac{D_n}{n!} = \sum_{i=0}^{n} \frac{(-1)^i}{i!}$$

Image by the author.

This sum is exactly the **Taylor expansion of e^x:**

$$e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

Image by the author.

Plug in $x = -1$:

$$e^{-1} = \sum_{i=0}^{\infty} \frac{(-1)^i}{i!}$$

Image by the author.

So $D_n/n!$ is nothing but the **partial sum** of the Taylor series for $1 / e$! As $n \to \infty$, the partial sum converges to $1 / e$.

The derangement formula is a finite truncation of the Taylor series for 1 / e.

And the convergence is absurdly fast. Already for $n = 5$, we have $D_5 / 5! \approx 0.3667$, which is nearly indistinguishable from $1 / e$. In practice, this means you can approximate

$$D_n \approx \mathrm{round}(n! / e)$$

and get the exact answer for all $n \geq 1$. How cool is that?

Implementation

Let us put the exact formula into Python and compare it to the rounding trick.

```
from math import factorial, e


def count_derangements(n: int) -> int:
    """Compute D_n using the inclusion-exclusion formula."""
    result = 0
    for k in range(n + 1):
        result += (-1) ** k * factorial(n) // factorial(k)
    return result

def derangement_probability(n: int) -> float:
    return count_derangements(n) / factorial(n)
```

Let us try it out:

```
for n in range(1, 11):
    d_n = count_derangements(n)
    prob = derangement_probability(n)
    approx = round(factorial(n) / e)
    print(f"n = {n:>2}: D_n = {d_n:>7}, D_n/n! = {prob:.6f}, round(n!/e) = {appr
```

```
n =  1: D_n =       0, D_n/n! = 0.000000, round(n!/e) =       0
n =  2: D_n =       1, D_n/n! = 0.500000, round(n!/e) =       1
n =  3: D_n =       2, D_n/n! = 0.333333, round(n!/e) =       2
n =  4: D_n =       9, D_n/n! = 0.375000, round(n!/e) =       9
n =  5: D_n =      44, D_n/n! = 0.366667, round(n!/e) =      44
n =  6: D_n =     265, D_n/n! = 0.368056, round(n!/e) =     265
n =  7: D_n =    1854, D_n/n! = 0.367857, round(n!/e) =    1854
n =  8: D_n =   14833, D_n/n! = 0.367897, round(n!/e) =   14833
n =  9: D_n =  133496, D_n/n! = 0.367879, round(n!/e) =  133496
n = 10: D_n = 1334961, D_n/n! = 0.367879, round(n!/e) = 1334961
```

The exact formula and the rounding trick agree every single time. And $D_n / n!$ locks in at 0.367879, which is $1 / e$ to six decimal places, by $n = 9$. The approximation is extremely good.

Back to the Party

So, what about our original birthday game? With 8 guests, the probability that nobody grabs their own present is about **36.8%**. Conversely, there is a roughly **63.2%** chance that at least one person ends up with their own gift. Round void.

Now, suppose the group keeps replaying the round until nobody has their own. How many rounds does it take on average? Since each round succeeds with probability ≈ $1/e$, the expected number of rounds is E[rounds] = e.

So on average, about **2.7 rounds** before you get a valid outcome. Not terrible, but enough to test everyone's patience, especially after the third blindfolded spin.

Homework for You

Two exercises to take this further:

1. **A recursion.** Show that $D_n = n \cdot D_{n-1} + (-1)^n$.
2. **Expected fixed points.** What is the expected number of fixed points in a random permutation of n elements?

I hope that you learned something new, interesting, and valuable today. Thanks for reading!

> If you have any questions, write me on LinkedIn!

And if you want to dive deeper into the world of algorithms, give my publication All About Algorithms a try!

Mathematics Probability Probability Theory Puzzle Party

Published in Science Spectrum

Follow

17.2K followers · Last published 3 hours ago

Science Spectrum is here to guide you on your personal path to understanding the fascinating world of science, mathematics, and related topics. Our goal is to make complex concepts accessible to everyone. We are happy to be a member of the Medium Boost family!

Written by Dr. Robert Kübler

5K followers · 102 following

Studied Mathematics, PhD in Cryptanalysis, working as a Data Scientist. Check out my new publication! https://allaboutalgorithms.com

No responses yet