Simulating the Monty Hall Problem

by Jakob Peters (they/them)

November 14th, 2024

About Me

About Me

- I really like Rust

About Me

- I really like Rust
- I have not written very much Rust

About Me

- I really like Rust
- I have not written very much Rust
- I should have chosen a problem that requires more than 13 lines

About Me

- I really like Rust
- I have not written very much Rust
- I should have chosen a problem that requires more than 13 lines
- That’s literally everything about me

These slides were made in Typst

blah blah blah

> Suppose you're on a game show, and you're given the choice of three doors: Behind one door is a car; behind the others, goats. You pick a door, say No. 1, and the host, who knows what's behind the doors, opens another door, say No. 3, which has a goat. He then says to you, "Do you want to pick door No. 2?" Is it to your advantage to switch your choice?
>
> — Craig F. Whitaker's letter quoted in Marilyn vos Savant's "Ask Marilyn" column in Parade magazine in 1990

Formal Proof

1.

2.

3.

Formal Proof

1. It obviously does not matter which door you pick
2.
3.

Formal Proof

1. It obviously does not matter which door you pick
2. This presentation has code on slide 7, promise
3.

Formal Proof

1. It obviously does not matter which door you pick
2. This presentation has code on slide 7, promise
3. See 1

Confirmation Bias

```
> monty_hall 1000000
P(success | switch) ≈ 0.67
P(success | keep) ≈ 0.33
```

Confirmation Bias

```
> monty_hall 1000000
P(success | switch) ≈ 0.67
P(success | keep) ≈ 0.33
```

boring stuff...

Doors			Strategy	
1	2	3	switch	keep
				
				
				

- Each row is a possible world where the initial choice is 1

boring stuff...

Doors			Strategy	
1	2	3	switch	keep
				
				
				

- Each row is a possible world where the initial choice is 1
- Each strategy has an opposite outcome

boring stuff...

Doors			Strategy	
1	2	3	switch	keep
				
				
				

- Each row is a possible world where the initial choice is 1
- Each strategy has an opposite outcome
- Each strategy's outcome is determined by the initial choice

Show Code Already

```rust
use rand::{thread_rng, Rng};

fn monty_hall(n: u32) {
    let (mut rng, mut counts) = (thread_rng(), [0, 0]);

    for _ in 0..n {
        counts[(rng.gen_range(1..4) == rng.gen_range(1..4)) as usize] += 1;
    }

    for (strategy, count) in std::iter::zip(["switch", "keep"], counts) {
        println!("P(success | {strategy}) ≈ {:.2}", count as f32 / n as f32);
    }
}
```

hire me

```rust
fn blazingly_fast_monty_hall() {
    println!("P(success | switch) = 2 / 3")
    println!("P(success | keep) = 1 / 3")
}
```

```
> blazingly_fast_monty_hall
P(success | switch) = 2 / 3
P(success | keep) = 1 / 3
```

github.com/jakobjpeters/monty_hall