# **Monte Hall Simulations Documentation**

# **Overview**

This was a simple test of the Monte hall problem. This was to show the counter intuitive nature of probability. This program demonstrates and highlights the fault in our logic by showing a significant gain in probability of success by utilizing the “switching” strategy.

## **Core Components**

### **Main Game Runner (MainGameRunner)**

* Handles **multiple rounds** of the Monty Hall problem.
* Allows for different **door counts** to be tested.
* Runs a specified **number of simulations per door count** and **compares results**.
* Uses **ANSI color codes** for better readability in the terminal.

### **2. Standard Game (PrizeGame)**

* Simulates a **player who does not switch** and stays with their initial door choice.
* Tracks **win and loss counts** to determine probability outcomes.

### **3. Switching Game (PrizeGameWithSwitch)**

* Extends the PrizeGame class.
* Implements a **switching strategy**, where the player **always** switches doors after the host reveals a non-winning option.
* Shows the advantage of switching in typical Monty Hall settings.

### **4. Randomized Monty Hall Environment**

* Uses Random to generate **randomly placed prizes and player choices**.
* Ensures the host reveals a door that **never** contains the prize.
* Implements **logic to select the remaining unopened door** when the player switches.

## **Usage Instructions**

By default, it runs 100,000 simulations for 3 doors, testing the probability of winning when staying vs. switching.

### **Changing Door Counts**

The program can analyze the Monty Hall problem with different numbers of doors. Modify the RunGames() function in Main to adjust parameters:

## **Technical Details**

## **Technical Details**

### **Simulation Approach**

* The program **randomly places the prize** behind one of the doors.
* The **player selects a random door**.
* The host **reveals a non-winning door** (never the prize).
* The player either:
  + **Stays** with their initial choice.
  + **Switches** to the remaining door.
* Results are recorded for **win percentages** across multiple trials.

### **Breaking Point of the Switching Effect**

* This program tests at what point increasing doors reduces the difference in win rates between staying and switching.
* With **very high door counts**, switching remains dominant but provides diminishing gains over time.

## **Demo Functionality**



