ZHONGYIN ZHANG
July 21, 2022

3 Briefcases Problem Simulation Report

Tango Technical Assessment

Abstract

Mathematically, we hypothesized the result is the contestant switching their original choice will result a double chance to win the price, and the paradigm of solving is do a paradigm shift to think the game process reversely. Base on that, a simulation was built to imitate the game, and play the game 2000 times and analyze the result.

In the simulation, the input data of each game is the winning briefcase, the contestant's original choice and do they choose to switch. An exclusive or logic (XOR) is performed in the game checking process. From the simulation, we obtain the winning rate of the contestant chose switch is 68.1%, and the winning rate of staying is 33.8%. The ratio of these rates is approximately 1:2. Also, the simulation provides the cumulative winning rate, which is 51%, close to a half, shows the game is fair, and the algorithm is preformed correctly. This program also provides a visualization of the results by plotting them into a choice vs. winning rate bar chart. and 3 pie charts for the winning rates.

The result of the simulation matches the mathematical hypothesis, explicitly shows switch the original choice will result a double probability to win. This program is also friendly to add more features to the current specifications, by modifying the constants, and change the data type of the parameters, specifically, a boolean to an integer.

Section 1 Mathematical Analysis

To handle this question, a good way is do a paradigm shift. Suppose the contestant may choose 2 briefcases initially. The probability of the prize in either one is 2/3, and the probability of the prize in the one not chosen is 1/3. Then, the host opened one of the empty one from the contestant's choices, and now the contestant's choice has only one briefcase left, the probability of the prize in that briefcase is 2/3. Finally, the host ask the contestant if they what to switch to the one left behind, and the probability of prize in that one is 1/3.

Now think back to this problem. In the new problem above, the initial choice can be treated as "switched" in the original problem, and switch in the new problem can be treated as "not switch" in the original problem. As a result, mathematically, the probability to win of stay into the original choice is 1/3, and switch the original choice is 2/3, which is doubled.

For the winning condition, we obtained 3 parameters. The first one is the briefcase originally chose, the second one is the winning briefcase, and the third one is contestant switched there choice or not. To win the game, there are 2 possible cases: one is the prize in the original choice and the contestant chose not switch, another one is the prize is not in the original chosen briefcase and the contestant switched. Simplify the conditions, it is a exclusive or logic, (the prize in the original choice) XOR (the contestant switched their choice).

Section 2 Program Design

The simulation program is divided into these following classes:

App.java - the application class to execute the simulation program

GameChecker.java - a class to imitate the game with user's choice as fields, and obtain the game result by getResult() method.

Simulation.java - a class to simulate the game in multiple times, and get the resultant data contains the number of wins and the winning rate for both switch and stay.

Visualization.py - a python file to visualize the data into bar chart and pie chart, so that the network's analytics department can clearly see the winning rate and the difference between the results of the contestant choses switch or stay.

Section 3 Conclusion

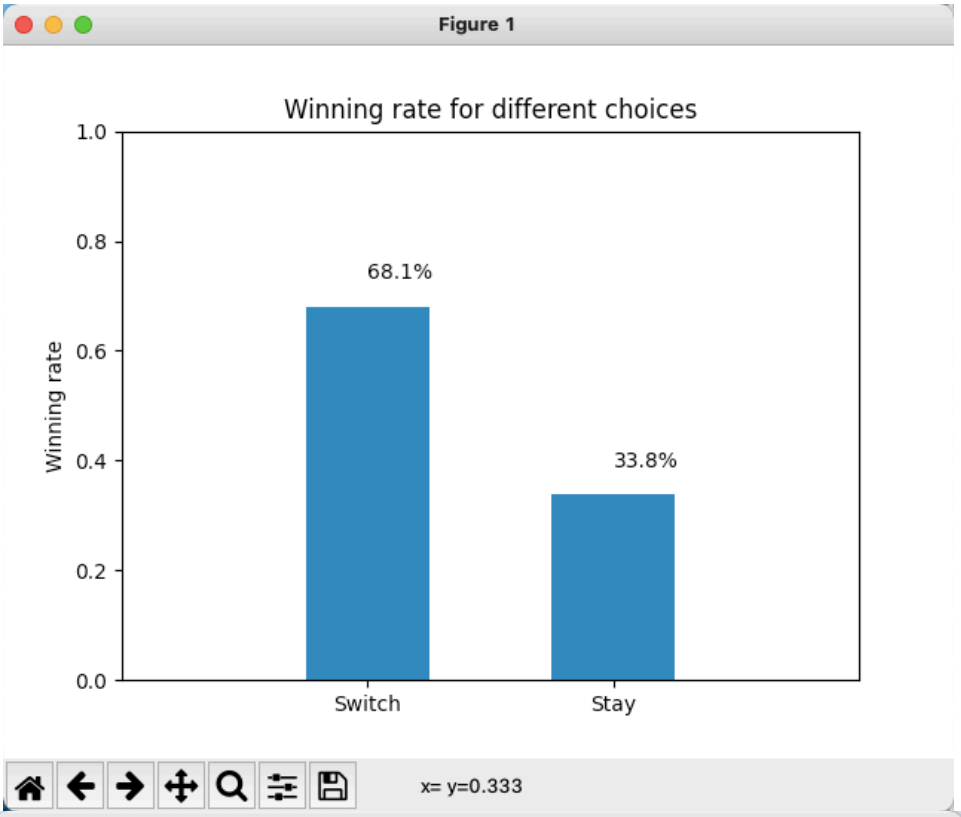
The simulation results shows, the contestant chooses to switch has a double probability to win than choose to stay for their initial choice.

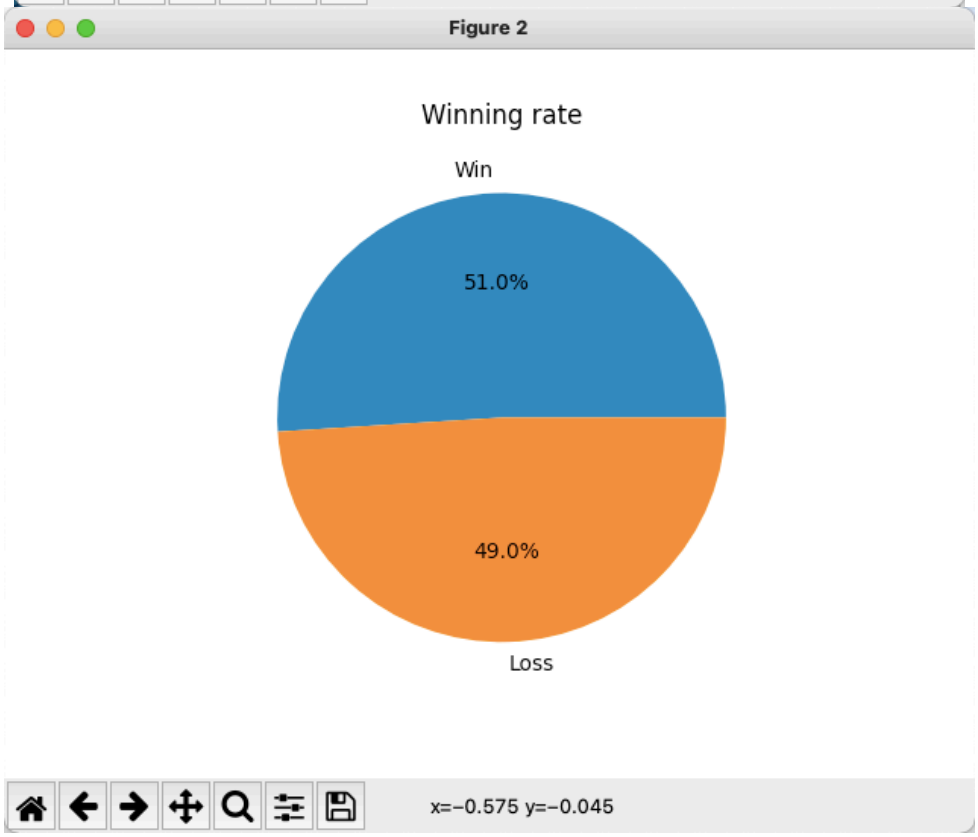
Here below is the result of a sample run of the simulation, which plays the game 2000 times, 1000 contestants choose switch and the rest 1000 choose stay.

Command line result in txt file:

```
Total simulations: 2000
Total number of the contastent chooses switch: 1000
Total number of the contastent chooses stay: 1000
Total number of wins: 1019
Number of wins chose switch: 681
Number of wins chose stay: 338
Winning rate: 0.510
Winning rate chose switch: 0.681
Winning rate chose stay: 0.338
Winning rate ratio: stay:switch = 1:2
```

Visualization of the result above:





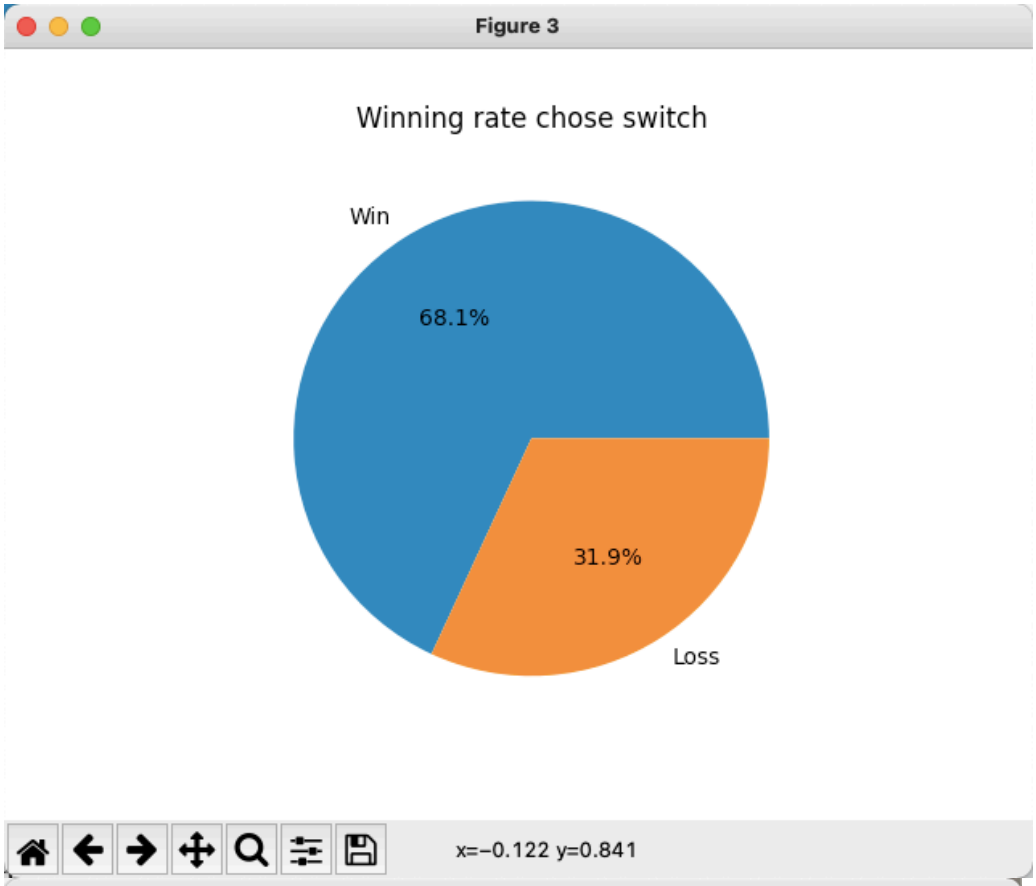


Figure 3
Winning rate chose switch
Win
68.1%
31.9%
Loss
x=−0.122 y=0.841

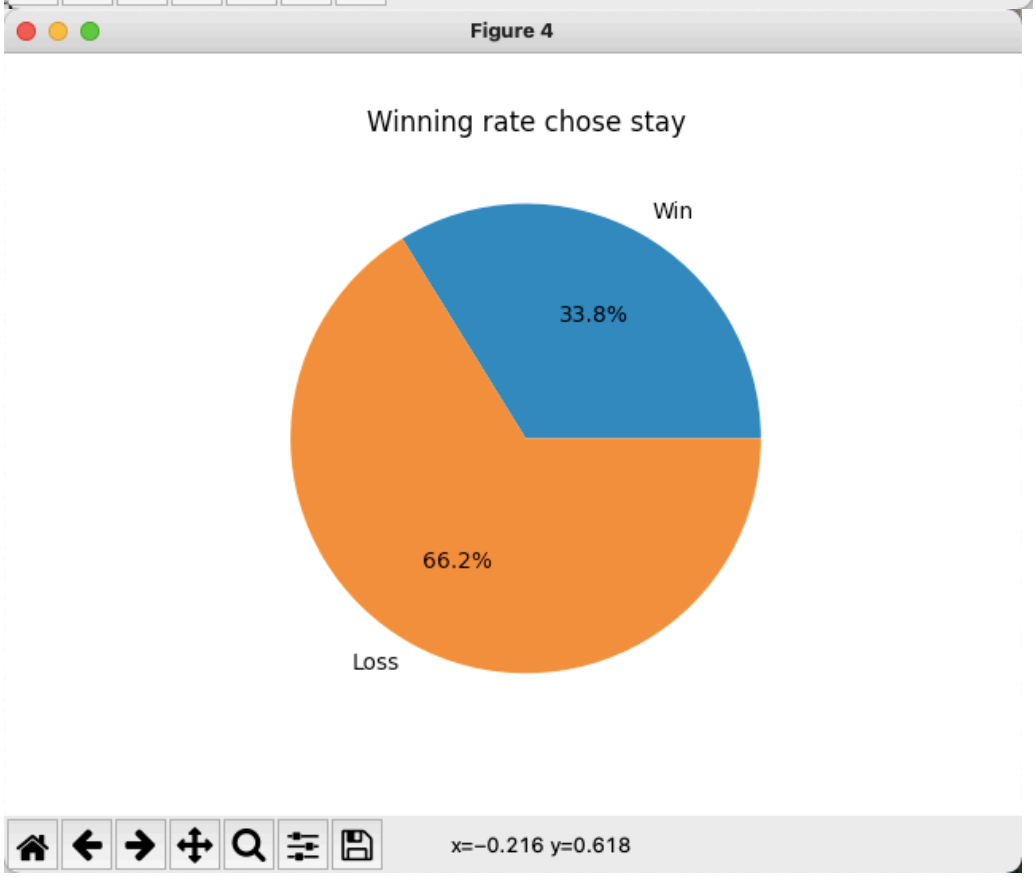


Figure 4
Winning rate chose stay
Win
33.8%
66.2%
Loss
x=−0.216 y=0.618

From the graphs above, figure 1 shows the winning chance of choose switch is much higher than choose stay, and the chance is approximately doubled. Figure 2 is the cumulative winning for all contestant, includes 1/2 choose switch and 1/2 choose stay, is around 50%, shows this is a fair game, and it also checks the correctness of the algorithm.

Section 4 Development thinking : n - briefcases problem

From above conclusion we can explicitly see, switch the choice will result a double chance in get the prize. But, what about there are n briefcases? This leads to a new problem:

There are n briefcases, and only one of them contains the prize. The contestant choose one from these briefcases, then the host eliminates one of the other briefcases the host knows does not contain the prize. If the contestant stays to their original choice, then open that one, otherwise, of they chooses to switch, the problem becomes "There are n-1 briefcases …", repeat this process. This is a recursing problem and the base case is either the contestant stays or switched to the final briefcase.

However, if we think the same way as 3 briefcases problem, we can easily get, always switch until the last one is the best. But this problem can be more challenging: if the price worths $p, and to open the chosen briefcase costs $x, and each switch costs $y, how many switches maximize the net income of the game, in terms of x, y and p? And what about the switch price ascends after each switch, such that the cost per switch becomes $y = (base fare + unit switch fare * previous switched times)?

These type of questions could be handled based on the current code base. There are constants at the beginning of the code, to change the number of briefcase, it could be done by change the constant, and change the simulate() method's parameter to from boolean isSwitch to int numSwitches. Then let the getResult() method in GameChecker class becomes recursive.

To visualize the data, we can add more indexes to the indexMap(), and add more BARS to the barChart(). To add the price of switching and the value of the prize, we can add these constants and add a getPrice() and a getNetIncome() method, and plot the net incomes by switching different times respectively, such that the y axis is the new income and x axis is the number of switches.