**Probability and Statistics**

**Project Report**

**Team member 1: Roxana Mafteiu-Scai**

**Group 2, Subgroup 4**

[**roxana.mafteiu97@e-uvt.ro**](mailto:roxana.mafteiu97@e-uvt.ro)

**Team member 2: Vlad Rochian**

**Group 3, Subgroup 5**

[**vlad.rochian98@e-uvt.ro**](mailto:vlad.rochian98@e-uvt.ro)

**Problem 1 – Matching Cards**

**Statement:**

Take two identical decks of cards, each deck containing m cards. (For example, from a standard deck of cards, use the 13 spades as one deck and the 13 hearts as the other deck.) Shuffle each of the decks and determine if at least one match has occurred by comparing the positions of the cards in the decks. If a match occurs at one or more than one position, call the trial a success. Thus, each experiment consists of comparing the positions of the m cards in each of the two decks. The outcome of the experiment is “success” if at least one match occurs, “failure” if no matches occur.

Simulate this experiment on the computer using permutations of the first m positive integers. For example, for m = 6, you could randomly order 1; 2; 3; 4; 5; 6 (by using randperm(6)) and if any of these integers occupies its natural position, a match occurs. For each of m = 4, 6, 8, 10, and 12, repeat the simulation 1000 times and calculate the relative frequency of success.

**Implementation in R:**

*checkIfSuccess = function(size) {*

*a = sample(size);*

*b = sample(size);*

*for (i in 1:size) {*

*if (a[i] == b[i]) {*

*return (1);*

*}*

*}*

*return (0);*

*}*

*successFrequency = function(count, size) {*

*success = 0;*

*for (i in 1:count) {*

*if (checkIfSuccess(size)) {*

*success = success + 1;*

*}*

*}*

*return (success / count);*

*}*

*successFrequency(1000, 4);*

*successFrequency(1000, 6);*

*successFrequency(1000, 8);*

*successFrequency(1000, 10);*

*successFrequency(1000, 12);*

**Experimental results:**

|  |  |
| --- | --- |
| **Length of permutation (m)** | **Relative frequency of success for 1000 simulations** |
| 4 | 0.610 |
| 6 | 0.626 |
| 8 | 0.635 |
| 10 | 0.630 |
| 12 | 0.643 |

**Theoretical deduction of the results:**

Let a, b be the two permutations. Let N be the size of the permutations. The probability of having a match on one fixed position is 1/N, as there are N possible values for that position in b, while only one of them coincides with the one in a. The probability of having matches on two fixed positions is 1 / (N \* (N - 1)), as there are N possible values for the first position in b and N-1 remaining possible values for the second position in b, while only one combination of them coincides with the values in a.

By applying this logic, we can deduce that for k fixed positions the probability for all of them to be matches is (N - k)! / (N!). In order to compute the probability to have at least one match, we can use the principle of inclusion and exclusion as follows: There are N positions, for every one of them the match probability is 1/N; There are N \* (N-1) / 2 pairs of positions, for every pair the perfect match probability is 1 / (N \* (N - 1))

On the general case, there are C(N, k) = N! / (k! \* (N - k)!) subsets of k positions, and for every one of them the perfect match probability is (N - k)! / (N!). That means that for a given k we have a cummulative probability of 1 / (k!).

Applying the principle, we have to add the cummulative probabilities for odd values of k and subtract them for even values of k, which gives us a probability of 1 - 1/(2!) + 1/(3!) - ... - (-1) ^ k / (k!). For N = 4, that will be 1 - 1/2 + 1/6 - 1/24 = 0.625. For N >= 6, it will be approximately 0.632, with minor further variations.

**Problem 2 – Monty Hall**

**Statement:**

On a TV game show there are three closed doors. Suppose that there is a new car behind one of the doors and a goat behind each of the other two doors. A contestant selects a door at random and wins the prize behind it unless he or she wants to switch doors as now described. Before opening the contestant’s door to reveal the prize, here are three possible rules that the host could use to decide whether to open the door:

(i) The host of the show always opens one of the other two doors and reveals a goat, selecting the door randomly when each of these doors hides a goat.

(ii) The host only opens one of the other two doors to reveal a goat, selecting the door randomly, when the contestant has selected the door with a car behind it.

(iii) When the contestant’s door hides a car, the host randomly selects one of the other two doors and opens it to reveal a goat. When the contestant’s door hides a goat, half the time the host opens the other door that is hiding a goat.

The contestant is allowed to switch doors before the contestant’s door is opened to reveal their prize.

Suppose that the contestant initially selects door 1. For each of the three strategies of the host, use computer simulations (with at least 100 repetitions) to estimate the probability that the contestant wins a car (a) without switching, (b) with switching.

**Implementation in R:**

*randRoom = function() {*

*return (sample(1:3, 1)[1]);*

*}*

*randRoom();*

*simulate1 = function(sw) {*

*win = randRoom();*

*if (win == 1) {*

*# The contestant has chosen the winning door*

*# Doesn't matter what door the host chooses,*

*# the outcome is the same*

*if (sw) {*

*return (0);*

*}*

*return (1);*

*}*

*# The contestant has chosen the losing door*

*if (sw) {*

*return (1);*

*}*

*return (0);*

*}*

*simulate2 = function(sw) {*

*win = randRoom();*

*if (win == 1) {*

*if (sw) {*

*return (0)*

*}*

*return (1);*

*}*

*if (sw) {*

*if (sample(1:2, 1) == 1) {*

*return (1);*

*}*

*return (0);*

*}*

*return (0);*

*}*

*simulate3 = function(sw) {*

*win = randRoom();*

*if (win == 1) {*

*if (sw) {*

*return (0);*

*}*

*return (1);*

*}*

*if (sw) {*

*if (sample(1:2, 1) == 1) {*

*return (1);*

*}*

*if (sample(1:2, 1) == 1) {*

*return (1);*

*}*

*}*

*return (0);*

*}*

*simulate = function(count, simfnc, sw) {*

*success = 0;*

*for (i in 1:count) {*

*if (simfnc(sw)) {*

*success = success + 1;*

*}*

*}*

*return (success / count);*

*}*

*simulate(1000, simulate1, 0);*

*simulate(1000, simulate2, 0);*

*simulate(1000, simulate3, 0);*

*simulate(1000, simulate1, 1);*

*simulate(1000, simulate2, 1);*

*simulate(1000, simulate3, 1);*

**Experimental results:**

|  |  |  |
| --- | --- | --- |
| **Strategy number** | **Probability of success for 1000 simulations** | |
|  | 1. **without switching** | 1. **with switching** |
| (i) | 0.376 | 0.66 |
| (ii) | 0.338 | 0.286 |
| (iii) | 0.341 | 0.551 |

**Theoretical deduction of the results:**

For all strategies, if the contestant does not switch, the winning probability should be 1/3. If the contestant is switching, then there are 2 possibilities. If he initially selected the winning room, he will certainly select a losing one. If he selected a losing room, he will certainly select the winning room, if other losing room is open and will have 1/2 probability of doing so, if all doors are closed. We need to analyze the 3 strategies in the case where the contestant switches.

In the first strategy, the host always opens a losing door. Therefore, the contestant will always reverse his initial outcome. That provides him a winning chance of 2/3.

In the second strategy, if the contestant initially selected the winning door, after switching he will always lose. But if he initially selected a losing door, as no other doors are open, he will have chance of 1/2 of winning. That gives a total of 1/3 \* 0 + 2/3 \* 1/2 = 1/3 chances of winning.

In the third strategy, if the contestant initially selected the winning door, after switching he will always lose. Meanwhile, if he initially selected a losing door, there are 1/2 chances of a win, if the doors are closed and represent a certain win, if the other losing door is open. Thus the chances of winning are 1/3 \* 0 + 2/3 \* (1/2 \* 1/2 + 1/2 \* 1) = 1/2.