Monty Hall Simulation

Greg DeVore

July 16th, 2017

Overview

The goal of this assignment is to determine the optimal strategy for the classic "Monty Hall Problem", based on the 1970's television show "Let's Make a Deal", hosted by Monty Hall.

In the game, a contestant is presented three doors to choose from. Behind one of the doors is a prize (a car, for exmaple), and behind the other two doors are items of nominal value (goats, for example). The flow of the game is as follows:

1. The contestant chooses a door at random.
2. Monty opens one of the other two doors. The door opened by Monty always contains a goat.
3. The contestant is given the opportunity to swtich their choice to the door that was not opened by Monty.
4. Monty opens the contestants chosen door, reavealing either a car or a goat.

The question is, is it more advantageous for the contestant to stick with their original door, or switch doors when given the opportunity? To determine this, we'll run a simulation in R to explore both strategies, and see if one has a clear advantage. At the conclusion of this simulation, it will be found that by switching doors, a contestant actually doubles their chance of winning a car, versus if they had stayed with their original choice.

Simulation

The simulation will consist of 1000 rounds, where each round simulates 100 iterations of the game. Overall, 100,000 games will be played, and the simulation will be run for the strategies of staying versus switching. A function was created in R to run the simulation. The function will be presented, and then dissected in detail.

```
# Monty Hall Simulation
MontyHall <- function(rounds, strategy) {

    doors <- 1:3 # Three doors to choose from
    win_car <- rep(0,rounds) # Empty vectors for success statistics
    win_goat <- rep(0,rounds)
    nTrials <- 100 # Number of trials in each round

    for(i in 1:rounds) {

    # Initialize counters
    wins_car <- 0
    wins_goat <- 0

      for(j in 1:nTrials) {
        # Step 0: Prize is located behind a random door
        unif <- runif(1)
        prize <- ifelse(unif <= 0.33,1,ifelse(unif <= 0.66,2,3))

        # Step 1: Contestant picks a door at random
        unif <- runif(1)
```

```
        door <- ifelse(unif <= 0.33,1,ifelse(unif <= 0.66,2,3))

        # Step 2: Monty opens a door
        if (door == prize) { # Scenario 1: Prize is behind same door contestant chose.
          # Open either of the two remaining doors with equal probability
          open <- sample(doors[-c(prize)],1)
        } else { # Scenario 2: Prize is behind a door the contestant did not choose.
          # Open the door not chosen by contestant, and not containing prize
          open <- doors[-c(door,prize)]
        }

        # Step 3: Contestant decides whether or not to switch
        if (strategy == 'Stay') { # Stick with original choice
          final <- door
        } else { # Switch to door not opened by Monty
          final <- doors[-c(door,open)]
        }

        # Step 4: Monty reveals the prize
        if (final == prize) {
          wins_car <- wins_car + 1
        } else {
          wins_goat <- wins_goat + 1
        }
    }
    # Record number of wins for cars and goats
    win_car_percentage <- wins_car/nTrials
    win_goat_percentage <- wins_goat/nTrials
    win_car[i] <- win_car_percentage
    win_goat[i] <- win_goat_percentage
  }

  # Create data frame of results
  results <- data.frame(win_car = win_car, win_goat = win_goat)

  # Plot histograms
  bw1 = (max(win_car) - min(win_car))/10
  bw2 = (max(win_goat) - min(win_goat))/10
  p1 = ggplot(results, aes(win_car)) + geom_histogram(binwidth = bw1) + xlim(0,1) +
    ggtitle(paste('Win Car (',strategy,')')) + xlab('Success Rate')
  p2 = ggplot(results, aes(win_goat)) + geom_histogram(binwidth = bw2) + xlim(0,1) +
    ggtitle(paste('Win Goat (',strategy,')')) + xlab('Success Rate')
  grid.arrange(p1, p2, nrow = 2)

  return(results)
}
```

When the function is called, variables are created for the doors (1,2,3), and empty vectors are initialized to store the simulation results. Also, the number of trials for each round is set to 100.

```
doors <- 1:3 # Three doors to choose from
win_car <- rep(0,rounds) # Empty vectors for success statistics
win_goat <- rep(0,rounds)
nTrials <- 100 # Number of trials in each round
```

There are two loops within the function, an outer and an inner. The outer loop iterates over the number of rounds, and each time a new round is started, the number of wins for cars and goats is reset to zero.

```
# Initialize counters
wins_car <- 0
wins_goat <- 0
```

The inner loop iterates over the number of trials in each round, and this is where the bulk of the simulation occurs. The steps described in the comments will be covered in detail here.

0. The prize is located at random behind one of the three doors. To simulate this, a random number is generatd from a uniform distribution, and the door is assigned based on the outcome. Door 1 is assigned for numbers less than 1/3, door 2 is assigned for numbers between 1/3 and 2/3, and door 3 is assigned for numbers greater than 2/3. The prize has an equal chance of being placed behind any door.

```
unif <- runif(1)
prize <- ifelse(unif <= 0.33,1,ifelse(unif <= 0.66,2,3))
```

1. The contestant picks a door at random. This is simulated in the same manner as Step 0.

```
unif <- runif(1)
door <- ifelse(unif <= 0.33,1,ifelse(unif <= 0.66,2,3))
```

2. Monty opens a door containing a goat. There are two scenarios here, one in which the contestant picked the door containing the car, and one in which the contestant picked a door containing a goat.
   - If the contestant picked the door containing the car, Monty can open either of the two remaining doors with equal probability, since they each contain a goat. To simulate this, a random sample is taken from a vector containing those two doors. This will return one of the doors with equal probability.
   - If the contestant picked a door containing a goat, Monty must open the door not chosen by the contestant, and not containing the car. To simulate this, the single remaining door is returned.

```
if (door == prize) { # Scenario 1: Prize is behind same door contestant chose.
  # Open either of the two remaining doors with equal probability
  open <- sample(doors[-c(prize)],1)
} else { # Scenario 2: Prize is behind a door the contestant did not choose.
  # Open the door not chosen by contestant, and not containing prize
  open <- doors[-c(door,prize)]
}
```

3. The contestant decides whether to stay with their initial choice, or switch doors.
   - If the strategy is to stay, the contestants final choice is equal to their initial choice.
   - If the strategy is to switch, the contestant chooses the remaining door not opened by Monty.

```
if (strategy == 'Stay') { # Stick with original choice
  final <- door
} else { # Switch to door not opened by Monty
  final <- doors[-c(door,open)]
}
```

4. The location of the car is revealed. If this matches the contestants final choice, a win is recorded for a car. Otherwise, a win is recorded for a goat.

```
if (final == prize) {
  wins_car <- wins_car + 1
} else {
  wins_goat <- wins_goat + 1
}
```

Once the inner loop is completed, the win percentages for cars and goats is stored by dividing the number of wins for each by the total number of trials. The statistics are then stored for that round, the outer loop iterates, and the inner loop is run again.

```
# Record number of wins for cars and goats
win_car_percentage <- wins_car/nTrials
win_goat_percentage <- wins_goat/nTrials
win_car[i] <- win_car_percentage
win_goat[i] <- win_goat_percentage
```

Once all rounds and trials are completed, the results for all rounds are stored as a data frame and returned.

```
# Create data frame of results
results <- data.frame(win_car = win_car, win_goat = win_goat)
```

Also, histograms are created to show the results for winning a car versus winning a goat.

```
# Plot histograms
bw1 = (max(win_car) - min(win_car))/10
bw2 = (max(win_goat) - min(win_goat))/10
p1 = ggplot(results, aes(win_car)) + geom_histogram(binwidth = bw1) + xlim(0,1) +
  ggtitle(paste('Win Car (',strategy,')')) + xlab('Success Rate')
p2 = ggplot(results, aes(win_goat)) + geom_histogram(binwidth = bw2) + xlim(0,1) +
  ggtitle(paste('Win Goat (',strategy,')')) + xlab('Success Rate')
grid.arrange(p1, p2, nrow = 2)
```

Results

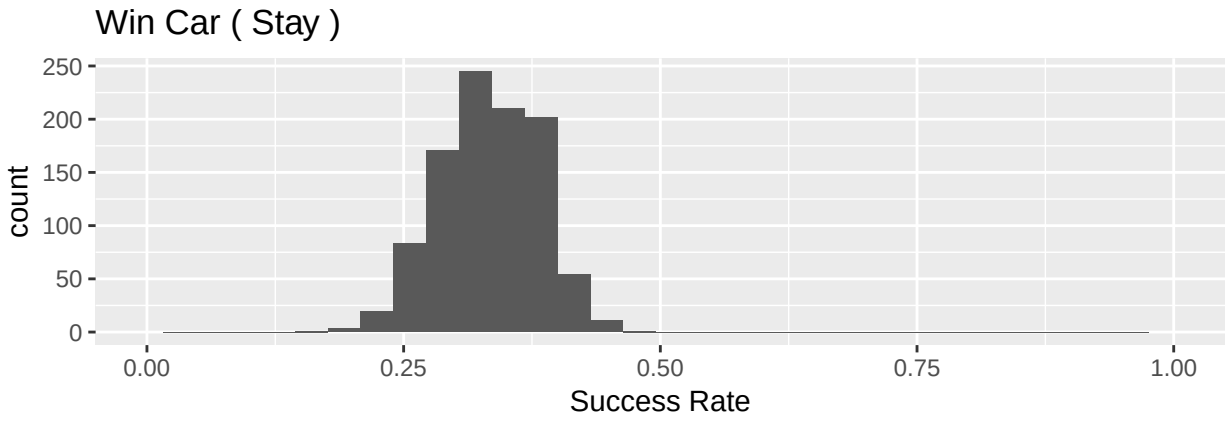
The simulation will now be run, both for the strategies of staying and switching. First, let's set the number of rounds to 1000.

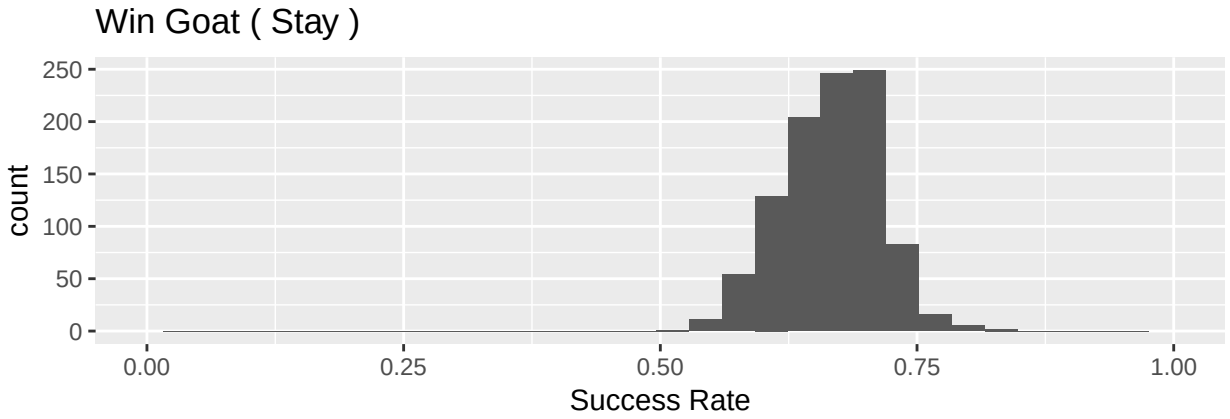
```
# Set number of rounds
rounds = 1000
```

Staying

Let's run the simulation using the strategy where the contestant stays with their initial guess.

```
# Strategy: Stay
results_stay <- MontyHall(rounds,'Stay')
```





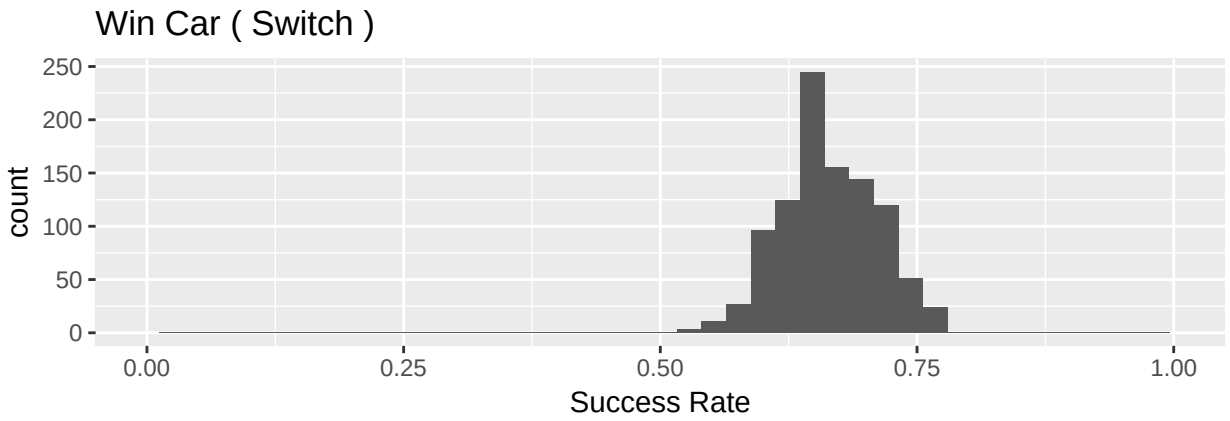
From the first histogram above, it appears that staying with your initial guess will result in winning the car roughly 1/3 of the time. That is, if you don't change your initial guess, you have a 1 in 3 chance of winning the car. This makes sense, because there are three doors and one prize, so your chance of picking the door containing the prize is 1 in 3.

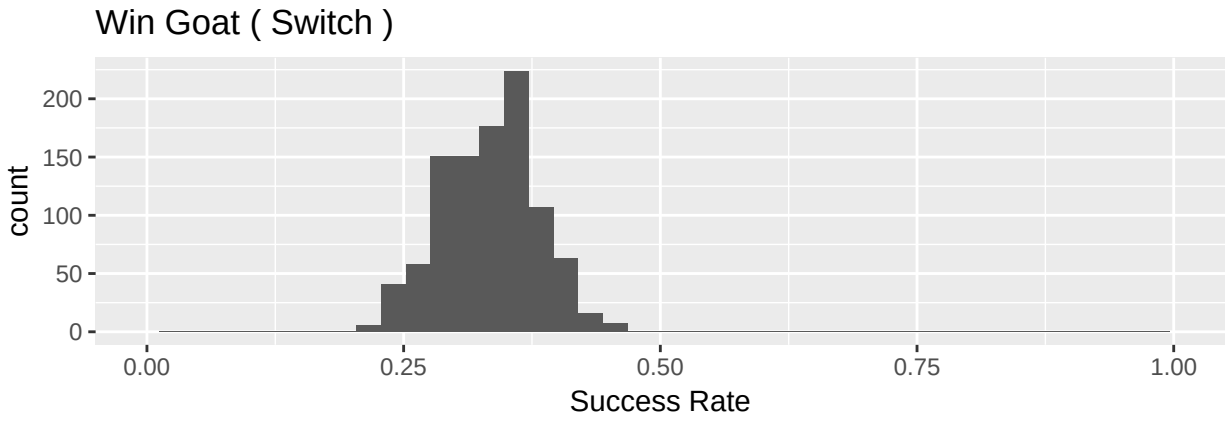
Also, since the probability of the entire sample space is 1, the chance of winning a goat with the strategy of staying with your initial guess should be one minus the chance of winning the car, which is 2/3. That is indeed the case, as shown in the second histogram.

Switching

Now, let's run the simulation using the strategy where the contestant switches from their initial guess when given the chance.

```
# Strategy: Switch
results_switch <- MontyHall(rounds,'Switch')
```





From the first histogram above, it now appears that switching from your initial guess when given the chance will result in winning the car roughly 2/3 of the time. That is, if you change your initial guess, you have a 2 in 3 chance of winning the car. This is a little harder to see at first, but one way to think of it is that your initial guess has a 1 in 3 chance of containing the car (as described in the previous section), and the other two doors combined have a 2 in 3 chance of containing the car. When Monty opens one of those two doors, the probability of that door containing the prize drops to zero (because that door always contains a goat). Since the probability of the entire sample space must be 1, the remaining door now has a 2 in 3 chance of containing the car. The door chosen initially still has a 1 in 3 chance of containing the car, since no new information was revealed about that door. So overall, switching increases the chances of winning the car from 1 in 3 to 2 in 3.

Also, as stated before, since the probability of the entire sample space is 1, the chance of winning a goat with the strategy of switching your initial guess should be one minus the chance of winning the car, which is 1/3. That is indeed the case, as shown in the second histogram.

Summary and Conclusion

To summarize, let's start by calculating and printing the summary statistics (mean and variance) of the two strategies:

```
stay_mean <- apply(results_stay,2,mean)
switch_mean <- apply(results_switch,2,mean)
stay_var <- apply(results_stay,2,var)
switch_var <- apply(results_switch,2,var)

summary_statistics <- rbind(Stay = c(stay_mean,stay_var),
                            Switch = c(switch_mean,switch_var))
colnames(summary_statistics) <- c('Win.Car.Mean','Win.Goat.Mean',
                                  'Win.Car.Var','Win.Goat.Var')
kable(summary_statistics, caption = 'Summary Statistics For Monty Hall Strategies')
```

Table 1: Summary Statistics For Monty Hall Strategies

	Win.Car.Mean	Win.Goat.Mean	Win.Car.Var	Win.Goat.Var
Stay	0.33378	0.66622	0.0021685	0.0021685
Switch	0.66531	0.33469	0.0020944	0.0020944

From the summary statistics above, it is clear that using a strategy of switching will result in winning a car twice as often as a strategy of staying. The variance for both strategies is very low, meaning the average results from any given game will not stray far from these computed values.

In conclusion, given the results of the simulation and computed summary statistics, it is recommended that the contestant adopt a strategy of switching doors to have the greatest chance of winning a car. By using a strategy of switching, the contestant has a 2 in 3 chance of winning the car, versus only a 1 in 3 chance if they were to use a strategy of staying with their initial guess.