Martingale Madness: The Gambler's Fate

Jolie Ng

2025-02-13

Casinos are famous for always having the upper hand in the long run, but that hasn't stopped people from trying to beat the odds. One strategy that often gets attention is the "Martingale" strategy, especially in games like roulette, one of the most iconic casino games. In roulette, players bet on where a small ball will land on a spinning wheel divided into numbered pockets, with options to bet on colors (red, black or green), odd or even numbers, or specific numbers.

How the Martingale Strategy Works

The Martingale strategy is simple and appeals to many players because it is straightforward: You start by betting $1. If you win, great! If you lose, you double your bet and continue until you win. The idea is that when you eventually win, you'll recover all previous losses, plus make a a profit. It sounds logical - after all, a win is bound to happen at some point, right? But what happens if you keep losing? And is this a reliable way to beat the casino?

Here, we'll take a deeper look at how this strategy works, the math behind it, and whether it's truly a viable approach. To understand its effectiveness (or lack thereof), we'll also run a computer simulation and see how this strategy plays out over time. Can you really "cheat" the casino, or is this just another way to lose money faster? Let's find out.

Simulating the Roulette Wheel

We first simulate the basic structure of a roulette game. In this version, there are 38 pockets on the wheel: 18 red, 18 black, and 2 green.

```
single_spin <- function(){
  possible_outcomes <- c(rep("red",18), rep("black",18), rep("green",2))
  sample(possible_outcomes, 1)
}
```

The `single_spin` function models a single spin of the roulette wheel, which randomly selects a pocket, which is either "red," "black," or "green." When using the Martingale strategy, we're particularly interested in the red or black outcomes since we only bet on black (or red).

Simulating the Martingale Strategy

```
martingale_wager <- function(previous_wager,
                             previous_outcome,
                             max_wager,
                             current_budget){

  if(previous_outcome == "red") return(1)
  min(2 * previous_wager, max_wager, current_budget)
}

one_play <- function(previous_ledger_entry, max_wager){
  out <- previous_ledger_entry
  out[1, "game_index"] <- previous_ledger_entry[1, "game_index"] + 1
  out[1, "starting_budget"] <- previous_ledger_entry[1, "ending_budget"]
  out[1, "wager"] <- martingale_wager(
    previous_wager = previous_ledger_entry[1, "wager"]
    , previous_outcome = previous_ledger_entry[1, "outcome"]
    , max_wager = max_wager
    , current_budget = out[1, "starting_budget"]
  )
  out[1, "outcome"] <- single_spin()
  out[1, "ending_budget"] <- out[1, "starting_budget"] +
    ifelse(out[1, "outcome"] == "red", +1, -1)*out[1, "wager"]
  return(out)
}
```

The `one_play()` function simulates the Martingale betting strategy in roulette for a single spin. The gambler places a bet, the wheel spins, and based on the outcome (red or black), the gambler either wins or loses. Then, it updates the player's wager and budget based on the results of that round, and makes a decision on how much to wager next: If the previous outcome was a win ("red"), the bet resets to $1. If the gambler lost, the next wager doubles. The bet is capped by the gambler's available budget and a maximum wager (set by the casino). By having these limits, it makes sure that the gambler doesn't bet more than they have.

Simulating Multiple Games and Stopping Rule

Now that we can simulate one game, let's see how this strategy performs over multiple games. The gambler will continue betting until one of the following conditions is met:

- The gambler's budget reaches a certain winning threshold (in our case, $300).
- The gambler runs out of money (bankruptcy).
- The gambler has played a maximum number of games (in our case, 1,000 plays).

```
one_series <- function(max_games,
                       starting_budget,
                       winning_threshold,
                       max_wager){
  ledger <- data.frame(
    game_index = 0:max_games,
    starting_budget = NA_integer_,
    wager = NA_integer_,
    outcome = NA_character_,
    ending_budget = NA_integer_
  )
  ledger[1, "wager"] <- 1
  ledger[1, "outcome"] <- "red"
  ledger[1, "ending_budget"] <- starting_budget

  for(i in 2:nrow(ledger)){
    ledger[i,] <- one_play(ledger[i-1,], max_wager)

    if(stopping_rule(ledger[i,], winning_threshold)) {
      break
    }
  }
  ledger[2:i, ]
}
```

When the function `one_play()` above is called, it takes into account the previous game's results and the maximum wager, and returns the results of the current play (this information is then logged in the corresponding row in the ledger for the next game).

```
stopping_rule <- function(ledger_entry, winning_threshold){
  ending_budget <- ledger_entry[1, "ending_budget"]
  if(ending_budget <= 0) return(TRUE)
  if(ending_budget >= winning_threshold) return(TRUE)
```

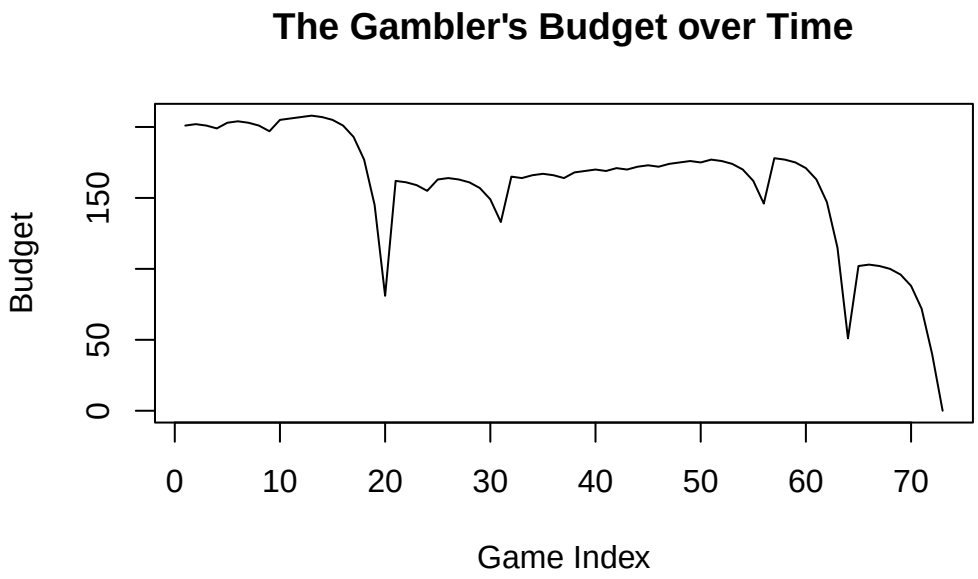
```
  FALSE
}
```

After each play, a `stopping_rule()` function is also called to check if any of the above stopping conditions are met. If so, the loop breaks and stops the simulation. This symbolizes the gambler's decision to stop playing. The `stopping_rule` function is above.

Visualizing the Gambler's Earnings

To visualize how the gambler's earnings evolve, we can run the simulation and plot the results:

```
set.seed(1)
ledger <- one_series(1000, 200, 300, 100)
plot(ledger[,c(1,5)],
     type = "l",
     xlab = "Game Index",
     ylab = "Budget",
     main = "The Gambler's Budget over Time")
```



This plot shows how the gambler's budget fluctuates over time. The x-axis represents the number of plays (or spins), and the y-axis shows the gambler's budget. With each spin, we

see how the Martingale strategy plays out. We see that as the number of plays increases, the gambler's budget never seems quite get back to, or go over, the original budget, suggesting that the gambler will lose money in the long run. Let's take a step further in our analysis.

Estimating the Average Profit & Number of Plays Before Stopping

Something else we can estimate from our simulation is the average profit and/or the average number of games the gambler can play before they either hit their win target or go bankrupt. The profit function below returns the gambler's earning for a single simulation.

```
profit <- function(ledger){
  n <- nrow(ledger)
  profit <- ledger[n, "ending_budget"] - ledger[1, "starting_budget"]
  return(profit) # to calculate average earnings
}

num_plays <- function(ledger){
  return(nrow(ledger))  # to calculate the average number of plays
}
```

We can then choose the number of times we run the `one_series()` command (in the code below I simulated 1000 runs), then take the average of the profits/plays simulated in each run. The more simulations we run, the more accurate our estimate will be.

```
average_num_plays <- replicate(1000,
                               one_series(1000, 200, 300, 100)
                               |> num_plays()) |> mean()
cat("Average Number of Plays: ", average_num_plays, "\n")
```

```
Average Number of Plays:  198.246
```

```
average_profit <- replicate(1000,
                            one_series(1000, 200, 300, 100)
                            |> profit()) |> mean()
cat("Average Profit: ", average_profit, "\n")
```

```
Average Profit:  -48.5
```

The fact that the average profit is negative means that on average, the gambler loses money to the casino. Now, let's look at how this amount differs based on different stopping conditions.

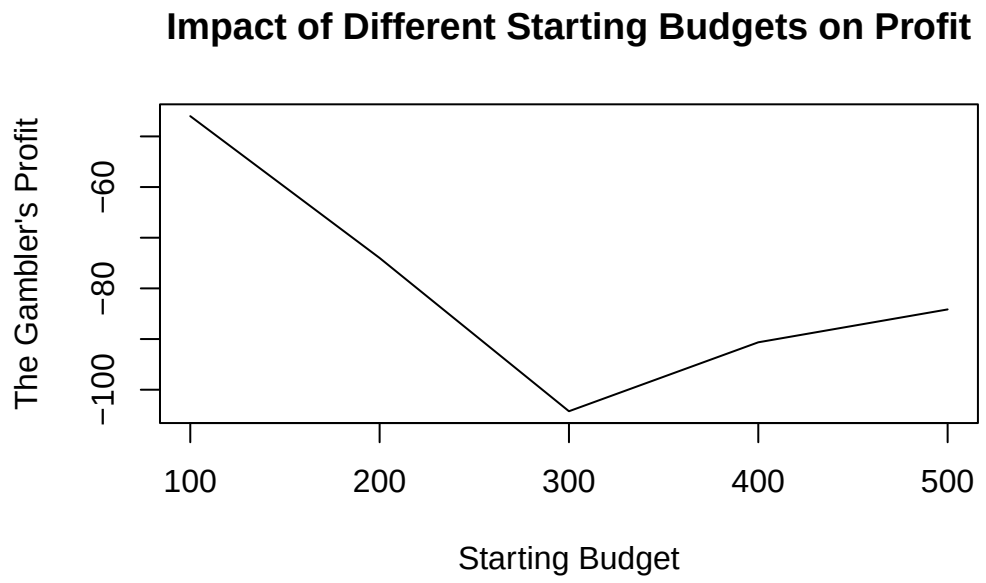
Impact of Changing Parameters

One of the key benefits of running a simulation is that we can tweak various parameters to see their impact on the outcome. Below, I have varied four parameters in our simulation:

1. **Starting Budget**: Increasing the starting budget gives the gambler more room to withstand a long losing streak, but also risks them losing more money.

```
starting_budget <- seq(100, 500, by = 100)
# produce a range of starting budgets from $100 to $1000 (in $100 increments)
earnings <- numeric(length(starting_budget))

for(n in 1:length(starting_budget)){
  earnings[n] <- replicate (100, one_series (1000,
                                             starting_budget[n],
                                             starting_budget[n] + 100,
                                             100) |> profit()) |> mean()}
plot(starting_budget, earnings,
     type = "l",
     main = "Impact of Different Starting Budgets on Profit",
     ylab = "The Gambler's Profit",
     xlab = "Starting Budget")
```

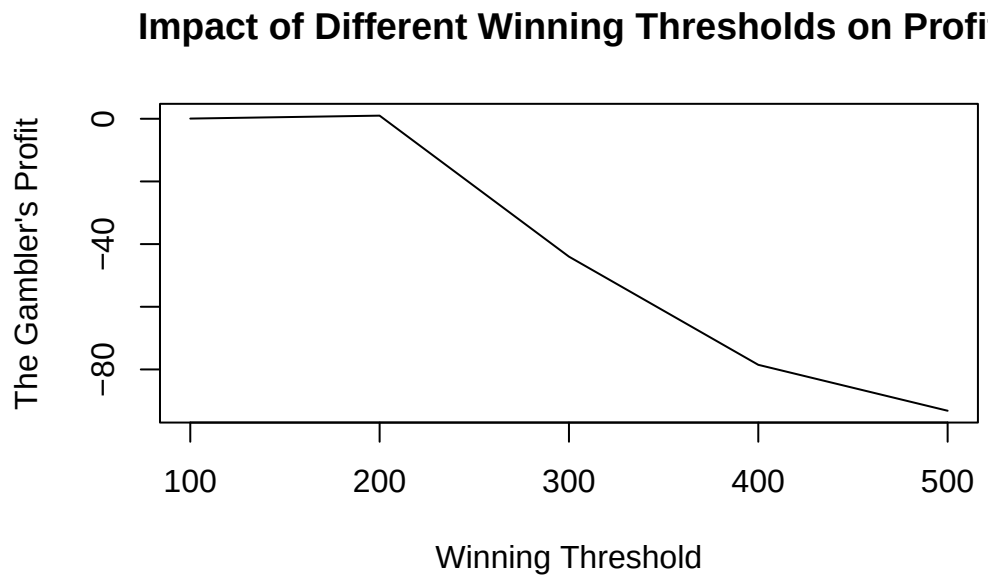


From above, we see that no matter the starting budget, the gambler will lose money. In particular, $300 seems to be the worst budget the gambler should start with.

2. **Winning Threshold**: Raising the winning threshold means the gambler can play for longer, which means they can potentially win/lose more.

```
winning_threshold <- seq(100, 500, by = 100)
# produce a range of winning thresholds from $100 to $1000 (in $100 increments)
earnings_threshold <- numeric(length(winning_threshold))

for(n in 1:length(winning_threshold)){
  earnings_threshold[n] <- replicate (100,
                                      one_series (1000,
                                                  200,
                                                  winning_threshold[n],
                                                  100) |> profit()) |> mean()}
plot(winning_threshold, earnings_threshold,
     type = "l",
     main = "Impact of Different Winning Thresholds on Profit",
     ylab = "The Gambler's Profit",
     xlab = "Winning Threshold")
```



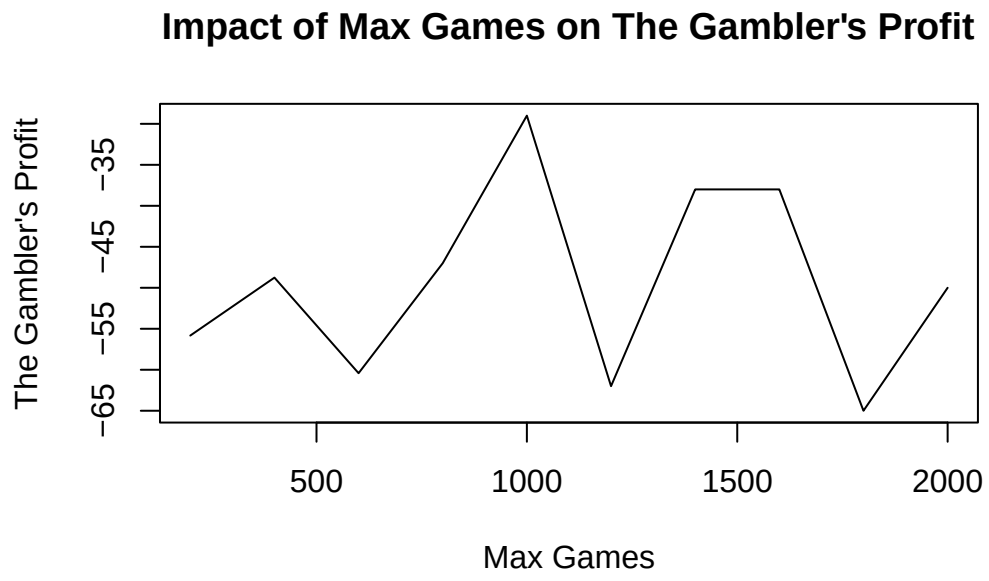
From above, we can see that the larger the gambler's winning threshold is, the more probable it is for them to lose large sums of money. When the gambler's winning threshold is small

(<200), their final earnings seems to be the same as or even a little more than their starting budget. This makes sense, because it means that the gambler is giving up on small wins to bet on winning a larger amount of money, which is a very risky endeavor.

3. **Max Games**: Setting a low maximum number of games limits how long the gambler can play, which can protect against large losses but also reduce the chances of a big win.

```
max_games <- seq(200, 2000, by = 200)
# produce a range of max games from 100 to 1000 (in 100 game increments)
earnings_games <- numeric(length(max_games))

for(n in 1:length(max_games)){
  earnings_games[n] <- replicate (100, one_series (max_games[n],
                                                   200,
                                                   300,
                                                   100) |> profit()) |> mean()}
plot(max_games, earnings_games,
     type = "l",
     main = "Impact of Max Games on The Gambler's Profit",
     ylab = "The Gambler's Profit",
     xlab = "Max Games")
```



Varying the maximum number of games the gambler plays does indeed vary the gambler's profit, but there is no correlation trend between the two.

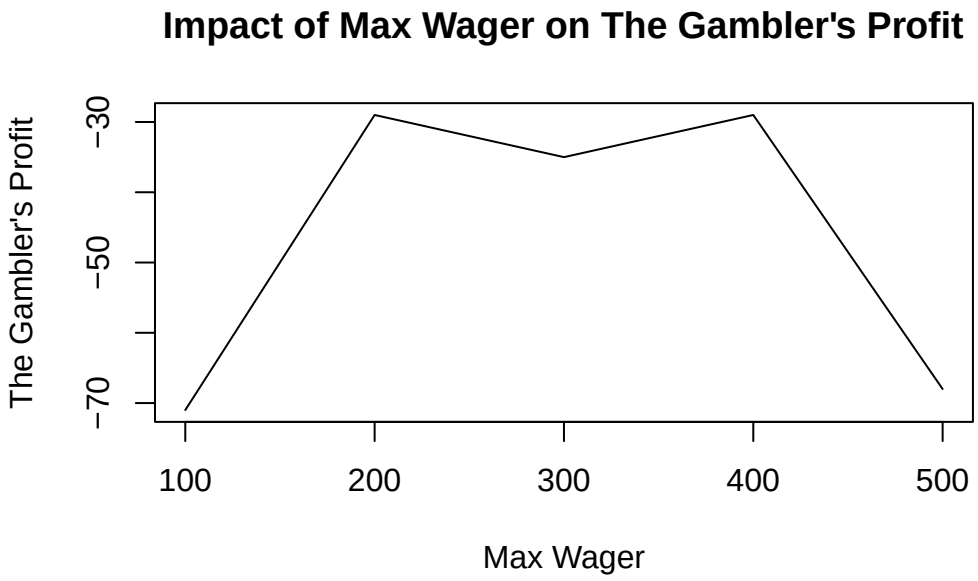
4. **Max Wager**: The casino's maximum bet rule limits the strategy's potential for recovery after several losses, reducing the gambler's chance of winning back previous losses.

```
max_wager <- seq(100, 500, by = 100)
# produce a range of max wager values from $10 to $500 (in $50 increments)
earnings_wager <- numeric(length(max_wager))

for(n in 1:length(max_wager)){
  earnings_wager[n] <- replicate (100,
                                  one_series (1000,
                                              200,
                                              300,
                                              max_wager[n])|> profit()) |> mean()
}

# the format above is:
# one_series (max_games, starting_budget, winning_threshold, max_wager)

plot(max_wager, earnings_wager,
     type = "l",
     main = "Impact of Max Wager on The Gambler's Profit",
     ylab = "The Gambler's Profit",
     xlab = "Max Wager")
```



As we can see from the graph above, changing the maximum number of games the gambler can play definitely changes the amount that they profit/lose. It also seems like when the maximum wager is between $200-$400, the gambler loses the least amount of money compared the a maximum wager of $100 or $500. This suggests that a casino can implement a maximum wager that statistically causes gamblers to lose more money, such that the house will always beat the odds.

Limitations and Simplifications

While simulations like this give us a good sense of how a strategy might play out, they come with limitations. For one, this simulation doesn't account for player behavior — real gamblers might not follow the strategy perfectly and make split-second decisions due to greed, other people's influence, etc. There's also the matter of casino rules (like the maximum bet limit) which can greatly impact the strategy's effectiveness. Additionally, the single spin function we have assumes that each roulette spin is fair and every pocket has an equal chance of being selected, which ignores the possibility that certain pockets might be rigged by the house and have a higher or lower chance of being selected. All of these limitations are warnings to tell us that although our simulation might show that in some (unlikely) cases with certain parameters, the gambler can gain a small amount of profit, the casino always implements restrictions to ensure that the house always wins in the long run.

Conclusion

The Martingale strategy may look tempting on paper, but as our simulation shows, it's a risky game, and gamblers tend to lose money over profiting. This is because the gambler needs deep pockets and plenty of time to survive long losing streaks, and even then, casinos often showing us that while the gambler may be able to win small profits in the short run, the odds are always stacked in the house's favor.