Simulation of Martingale Strategy on Roulette

1. Introduction to Roulette Game

1.1 Roulette Rules

A roulette table composed of 38 (or 37) evenly sized pockets on a wheel. The pockets are colored red, black, or green. Roulette is a game of chance in which a pocket is randomly selected. Gamblers may wager on several aspects of the outcome. For example, one may place a wager that the randomly selected pocket will be red or odd numbered or will be a specific number.

For this blog post, we only consider betting on colors. On the roulette, there are 38 pockets of which 2 are green, 18 are red, and 18 are black. The payout for a bet on black (or red) is $1 for each $1 wagered. This means that if a gambler bets $1 on black and the randomly selected pocket is black, then the gambler will get the original $1 wager and an additional $1 as winnings.

1.2 The "Martingale" Strategy

2. Simulation

2.1 Simulating a Single Play

To start, we first try to simulate one single play. We bet on red, if the result turns out to be red, then we win; if the result is black, then we lose. We change the status according to our result.

Status includes:

Parameter	Description
B	Starting budget
W	Winnings threshold for stopping
L	Time threshold for stopping
M	Casino's maximum wager

```
#needed to use the pipe '%>%'
library(magrittr)

#' A single play of the Martingale strategy
#'

#' Takes a state list, spins the roulette wheel, returns the state list with updated values (for exampl
#' @param state A list with the following entries:
#'   B              number, the budget
#'   W              number, the budget threshold for successfully stoping
#'   L              number, the maximum number of plays
#'   M              number, the casino wager limit
#'   plays          integer, the number of plays executed
#'   previous_wager number, the wager in the previous play (0 at first play)
#'   previous_win   TRUE/FALSE, indicator if the previous play was a win (TRUE at first play)
#' @return The updated state list
one_play <- function(state){

    # Wager
```

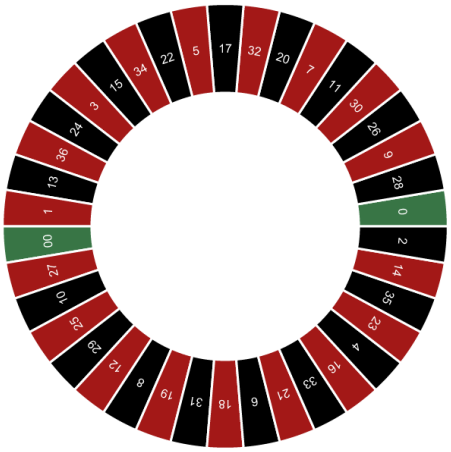


Figure 1:

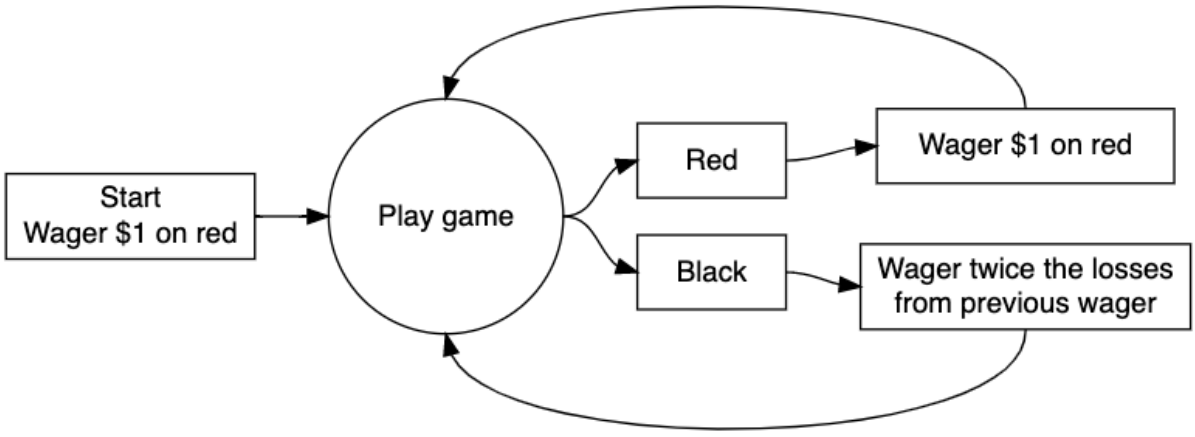


Figure 2:

```
    proposed_wager <- ifelse(state$previous_win, 1, 2*state$previous_wager)
    wager <- min(proposed_wager, state$M, state$B)

    # Spin of the wheel
    red <- rbinom(1,1,18/38)

    # Update state
    state$plays <- state$plays + 1
    state$previous_wager <- wager
    if(red){
      # WIN
      state$B <- state$B + wager
      state$previous_win <- TRUE
    }else{
      # LOSE
      state$B <- state$B - wager
      state$previous_win <- FALSE
    }
  state
}
```

2.2 Stopping Rules

A player stops when either one of the three conditions happenes:

1. the player has W dollars
2. the player goes bankrupt
3. the player completes L wagers (or plays)

```
#' Stopping rule
#'
#' Takes the state list and determines if the gambler has to stop
#' @param state A list.  See one_play
#' @return TRUE/FALSE
stop_play <- function(state){
  if(state$B <= 0) return(TRUE)
  if(state$plays >= state$L) return(TRUE)
  if(state$B >= state$W) return(TRUE)
  FALSE
}
```

2.3 Simulating a Single Series

We assume that a player starts playing with $200 budgets. The player continues to play until one of the stopping rules is met.

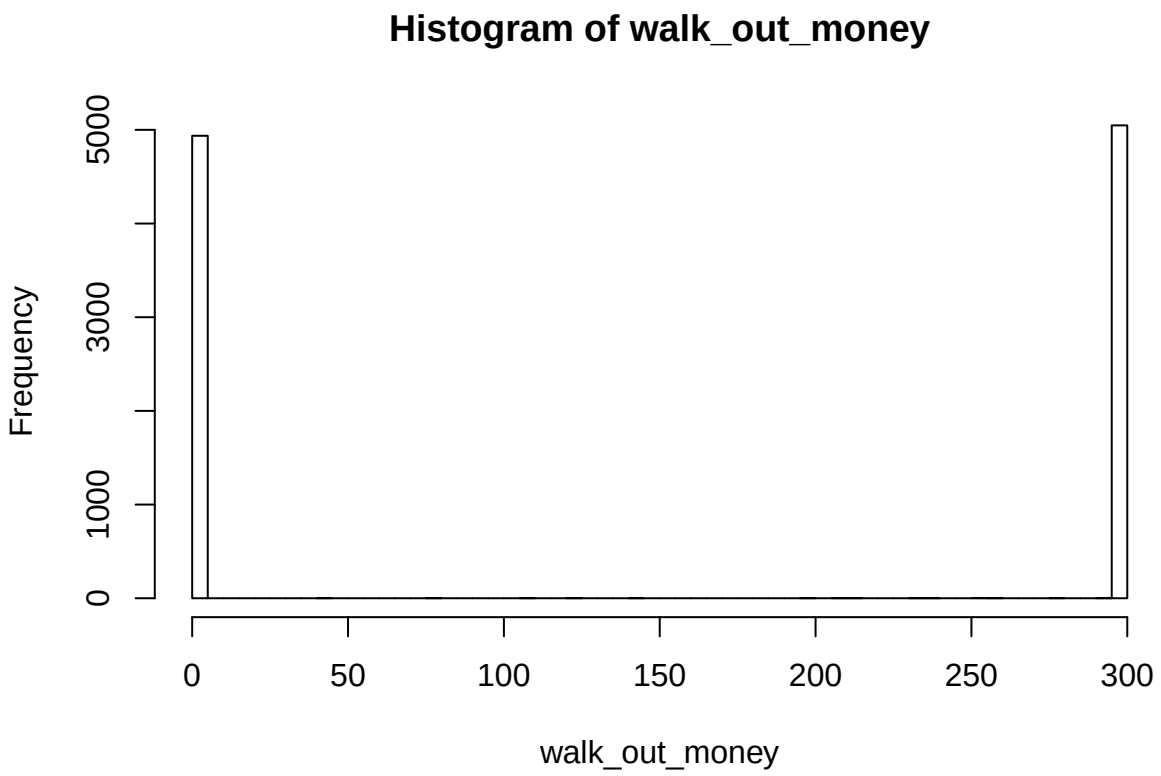
```
#' Play roulette to either bankruptcy, success, or play limits
#'
#' @param B number, the starting budget
#' @param W number, the budget threshold for successfully stoping
#' @param L number, the maximum number of plays
#' @param M number, the casino wager limit
#' @return A vector of budget values calculated after each play.
one_series <- function(
    B = 200
```

```
  , W = 300
  , L = 1000
  , M = 100
){

  # initial state
  state <- list(
    B = B
  , W = W
  , L = L
  , M = M
  , plays = 0
  , previous_wager = 0
  , previous_win = TRUE
  )

  # vector to store budget over series of plays
  budget <- rep(NA, L)

  # For loop of plays
  for(i in 1:L){
    new_state <- state %>% one_play
    budget[i] <- new_state$B
    if(new_state %>% stop_play){
      return(budget[1:i])
    }
    state <- new_state
  }
  budget
}

# helper function
get_last <- function(x) x[length(x)]
```

2.4 Estimating the average earning/loss

We run the simulation for single series 10K times in order to get the a approximated earning/loss.

```
# Simulation
walk_out_money <- rep(NA, 10000)
for(j in seq_along(walk_out_money)){
  walk_out_money[j] <- one_series(B = 200, W = 300, L = 1000, M = 100) %>% get_last
}

# Walk out money distribution
hist(walk_out_money, breaks = 100)
```



We can see that among the 10K times, the player either walk out with $300 (winning threshold) or $0 (losing everything).

```
# Estimated probability of walking out with extra cash
mean(walk_out_money > 200)
```

```
## [1] 0.5057
```

```
# Estimated earnings
mean(walk_out_money - 200)
```

```
## [1] -48.2686
```

About 50% of the time, the player actually has a earning, and 50% chance is that the player has a loss. Since half of the time, the player wins $100 ($300 walk out money - $200 starting budget), and half of the time, the player loses $200 (all the starting budget), the average earning is around -$45.

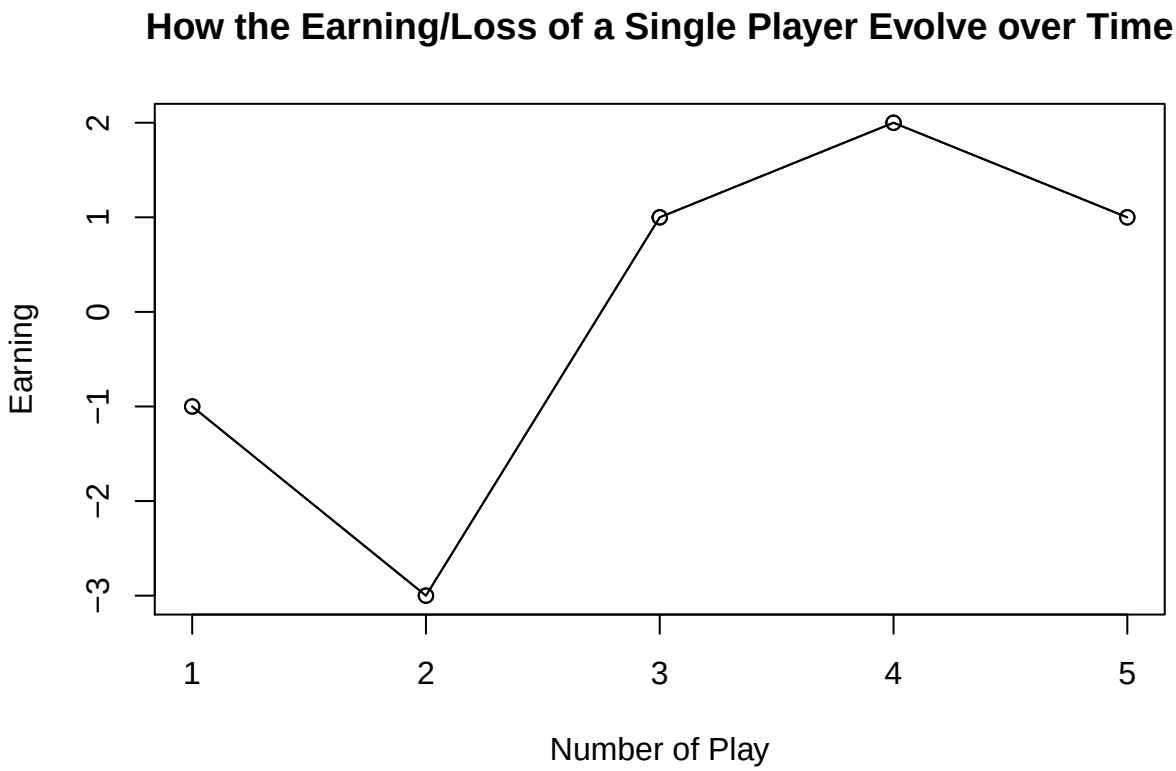
3. More analysis

3.1 How does the player's earning/loss evolve over his course of playing?

We now see how one player's earning/loss evolve. Take a look of the budet change for the first 5 wagers:

```
# How the earning of one player vary over the whole time of his playing
# Plot one player's budget change over the time of wagers

  set.seed(8)
#calculate earning and time of play
  earning <- one_series(B = 200, W = 300, L = 1000, M = 100) - 200
  time_of_play <- (1:length(earning))

# Make a line chart for first 5 plays
```

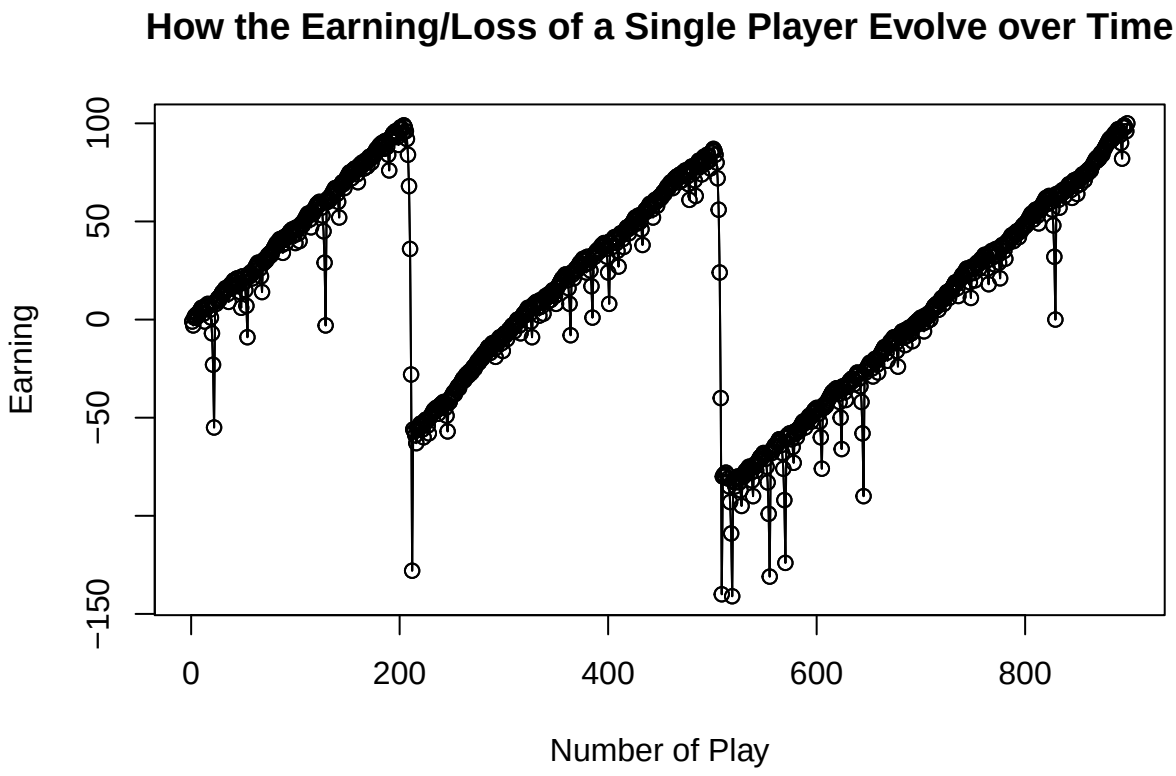
```r
  plot (time_of_play[0:5], earning[1:5], type = "o", main = "How the Earning/Loss of a Single Player Ev
  lines(time_of_play[0:5], earning[1:5], type = "o")
```



Play	Wager	Outcome	Earnings
1	1	Black	-1
2	2	Black	-3
3	4	Red	1
4	1	Red	2
5	1	Black	1

Then we want to the change of earning/loss for the whole process, from the point the player starts to the time one of the stopping conditions is met.

```r
# Make a line chart
  set.seed(8)
  plot (time_of_play, earning, type = "o", main = "How the Earning/Loss of a Single Player Evolve over
  lines(time_of_play, earning, type = "o")
```



It's clear that from the graph that when the player is winning in a row, his/her earning increases in a linear order. On the other side, there are two obvious drops of the earning happening around 200 and 450 plays. Notice that when losing several plays all at once can cause exponential decrease in the earning. So the speed of losing money is faster than that of winning.

3.2 How does a change in parameter impact average earnings?

For convenience, I wrote a function to generate one simulation, which includes 1000 series samples, and output the average earning for those samples.

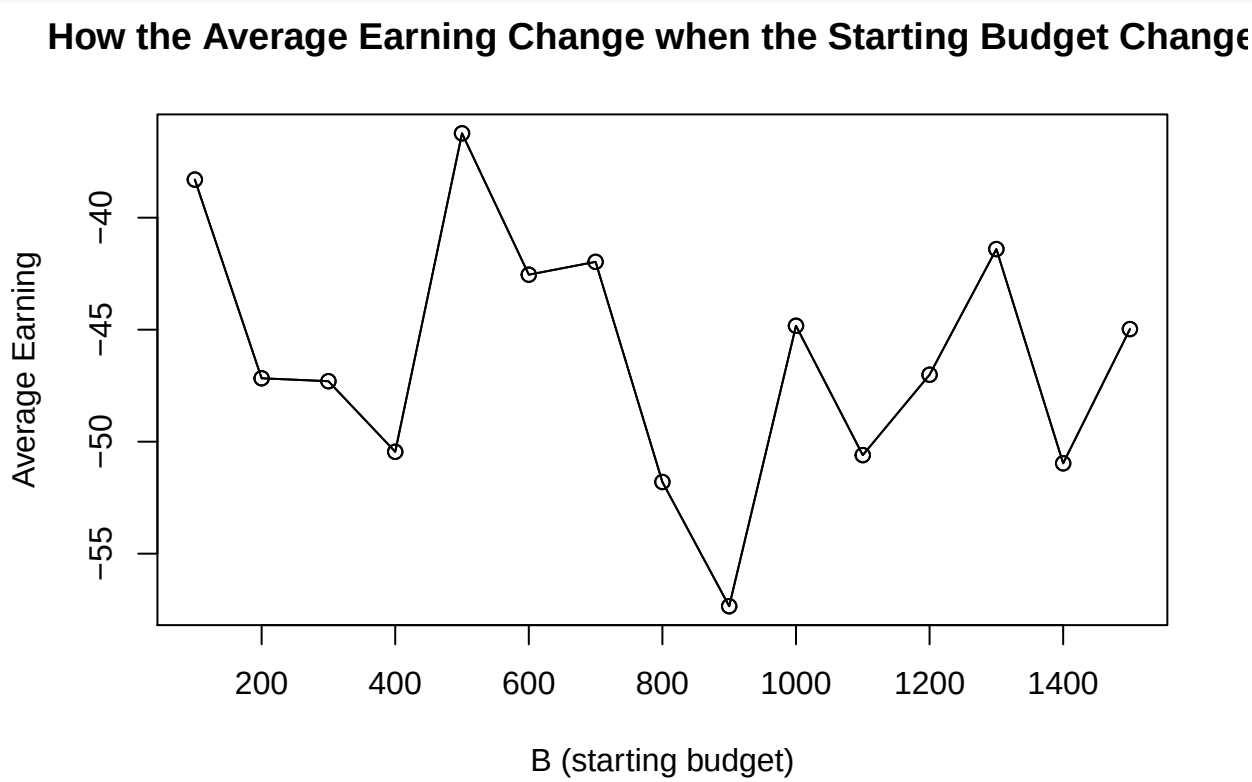
```
# how changing a parameter of the simulation (see table below) does or does not have an impact on avera
#' @param B number, the starting budget
#' @param W number, the budget threshold for successfully stoping
#' @param L number, the maximum number of plays
#' @param M number, the casino wager limit

# Simulation
# Output average earning
one_simulation <- function(
  B = 200
  , W = 300
  , L = 1000
  , M = 100
){
  walk_out_money <- rep(NA, 1000)  #change the rep times to 10000
  for(j in seq_along(walk_out_money)){
    walk_out_money[j] <- one_series(B = 200, W = 300, L = 1000, M = 100) %>% get_last
  }

  return(mean(walk_out_money - 200))
```

```
}
```

Then, we want to take a look at the way earnings change when parameter B, the starting budget, increases from $100 to $1,500.

```
# The variation time of one variable
time_of_variation = 15
set.seed(150)
# Change B, the starting budget
b <- seq(100, 100*time_of_variation, by = 100)

earning <- rep(NA, time_of_variation)

for(i in 1:time_of_variation){
  earning[i] <- one_simulation(
    B = b
  , W = 300
  , L = 1000
  , M = 100)
}
```

```
  plot (b, earning, type = "o", main = "How the Average Earning Change when the Starting Budget Changes
  lines(b, earning, type = "o")
```



Despite some fluctuations, we can see a decreasing trend for average earning when the starting budget are getting bigger. This phenomena might be explained by the fact that when the player has more starting budget, they potentially lose more money (remember there is a 50% chance that the player walks out of the casino with no money)! In the meantime, the threshold for winning does not change, so when the player wins, he/she wins the same amount. Losing more and winning the same can be the cause of the decreasing average earning.

3.3 How many times do you play before the game ends?

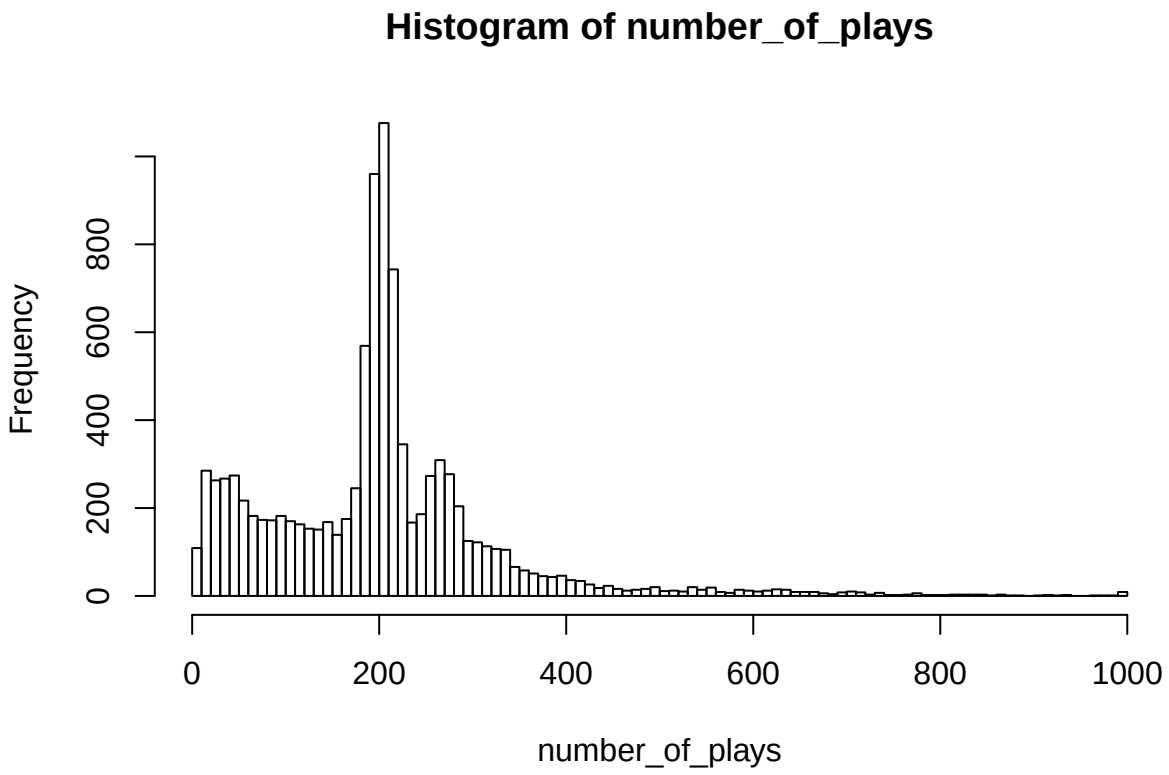
Here, we run the simulation 10K times, for each time we record the number of plays before the game ends, and then take the average.

```
# Estimate the average number of plays before stopping

number_of_plays <- rep(NA, 10000)
for(j in seq_along(walk_out_money)){

  one_series_result <- one_series(B = 200, W = 300, L = 1000, M = 100)

  number_of_plays[j] <- length(one_series_result)
}

mean(number_of_plays)
```

```
## [1] 200.9309
```

In average, a player would play 200 times before one of the stopping conditions are satisfied and thus ends the game. The distribution is as follows:

```
# Number of plays distribution
hist(number_of_plays, breaks = 100)
```



4. Limitations and Uncertainties

1. Due to limiting computing power, when running simulations for section 3.2 (how does changing a parameter would change the average earning), only 1K samples (series) are used, comparing the 10K samples for other parts. In addition, plotting only 15 points might be oversimplification, as we can't predict the trend outside this range. As a result, the obeservation might not be accurate enough.

2. There are always uncertainties generated by randomness in the process of simulation. For example, in section 2.4, when we try to estimate the average earning, the result is -$45, but mathematically speaking, the expected earning should be -$50.

3. In section 3.1, we plotted a player's earning during the course of his game. In the example we give, the play stopped when he/she reaches 1000 plays. This example might not be representative, since the average time of play we calculated in section 3.3 is around 200. Also, the chance of the player walk out of casino with earning or losing is roughly the same, and the graph only shows one of the two scenarios. (If the player walk out with $0, then it's likely that he/she continues to lose such that all his/her budgets are depleted before a single win.)