roulette simulation

Probability and Inference — Deliverable 01

Roulette Simulation

Rastko Stojsin

Simulations are commonly used when the math required to answer a problem is complex and would take a lot of time to work out. Simulations can be used and may be quicker to run than figuring out how to setup and solve complex math problems.

The Martingale strategy for roulette is an interesting strategy that due to its evolving and iterative nature, is better suited to be evalued through simulations. The simulations are evolving due to their operating characteristics, which are influential to the outcome and change throughout/between the simulation(s).

Below is the R code required to setup a simulation of the Martingale strategy at a roulette table. Feel free to step through the code and try to understand the moving pieces and what each code block does. This will be harder without a knowledge of R, but you should be able to get a fundamental understanding nonetheless.

This code is broken down into many little manageable pieces based on tasks, allowing for easy understanding, debugging, and editing.

```
#' 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
    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
}


#' 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
}


#' 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
  }
```

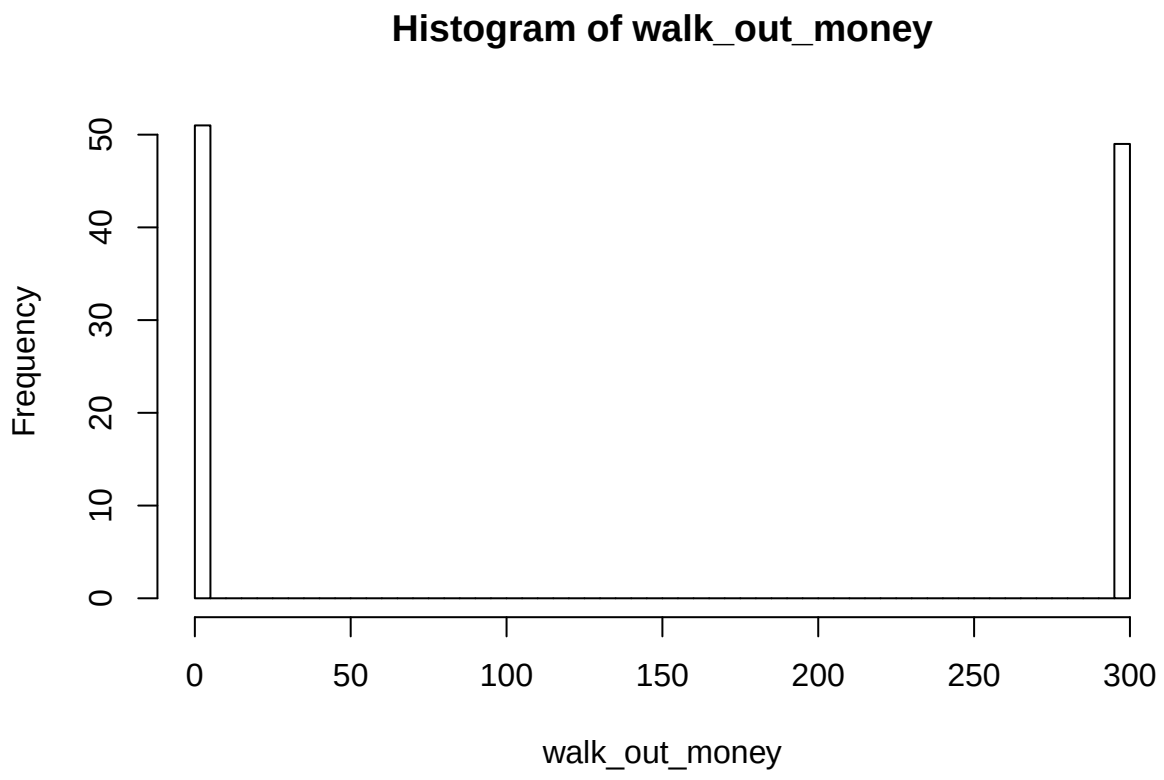
```
    budget
}

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

Original Simulation

Below is the code of a loop of 1000 simulations where the player enters the casino with 200, will leave if they reach 300, can play a maximum 1000 number of times (reasonable time constraint), and plays at a table with a 100 wager limit (standard at casinos). I used and will continue to use a seed value, for reproductability so those following along may reach the same results

```
# Simulation
set.seed(238)
walk_out_money <- rep(NA, 100)
for(j in seq_along(walk_out_money)){
  walk_out_money[j] <- one_series(B = 200, W = 300, L = 1000, M = 100) %>% get_last
}
```




```
## [1] "Estimated probability of walking out with extra cash"
```

```
## [1] 0.49
```

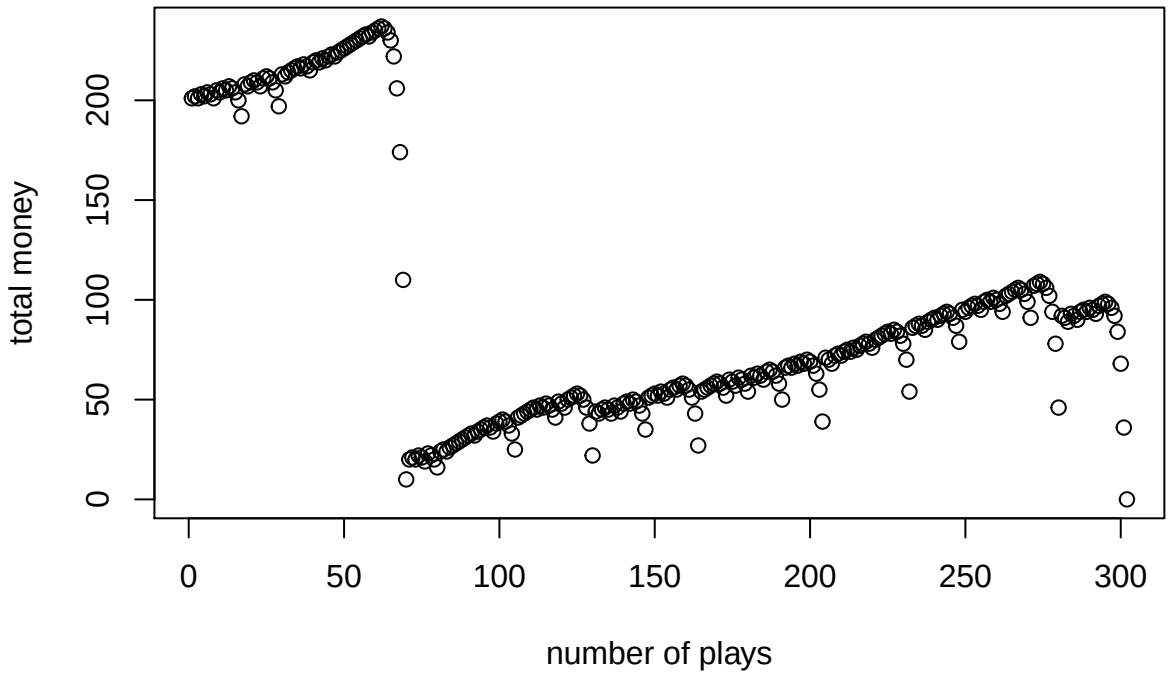
```
## [1] "Estimated earnings"
```

```
## [1] -53
```

The histogram and the R console show that the probabilities of walking out with 300 and 0 are both nearly 0.5. This means the average amount players can leave with is about 150, about 50 less than what they came in with.

The earning trajectory of a random player could look like the graphic below.

```
set.seed(238)
plot(one_series(B = 200, W = 300, L = 1000, M = 100), xlab="number of plays", ylab="total money")
```



This individual obviously lost all of their money soon after 300 plays. It is pretty easy to see why this strategy might not be as effective as some people would think. It is very easy for us to think linearly, i.e. we can easily visualize the winning areas of the Martingale strategy. But we often have trouble grasping how much exponential growth actually grows. This graph illustrates how easily and quickly a player looses money in a string of bad bets compared to the slow gains of the good bets.

Original Simulation — Number of Plays

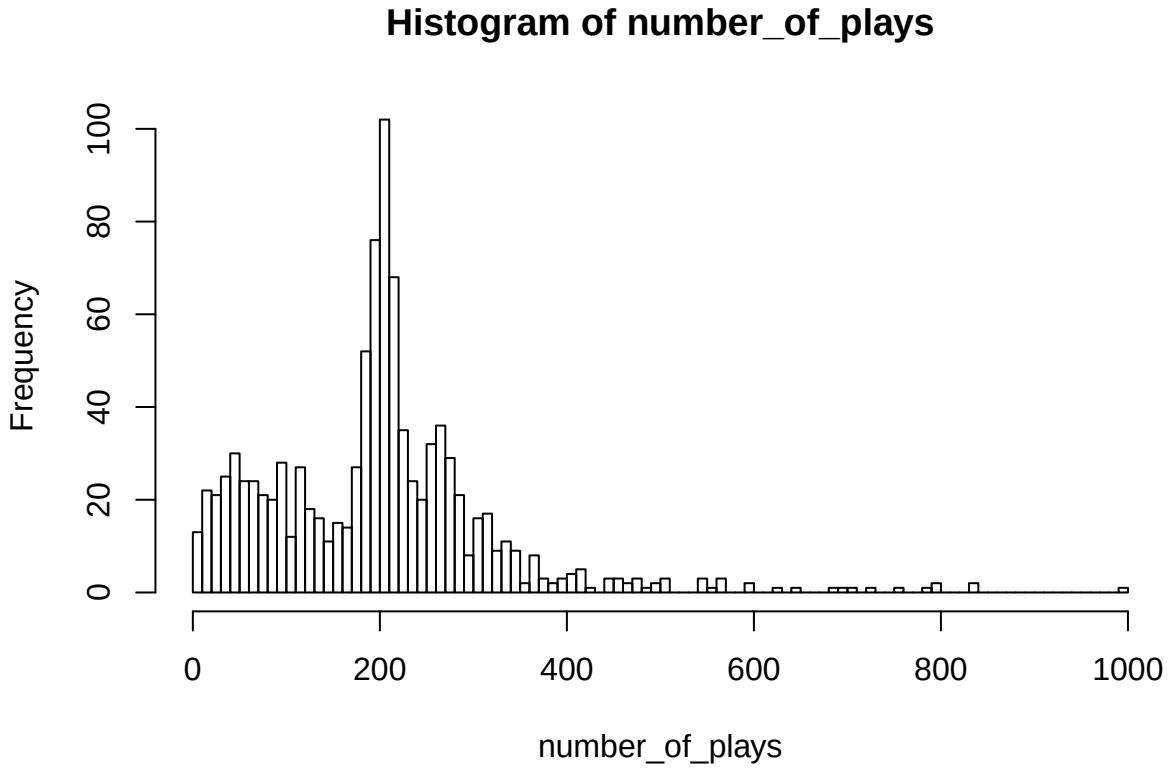
Below we evaluate the the how long a player plays the game by looking at turns played.

```
# Simulation
set.seed(25)
number_of_plays <- rep(NA, 1000)
```

```
for(j in seq_along(number_of_plays)){
  number_of_plays[j] <- one_series(B = 200, W = 300, L = 1000, M = 100) %>% length
}
```

The simulation below shows the number of turns playing in the 1000 simulations. It takes at least 100 turns to win (best scenario of 100 straight wins), so anything in the distribution within 100 is a loss. Around 0.30 of all games.

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




```
"mean number of plays"
```

```
## [1] "mean number of plays"
```

```
mean(number_of_plays)
```

```
## [1] 198.623
```

```
"percentage of people with no chance"
```

```
## [1] "percentage of people with no chance"
```

```
sum(number_of_plays<=100)/sum(number_of_plays>100)
```

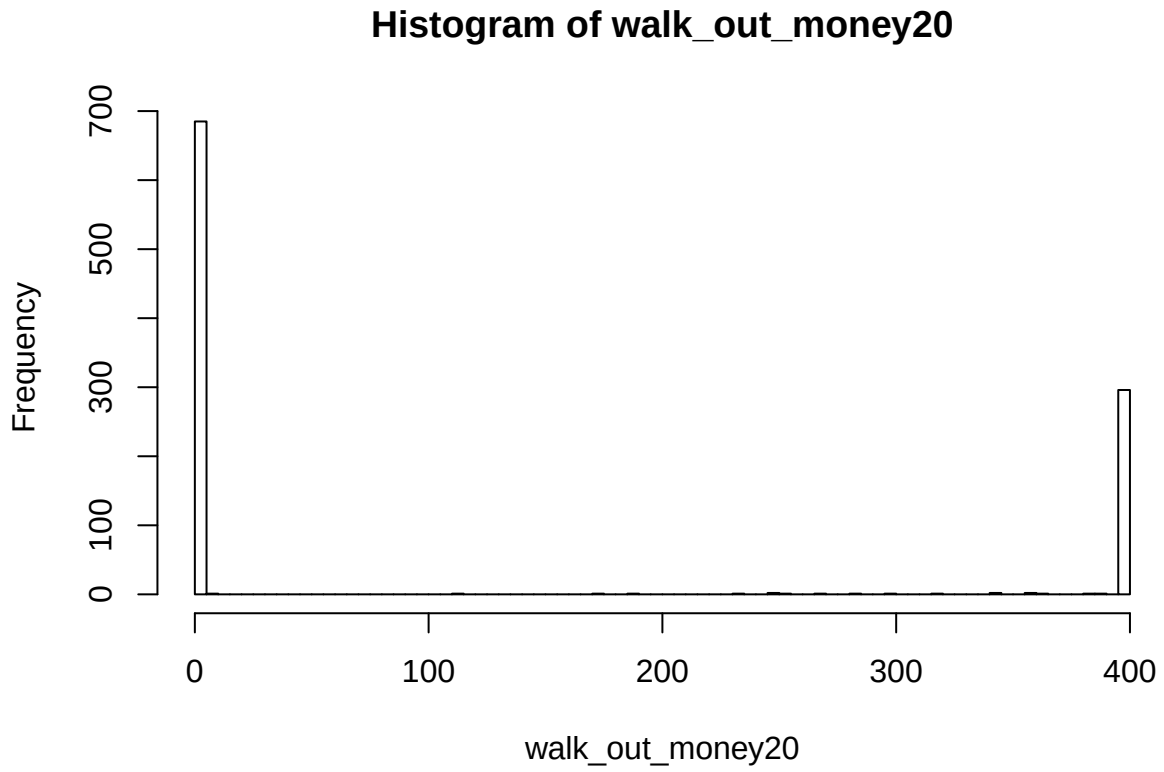
```
## [1] 0.2953368
```

All of the analysis has been centered on the original parameters. Next lets look at players trying to double their money from 200 to 400 not 300. I will explain after the results why I decided to evaluate the "double your money" strategy.

Simulation With Modified Parameters — "Double Your Money"

Below is loop of 1000 simulations where the player enters the casino with 200, will leave if they reach 400, can play a maximum 1000 number of times (reasonable time constraint), and plays at a table with a 100 wager limit (standard at casinos).

```
# Simulation
set.seed(238)
walk_out_money20 <- rep(NA, 1000)
for(j in seq_along(walk_out_money20)){
  walk_out_money20[j] <- one_series(B = 200, W = 400, L = 1000, M = 100) %>% get_last
}
```




```
## [1] "Estimated probability of walking out with extra cash"
```

```
## [1] 0.311
```

```
## [1] "Estimated earnings"
```

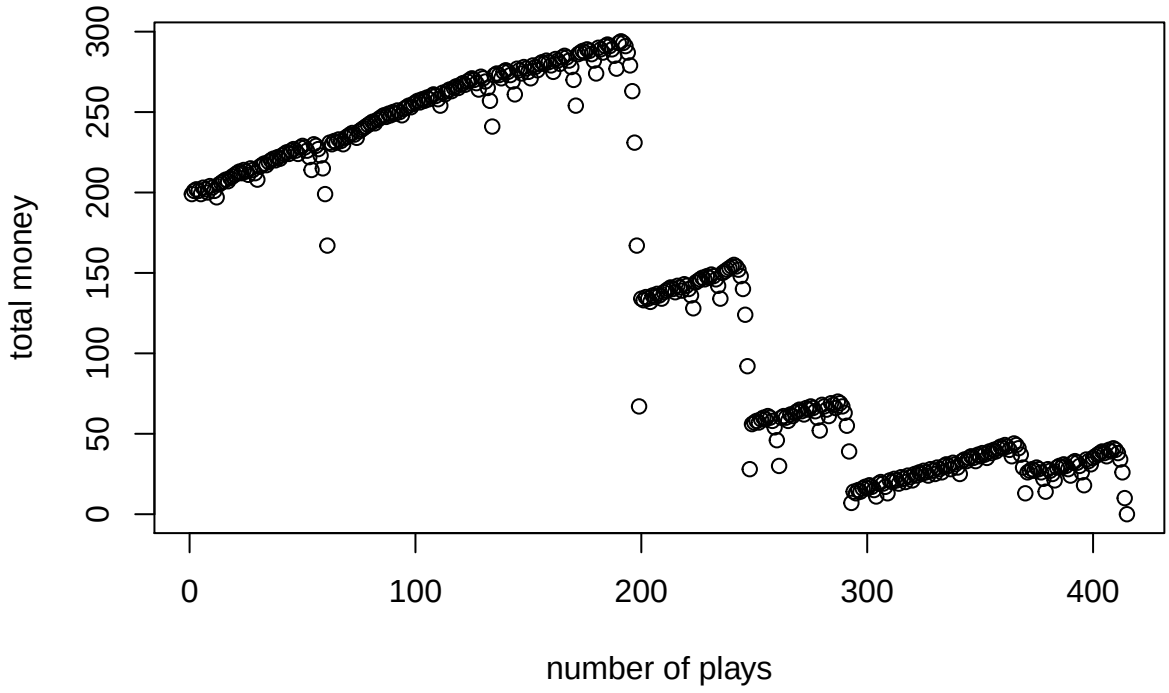
```
## [1] -76.432
```

The histogram and the R console show that the probabilities of walking out with 400 is around only 0.3, while the probability of leaving empty handed is around 0.7. The average player of the game with these parameters will leave around $75 poorer.

T

The earning trajectory of a random player could look like the graphic below.

```
set.seed(240)
plot(one_series(B = 200, W = 400, L = 1000, M = 100), xlab="number of plays", ylab="total money")
```



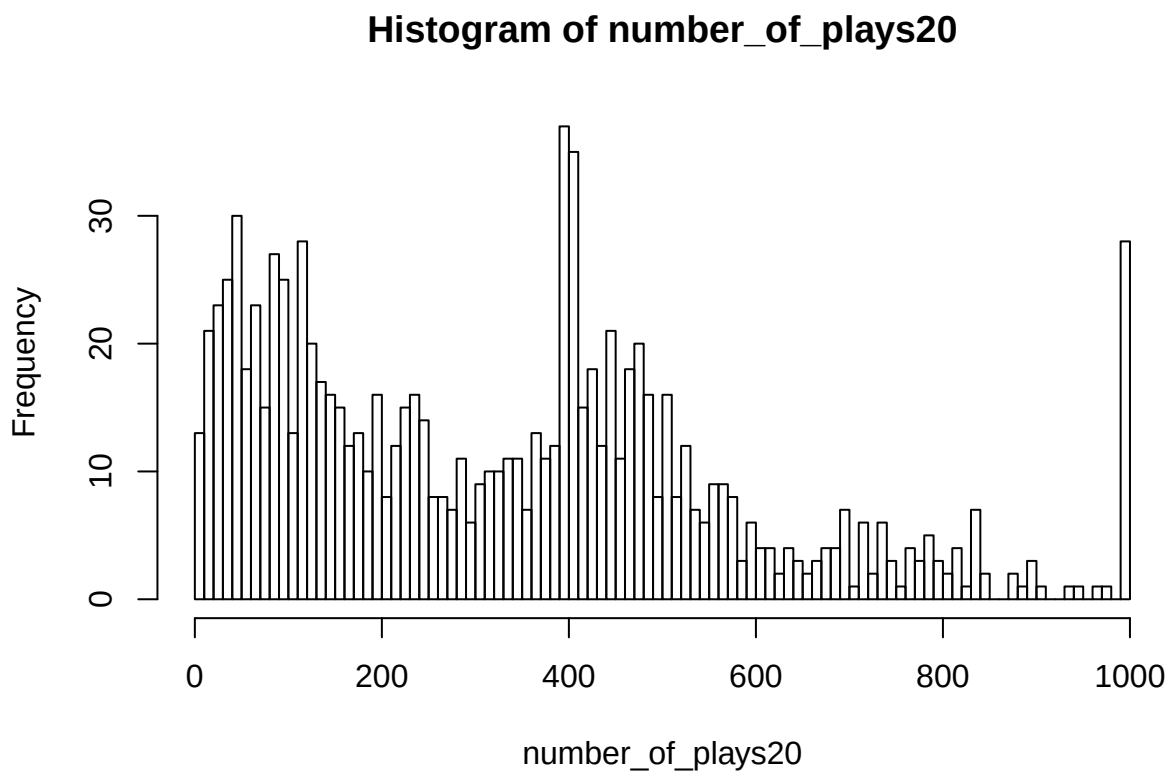
Modified Parameter Simulation — Number of Plays

Below we evaluate the the how long a player plays the game by looking at turns played.

```
# Simulation
set.seed(25)
number_of_plays20 <- rep(NA, 1000)
for(j in seq_along(number_of_plays20)){
  number_of_plays20[j] <- one_series(B = 200, W = 400, L = 1000, M = 100) %>% length
}
```

The simulation below shows the number of turns playing in the 1000 simulations. It takes at least 200 turns to win (best scenario of 200 straight wins), so anything in the distribution within 200 is a loss. Around 0.30 of all games.

```
# Walk out money distribution
hist(number_of_plays20, breaks = 100)
```




```
"mean number of plays"
```

```
## [1] "mean number of plays"
```

```
mean(number_of_plays20)
```

```
## [1] 332.017
```

```
"percentage of people with no chance"
```

```
## [1] "percentage of people with no chance"
```

```
sum(number_of_plays20<=100)/sum(number_of_plays20>100)
```

```
## [1] 0.2820513
```

The reason I chose the "double my money" for the win strategy because it allows for easy comparison of the martingale strategy to the strategy of simply putting all your money on red/black on the first hand. If you put all your money on red in the first hand you will double your money with the probability of 0.474. This is a lot better than using the Martingale strategy with the above parameters.

"Infinite" Martingale

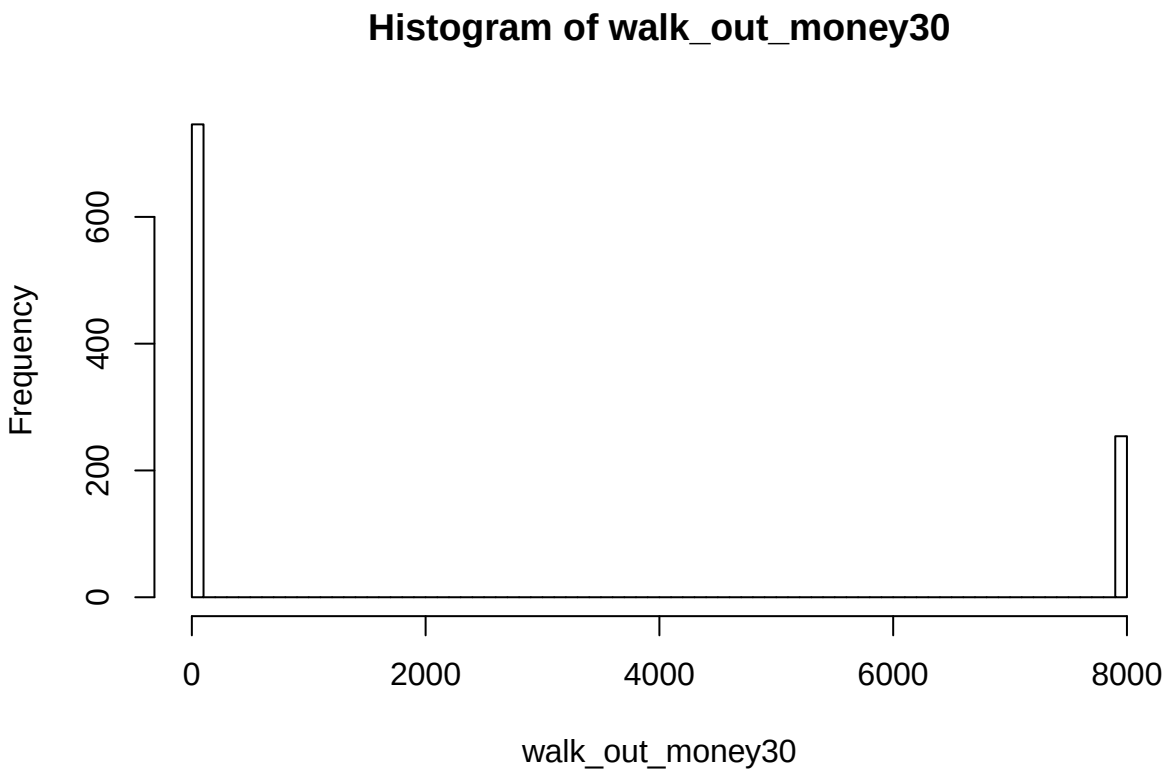
I do believe that the Martingale strategy would work if you had infinite money and there was no wager limit and time parameter (why you would be playing roulette if you have infinite money is beyond me). But lets see if the strategy works better as the player "approaches" having infinite money. I am interested in if there is a point where the strategy works better i.e. how close to infinity do you need to be. I would guess very close as exponents grow very quickly and you would need to play for more and more turns allowing for many consecutive losses.

B = 4000, W = 8000, L = 70000, M = 1000000

Below is loop of 1000 simulations where the player (now "approaching" infinity money, my computer can't handle more) enters the casino with 4000, will leave if they reach 8000, can play a maximum 70000 number of times, and plays at a table with a 1000000 wager limit (very relaxed wager limit).

```
# Simulation
set.seed(238)
walk_out_money30 <- rep(NA, 1000)
for(j in seq_along(walk_out_money30)){
  walk_out_money30[j] <- one_series(B = 4000, W = 8000, L = 70000, M = 1000000) %>% get_last
}
```

```
# Walk out money distribution
hist(walk_out_money30, breaks = 100)
```




```
# Estimated probability of walking out with extra cash
"Estimated probability of walking out with extra cash"
```

```
## [1] "Estimated probability of walking out with extra cash"
```

```
mean(walk_out_money30 > 4000)
```

```
## [1] 0.254
```

```
# Estimated earnings
"Estimated earnings"
```

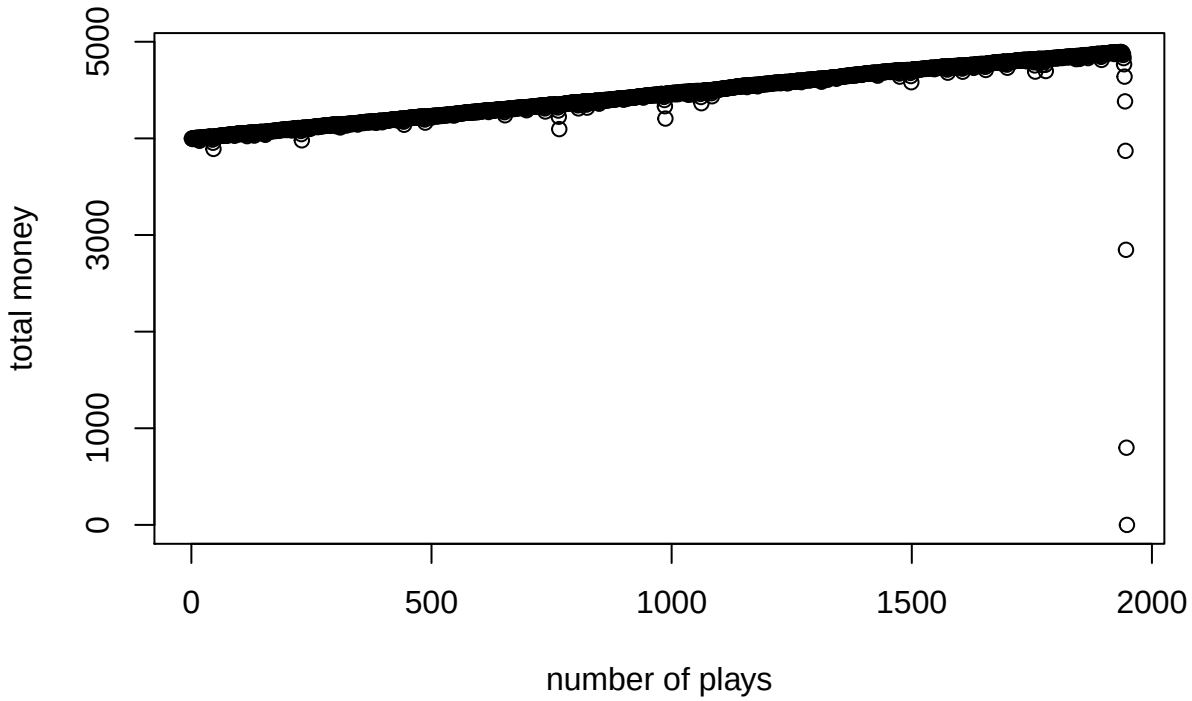
```
## [1] "Estimated earnings"
```

```
mean(walk_out_money30 - 4000)
```

```
## [1] -1968
```

This is worse for players than the previous two Martingale strategies. The more turns/opportunities means more chances for long strings of losses. Again it would be very interesting to see where between this and infinity does this strategy become plausible.

```r
set.seed(243)
plot(one_series(B = 4000, W = 8000, L = 70000, M = 1000000), xlab="number of plays", ylab="total money")
```



In conclusion, the best roulette betting strategy (aside from not betting at all) is betting all your money on first hand on either red or black. The Martingale strategy is not a viable long term profitable strategy ### Model Limitations 1. We don't have infinite computing power/time to run the simulations with more lenient parameters 2. I also did not take into account quirks about specific roulette tables. I am sure that although they try to keep them fair, some aren't.