# Roulette Strategy B

October 28, 2023

## 1 Simulating Strategy B: Placing a Bet on a single Roulette Number

### 1.1 Strategy B1

In this strategy, we will be betting a constant amount of money on a single number in a game of roulette.

There are a couple of assumptions I'd like to clarify before designing and simulating our playing strategy. Firstly, we will be playing European roulette, where the wheel has only one zero, and therefore the probability of a win at any given spin is 1/37.

We will start with an initial bankroll of $1,000 and will be betting $10 on a single number. Our game will cease either after we reach 500 spins or if we run out of money.

We will simulate this scenario over 1,000 games and analyse the cumulative profit and loss balance.

[ ]:
```
# Create a function that simulates the strategy of betting $10 on a single␣
↪number for 500 spins,
# and returns the amount of money left in the bank after the 500 spins.

simulate_roulette <- function(initial_bank, max_spins = 500, actual_bet = 10) {
  bank <- initial_bank
  spins <- 0
  bet <- actual_bet


  while (bank > 0 && spins < max_spins) {
    spins <- spins + 1
    outcome <- sample(c("win", "lose"), 1, prob = c(1 / 37, 36 / 37))

    if (outcome == "win") {
      bank <- bank + 35 * bet
      bet <- 10
    } else {
      bank <- bank - bet
      bet <- 10

      if (bet > bank) {
```

```
        bet <- bank
      }
    }



    if (bank <= 0) {
      return(list("bank" = 0, "spins" = spins))
    }
  }

  return(list("bank" = bank, "spins" = spins))
}
```

[ ]:
```
# simulate the strategy with $1000 as initial bank, 500 max spins and $10 bet

result <- simulate_roulette(1000, 500, 10)
```

[ ]:
```
# print the results from the simulation

print(result)
```

```
$bank
[1] 0

$spins
[1] 244
```

[ ]:
```
# Run the simulation 1000 times and store the results in results_total

results_total <- replicate(1000, simulate_roulette(1000, 500, 10))
```

[ ]:
```
# Convert and transpose the results_total into a data frame called outcomes

outcomes <- as.data.frame(t(results_total))
```

[ ]:
```
# Print the first 6 rows of outcomes

print(head(outcomes))
```

```
  bank spins
1    0   316
2    0   100
3  320   500
4 2480   500
5 3200   500
6 2840   500
```

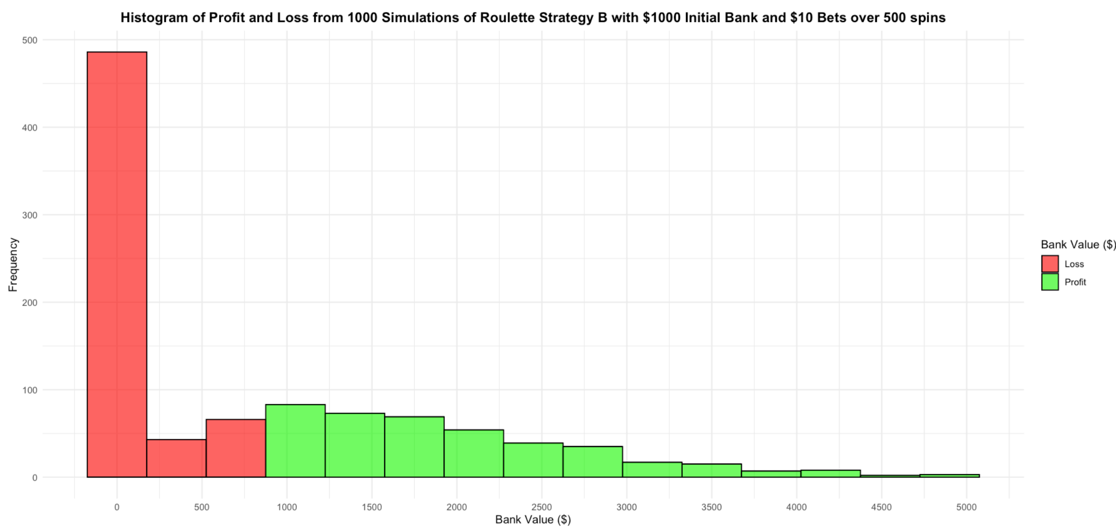
[ ]:
```r
# Convert the list in the 'bank' column to a numeric vector
outcomes$bank <- unlist(outcomes$bank)
```

[ ]:
```r
# Load the ggplot2 library
library(ggplot2)
```

[ ]:
```r
# Set the dimensions for the plot
options(repr.plot.width = 15, repr.plot.height = 7)
```

[ ]:
```r
# Produce histogram of bank values

ggplot(outcomes, aes(x = bank, fill = ifelse(bank > 1000, "Profit", "Loss"))) +
  geom_histogram(binwidth = 350, color = "black", alpha = 0.7) +
  scale_fill_manual(name = "Bank Value ($)",
                    values = c("Profit" = "green", "Loss" = "red")) +
  labs(title = "Histogram of Profit and Loss from 1000 Simulations of Roulette Strategy B with $1000 Initial Bank and $10 Bets over 500 spins",
       x = "Bank Value ($)",
       y = "Frequency") +
  scale_x_continuous(breaks = seq(0, max(outcomes$bank, na.rm = TRUE), by=500)) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))
```




[ ]:
```r
summary(outcomes$bank)
print(var(outcomes$bank))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
      0       0     320     873    1400    5000 
```

```
[1] 1221886
```

[ ]:
```
# Calculate the probability of losing all of your money

prob_going_bust <- sum(outcomes$bank == 0) / length(outcomes$bank)

# Calculate the probability of making a profit

prob_making_profit <- sum(outcomes$bank > 1000) / length(outcomes$bank)

# Calculate the probability of making a loss

prob_making_loss <- 1 - prob_making_profit

print(prob_going_bust)
print(prob_making_profit)
print(prob_making_loss)
```

```
[1] 0.486
[1] 0.405
[1] 0.595
[1] 0.405
[1] 0.595
```

[ ]:
```
# Calculate the average profit and average loss

average_profit <- mean(outcomes$bank[outcomes$bank > 1000]) - 1000
average_loss <- 1000 - mean(outcomes$bank[outcomes$bank < 1000])
max_profit <- max(outcomes$bank[outcomes$bank > 1000]) - 1000
max_loss <- 1000 - min(outcomes$bank[outcomes$bank < 1000])

print(average_profit)
print(average_loss)
print(max_profit)
print(max_loss)
```

```
[1] 1010.667
[1] 901.4454
[1] 4000
[1] 1000
[1] 901.4454
[1] 4000
[1] 1000
```

### 1.2 summary of strategy B1

From the findings above, we can deduce that a gambler can expect to lose $127 on average in any given game. Since we simulated 1,000 games, the total loss amounts to $127,000.

The probability that a gambler will lose all of their money in any given game is 48.6%.

The probability that a gambler will be in profit at the end of a game is 40.5%, with the average win being $1,010 and the maximum winnings reaching $4,000.

On the other hand, the probability a gambler will be at a loss after a game is 59.5%, with the average loss being $901 and the maximum loss amounting to $1,000.

### 1.3 Strategy B2

In this scenario we will start with an initial bankroll of $3000 and will be betting $5 on a single number. Our game will cease either after we reach 300 spins or if we run out of money.

We will simulate this scenario over 1,000 games and analyse the cumulative profit and loss balance.

[ ]:
```
# simulate the strategy with 3000 initial bank, 300 max spins and bet 5

result2 <- simulate_roulette(3000, 300, 5)
```

[ ]:
```
# print the results from the simulation

print(result2)
```

```
$bank
[1] 2885

$spins
[1] 300
```

[ ]:
```
# Run the simulation 1000 times and store the results in results_total2

results_total2 <- replicate(1000, simulate_roulette(3000, 300, 5))
```

[ ]:
```
# Convert the results_total2 into a data frame called outcomes2

outcomes2 <- as.data.frame(t(results_total2))
```

[ ]:
```
# Print the first 6 rows of outcomes2

print(head(outcomes2))
```

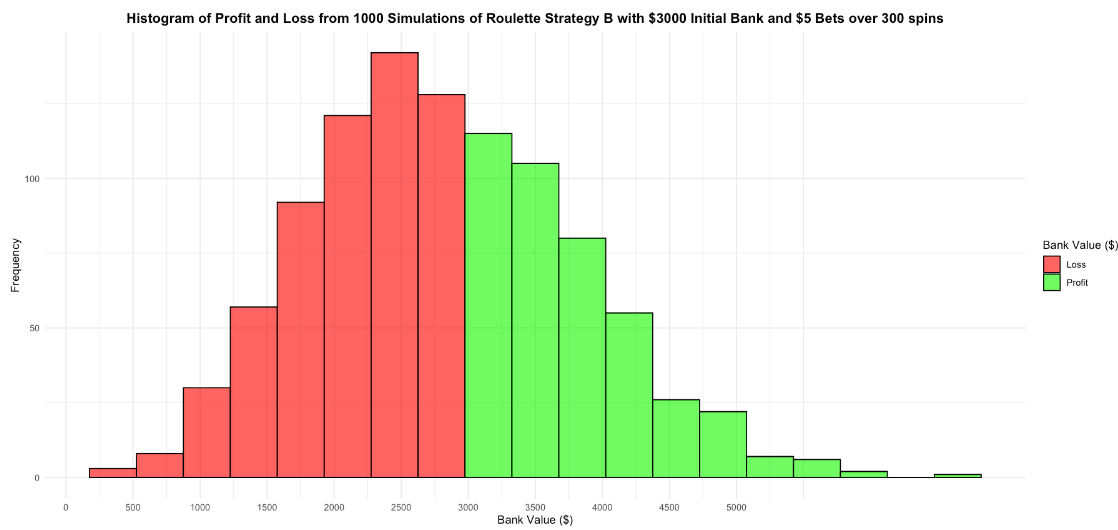
```
  bank spins
1  725   300
2 2165   300
3 1805   300
4 3965   300
5 5405   300
6 2525   300
```

[ ]:
```
# Convert the list in the 'bank' column to a numeric vector
outcomes2$bank <- unlist(outcomes2$bank)
```

```
# Convert the list in the 'spins' column to a numeric vector
outcomes2$spins <- unlist(outcomes2$spins)
```

[ ]:
```
# Produce histogram of bank values

ggplot(outcomes2, aes(x = bank, fill = ifelse(bank > 3000, "Profit", "Loss"))) +
  geom_histogram(binwidth = 350, color = "black", alpha = 0.7) +
  scale_fill_manual(name = "Bank Value ($)",
                    values = c("Profit" = "green", "Loss" = "red")) +
  labs(title = "Histogram of Profit and Loss from 1000 Simulations of Roulette Strategy B with $3000 Initial Bank and $5 Bets over 300 spins",
       x = "Bank Value ($)",
       y = "Frequency") +
  scale_x_continuous(breaks = seq(0, max(outcomes$bank, na.rm = TRUE), by=500)) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))
```




[ ]:
```
summary(outcomes2$bank)
print(var(outcomes2$bank))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
    365    2165    2885    2904    3605    6485 

[1] 1048229
```

[ ]:
```
# Calculate the probability of losing all of your money

prob_going_bust2 <- sum(outcomes2$bank == 0) / length(outcomes2$bank)
```

```
# Calculate the probability of making a profit

prob_making_profit2 <- sum(outcomes2$bank > 3000) / length(outcomes2$bank)

# Calculate the probability of making a loss

prob_making_loss2 <- 1 - prob_making_profit2

print(prob_going_bust2)
print(prob_making_profit2)
print(prob_making_loss2)
```

```
[1] 0
[1] 0.419
[1] 0.581
[1] 0.419
[1] 0.581
```

[ ]:
```
# Calculate the average profit and average loss

average_profit2 <- mean(outcomes2$bank[outcomes2$bank > 3000]) - 3000
average_loss2 <- 3000 - mean(outcomes2$bank[outcomes2$bank < 3000])
max_profit2 <- max(outcomes2$bank[outcomes2$bank > 3000]) - 3000
max_loss2 <- 3000 - min(outcomes2$bank[outcomes2$bank < 3000])

print(average_profit2)
print(average_loss2)
print(max_profit2)
print(max_loss2)
```

```
[1] 884.2363
[1] 803.7091
[1] 3485
[1] 2635
[1] 803.7091
[1] 3485
[1] 2635
```

### 1.4 summary of strategy B2

By increasing our initial bank size from $1,000 to $3,000, reducing the number of spins from 500 to 300, and decreasing the bet size from $10 to $5, we discovered that a gambler can expect to lose $96 on average in any given game. After simulating 1,000 games, the cumulative loss was $96,000.

The likelihood of a gambler losing all their money in a single game dropped dramatically from 48.6% to 0%.

The probability that a gambler will be in profit at the end of a game is 41.9%, with the average win being $884 and the maximum winnings reaching $3485.

On the other hand, the probability a gambler will be at a loss after a game is 58.1%, with the average loss being $803 and the maximum loss amounting to $2635.

By augmenting our initial bankroll, decreasing the bet size, and limiting the number of spins per game, we succeeded in reducing our losses considerably. However, we still did not identify a profitable strategy.

### 1.5 deeper look into strategy B2

[ ]:
```
simulate_roulette3 <- function(simulation_id, initial_bank, max_spins = 300,␣
↪actual_bet = 5) {
  bank <- initial_bank
  spins <- 0
  bet <- actual_bet
  bank_history <- numeric(max_spins)  # Initialize a vector to store bank␣
↪balance after each spin

  while (spins < max_spins) {
    spins <- spins + 1

    # Check if bank is still positive; else, bet remains zero
    if (bank > 0) {
      outcome <- sample(c("win", "lose"), 1, prob = c(1 / 37, 36 / 37))

      if (outcome == "win") {
        bank <- bank + 35 * bet
      } else {
        bank <- bank - bet
      }

      # Adjust bet for next round
      bet <- actual_bet

      if (bet > bank) {
        bet <- bank
      }
    }

    bank_history[spins] <- bank  # Record bank balance

    # Check for bankruptcy, if so, record 0 for remaining spins
    if (bank <= 0) {
      bank_history[spins:max_spins] <- 0
      break
    }
  }

  return(data.frame(
```

```
    simulation_id = rep(simulation_id, length(bank_history)),
    spin = 1:length(bank_history),
    bank_balance = bank_history
  ))
}
```

[ ]:
```
result <- simulate_roulette3(1, 3000, 300, 5)
```

[ ]:
```
print(head(result))
```

```
  simulation_id spin bank_balance
1             1    1         2995
2             1    2         2990
3             1    3         2985
4             1    4         2980
5             1    5         2975
6             1    6         2970
```

[ ]:
```
set.seed(123) # For reproducibility
all_simulations <- do.call(rbind, lapply(1:10000, function(x)
 ↪simulate_roulette3(x, 3000, max_spins = 300, actual_bet = 5)))
```

[ ]:
```
print(head(all_simulations))
```

```
  simulation_id spin bank_balance
1             1    1         2995
2             1    2         2990
3             1    3         2985
4             1    4         2980
5             1    5         2975
6             1    6         2970
```

[ ]:
```
library(dplyr)

balance_statistics <- all_simulations %>%
  group_by(spin) %>%
  summarize(
    average_balance = mean(bank_balance, na.rm = TRUE),
    max_balance = max(bank_balance, na.rm = TRUE),
    min_balance = min(bank_balance, na.rm = TRUE),
    standard_dev_balance = sd(bank_balance, na.rm = TRUE)
  )
```

[ ]:
```
print(head(balance_statistics))
```

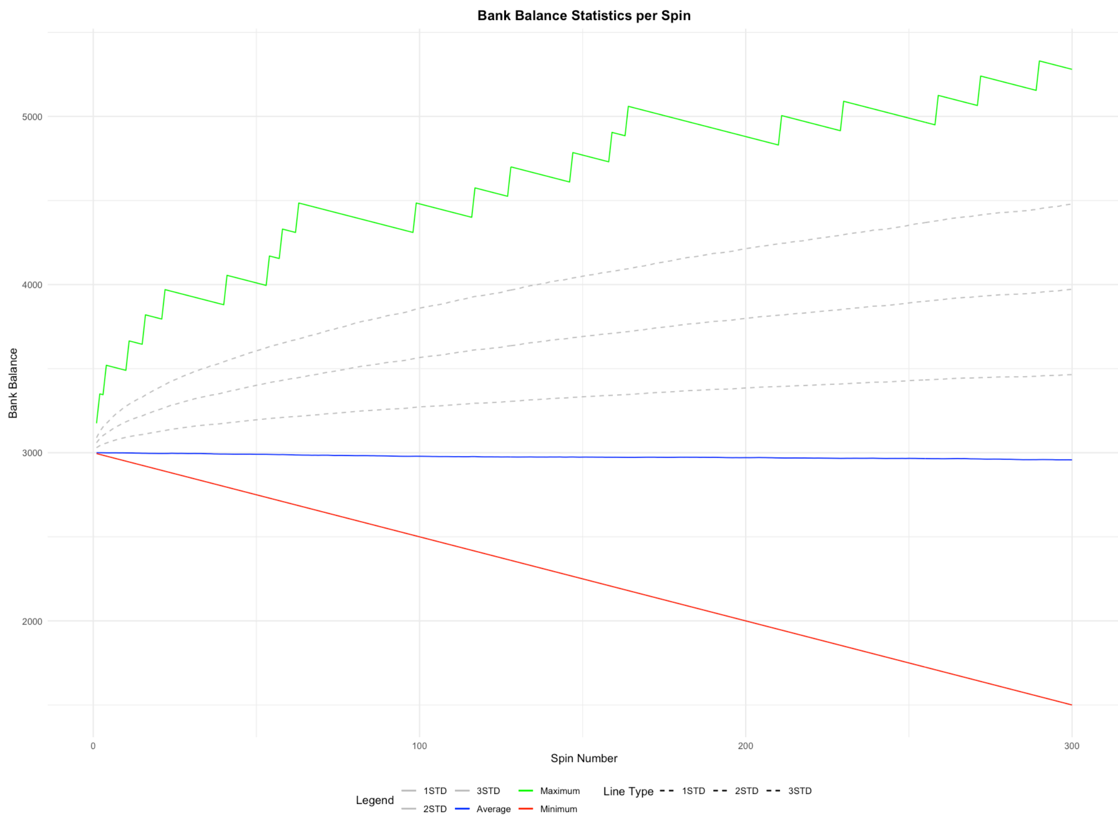
```
# A tibble: 6 x 5
   spin average_balance max_balance min_balance standard_dev_balance
  <int>           <dbl>
<dbl>       <dbl>
```

```
<dbl>
1      1           3000.         3175
2995                 29.7
2      2           3000.         3350
2990                 42.1
3      3           3000.         3345
2985                 51.3
4      4           2999.         3520
2980                 58.2
5      5           2999.         3515
2975                 64.8
6      6           2999.         3510
2970                 71.4
```

[ ]:
```
# Set the dimensions for the plot
options(repr.plot.width = 15, repr.plot.height = 11)
```

[ ]:
```
ggplot(balance_statistics, aes(x = spin)) +
  geom_line(aes(y = average_balance, color = "Average")) +
  geom_line(aes(y = max_balance, color = "Maximum")) +
  geom_line(aes(y = min_balance, color = "Minimum")) +
  geom_line(aes(y = average_balance + standard_dev_balance, color = "1STD",␣
↪linetype = "1STD")) +
  geom_line(aes(y = average_balance + 2 * standard_dev_balance, color = "2STD",␣
↪linetype = "2STD")) +
  geom_line(aes(y = average_balance + 3 * standard_dev_balance, color = "3STD",␣
↪linetype = "3STD")) +
  scale_color_manual(values = c(
    "Average" = "blue",
    "Maximum" = "green",
    "Minimum" = "red",
    "1STD" = "grey",
    "2STD" = "grey",
    "3STD" = "grey"
  )) +
  scale_linetype_manual(values = c(
    "1STD" = "dashed",
    "2STD" = "dashed",
    "3STD" = "dashed"
  )) +
  theme_minimal() +
  labs(
    title = "Bank Balance Statistics per Spin",
    x = "Spin Number",
    y = "Bank Balance",
    color = "Legend",
    linetype = "Line Type"
```

```
) +
theme(
  legend.position = "bottom",
  plot.title = element_text(hjust = 0.5, face = "bold")  # Center and bold the title
)
```




## 1.6 summary

Since it is known that roulette is a negative expectation game, which means the longer one plays, the more likely one is to lose, we can try to find the optimal time to stop playing if we decide to play after all. From the chart above, we can see the minimum, the maximum, and the average value of our bank balance at each spin. The dashed lines represent the standard deviation from the average to the upside.

The recommendation is, if we decide to play and our balance is above the second standard deviation line, we should take our winnings and stop playing, as the probability that we will make much more is small relative to what we can lose.