Word Stand or Hit: Simulation of Blackjack by Programming

Yiqi Huang
School of Natural Sciences, University of Manchester, Manchester, M13 9PL, U.K.

Keywords: Blackjack, Game Theory, Optimal Strategy.

Abstract: Blackjack is a global gambling game with great influence and popularity where players aim to have a hand total closer to 21 than the dealer's hand without exceeding it. It is a game played against the dealer or the casino. In Blackjack, the player's subjective decision almost determines the outcome of the game, while the dealer follows a fixed rule resulting in less subjective impact on the game. As a result, players strive to find the optimal strategy to gain profit. This paper designs a computer program to discuss whether choosing to continue or stop drawing cards is statistically wiser for different initial cards. The paper also emphasises how probability principles can be applied to Blackjack, providing guidance for practical decision-making in casinos. The optimal player actions for each combination of initial dealer and player hand types calculated by the program are summarised in a table to provide players with effective references.

1. INTRODUCTION

Blackjack, also known as 21, is one of the most popular gambling games nowadays. The player is aiming to have a hand total that is closer than the dealer's hand but without going over 21. Each player is dealt two cards initially and can choose to take more cards or stop at the current total, which is called "hit" or "stand" respectively. Face cards such as Kings, Queens, and Jacks are counted as 10 points, Aces can be counted as either 1 or 11 points, while all other cards are worth their face value. The name "Blackjack" of the game comes from the best possible hand, which is an Ace and a 10-point card. If the player's hand exceeds 21 points, they automatically lose regardless of the dealer's hand, which is called "bust". On the contrary, if the dealer busts cards, then all players who did not bust their cards are considered winners.

Blackjack is ancient and simple. The game could be traced to the 15th century when Gutenberg's printing press was invented (Snyder, 2013). It requires very few props: only chips and playing cards, and the rules are simple and uniform. As a result, the game has a unique charm that attracts people all over the world and can be found in almost any casino. According to statistics, as the second most profitable gambling game in America, 31% of table game action is made up by Blackjack (Lindner, 2023). Another important reason why this casino game is popular among players is that using reasonable strategies can effectively increase the winning rate. In general, the games operated in the casino are games with negative player expectations, because the casino needs to make a profit. However, the disadvantages of blackjack are smaller than other popular casino games, such as American Roulette (Baldwin et al., 1956). With the development of game theory and the standardization of the game, more and more Blackjack optimal strategies have been proposed. The concept of the Game Theory Optimal (GTO) strategy, which is central to game theory, was first developed by John von Neumann (Neumann and Morgenstern, 2007). Culbertson proposed a strategy in 1952 that brought the player's expectations to -0.036 (Culbertson, 1952). This makes blackjack almost a fair game and even means there is a chance of making money from the casino. Manson proposed a strategy for four decks used in 1975 (Manson, Barr and Goodnight, 1975). In addition to the analysis of decisions for action, there is also a lot of analysis of decisions for betting based on card counting, which was first proposed by Thorp (Thorp, 2016). Later in 1963, Dubner proposed the Hi-Lo system at a conference in Las Vegas, which is a simplification of the complex strategies that had been previously developed by Thorp, which made card counting more accessible to the general public (Snyder, 2013). The Hi-Lo card counting method is widely recognized and its practical application has

Huang, Y.
Word Stand or Hit: Simulation of Blackjack by Programming.
DOI: 10.5220/0013055800004601

In *Proceedings of the 1st International Conference on Innovations in Applied Mathematics, Physics and Astronomy (IAMPA 2024)*, pages 253-257
ISBN: 978-989-758-722-1

been extensively studied (Wong, 1994). Generally, Game theory gives ideal frameworks that can be used to provide guidance on the allocation of limited resources and can be applicable such as solving transportation problems (Hughes and Chen, 2021). It can also be used to develop pricing models aimed at balancing loads in the grid (Ghosh et al, 2004). Therefore, game theory and optimal strategy research are considered broadly applicable and meaningful.

This paper will introduce a method of using an open-source programming language and software environment: R language to solve the Blackjack strategy problem. That is, R language will be used to give expectations for selecting hit or stand under different player versus dealer cards situations to give players a reference for decision-making.

2. METHODOLOGY

2.1. Game Description

Specifically, the process of the Blackjack is as follows: the dealer first deals two cards to each player, and then deals one card to himself. These cards are all shown publicly, which means that players also have the information of the two cards of other players. After that, players make decisions in order from right to left. The player can choose to "stand", for which the sum of the two cards in hand will become the final points; or choose to "hit", for which the dealer will deal one more card to the player to increase the sum, and players are allowed to hit multiple times. Mentionable, players aim to not exceed 21 points, or they will be eliminated immediately. The dealer will take action after all players have ended their action round. Unlike the players, the dealer's actions do not depend on subjective will, but on a fixed rule: the dealer must hit if the dealer's card sum is less than or equal to 16 points and must stand if greater than or equal to 17 points. At this moment, all surviving players will compare their points with the dealer and whose points are not as high as the dealer's will lose all bets. With a bigger card sum, the player will win the game, and the payout varies according to the odds specified by the table and casino. If the dealer's hand exceeds 21, then all players who survived, which means those who did not lose at the player action, win. If the points are equal, it is a tie, with no profit or loss for the players.

There will be different rules according to different casino regulations. For example, if the player believes that his two initial cards are very favourable compared to the dealer's one initial card, then the player can choose to double in his round, that is, double his original bet. In this case, the dealer gives the player one card only, which means there are no multiple hits. Therefore, players need to consider whether their winning odds are worth doubling down on, as either gains or losses are doubled. Some casinos allow betting on flushes, straights, or triple 7s. Although the probability might be extremely low, the payment could be very impressive, with some odds can reach 1: 500.

2.2. Simulate Process and Convert to Code

In summary, it is crucial for players to correctly analyse both sides' initial cards and take action. This article will use R to produce an answer of whether the player should choose to hit or stand in terms of probability by various player points and dealer points in the situation of one deck of cards and one player playing. First, an environment with one deck of cards is needed to be created. The code shown in Table 1 will create a vector: numbers from 2 to 10 represent the number cards, three 10s represent the face cards, and an 11 represents the ace card. This vector is then repeated 4 times to form an environment of 52 standard playing cards. The user could simply change the variable in the repeat code to a multiple of 4 in simulation of multiple decks of cards. In theory, using multiple decks will slightly increase the dealer's winning rate, which is tiny enough to be ignorable.

Table 1. R code Function call for one standard deck.

Algorithm 1	
Step 1	Import the dplyr library
Step 2	Create an array 'deck' - Initialize 'deck' with card values: 2,3,4,5,6,7,8,9,10,10,10,10,11.
Step 3	Create an array 'full_deck' by replicating 'deck' four times.

a check action is needed to be proposed for the ace card because of that it can be either 1 or 11. Logically, the ace card could be considered that it only represents 1 point after busting. This allows the R language to subtract 10 points from the hand card

satisfying two conditions: busted cards and ace cards included, to achieve changing the aces from representing 11 points to representing 1 point. Finally, the code checks if the card is busted and returns the outcomes. The initialization of the player's hand is the same as the dealer's, but the player's actions need to be defined further.

Table 2. R code Function call for dealer and player simulation.

Algorithm 2	
Step 1	Define a function that takes the dealer's initial card value input by user.
Step 2	Start an infinite loop - If dealer's hand is less or equal to 16, draw a new card from the deck and update dealer's hand by adding the value of the card drawn. - If dealer's hand is greater or equal to 17 with an Ace, minus dealer's hand by 10. - If dealer's hand is greater or equal to 17, exit the loop.
Step 3	Return the final value of dealer's hand once the loop is exited (the dealer either stands or busts).
Step 4	Define a function that stores the dealer's initial card value input by user.
Step 5	Initialize a data frame named outcomes - Create an empty data frame called outcomes with three columns: - Strategy: to store different playing strategies the player might use, expected as character data. - Result: to store numerical results related to the strategies, expected as double-precision numbers. - Win: to store the win or loss outcome for each strategy, also expected as double-precision numbers.

In this next step, the algorithm shown in Table 3 simulates the outcome of choosing to stand or hit by the given initial card which is input by the users. If the dealer busts or the player's hand is higher, the player wins and is recorded as (1). If the player's hand is smaller, the player loses (-1). If the hands are equal, the result is a draw (0). Finally, to record to result a vector is created. The programming for a hit will be significantly more complicated. Firstly, an order is given to randomly draw a card from the deck set before, then add its points to the player's initial hand, with the same special design of the aces. If the player does not bust, repeat the previous recording method. If the player busts the card, it will be directly recorded as (-1). Similarly, the results are recorded in the result vector as well.

Table 3. R code Function call for hit or stand simulation.

Algorithm 3	
Step 1	Store the result of dealer's hand after simulation.
Step 2	For player choosing to stand, define an if-else function. - If dealer's hand exceeds 21, output (1). -Otherwise: - If player's hand is greater than dealer's hand, output (1). - If player's hand is less than dealer's hand, output (-1). - If player's hand is equal to dealer's hand, output (0).
Step 3	For player choosing to hit, draw a new card from the deck and update player's hand by adding the value of the card drawn.
Step 4	Define an if-else function to check Ace card. - If player's hand exceeds 21 with Ace card, minus 10 and output. -Otherwise, output directly.
Step 5	Define a function for comparison. - If player's hand exceeds 21, output (-1). - If player's hand is less than 21: - If dealer's hand exceeds 21, output (1). - Otherwise: - If player's hand is greater than dealer's hand, output (1). - If player's hand is less than dealer's hand, output (-1). - If player's hand is equal to dealer's hand, output (0).
Step 6	Return outcomes for both stand or hit.

It is not enough to so far since the aim is to design code that can output probability which could be achieved by simply simulating a large number of times and calculating expectations. In Table 4, the program is ordered to simulate 10000 times with the input player's card is 15 points and the input dealer's card is 7 points. These three parameters can be input freely by users to meet the needs of real-time simulation. The output gives two probabilities as a reference, which are the expectations of selecting hit and stand for action in the situation of the input. Players should choose the action with the larger output to ensure a higher winning rate.

Table 4. R code Function call for expectation.

Algorithm 4	
Step 1	Use the replicate function to run the hit and stand simulation function 10,000 times with initial player hand and dealer card input by user. Combine all individual simulation results into a single data frame
Step 2	Combine all individual simulation results into a single data frame.
Step 3	Group and summarize the data by strategy - Group results by the 'Strategy' column. - Calculate the mean of the 'Win' column for each strategy group.
Step 4	Print the summary data frame

3. RESULTS AND DISCUSSION

3.1. Example of player 15 versus dealer 9

Suppose in a game, the player gets a King and a 5, and the dealer gets a 9. This brings the player's points to 15 and the dealer's points to 9. The question interested in is whether a hit or a stand has a statistically higher chance of winning in this case (Table 5).

Table 5. An example of running the algorithms.

Input	**Out put**
Replicate (10000, simulate_player_actions (15, 9), simplify = FALSE)	Strategy AverageWin <chr> <dbl> 1 Hit Once -0.480 2 Stand -0.547

According to the results, the winning rate of hit is -0.480, and the winning rate of stand is -0.547. This demonstrates that this is a favourable situation for the dealer because both actions have negative expectations for the player, which means that in the long run, the player expects to lose. However, the player should choose to hit in this situation as it is less likely to lose by comparison. According to the rules, a player can hit multiple times. Therefore, if there is no bust after a hit, the player can use the updated points as input to calculate the expectation again to decide whether a second hit is needed.

3.2. Solution on other simulations

Similarly, it is reasonable to list every possible hand situation and the corresponding winning rates of hit and stand. This is summarised in Table 6, with the dealer's hand in the row and the player's hand in the column.

The coloured options are the actions that are more likely to win. It is worth noting that most of the optimal choices are still negative. Only in very lucky cases, one can get a large positive expectation close to 1 with a clear advantage. This proves that blackjack is a game that favours the dealer, even if it is not obvious. Situations where the player's hand is equal to or less than 10 are not recorded. While expectations can be calculated in these situations, they are meaningless. Assuming the player's hand is equal to or less than 10, then a hit will increase the points without any possibility of resulting in a bust. Therefore, when the player encounters these situations, they should choose to hit the card no matter what the expectation is.

The code in this article gives the optimal solution considering only hit and stand. In most casinos, double or split is allowed to improve the player's winning rate, which is not considered in the probability calculation.

Table 6. Expected value in various situations.

	Expected value table									
	2		**3**		**4**		**5**		**6**	
	Hit	**Stand**	**Hit**	**Stand**	**Hit**	**Stand**	**Hit**	**Stand**	**Hit**	**Stand**
12	-0.22	-0.26	-0.22	-0.22	-0.19	-0.18	-0.17	-0.11	-0.17	-0.15
13	-0.28	-0.24	-0.27	-0.21	-0.27	-0.17	-0.24	-0.13	-0.21	-0.15
14	-0.34	-0.24	-0.34	-0.22	-0.31	-0.17	-0.30	-0.12	-0.30	-0.16
15	-0.41	-0.25	-0.39	-0.19	-0.39	-0.19	-0.37	-0.11	-0.37	-0.16
16	-0.47	-0.25	-0.46	-0.20	-0.45	-0.15	-0.44	-0.12	-0.44	-0.16
17	-0.53	-0.11	-0.53	-0.08	-0.51	-0.05	-0.52	-0.01	-0.52	0.01
18	-0.61	0.14	-0.61	0.19	-0.73	0.44	-0.62	0.23	-0.62	0.27

19	-0.71	0.41	-0.73	0.43	-0.63	0.20	-0.73	0.45	-0.73	0.49
20	-0.85	0.65	-0.86	0.65	-0.86	0.68	-0.85	0.69	-0.86	0.70
21	-1.00	0.88	-1.00	0.89	-1.00	0.90	-1.00	0.90	-1.00	0.90
	7		**8**		**9**		**10**		**Ace**	
	Hit	**Stand**	**Hit**	**Stand**	**Hit**	**Stand**	**Hit**	**Stand**	**Hit**	**Stand**
12	-0.24	-0.47	-0.30	-0.51	-0.37	-0.54	-0.43	-0.58	-0.48	-0.57
13	-0.27	-0.47	-0.33	-0.51	-0.40	-0.53	-0.47	-0.59	-0.49	-0.57
14	-0.34	-0.48	-0.37	-0.52	-0.44	-0.55	-0.51	-0.59	-0.55	-0.58
15	-0.37	-0.50	-0.43	-0.51	-0.48	-0.55	-0.55	-0.57	-0.58	-0.58
16	-0.42	-0.48	-0.48	-0.52	-0.51	-0.56	-0.55	-0.57	-0.61	-0.57
17	-0.48	-0.11	-0.50	-0.38	-0.57	-0.41	-0.61	-0.46	-0.64	-0.47
18	-0.59	0.40	-0.59	0.12	-0.62	-0.18	-0.67	-0.25	-0.71	-0.24
19	-0.72	0.61	-0.71	0.59	-0.72	0.28	-0.74	-0.02	-0.78	-0.02
20	-0.87	0.79	-0.85	0.80	-0.85	0.77	-0.86	0.42	-0.87	0.21
21	-1.00	0.93	-1.00	0.93	-1.00	0.94	-1.00	0.89	-1.00	0.65

4. CONCLUSION

Due to its high subjectivity, people always want to improve their odds in Blackjack through personal decisions. However, in some seemingly even situations such as 16 versus 10, decisions are often hard to make. But from a probabilistic perspective, each hand will have an optimal decision that gives the player a higher odds of winning. In other words, there is a set of optimal strategies that will make the player win the most or lose the least if the player follows it for a long term. This paper proposed a probability analysis of strategies used in Blackjack. Although some minor part of the rule is not considered, the simulation is still close to real-life games. By using the R program, every situation a player would encounter was simulated and the optimal strategy was given. This provides an important reference for real casino decision-making. At the same time, it proves the minor bias of the game. Without considering additional strategies, it is difficult for 21 points to become a profitable game for players.

REFERENCES

Snyder A 2013 Big Book of Blackjack. *Cardoza Publishing*.

Lindner J 2023 Must-know blackjack statistics. *GITNUX*.

Baldwin R R, Cantey W E, Maisel H and McDermott J P 1956 The optimum strategy in Blackjack. *Journal of the American Statistical Association*, **51(275)** 429.

Neumann J V and Morgenstern O 2007 Theory of games and Economic Behavior. *Princeton University Press*.

Culbertson E 1952 Culbertson's card games, complete, with Official Rules. *Greystone Press*.

Manson A R, Barr A J and Goodnight J H 1975 Optimum zero-memory strategy and exact probabilities for 4-deck blackjack. *The American Statistician*, **29(2)** 84-88.

Thorp E O 2016 Beat the dealer: A winning strategy for the game of twenty-one. *Vintage*.

Wong S 1994 Professional blackjack. *Pi Yee Press*.

Hughes J and Chen J 2021 Resilient and distributed discrete optimal transport with deceptive adversary: A game-theoretic approach. *IEEE Control Systems Letters*, **6** 1166-1171.

Ghosh P, et al. 2004 A game theory based pricing strategy for job allocation in mobile grids. *In 18th International Parallel and Distributed Processing Symposium, 2004. Proceedings*.