

Derive Poker Winning Probability by Statistical JAVA Simulation

Mason Chen

Stanford OHS, Palo Alto, CA, USA
mason05@ohs.stanford.edu

Abstract

This STEM project is to simulate the Poker Probability. The game rule is to select three from four shared cards plus two own cards to form the BEST Five to determine who can be the winner. In order to simplify the probability scenario, partial deck (9, 10, J, Q, K, and A) are drawn to increase matching probability on higher ranked patterns such as Four of a Kind and Full House. Author used JAVA random simulation method and conducted Combination and Conditional Probability to calculate each player's winning probability. Author used a simple case study to derive the winning probability between two players. JAVA random simulation is programmed to generate the random cards for both players. JAVA programming can randomly duplicate the real Poker scenario in a random way to distribute the cards to players randomly. Authors compared the JAVA simulated results against the expected probability derived based on combination and conditional probability. Contingency table, Chi-Square proportions tests were conducted and confirmed that the JAVA simulated probability could match the expected Poker probability very well. This is a very successful STEM project which has integrated JAVA Computer Science and Statistics/Probability on the Poker application.

Keywords

Java, Statistics, Poker Probability, Monte Carlo Simulation

1. Introduction and Literature Research

Most Poker players lost Money in Poker Gambling since they played blind gambling without applying the poker probability and assess their risk on each play. The objective of this paper is to use JAVA to simulate Poker Probability and study Sample Size effect on Statistics and decision making. The project scope is for learning purpose, not for gambling purpose. Authors used partial deck (9, 10, J, Q, K, A) of 24 cards to simplify JAVA poker simulation. Figure 1 has listed the rankings of different matched patterns for the full deck (52 cards) scenario. The full deck poker for 6 to 7 random cards is very popular in most Poker tournament ^[1,2]. Several research papers have demonstrated and simulated the poker probability by using Monte Carlo Simulation ^[3], Evolutionary Computing ^[4] and Artificial Intelligence ^[5]. There is also an US Patent ^[6] studied the partial deck on Royal Flush probability. In this paper, the authors will study the Poker Probability on the 24-cards Partial Deck and use JAVA Monte Carlo Simulation on a special case study to verify the winning probability between two players. The ranking of Partial Deck may be different from the Full Deck.

2. Study Partial Deck Probability

Authors used partial deck (9, 10, J, Q, K, A) of 24 cards to simplify JAVA poker simulation... Partial Deck can increase the matching probability especially on higher ranked patterns such as Four of a Kind, and Full House. Partial Deck Poker may also simplify JAVA simulation process concentrated on higher ranked patterns which may be critical for Poker Players in real time decision making on each betting move.



Figure 1. Ranking of Matched Patterns for Full Deck.

2.1 Probability Comparison of Four of a Kind

When the Poker Cards have been reduced from the Full Deck (52 Cards) to Partial Deck (24 Cards), as shown in Figure 2, the trial space is reduced by 60X factor from Combination (52, 5) to Combination (24, 5). The event of “Four of a Kind” is also reduced by 5X from 624 to 120. The even matching probability has been increased by 12X from <0.0001% to 0.002%.

Probability: Matching Four of a Kind


Full Deck	60X Higher Trials	Partial Deck
Total Permutations $C\binom{52}{5} = \frac{52!}{(52-5)!}$ $= 2,598,960$	 Four of a Kind	Total Permutations $C\binom{24}{5} = \frac{24!}{5!(24-5)!}$ $= 42,504$
$C\binom{13}{1} * C\binom{48}{1}$ $= 624$	5X Higher Events	$C\binom{6}{1} * C\binom{20}{1}$ $= 120$
Probability= 624/2,598,960 < 0.001%	12X	Probability= 120/42,504= 0.002%

Figure 2. Probability Comparison: Four of a Kind.

2.2 Odds Ratio Comparison

To extend the probability change from Full Deck to Partial Deck, as shown in Figure 3, the odds ratio has been derived. Partial Deck has significantly increased the matching probability except for “Flush” and “Nothing” Cases. These calculations are based on between Full Deck and Partial Deck of 24 Cards. The ranking of matching probability is also changed. For example, for 24-cards partial deck, the probability of matching “Flush” is lower than the probability of matching “Full House”, same as “Nothing” lower than “One Pair”. This ranking order change is critical if the Poker Game has used only partial deck.

	Full Deck			24 Partial Deck			Ratio
	Trial	Event	Probability	Trial	Event	Probability	
Royal Straight	C(52, 5) 2,598,960	C(4,1)	0.000%	C(48,5) 42,504	C(4,1)	0.009%	61.1
Straight Flush		C(4,1)*C(9,1)	0.001%		C(4,1)*C(1,1)	0.009%	6.5
Four of a Kind		C(13,1)*C(12,1)*C(4,1)	0.024%		C(6,1)*C(5,1)*C(4,1)	0.282%	11.7
Full House		C(13,1)*C(12,1)* C(4,3)*C(4,2)	0.144%		C(6,1)*C(5,1)* C(4,3)*C(4,2)	1.694%	11.8
Flush		C(4,1)*C(13,5)- C(4,1)*C(10,1)	0.197%		C(4,1)*C(6,5)- C(4,1)*C(2,1)	0.038%	0.2
Straight		C(10,1)*[C(4,1)^5- C(4,1)]	0.392%		C(2,1)*[C(4,1)^5- C(4,1)]	4.800%	12.2
Three of a Kind		C(13,1)*C(12,2)* C(4,3)*C(4,1)*C(4,1)	2.113%		C(6,1)*C(5,2)* C(4,3)*C(4,1)*C(4,1)	9.034%	4.3
Two Pair		C(13,2)*C(11,1)* C(4,2)*C(4,2)*C(4,1)	4.754%		C(6,2)*C(4,1)* C(4,2)*C(4,2)*C(4,1)	20.327%	4.3
One Pair		C(13,1)*C(12,3)*C(4,2)* C(4,1)*C(4,1)*C(4,1)	42.257%		C(6,1)*C(5,3)*C(4,2)* C(4,1)*C(4,1)*C(4,1)	54.207%	1.3
Nothing		[C(13,5)-10] * [C(4,1)^5-4]	50.118%		[C(6,5)-2] * [C(4,1)^5-4]	9.599%	0.2

Figure 3. Odds ratio comparison of Full Deck to Partial Deck.

2.3 Poker Partial Deck Case Study

In order to demonstrate Poker probability and simulation simply, a special case study has been created in Figure 4.



Figure 4. Case Study

Four cards are shown in the shared dealer field; two players have one card shown and one card hidden as in Figure 4. Each player will calculate the winning probability by guessing the other unknown card of the opponent's hand. To keep in simple, the authors will consider "tie" if the matching pattern is the same and the card numbers are the same even the card category is different (for example, Spade A will be treated the same as Heart A as tie).

2.4 Derive Poker Winning Probability for the Special Case

In Figure 5, the winning scenario has been thoroughly explored for each condition. There are three outcomes: (1) Player A won, (2) Player B won, and (3) both players tied. The expected probability is calculated based on 24-cards partial deck. In summary, Player A has 20.3% winning chance; Player B has 19.7% winning chance and the remaining 60% chance of tie.

A	B	Who won?	Formula	Expected Probability
A Full House (A)	A Full House (A)	Tie	$=(2/18)*(1/17)$	0.7%
A Full House (A)	J Full House (J)	A	$=(2/18)*(2/17)$	1.3%
A Full House (A)	Not Full House (K,Q,10,9)	A	$=(2/18)*(14/17)$	9.2%
K Full House (K)	A Full House (A)	B	$=(2/18)*(2/17)$	1.3%
K Full House (K)	J Full House (J)	A	$=(2/18)*(2/17)$	1.3%
K Full House (K)	Not Full House (K,Q,10,9)	A	$=(2/18)*(13/17)$	8.5%
Not Full House (Q,J,10,9)	A Full House (A)	B	$=(14/18)*(2/17)$	9.2%
Not Full House (Q,J,10,9)	J Full House (J)	B	$=(14/18)*(2/17)$	9.2%
Not Full House (Q,10,9)	Not Full House (K,Q,10,9)	Tie	$=(12/18)*(13/17)$	51.0%
Not Full House (J)	Not Full House (K,Q,10,9)	Tie	$=(2/18)*(13/17)$	8.5%
				100%

Figure 5. Winning Opportunity of the Special Case Study.

So far, authors have derived the expected winning probability of two players or tie. In the next session, authors will run Monte Carlo Simulation to verify these expected values by random generation like playing the real games.

3. JAVA Monte Carlo Simulation

In order to simulate the real Poker Games (random variation), authors have adopted the Monte Carlo Simulation algorithm by JAVA random generation programming as shown in Figure 6 of JAVA Flow Chart. There are six major steps in JAVA simulation: (1) Create 24-Cards Partial Deck, (2) Create 6-Cards Case Study, (3) Develop Shuffle Program for remaining 18 cards, (4) use Random Generation to pick two hidden cards, (5) print out the result, and (6) repeat 25 times of random process.

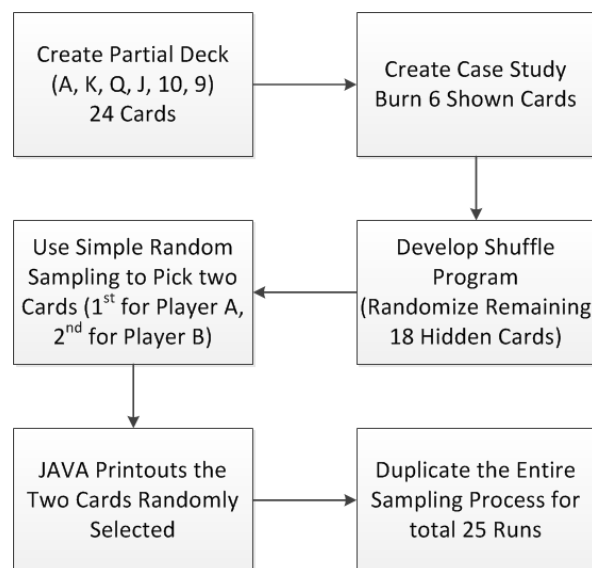


Figure 6. JAVA Flow Chart

3.1 JAVA Random Generator Simulation

Authors used JAVA Random Generation method to randomly pick two cards: one for player A and one for player B as shown in Figure 7. Then, add this new card with previous two share cards and two own cards (total five cards) to determine whether Player A or/and Player B have Full House (A, K, 0r/and J). Based on the Full House result, the winner can be determined in the last column in Figure 7. Total 25 Monte Carlo Simulation Runs were generated.

JAVA	JAVA Random Card		Full House?		Who Won
	Player A	Player B	Player A	Player B	
1	9 of Heart	10 of Spade	Not	Not	Tie
2	Queen of Heart	9 of Club	Not	Not	Tie
3	9 of Heart	9 of Spade	Not	Not	Tie
4	Queen of Spade	9 of Heart	Not	Not	Tie
5	10 of Spade	9 of Diamond	Not	Not	Tie
6	9 of Club	Jack of Heart	Not	J	B
7	9 of Club	King of Club	Not	Not	Tie
8	Jack of Club	9 of Heart	Not	Not	Tie
9	9 of Diamond	9 of Spade	Not	Not	Tie
10	King of Heart	10 of Heart	K	Not	A
11	Ace of Club	Jack of Diamond	A	J	A
12	King of Club	Jack of Heart	K	J	A
13	Queen of Spade	Ace of Diamond	Not	A	B
14	Jack of Club	King of Heart	Not	Not	Tie
15	King of Heart	Jack of Heart	K	J	A
16	Jack of Diamond	Queen of Spade	Not	Not	Tie
17	Jack of Club	10 of Spade	Not	Not	Tie
18	Jack of Club	Queen of Club	Not	Not	Tie
19	9 of Club	Queen of Heart	Not	Not	Tie
20	9 of Heart	Queen of Club	Not	Not	Tie
21	Ace of Diamond	10 of Spade	A	Not	A
22	9 of Heart	Ace of Club	Not	A	B
23	1o of Club	9 of Diamond	Not	Not	Tie
24	King of Heart	Queen of Club	K	Not	A
25	Jack of Diamond	Ace of Club	Not	A	B

Figure 7. JAVA Random Generation and Poker Result.

3.2 Verify the JAVA Simulation Accuracy on Individual Matching Scenario

Authors used Minitab Tabulated Statistics on the JAVA simulation data in Figure 7 and created a summary table as shown in Figure 8.

Tabulated Statistics: A, B

```

Rows: A    Columns: B

      A  J  Not  All
A      0  1   1   2
K      0  2   2   4
Not    3  1  15  19
All    3  4  18  25

Cell Contents:      Count

```

Figure 8. Tabulated Statistics of JAVA Simulation Summary.

Then, compared the JAVA simulated probability (Figure 8) against the expected probability (Figure 5) on each scenario. Run 1-Proportion Z test to verify JAVA simulation algorithm. Except (Player A- K Full House, Player B- J Full House), all the other scenarios showed good matching (P-Values > 0.05) as shown in Figure 9. In general, the JAVA simulation results matched the expected probability very well. The matching performance can be further improved by increasing the sample size of JAVA simulation run (for example from current 25 runs to 50 runs).

A	B	Who won?	Formula	Expected Probability	Simulated Probability	1- Proportion P-Value
A Full House (A)	A Full House (A)	Tie	$=(2/18)*(1/17)$	0.7%	0%	1.000
A Full House (A)	J Full House (J)	A	$=(2/18)*(2/17)$	1.3%	4%	0.279
A Full House (A)	Not Full House (K,Q,10,9)	A	$=(2/18)*(14/17)$	9.2%	4%	1.000
K Full House (K)	A Full House (A)	B	$=(2/18)*(2/17)$	1.3%	0%	1.000
K Full House (K)	J Full House (J)	A	$=(2/18)*(2/17)$	1.3%	8%	0.042
K Full House (K)	Not Full House (K,Q,10,9)	A	$=(2/18)*(14/17)$	9.2%	8%	1.000
Not Full House (Q,J,10,9)	A Full House (A)	B	$=(14/18)*(2/17)$	9.2%	12%	0.724
Not Full House (Q,J,10,9)	J Full House (J)	B	$=(14/18)*(2/17)$	9.2%	4%	1.000
Not Full House (Q,J,10,9)	Not Full House (K,Q,10,9)	Tie	$=(14/18)*(13/17)$	58.8%	60%	1.000
				100%	100%	

Figure 9. Compare JAVA Simulation Result against the Expected Probability.

3.3 Verify the JAVA Simulation Accuracy on Player Winning Probability

Next, the winner probability by JAVA simulation (Figure 8) was compared to the expected winning probability (in section 2.4). In Figure 10, JAVA Random Simulation method can match the expected probability reliably. Player A has a slightly higher chance to win over Player B (Because Player A K Full House > Player B J Full House).

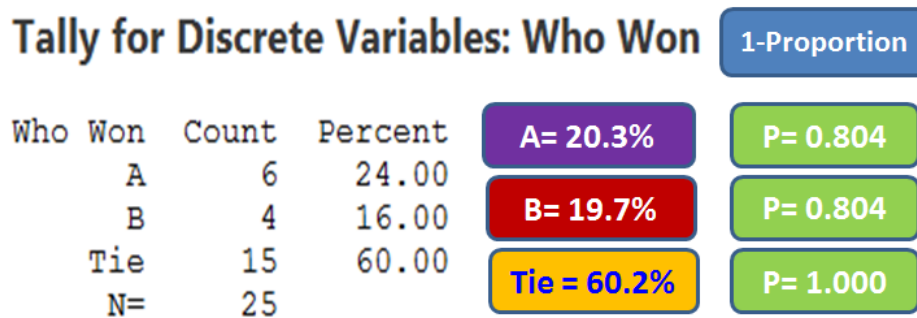


Figure 10. Winning Probability: JAVA Simulation vs. Expected Probability

4. Results and Conclusions

Authors have successfully applied both Probability and JAVA programming on simulating Poker Winning Probability. The JAVA simulation can predict the expected matching probability and players' winning probability very well based on 1-Proportion tests. Partial Deck Poker was used to simplify the winning patterns and probability simulation. The ranking of the matching patterns has been changed from the Full Deck to Partial Deck as unexpectedly. Authors have created a Case Study and focus on Full House pattern scenario to demonstrate the JAVA Monte Carlo simulation.

5. Future Work

This paper can be further polished by the following opportunities: (1) extend the partial deck from 24 cards to in general, (2) expand it from current 6-cards to 7-cards (five shared, 2 owned), and (3) consider the winning probability of all possible matching (not just on Full House).

Acknowledgements

Need to acknowledge Dr. Charles Chen and Dr. Ying Huang here for the guidance and mentorship here....

References

- [1] D. Billings, et al., "The challenge of poker," *Artif. Intell.*, vol. 134, pp. 201-240, 2002.
- [2] Wenkai Li and Lin Shang, "Estimating Winning Probability for Texas Hold'em Poker", *International Journal of Machine Learning and Computing*, Vol. 3, No. 1, February 2013
- [3] N. M. S. Ulam, "The Monte Carlo Method," *Journal of the American Statistical Association*, vol. 44, pp. 335-341 1949.
- [4] H. Quek, C. Woo, K. Tan, and A. Tay "Evolving Nash-optimal poker strategies using evolutionary computation", *Frontiers of Computer, Science in China*, pp. 73–91, 2009.
- [5] D. Billings, D. Papp, J. Schaeffer, and D. Szafron "Opponent modeling in poker," in *Proceedings of the National Conference on Artificial Intelligence*, 1998, pp. 493–499
- [6] Bradley Ward, Anthony Cabot, (2004) "Partial-deck poker game with guaranteed royal flush opportunity", US Patent 20040212147 A1

Biography

Mason Chen is student in Stanford On-Line High School Program. Mason has certified Lean Six Sigma Black Belt through IASSC (International Associate of Six Sigma Certification), and also certified IBM SPSS/Modeler Statistics and Data Mining Certificates. Mr. Chen has been invited to several conferences like IEOM, ASQ, AQI, ASA, JMP/SAS and local ASQ SV statistics group to present his STEM Projects. His STEM projects have drawn interest in Robotics/EV3, JAVA Science, Poker Probability, Powerball Lottery, Sports Analytics, Biostatistics and Healthcare Statistics... Mason is familiar with Lean Six Sigma DMAIC, DFSS, and Minitab 17, JMP 13, SPSS 24, and Modeler 18 Statistics Software. Mason has also been learning Data Mining and Big Data Analytics through several STEM Projects. As a Stanford High School Student, he has published several Conference Proceeding Papers in IEOM, ISF, IWSM, FSDM conferences.