Blackjack Write Up

April 27, 2012

Michael Adler

Cole Griesbach

Josie Trafelet

Isaac Wilda

Throughout the majority of the semester, we have been working on how to simulate the game of blackjack using Mathematica. The main goals of this project were to create a simulation that included rules commonly found in casinos, write code in a way that would be easily understood and modified to fit different rule variations, and to include a card counting strategy within that code. Some things we wanted to investigate using these simulations are the advantages associated with different card counting schemes and the advantages associated with different rule variations.

The first thing we did was identify how the game of blackjack is actually played in casinos. The main objective of the game is to beat the dealer's score without busting, or having a score over 21. Your score is determined by the cards you are dealt, with the numbered cards having a value equal to their number, face cards being worth ten, and aces being worth either 11(soft) or 1(hard). Whether an ace is soft or hard depends on the other cards. If the value of the other cards is ten or less, the ace is soft. If the total of the other cards is more than ten, the ace is hard.

Placing your bet is the first thing that happens in a game of blackjack. Most tables that one will play at in a casino have a bet minimum and maximum, and you have to stay inside those constraints. After your bet is placed, the dealer deals one face up card from the shoe, or stack of multiple decks of cards, to every player, followed by one face down card to him/herself. The dealer then deals another face up card from the shoe to each player, followed by one face up card to him/herself. If the dealer has a natural, or a hand consisting of only two cards that has a value of 21, every player that does not have a natural loses, and every player that does have a natural ties, and their bets are transferred to the next hand. If the dealer does not have a natural, players make decisions about their hand in order to try and beat the dealer's score.

There are five decisions a player can make: hit, stand, double, split, and surrender. Hitting is asking the dealer for another card from the shoe to be added to your total. You can hit as many times as you would like to until you either bust or decide to stand. Standing is being content with the score of your hand, and is the last move of the hand for a player. Doubling is simply doubling your original bet. Usually this is the only move you make and can only be the first decision of your turn. Splitting can only be done

when a player is dealt a pair of cards. When a player splits, they divide their original hand into two hands, each with the same bet as the original bet for the hand. These two hands are then played like any other hand. Surrendering is giving up a hand in return for half of the original bet. After every player plays their hand, the dealer then flips the face down card over, and continues to hit until his or her hand reaches a score of 17, at which point he or she must stay. Because of this, if the dealer has a score of 16, he or she must hit. When the dealer either busts or has a score between 17 and 21, each player compares their hands to the dealers. If the player wins with a natural, he or she receives a payout of 3:2. If the player wins without a natural, the payout is 1:1. If the player loses, he or she loses his or her bet.

Once we understood how blackjack was played, we looked at how to count cards. We found that there are many different schemes, but they all work the same way. Each card is given a count value, and as cards are played, the person counting cards is keeping a running total (starting at zero), adding the count values of the cards played in order to make inferences about the remaining cards in the shoe. In the card counting schemes that we looked at, lower valued cards were given positive values and higher valued cards were given negative values. Keeping this in mind, if the running total, or count, is positive, one can infer that there are more higher valued cards remaining in the shoe than lower valued cards. Knowing this count also influences how you bet. When the count is positive, your probability of winning is supposed to be over 0.5, which means you should bet higher. The three counting strategies we used in our simulation were Hi Lo, Zen, and Revere RAPC. The count value of each card for each strategy can be seen in the table below.

Count Values

Strategy ⇓ / Card ⇒	2	3	4	5	6	7	8	9	10/J/Q/K	A
Hi Lo	+1	+1	+1	+1	+1	0	0	0	-1	-1
Zen	+1	+1	+2	+2	+2	+1	0	0	-2	-1
Revere RAPC	+2	+3	+3	+4	+3	+2	0	-1	-3	-4

When we figured out how to play blackjack, the rules involved in playing, and how card counting worked, we started to assemble our code. The first things we coded were how to create a new deck and how to deal cards. When we were doing this, we also coded a shuffle barrier function to mimic when the dealer would clear the existing shoe and start a new one in reality. Once we set up the deck and dealing, we coded how to score a hand, keeping hard and soft totals in mind, as well as identifying if a hand was a pair or a natural. After that was done, we set up card counting functions that were easily modified to suit the three different card counting strategies. Along with those, we coded three different decision tables found on wikipedia that told the player what to do for every possible hand in regards to the dealer's face up card. These tables can be seen below.

Hard Totals Decision Table

Player hand	Dealer's face-up card									
	2	3	4	5	6	7	8	9	10	A
Hard totals (excluding pairs)										
17-20	S	S	S	S	S	S	S	S	S	S
16	S	S	S	S	S	H	H	SU	SU	SU
15	S	S	S	S	S	H	H	H	SU	H
13–14	S	S	S	S	S	H	H	H	H	H
12	H	H	S	S	S	H	H	H	H	H
11	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	H
10	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	H	H
9	H	Dh	Dh	Dh	Dh	H	H	H	H	H
5–8	H	H	H	H	H	H	H	H	H	H

Soft Totals Decision Table

Soft totals										
	2	3	4	5	6	7	8	9	10	A
A,8–A,9	S	S	S	S	S	S	S	S	S	S
A,7	S	Ds	Ds	Ds	Ds	S	S	H	H	H
A,6	H	Dh	Dh	Dh	Dh	H	H	H	H	H
A,4–A,5	H	H	Dh	Dh	Dh	H	H	H	H	H
A,2–A,3	H	H	H	Dh	Dh	H	H	H	H	H

Pairs Decision Table

Pairs										
	2	3	4	5	6	7	8	9	10	A
A,A	SP	SP	SP	SP	SP	SP	SP	SP	SP	SP
10,10	S	S	S	S	S	S	S	S	S	S
9,9	SP	SP	SP	SP	SP	S	SP	SP	S	S
8,8	SP	SP	SP	SP	SP	SP	SP	SP	SP	SP
7,7	SP	SP	SP	SP	SP	SP	H	H	H	H
6,6	SP	SP	SP	SP	SP	H	H	H	H	H
5,5	Dh	Dh	Dh	Dh	Dh	Dh	Dh	Dh	H	H
4,4	H	H	H	SP	SP	H	H	H	H	H
2,2–3,3	SP	SP	SP	SP	SP	SP	H	H	H	H

After the more basic functions of our code were in place, we set up some more complicated functions. Those included player actions, dealer actions, and the betting function. In our simulations, the player started out with 100 dollars, and the table limits were a one dollar minimum and a ten dollar maximum. The betting function also bet the minimum when the count was zero or negative (unfavorable), and the maximum when the count was positive (favorable).

Once the basic coding was in place, we coded a repeat function that would repeat the playing process. Along with that, we set up analysis for six different variations: NoCount, TwoDeck, Soft17, HiLo, Zen, and RAPC. Those variations were based off of the same bit of code with minor adjustments to accommodate the different variations. Although there were differences between variations, each variation included data collection parameters, code that would write data to a file, stack and count plots, code that could construct a bar chart that graphed the amount of hands at a given count, the average income per hand, and code that recorded the total number of hands played. The first variation, NoCount, was set up with the following parameters: 5 deck shoe with the shuffle barrier 80% of the way through the shoe, infinite doubling (if it was an option on the decision table), infinite splitting (if an option on the decision table), surrendering whenever it was an option on the decision table, and the dealer stayed on a score of 17, regardless if it was a hard total or a soft total. The player started with 100 dollars and

always bet the minimum bet. The NoCount variation acted as our control, and was the variation that we compared the other variations to. The TwoDeck variation had the same parameters as the NoCount variation did, but instead of a 5 deck shoe, there was a two deck shoe. Similarily, the Soft17 variation had the same parameters as the NoCount variation did, except the dealer hit on a score of 17 if it was soft.

For the last three variations, HiLo, Zen, and RAPC, every parameter, except for betting, was the same as the NoCount variation. However, along with those parameters, each variation included their respective counting strategies, as well as betting the maximum if the count was positive and betting the minimum when the count was negative.

When we set up all six variations and ran them for some time, we collected some general data. That data included the simulation time, number of hands played, and average winnings per hand. These results are shown in the table below.

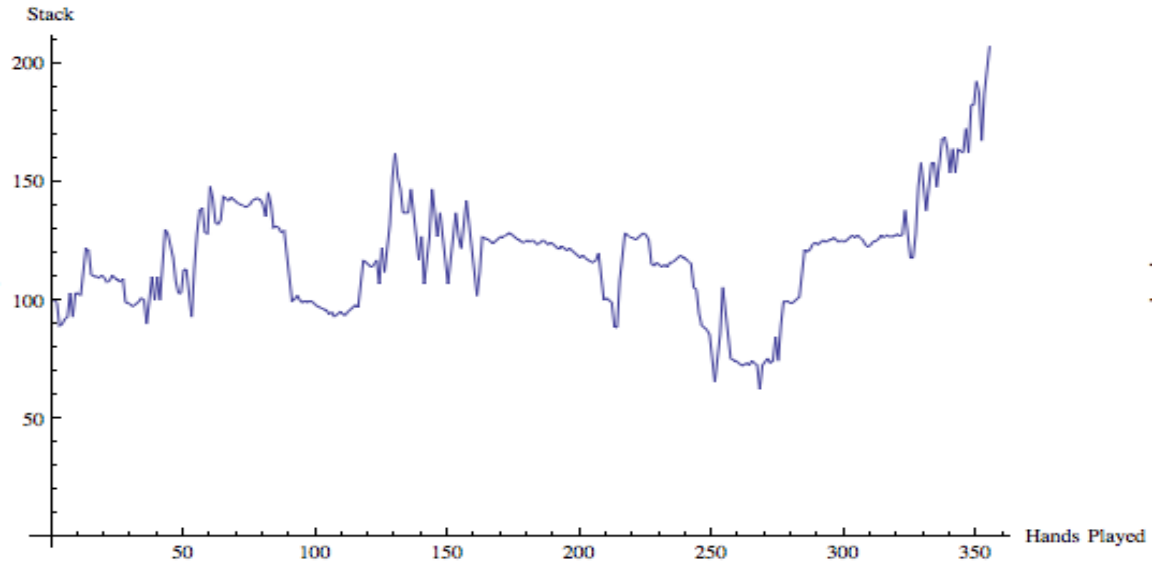
General Data Table

Variation	Simulation Time	Hands Played	Average Winnings Per Hand (% of dollar)
NoCount	6 hours 2 min.	50 348 521	0.3376
TwoDeck	15 hours 45 min.	49 656 627	0.4305
Soft17	15 hours 24 min.	49 623 701	0.5590
HiLo	45 hours 9 min.	116 121 585	7.193
Zen	15 hours 20 min.	39 375 050	7.494
RAPC	14 hours 58 min.	38 423 377	7.520

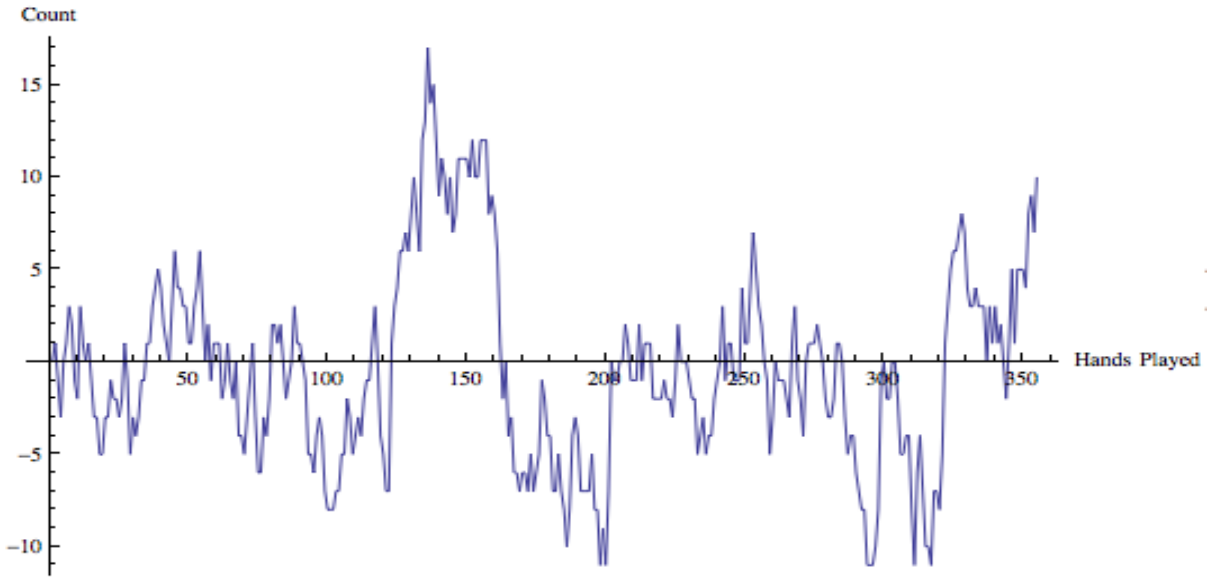
We also collected data that was more specific to the betting aspect of each of the counting variations. For example, the graphs on the next page show the chip stack (top) and the count (bottom) over a game of blackjack that lasted a little more than 350 hands. The game lasted that long because on the last turn, the player reached a chip count greater than or equal to double the starting amount, which was 200 dollars. As you can see, the chip stack graph is similar to the count graph, but when the count is negative the chip

stack doesn't change as drastically as it does when the count graph is positive. This is because when the count is negative, the bet is 1, and when the count is positive, the bet is 10.

Chip Stack

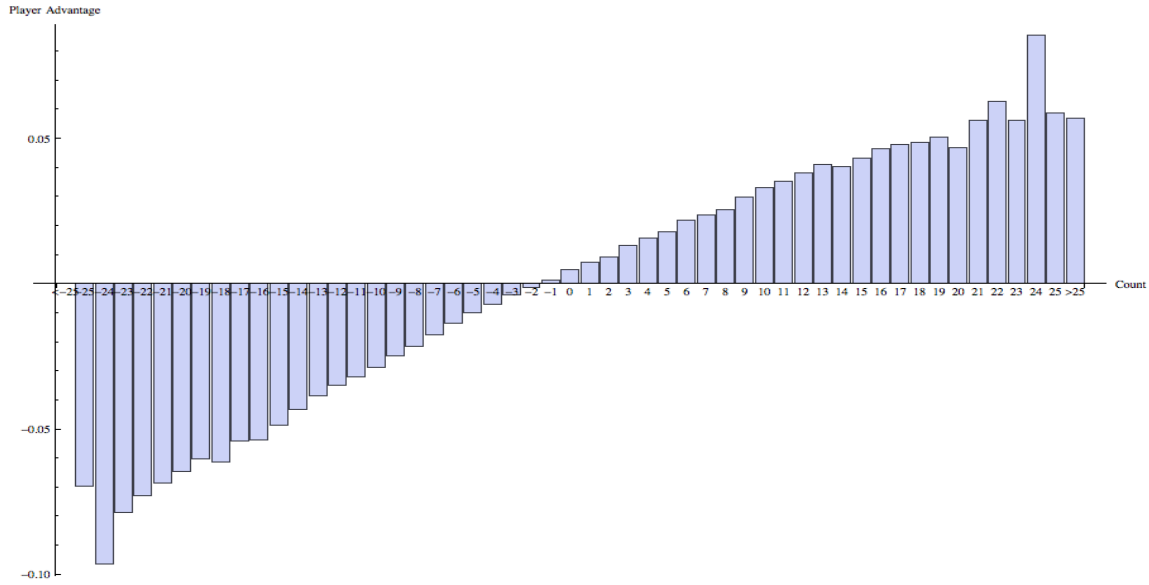


Count

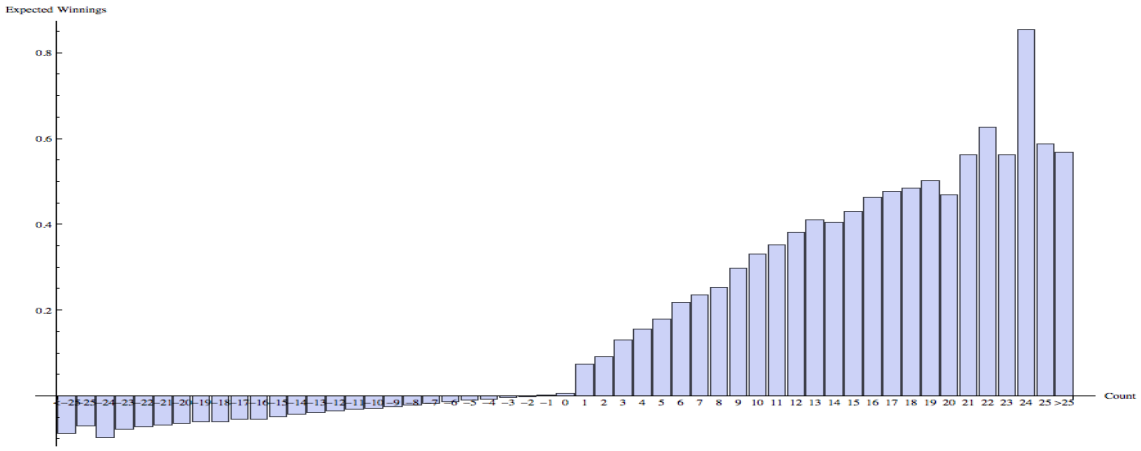


Some other types of data that we collected were the player advantage for each count, the expected winnings for each count, and the number of hands for each count. This data can be seen in the bar graphs below. Notice that the player advantage and expected winnings graphs are similar, with the expected winnings graph scaling the count bars above zero by a factor of ten. This is because when the count was positive, the bet was raised from one dollar to ten dollars. Also, the count distribution bar graphs are normally distributed. Along with being normally distributed, the count distribution bar charts have more and more hands being played above a count of 25 and below a count of -25. This is because as the counts get more complex, the count values for each card go from -1 to 1 in the HiLo variation to -4 and 4 in the RAPC variation, which resulted in more hands at higher and lower counts. The y-axis of the count distribution graphs are in a logarithmic scale.

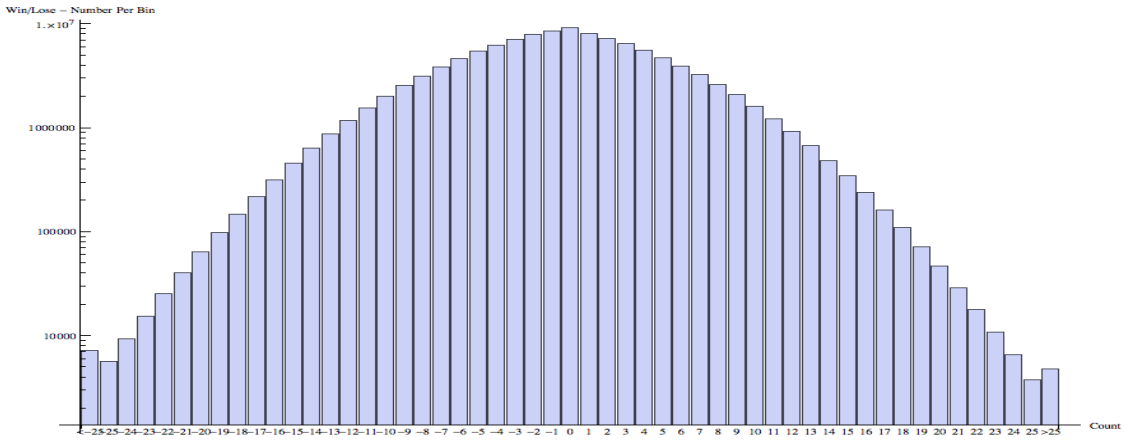
HiLo Player Advantage



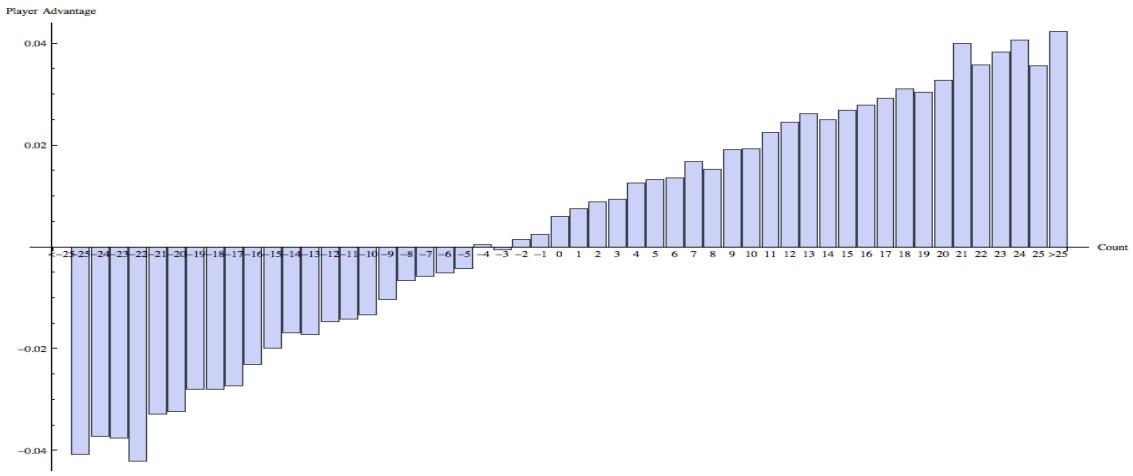
HiLo Expected Winnings



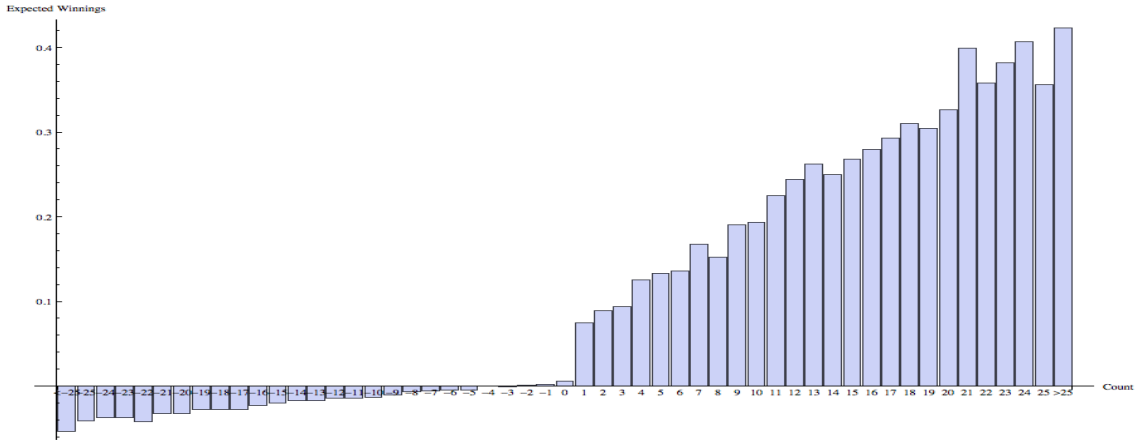
HiLo Count Distribution



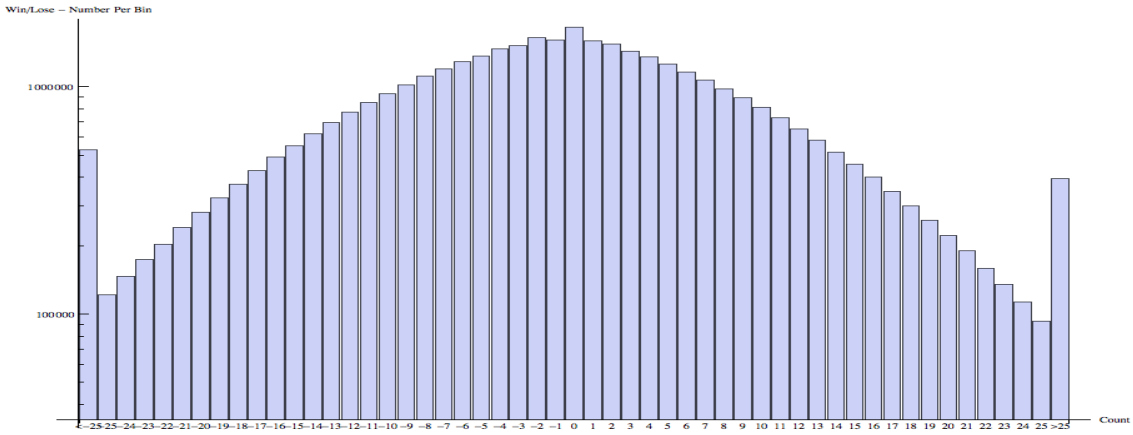
Zen Player Advantage



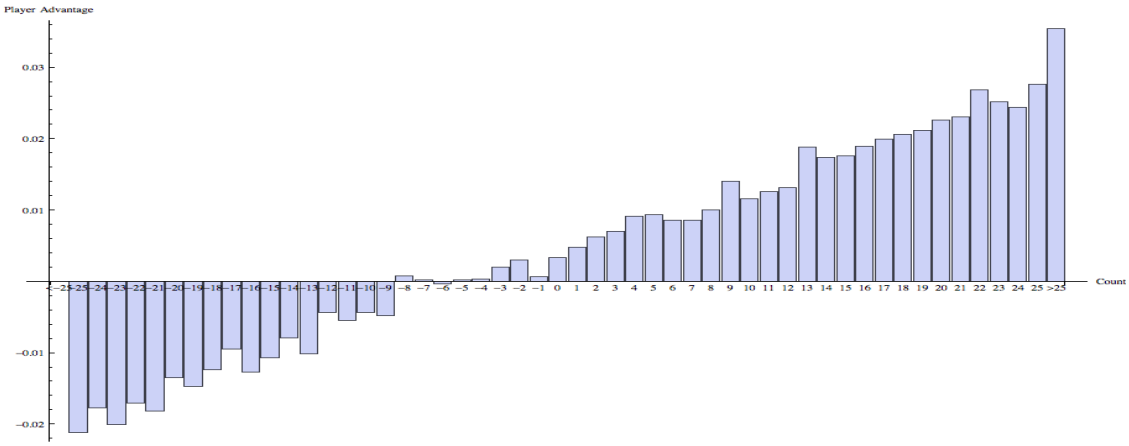
Zen Expected Winnings



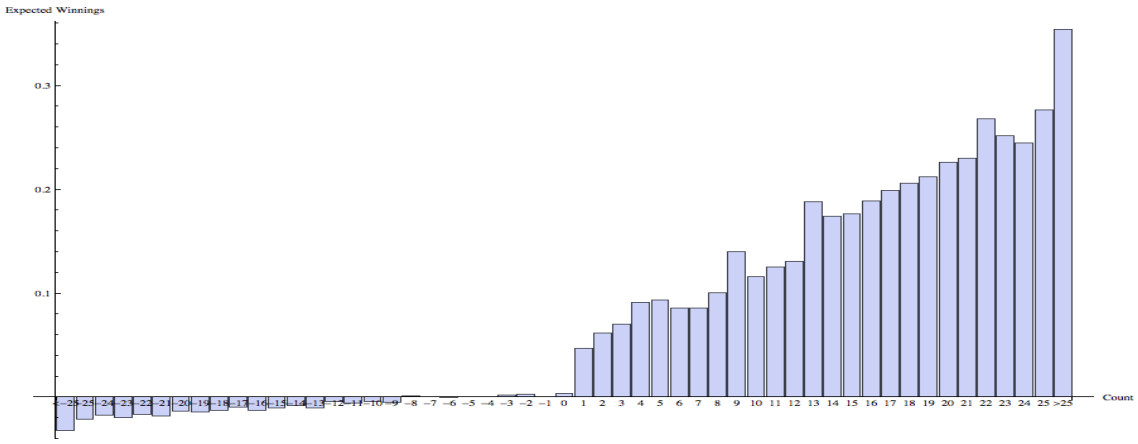
Zen Count Distribution



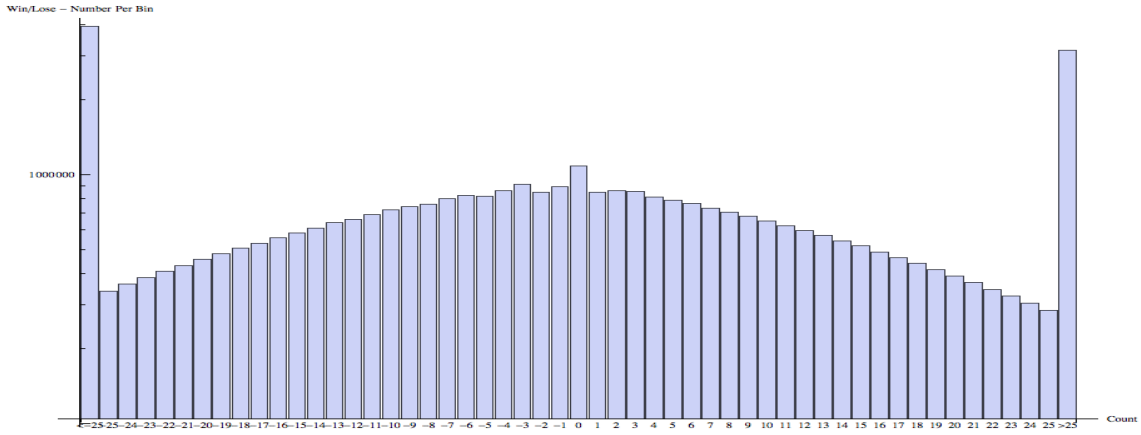
Revere RAPC Player Advantage



Revere RAPC Expected Winnings



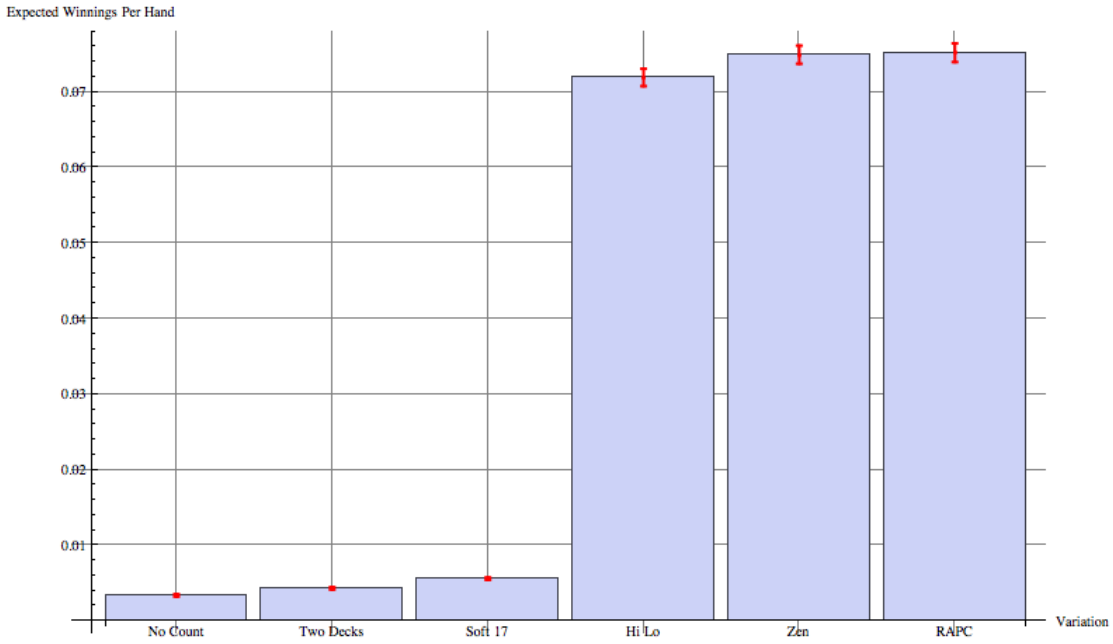
Revere RAPC Cound Distribution



We made a couple major observations when we were interpreting these graphs. The first was the player advantages. As the variations get more complex, the range of player advantages as the counts go from -25 and below to 25 and above get narrower as the counts get more complex. The HiLo variation saw a player advantage peak of 58.54% at a count of 24 and a player advantage low of 40.36 at a count of -24. The Zen variation saw a player advantage peak of 54.23% when the count was 25 and higher and a player advantage low of 45.79% at a count of -22. Finally, the RAPC variation saw a player advantage peak of 53.54 when the count was 25 and higher and a player advantage low of 47.79 when the count was -25 and below. The second major observation was the

count at which the player advantage went from above 50% to below 50% shifted as the counts got more complex. Notice that the count at which the player advantage went from favorable to unfavorable was -1 for the HiLo variation, -3 (arguably) for the Zen variation, and -8 (arguably) for the RAPC variation. When you take both of these observations into account, along with the fact that the betting system for all three variations was the same (betting the minimum when the count was zero or negative and betting the maximum when the count was positive), the graph of the average expected winnings per hand, shown below, is skewed. Had we known beforehand that the player advantages were favorable at different counts, we would have adjusted our betting function to better suit each variation. Because we did not do this, the average expected winnings per hand are not completely accurate. Regardless, they graph showing this data below still shows us our results.

Average Expected Winnings Per Hand



Over the course of the semester, we put together code that allowed us to simulate the game blackjack, count cards, and produce data regarding the count and chip stack. Along with that, we coded our simulation in such a way that making minor adjustments

to the rules and changing the card counting scheme was easily done. The results of the simulations we ran were pretty convincing. Counting cards and betting with regard to the count did increase the expected winnings of a player playing blackjack. On top of this, our data suggests that if you use a more complex count, the expected winnings will get higher.

Work Cited

"Blackjack." *Wikipedia*. N.p., 20 Apr. 2012. Web. 26 Apr. 2012. <http://en.wikipedia.org/wiki/Blackjack>.