Simulation 6644 Project: Yahtzee Simulation

Krista Radecke

Abstract:

In this project, I created a full Yahtzee simulation with four distinct strategies to determine which strategy performs better than the others. The four strategies varied which sections were filled out first and if point opportunities were checked in between each roll. The strategy that performed the best was filling out the bottom section of the scorecard first while checking for point opportunities. The 95% confidence interval for the mean is [191.121, 191.403]. This strategy performed much better than the other strategies so is therefore the optimal strategies of the ones I coded.

Background and Description of Problem:

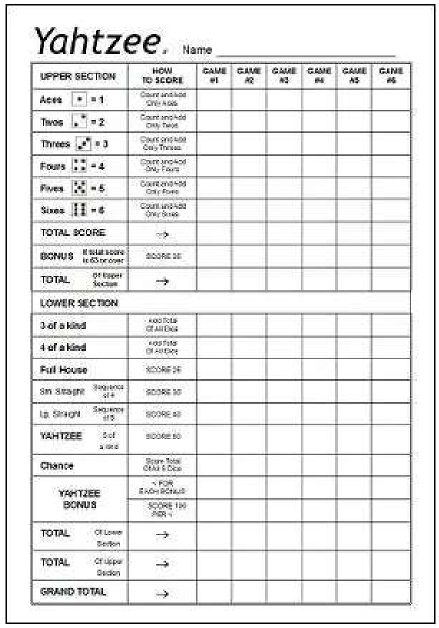
For this project I have chosen to program a Yahtzee simulation along with different strategies of playing Yahtzee to determine the optimal method. Yahtzee is a complex game to simulate, because of all the decisions that someone could make when they play. Each roll, the player could change the dice that they are hoping to roll based on the last roll and make changes up until the last second depending on how the game has been played. For example, a player could be going for a large straight and instead end up with three 3's after the second roll and change their mind in the hope of scoring a Yahtzee with 3's. While these decisions are obvious to our human brain, they all must be carefully programmed into the simulation's decision-making functions. To find the optimal way to play Yahtzee, I have coded in four different strategies and ran them 1000 times against each other to find the optimal strategy.

Main Findings:

I had two main strategies to code, fill out the top of the scorecard first or fill out the bottom of the scorecard first. I also wanted to try each strategy with the addition of another way to record points by checking in between each roll for ways we could record points. The strategies are as follows:

1. Fill in the upper part of the scorecard first. If that is complete, begin filling out the bottom.
2. Fill in the bottom of the scorecard first. If the roll does not fit the requirements for any bottom scores, then can record score in upper part if it is available.
3. Fill in the top of the scorecard first but check for point scoring opportunities between each roll.
4. Fill in the bottom of the scorecard first but check for point scoring opportunities between each roll.

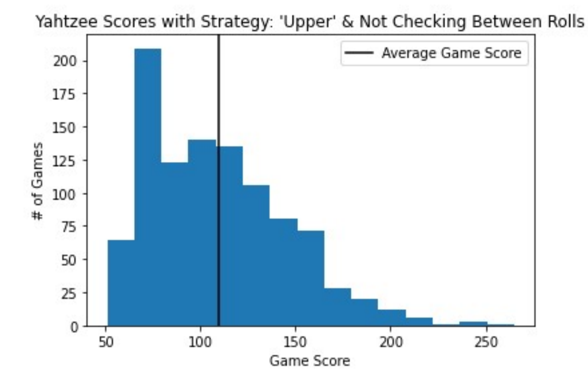
In case you aren't familiar with the Yahtzee scorecard, I have included a copy below. In both strategies, if the roll doesn't fit into any of the sections, then the program is permitted to see if the roll fits in the other section.

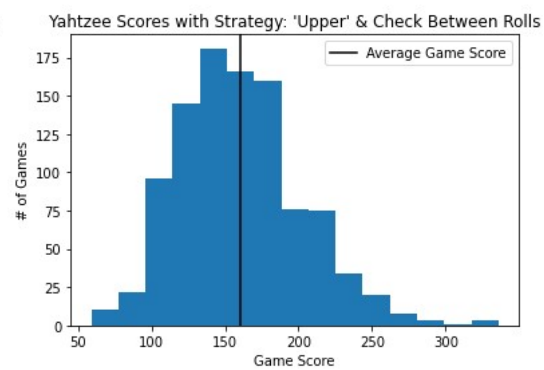


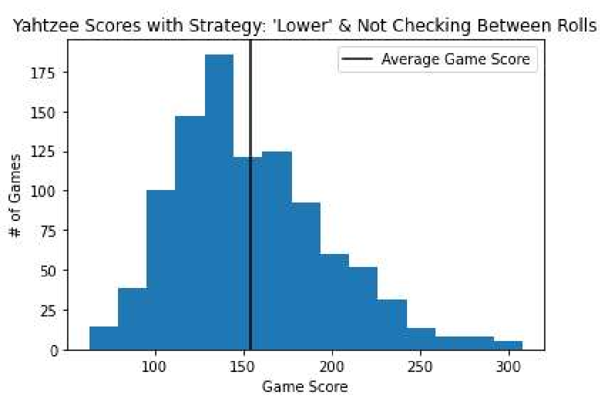
Yahtzee Name

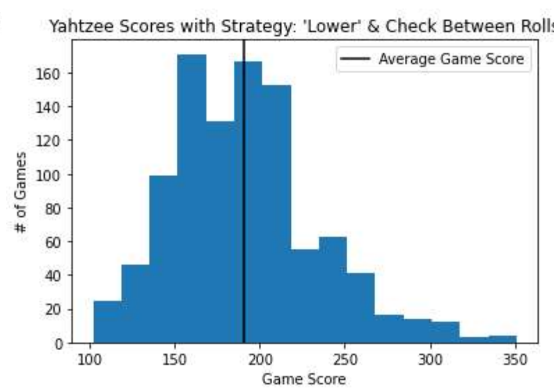
UPPER SECTION	HOW TO SCORE	GAME #1	GAME #2	GAME #3	GAME #4	GAME #5	GAME #6
Aces = 1	Count and Add Only Aces						
Twos = 2	Count and Add Only Twos						
Threes = 3	Count and Add Only Threes						
Fours = 4	Count and Add Only Fours						
Fives = 5	Count and Add Only Fives						
Sixes = 6	Count and Add Only Sixes						
TOTAL SCORE	→						
BONUS If total score is 63 or over	SCORE 35						
TOTAL Of Upper Section	→						
LOWER SECTION							
3 of a kind	Add Total Of All Dice						
4 of a kind	Add Total Of All Dice						
Full House	SCORE 25						
Sm. Straight Sequence of 4	SCORE 30						
Lg. Straight Sequence of 5	SCORE 40						
YAHTZEE 5 of a kind	SCORE 50						
Chance	Score Total Of All 5 Dice						
YAHTZEE BONUS	✓ FOR EACH BONUS						
	SCORE 100 PER ✓						
TOTAL Of Lower Section	→						
TOTAL Of Upper Section	→						
GRAND TOTAL	→						

The application is set up to run each strategy 1000 times and then present a histogram of the scores for that strategy. It is also set up to print out the mean and the standard deviation for each run. To run this program, just open it on your favorite IDE, I use Spyder, and press "run". The charts will not appear the same each time due to the inherit randomness of the game, but the means are always close to each other. Below, I have presented all the histograms and the statistics that the program returns.



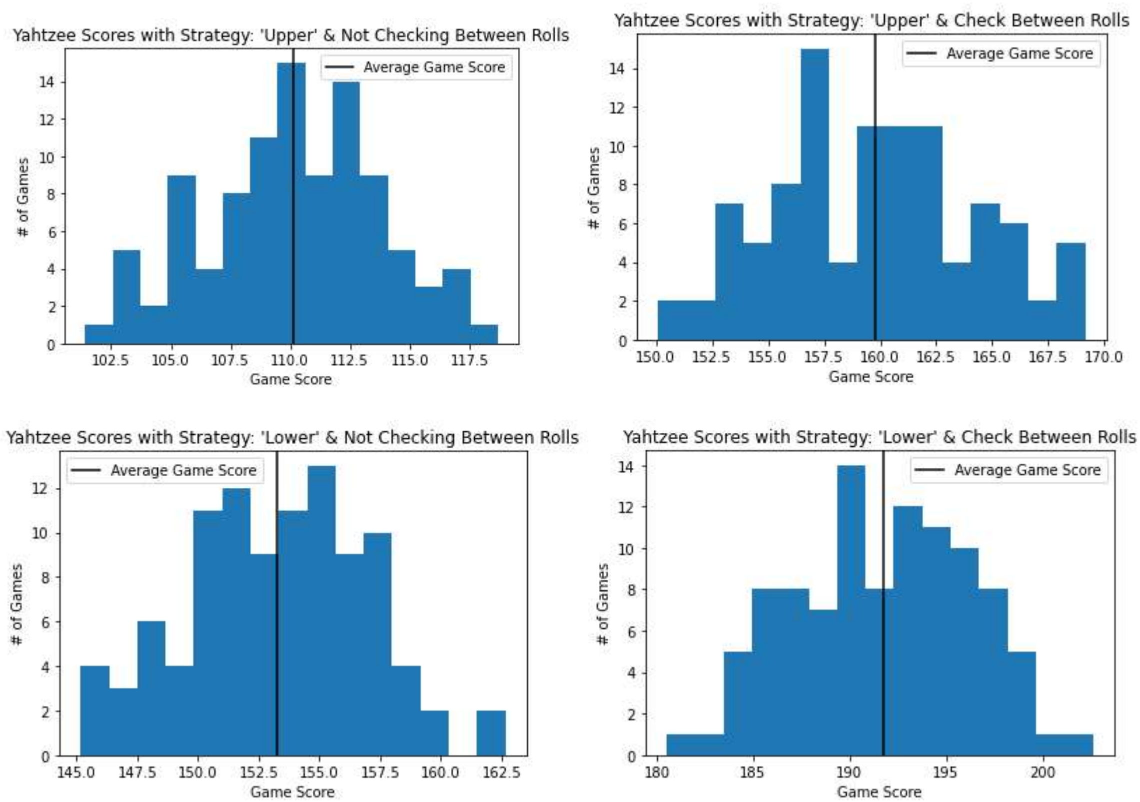






Strategy	Upper		Lower	
Check Between Rolls?	No	Yes	No	Yes
Average Score	109.209	159.492	153.352	189.524
Standard Deviation	36.61	42.54	43.52	40.80

These histograms and statistics give us an idea of how the simulations perform, but to truly analyze the performance we must perform independent replications. This ensures that the sample means are independent and identically distributed. The following results are from 50 independent replications of 200 simulation runs.



Strategy	Upper		Lower	
Check Between Rolls?	No	Yes	No	Yes
Average Score	110.104	159.79	153.256	191.762
Standard Deviation	2.38	2.501	2.655	3.203

Using these statistics, I can then obtain a confidence interval around the means for each different strategy. The following calculations are made with 49 degrees of freedom and a 95% confidence level. From these intervals, we can see that the best strategy is clearly the fourth one, filling out the lower first while checking in between each roll for scoring opportunities. While I can't promise a win against humans every time with this strategy, it will certainly perform better than the other ones! For future work, I could try adding more complex decision making to more mimic how a human would play and compare that against the strategies from this project.

Upper, No Check Between:

$$S_Z^2 \approx \frac{2.38^2 * 34.764}{49} = 4.019$$

$$\theta \in 110.104 \mp 1.68 * \sqrt{\frac{4.019}{50}} = [109.628, 110.58]$$

Upper, Check Between:

$$S_Z^2 \approx \frac{2.501^2 * 34.764}{49} = 4.43$$

$$\theta \in 159.79 \mp 1.68 * \sqrt{\frac{4.43}{50}} = [159.29, 160.29]$$

Lower, No Check Between:

$$S_Z^2 \approx \frac{2.655^2 * 34.764}{49} = 5.001$$

$$\theta \in 153.256 \mp 1.68 * \sqrt{\frac{5.001}{50}} = [152.725, 153.787]$$

Lower, Check Between:

$$S_Z^2 \approx \frac{3.203^2 * 34.764}{49} = 7.279$$

$$\theta \in 191.762 \mp 1.68 * \sqrt{\frac{7.279}{50}} = [191.121, 192.403]$$