Walter Limm

Expected Runs per Game of Across the Board Baseball

Essential Question

The Across the Board Baseball game is incredibly simple. It is a two-player game played on a miniature, wooden baseball field, with peg holes where each base would be, including home plate. The player who is at bat rolls two die. Each die correlates to a different outcome: for example, if the player rolls a 1 and a 2, they have “hit” a double. They will then move their peg from home plate to 2nd base and roll again until three outs have been reached. Once the at-bat player rolls 3 “out” outcomes, the opposing player is now at-bat. This pattern continues, each player rolling the dice and moving their pegs. A player scores a run when one of their pegs makes it all the way around the bases and back to home plate. I want to see how many runs a player is expected to score. To do this, I will code the game in Python and simulate it many times over to find the average runs per game.

Rules

Before we get into the coding methodology, we need to explore the details of the rolls and their outcomes. There are 21 different possible roll outcomes because the order of the two die does not matter (rolling a 4 and 6 is identical to rolling a 6 and 4). Among these 21 rolls, there are 9 distinct outcomes. You can walk, single, single and advance runners 2 bases, double, triple, homer, hit into a double play, sacrifice fly, and get out. The game uses many different words to describe an out-causing play, but they are all essentially the same thing (e.g., 1 and 3 is a flyout, and a 3 and 5 is a ground out, but the result is the same – 1 out). Let’s clarify the other rolls: a walk advances the batter to 1st base, and baserunners move forward if they are forced to leave their base to make space for the walked batsman (or another displaced runner). A single advances baserunners 1 base forward, regardless of whether they are forcefully displaced and the batsman advances to 1st base. A single that advances runners 2 bases is self-explanatory, and will henceforth be referred to as a “single2” roll. A double advances baserunners 2 bases and the batsman advances to 2nd base. A triple advances runners 3 bases and the batsman advances to 3rd base. A homerun scores all runners on base, as well as the batsman. The interesting rolls are the double play and the sacrifice fly. The double play outcome depends on the initial base state. If no one is on base, a double play roll simply results in one out. If there is anyone on base, 2 outs are tallied and the lead runner is the runner considered out. Additionally, all the runners behind the lead runner advance 1 base. For example, if the bases are loaded with no outs and a double play is rolled, the resultant state is runners on 2nd and 3rd, with 2 outs and no runs scored on that play. The game does not explicitly state that this is how double plays are handled, but I believe this is the most faithful approximation of the sport. Side note: originally, I had a double play always count two outs when any runners were on, and the base state remained the same before and after the play; after modifying the code to make the new base state reflect the additional out the expected run value decreased by just around 0.03 runs! The sacrifice fly outcome is much simpler. Like the double play, if no one is on base it simply counts as an out. If anyone is on base, ALL runners are moved over and an out is recorded. There are 3 walk rolls, 2 singles, 1 single2, 1 double, 1 triple, 1 homerun, 1 double play, 1 sacrifice fly, and 10 outs. Let’s now move on to the simulation.

Method

I didn’t program a two-player Across the Board Baseball game in Python, where you can run the game and a winner is decided. I instead simplified the problem– because I only care about expected run value, I can simply code the game for one player, resetting the bases every time three outs are made while only tracking a singular run counter. This streamlined the coding process because I didn’t have to worry about switching the player who is at-bat. It also streamlined the simulation process because my computer had to compute half the amount of outs. The final play() function was almost entirely comprised of helper functions. I had a function generate a random dice roll, and then a tree of if statements ensured that the correct outcome was simulated. Each outcome was a function of their own, and all accepted the base state, number of outs, and number of runs as their argument. Each outcome function was a similar tree of if statements, but instead of dice rolls being used as the parameter, I used base states. Base states were represented by a list, with [0, 0, 0] representing bases empty, and [1, 1, 0] representing runners on 1st and 2nd. Each outcome function would check the base state and then modify the base state, outs, and runs accordingly. The outcome function then returns those new parameters, and the play() function continues. To simulate 9 innings of play, I nested a while loop inside of a for loop. The condition for the while loop was as follows: while outs < 3:. I used an inequality to check the number of outs in the event there were two outs and a double play was rolled with a runner or runners on base. The outs counter would be set to “4”, so an inequality handles that case easily. The for loop simply iterated 9 times with for x in range(10) to simulate 9 innings. At the end of this, the function returns the final total runs scored. To find the average, I programmed a simple function, averageRuns(n). It takes in an integer, n, and runs the play() function n number of times. It adds together the output for each iteration of play(), and divides this by n to find the average.

Results/Comparison

averageRuns(1000000) returns an expected value of 7.570015. To be even more accurate, I ran averageRuns(10000000), which returns 7.5689275. How does this compare to real life? According to baseball-reference.com, the average runs scored per game in the 2020 season was 4.65 runs, with the Los Angeles Dodgers scoring a league leading 5.82 runs per game. Since integration in 1947, the highest MLB average of runs scored per game in any season was 5.14 runs per game set in the year 2000. Without looking to deeply into this, the year 2000 holding the highest average runs per game makes sense. The “run scoring environment” changes from year to year, and 2000 was in the heart of the “steroid era” in which steroid use was rampant. Thus the “run scoring environment” heavily favored hitters. If we ignore integration, the highest ever average runs per game was 10.47 in 1871. However, in 1871 batters had “the privilege of calling for a low or high pitch” and pitchers threw from 50 feet away, not 60 feet, 6 inches (“Baseball Rule…”). It suffices to say that the game was quite different back then.

Batting Statistics Approach/Batting Stats Primer

To better understand why the average runs per game is so much higher in the simulation, let’s imagine a player who’s at-bat outcomes are the same as our dice rolls. We can find the expected batting statistics of this player without having to do any sophisticated simulating. For our batting statistics, we will use batting average (BA), on-base percentage (OBP), and slugging percentage (SLG). BA gives the rate in which a player gets a hit. It is computed by dividing total hits by total at-bats. OBP is the rate in which a player gets on base; it is found by dividing the amount of times a player reaches base (hits + walks) by the players total plate appearances. A small distinction – a plate appearance in which the batter walks does not count as an at-bat. SLG tries to measure how often the player hits for extra bases (doubles, triples, and homeruns). It is found by dividing the total number of bases a batter traverses with total at-bats. We will also look at another stat – on-base plus slugging, or OPS. OPS is found simply by adding OBP and SLG together. OPS isn’t a perfect stat for many reasons, most significantly because it weights OBP and SLG equally despite OBP being slightly more offensively valuable. SLG is also a fundamentally flawed stat as it weighs hits linearly and this weighting system isn’t perfectly accurate.(e.g., SLG says that a triple is worth 3 times as much as a single because it is worth 3 times as many bases). However, despite all of its flaws, OPS does give a good overall idea of a player’s offensive value. OPS+, a version of OPS that normalizes the OPS of all major-leaguers on a scale of 100 while also adjusting for the impact of playing in differently shaped ballparks, highly correlates with wRC+, or weighted runs-created plus (Slowinski). wRC+ is hitting statistic that doesn’t treat OBP and SLG equally and uses a more sophisticated system of weights, resulting in “the most comprehensive rate statistic used to measure hitting performance” (Slowinski). This is all to say that looking at the OPS of our imagined player is a fairly accurate way to judge their offensive value. Another point – a “good” OPS varies based on the run-scoring environment. For example, an OPS of 0.600 might’ve been great in 1900, but is terrible now. For our current run-scoring environment, thinking about OPS like letter grades is a convenient way to assess value. An OPS over 1.000 is an A+ grade. 0.900+ is an A, 0.800+ is a B, 0.700+ is a C, 0.600+ is a D, and 0.500+ is an F. There are few hitters with OPS’s any lower than .500 because major league teams will often cease giving these hitters playing time. This variance is why stats like OPS+ and wRC+ are better; they scale every player’s OPS to the league average OPS. An OPS of 120 means that player was 20% better than league average in the OPS statistic. Unfortunately, this isn’t possible in this case as there is no league average to scale with.

Expected Batting Statistics

Let’s get back on track. A player who’s batting outcomes match the dice outcomes would have 21 plate appearances, one for each dice roll, and 17 at-bats (the difference of 4 is because walks and sacrifices don’t count as at-bats). 6 hit outcomes – 3 singles, 1 double, 1 triple, and 1 home run – gives our batter an BA of .353. This player gets on base 6 times from those hits and 3 more times from walks totaling to an OBP .428. Those 6 hits add up to 12 total bases, giving a SLG of 0.706. This gives our player an OPS of 1.129, which is elite. To put this into perspective, only one player has a higher career OPS than this (with a minimum of 3000 plate appearances). This is Babe Ruth, who posted a gaudy 1.1636 over the course of his career (“Career Leaders…”). Ted Williams comes in a close second with 1.1155 (“Career Leaders…”). This makes the ridiculous runs per game make sense; if a team’s lineup consisted of 9 of baseball’s all-time greats, one would expect a high-scoring game.

Conclusion

The scoring in Across the Board Baseball is far higher than real, post-integration baseball. The expected runs per game of around 7.56 was found by writing a program that played the game for one player and simulating that game 1 billion times to find the average score. This higher scoring rests solely on the offensive-inclined dice rolls - a player who hits like the dice outcomes for is comparable to Ted Williams. I believe this is because the dice rolling system necessitates simplicity. To approximate baseball, each basic outcome needs to be included for completeness, and with only 21 outcomes the simulated batter is going to have decent stats. With the rule that each hit needs to be included, our worst possible batter has 4 hits, one of each kind, and 1 walk in 21 plate appearances. I simulated this set of outcomes, and the runs per game over 1 million games was 3.54, below the 2020 average. This player has a BA of .200, OBP of .238, and SLG of .500, resulting in a solid OPS of .738. In 2020, 2 players had an OPS of .738, ranking 97 and 98 on the OPS leaderboard (“MLB Stats:…”). These players are the AL Rookie of the Year Runner-up, Luis Robert, and 5-time All-star, 4-time Silver Slugger award winner, Nolan Arenado. While this runs per game number was much closer to the 2020 average than the ludicrous 7.56, I think it would make Across the Board Baseball far more boring. This emphasis on offense is a way to keep the game engaging, especially for those unconcerned with realism. If I fantasize playing baseball, I don’t picture myself hitting like a slumping Nolan Arenado, I see myself slugging homers like Babe Ruth – Across the Board Baseball allows me to approximate this fantasy.

Works Cited

“Baseball Rule Change Timeline.” Baseball Almanac. Baseball Almanac Inc., n.d. https://www.baseball-almanac.com/rulechng.shtml.

Baseball-Reference. “Major League Baseball Batting Year-by-Year Averages”. Sports Reference. Web. 11 Nov 2020. <https://www.baseball-reference.com/leagues/MLB/bat.shtml>

Slowinski, Steve. “WRC and WRC+.” wRC and wRC+ | Sabermetrics Library. Fangraphs, February 16, 2020. <https://library.fangraphs.com/offense/wrc/>.

“Standard Stats: Glossary.” *Major League Baseball*, MLB, m.mlb.com/glossary/standard-stats.

“Career Leaders & Records for On-Base Plus Slugging.” *Baseball Reference*, Sports Reference, [www.baseball-reference.com/leaders/onbase\_plus\_slugging\_career.shtml](http://www.baseball-reference.com/leaders/onbase_plus_slugging_career.shtml).

“MLB Stats: Baseball Stats.” *MLB.com*, MLB Advanced Media, www.mlb.com/stats/?page=4.