FINAL PROJECT

Baseball Game Discrete Event Simulation

Justin Doyle

MSDS 460 DL, Spring 2024

Northwestern University, Decision Analytics

5/30/2024

Abstract

Using sports as a general springboard for this project, the aim was to develop a discrete event simulation to model the changing states within a given game using Python and the SimPy library. After landing on baseball as the sport of choice, we began by defining the scope and end goal of the simulation. There are levels of complexity that could be achieved with this project, as the scope could range from batter-to-batter all the way to season-to-season simulations, but based on the timeframe and depth of the code, we settled on conducting the simulation of a singular game. By leveraging simple statistical measures, like Batting Average, On Base Percentage, and Slugging Percentage, we would, in theory, be able to simulate the batter-to-batter results of each inning and subsequently the game overall. The simulation framework is designed to capture the intricacies of a game and the dynamic changes that occur in score, outs, and base runner positions with each at-bat. Additionally, we are able to maintain an event log that allows us to access the results of each individual state and identify where changes in the game state occur. This project demonstrates the applicability of discrete event simulation in sports analytics, providing insights into in-game decision making and potentially player performance evaluation with further development of the source code. As the scope of the project expands, it allows for more potential for exploring "what-if" scenarios to inform game strategy and roster construction. However, in its current state, the project is simply an introduction into the way discrete event simulation can be used to model a game on a very basic level.

Literature Review

While there were a number of resources available that delved into the intricacies of discrete event simulation itself, perhaps the most valuable article for this project was a piece from Matt Hunter, founder of SaberSim. In his article, "10 Lessons I Learned From Creating a

Baseball Simulator", he covers a number of considerations that provided answers to many of the initial questions for this project. He notes that, despite the number of inputs that can actually be included, baseball is a simple game where, at any point, it can be defined by a few states: inning, outs, baserunners, personnel, and score (Hunter 2014). Because of this, it is very simple to create base level code and it is possible to maintain event logs that allow for analysis of any given state within a game. Despite this, it is also possible to expand and include more detailed inputs. Within the scope of this project, we only consider batter statistics and not pitcher statistics, because there is a relationship between the two that is far more difficult to capture. Additionally, there are other considerations, such as weather, stadium, umpires, handedness, fielding, etc. that can be included in the modeling, but require far more knowledge and expertise in the structure of the code to implement. Hunter's findings drove this project in the right direction and provided insights that allowed us to refine the overall scope.

Research Design and Modeling

Before we began our modeling process, we needed to establish the inputs that would be considered in the discrete event simulation. While there are a number of potential factors, such as pitching statistics, environment, and game leverage, we wanted to begin with the most basic model. This includes batter statistics, like batting average, on-base percentage, and slugging percentage. Additionally, we determined that the states that we would track would include outs, base runners, and score. Conducting this project on a pitch-by-pitch basis would require far more computational power and would expand beyond our established scope.

We decided to employ the use of a Large Language Model in order to assist with the structure of the code. Throughout the course of the quarter, we have seen the capabilities of these LLMs in the solving of various optimization and modeling problems, like the Diet Problem and

the Knapsack Problem. In this case, we can utilize these tools in order to solve the issue at hand, which in our case is creating a discrete event simulation of a baseball game. Part of this process is understanding what needs to be prompted of the LLM in order to achieve the optimal end result. This includes fully understanding all necessary inputs, like player statistics, game states, game flow, rules, and anything else that replicates a real-life game within the scope that we have chosen. After rounds of trial and error, we eventually landed on a model that we felt comfortable with in terms of applicability to the problem statement.

Within the structure of the code, we utilize two packages, SimPy and Random, that help us carry out the simulation. Additionally, we created three classes that serve distinct purposes. The first class is “Player”, which initializes each player within both lineups and their respective statistics. The second class is “Team”, which establishes the respective lineups and also initializes the queue for each lineup. This is key in the repeating nature of lineup turnover and ensures that players bat in their proper order. The final class is “BaseballGame”, which actually carries out the flow of the simulation with many of the rules and assumptions that we have set forth. This includes inning flow, at-bat results, assumptions of hit probabilities, baserunning, scoring, and the logging of outcomes. Essentially, this is what takes us from state to state within the game, and is the core of the simulation itself. As an example, we used the current lineups of the New York Yankees and the Boston Red Sox for our lineups. We pulled real player data as of March 29th, 2024. After each run of the simulation, we receive a game log that shows us each state that occurred within the game.

Results, Implementation and Future Considerations

The results of the model proved to be very promising for further research. We were able to achieve a simulation that took player batting statistics and produced scores that were feasible

in the context of a typical game. Scores were often within reasonable ranges, and the game logs showed that the games followed a realistic flow.

For next steps, we can take this further and run the simulation a large number of times and view the distribution that exists. This includes score distributions, game outcomes, and player outcomes. We would expect scores to follow a Poisson distribution, as scoring can be considered a series of rare, discrete events occurring independently over time. Game results would follow a binomial distribution, with the assumption that ties can not occur. Finally, on a basic level, player outcomes can also be modeled with a multinomial distribution, since we are assuming that the outcomes are not just safe or out, but rather single, double, triple, HR, walk, etc.

These distributions and game logs can allow us to analyze probable game outcomes based on the outcomes of each state within a game. For example, we can determine the win probability added of a bunt with no outs and a runner on first or a walk with two outs. These insights allow us to better manage the decisions within a game as well as the overall construction of a roster based on what players are adept at. We can run these simulations to uncover any potential holes within a lineup.

Moving forward, we would also be interested in creating a more complex model that captures the reality of a game better, like pitching statistics, additional outcomes, environmental factors, etc., but this is certainly a good base that allows us to expand in any given direction.

References

Hunter, Matt. “10 Lessons I Learned from Creating a Baseball Simulator.” The Hardball Times, April 28, 2014. https://tht.fangraphs.com/10-lessons-i-learned-from-creating-a-baseball-simulator/.

Stark, Steven, and Deepankar Sinha. “How Can You Use Simulation Modeling to Improve Sports Analytics?” How Simulation Modeling Enhances Sports Analytics, November 7, 2023. https://www.linkedin.com/advice/0/how-can-you-use-simulation-modeling-improve-lzxde.

“A Gentle Introduction to Discrete-Event Simulation.” Software Solutions Studio, January 6, 2024. https://softwaresim.com/blog/a-gentle-introduction-to-discrete-event-simulation/.

Beazley, David M. 2022. *Python Distilled.* Boston: Addison-Wesley/Pearson Education. [ISBN-13: 978-0-13-417327-6] Chapter 6. Generators, pages 139–152. Available in Course Reserves.

Buss, Arnold H. 1996. "Modeling with Event Graphs." In Proceedings of the 1996 Winter Simulation Conference. John M. Charnes, Douglas J. Morrice, Daniel T. Brunner, and James J. Swain (eds.), Washington, DC: IEEE Computer Society, 153-160. https://dl-acm-org.turing.library.northwestern.edu/doi/pdf/10.1145/256562.256597

Hillier, Frederick S., and Gerald S. Liebermann. 2021. *Introduction to Operations Research* (eleventh ed.). New York: McGraw-Hill. [ISBN-13: 978-125987299-0] Chapter 20. Simulation, pages 866–912.

Ingalls, Ricki G. 2008. “Introduction to Simulation.” In Scott J. Mason, Ray R. Hill, Lars Mönch, Oliver Rose, Thomas Jefferson, and John W. Fowler (eds). *Proceedings of the 2008 Winter Simulation Conference,* Miami, FL.17–26.

Menner, William A. 1995. “Introduction to Modeling and Simulation.” *Johns Hopkins APL Technical Digest.* 16(1): 1–17. https://www.jhuapl.edu/Content/techdigest/pdf/V16-N01/16-01-Menner.pdfLinks to an external site.