Modeling Baseball Games and Measuring Player Performance Using Monte Carlo Simulations

Hamzeh Hamdan

APM 115: Mathematical Modeling

May 2024

Abstract

Baseball analytics is becoming increasingly important within the sport. Despite this, some leagues only report match and league statistics. This makes traditional analyses that require coordinates of the ball impossible. Using player-level league-averaged reported data that included 20 traditional statistics, we simulate a baseball match using Monte Carlo simulations. Analyzing the top national high school for baseball, Corona High School, using the simulations, we find that their lineup does not outperform a randomly-generated lineup significantly, that their recruitment of the Ebel brothers did not significantly increase runs, and that a multilinear regression on simulated data is inconclusive.

Contents

1 Introduction

Since the release of Moneyball in 2011, there has been a large increase in the use of data analytics to estimate the value of a player. Often, players' performance is reported through one-dimensional statistics—which have evolved over the years—that are over-simplistic. While more advanced methods can be employed using play-by-play data, this data is usually only available for college or professional players.

This poses a problem for high-school players, where players only have one-dimensional statistics available to represent their performance. This holds for the best players in the nation as well. Take brother Brady and Trey Ebel, for example, who both played on the 15U USA National Team and are projected to be signed into the Minor Leagues out of high school. Despite attending Corona High School—the number one rated high school in the US—the brothers don't get play-by-play data or a quantitative sense of how their statistics impact the overall game; due to the complex nature of baseball, the order of the lineup (including who hits before or after a player) impact the their own reported statistics.

This project aims to model the overall impact of switching players using only the statistics reported for high school students on MaxPreps, the standard software used for publishing player and game statistics for high school and college students.

Using this model, we will measure player performance impact by estimating how much the team would be impacted if a player were replaced by another player, if the current lineup of a team outperforms randomly generated lineups, and generate synthetic datasets to estimate the overall effect of team-wide statistics on a team's performance.

Note that while MaxPreps provides extensive data on traditional statistics for every player, they do not include the batting order in their match reporting. This causes an issue when trying to simulate matches. In a call with Brady and Trey Ebel, they reported Corona High School's typical batting order, claiming that the team rarely deviates from it within and in-between matches. This project will focus on validating using Corona High School, as they were willing to answer questions should I have any questions that would help with the model.

Lastly, this project will use, as a tool, GitHub user kevjiang's repository on finding the optimal lineup order using Monte Carlo simulations. While the repository is very out of date (its latest updates are 5-9 years ago) and required a lot of modifications, it had a very great backbone that simulated matches well and had already integrated the logic behind the statistics. I updated the code's syntax and resolved outdated errors, rewrote some parts of the logic because of a discrepancy between the statistics the code used and those provided by MaxPreps, and changed portions of the code to allow for the match scores to be returned rather than just the summary. The initial code was around 800 lines and this was a lengthy, but rewarding, process.

2 Modeling Approach

2.1 Methodology

The aim of this project—to model the overall impact of switching two players—can be used to estimate the value of a player over another, but can also be used to better understand the importance of specific metrics on overall team performance.

Using Monte Carlo simulations that return the number of runs, we can compare two players by measuring the Kullback-Leibler divergence between the distributions returned by the simulations. We can also generate data where we slightly shift some statistics (one at a time) of a singular player on a team and fit a multilinear regression, aiming to understand the importance of specific statistics

for a specific player on a team; this isn't necessarily consistent across players as their position in the lineup and own skill set might make some statistics more important than others. A multilinear regression is used because of the large number of statistics reported (20) and its computational simplicity; note that some statistics might be excluded if they are functions of others to ensure independence of the independent variables.

2.2 Code Review

The Monte Carlo simulations are grouped by season, with each season including 100 simulated matches. For each match, we use the real-world season-averaged statistics to simulate an outcome for the number of runs scored. The code is organized in four Python classes: players, games, and seasons. The players class structures the data retrieved from the input CSV file (with data from MaxPreps), the games class has methods to simulate innings and handles positioning of players, and the seasons class simulates an entire season and returns the data from the season.

During the player initialization simulation, we estimate the outcome probabilities for each state (a single, double, triple, home run, out, etc.) from the observed data; these are then easily called during the simulation of each inning in the game. The data that was used was from the most recent season Corona High School played in.

Using the code involves having a CSV file of the right structure. From there, you can load the lineup with "csv_to_lineup(filepath)" and the simulate the scores using "season_scores(lineup, iterations)". The season scores function was one I implemented, as previously the code would run the simulations (after fixing the outdated errors) and return optimal lineups.

2.3 Simulations

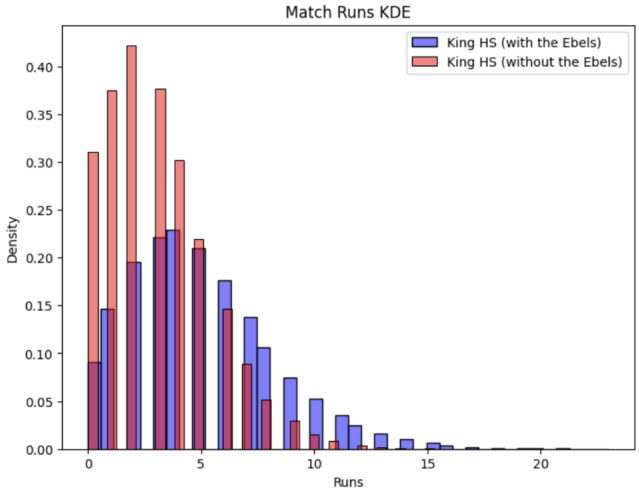
Once the simulation was set, most of the remaining mathematics relied on the Law of Large Numbers, which lies at the heart of Monte Carlo simulations; the standard number of simulations per cell was 40,000 matches (400 seasons, 100 matches per season). We ran simulations and estimated the empirical density functions of the returned match runs.

When attempting to understand the importance of specific statistics, we generated 400 bootstrapped datasets with normally distributed noise with a variance uniformly distributed from 0 to 0.15 times the observed mean for each statistic. For each of these, we ran 40,000 simulated matches (with the same breakdown as above, yielding a total of 16,000,000 matches). For each of the 400 datasets, we then calculated the average deviation from the original dataset for each statistic and the mean and standard deviation of the resulting distribution of scores. Although the fitted OLS regressions of the average deviations on mean and std yielded very high R-squared values (0.960 and 0.914, respectively), the standardized non-zero coefficients had inconclusive explanations. The residuals were checked for heteroskedasticity.

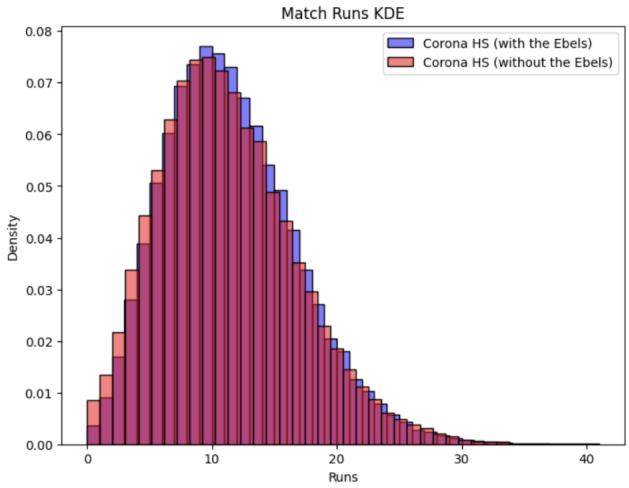
3 Analysis

There were several methods of analysis using this model relevant to recent events concerning the Ebel brothers, who have recently switched schools to join Corona High School. The first exploration was the marginal benefit Corona added to their team by recruiting Brady and Trey, compared with the benefit a lower-performing school in the same league would have gained from the recruitment.

For King High School, a recruitment of the Ebel brothers would have severely improved their game. Their average runs would have increased their average runs per game from 3.05 to 4.89. Additionally, their probability of scoring more than 5 runs per game would have increased from 14.9% to 37.22%. Additionally, their probability of scoring 0 would have decreased by more than 70%. See figure below for the distribution.



Despite this being a huge loss for King High School, Corona's adoption of the Ebel brothers did not severely increase its runs. In fact, it only increased the average runs per game by 0.47 (from 11.01 to 11.48). See figure below for the distribution.

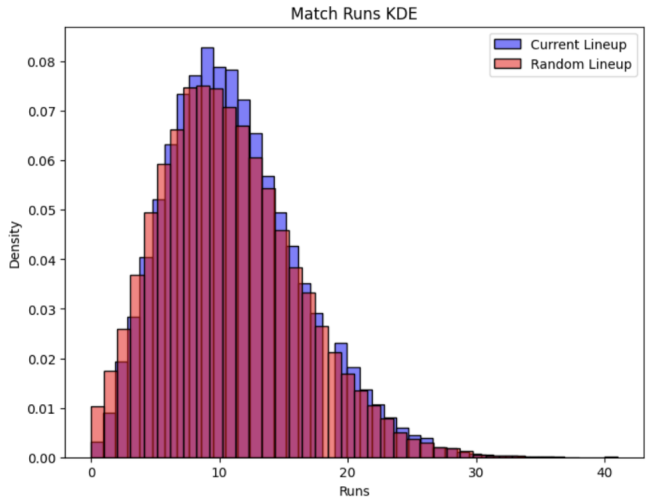


This very slight decrease revealed that Corona High's team is composed of the best talent; thinking of other ways to optimize their runs, they could also try a different batting order. We were curious to see how much better their current batting order was over a randomly generated one.[1]

In order to test this, we ran 400 one-season simulations, where each season had 100 matches, twice: the current lineup was used in the first set of seasons and a randomly generated lineup was chosen for each season in the second set. If it doesn't seem intuitive, the lineup order *should* matter in a baseball game as you can optimize for having specific types of players play after each other (a naive example would be to spread out frequent homerun hitters to maximize the number of loaded bases).

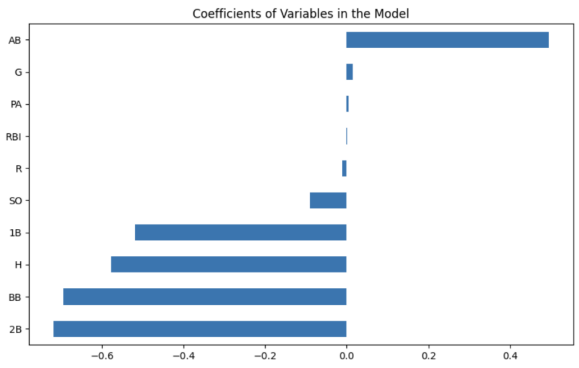
Our results, depicted in the figure below, were quite interesting; on average, a random lineup resulted in a 0.01 increase in the number of runs. This suggests that the lineup is not well optimized, and presents a great opportunity for the team.

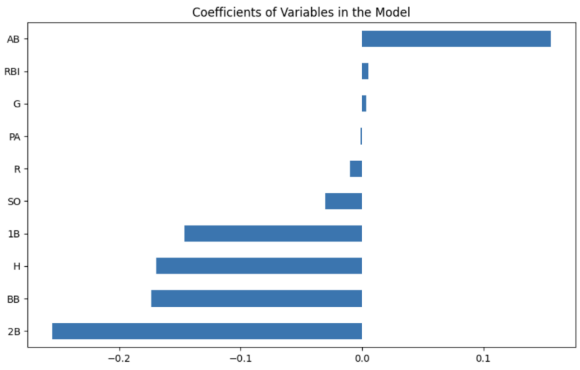
[1]This was slightly inspired by Professor Brenner's pandemic story; who knew the results would be similar!



Lastly, in an attempt to better understand the importance of predictors, we simulated noise in the data and generated 400 datasets. Now shifting to an analysis of the effect of changes of specific statistics across the entire team, we find two statistically significant variables in the OLS analysis: number of hits and the number of singles, both of which are negatively correlated with the average number of runs within the simulations.

While it might seem counter-intuitive, we must consider these holding all other statistics constant. If the number of hits was to increase but all other variables were the same, it could signify a lower skill level (since the team would have taken more failing shots, yielding a lower accuracy). Similarly, if the number of singles increases, but all other variables are constant, this could signify a larger failure rate at benefiting from those opportunities. The results from the other variables are shown below on the left;[2] note that this is standardized since the descriptors are the average change in a statistic after adding noise, and the noise was dependent on the scale of the variable.





Running the regression on the standard deviation of the data (see figure on the top right), we find statistically significant results for 1B, H, and AB. The only significant increase in variability is AB (at-bats), which signify the number of times in which players went to bat; this matches our intuition as plays have many possible outcomes, and thus increase variability in the statistics. The other two were shown to decrease, as above, variability.

[2]Statistic Definitions: AB: at-bat, G: games played, PA: plate appearance, RBI: runs batted in, R: runs scored, SO: strike outs, 1B: singles, H: hit, BB: base on balls (walk), 2B: doubles, AB: at-bats.

4 Discussion

The essence of the model is intuitive, and the results are generally intuitive. By simulating the matches, we can put a quantitative definition on intuitive trends and discover others. The balance between exploring known trends for validation and discovering others is an important one within emerging sports analytics, especially since the coaches are usually skeptical of its suggestions. This project produced some intuitive results and some, at least to me, surprising ones (like that about the lineup), helping us get a sense of validation of the model and extract other useful insights.

The main limitation of this model is that it is only capable of reporting results averaged over the season as the data is too granular on the match level. As such, many of the results do correspond, in some way or another, to the distribution of average runs. Additionally, the limited available data makes it hard to accurately validate the model, especially given the one-sided nature of the simulations. There are two potential extensions of this project: 1) checking empirical WAR (wins above replacement [player]) statistics for players using the simulations, and 2) modeling games using these simulations and testing for an error bound in Leagues with more data. This method would inherently be subject to error as these simulations only model the number of runs averaged over the season, rather than an interactive offense and defense set of actions.

Note: I know the rubric mentions commenting on why the behavior of the model makes sense and contextualizing results. I avoided repeating this from the Analysis section in order to be more concise.

5 Collaboration/Acknowledgements

I also used GPT4 throughout the coding portion of this project. It was very helpful in dealing with pandas dataframes and using numpy to optimize for speed. Towards the beginning of the project, I also used it to get a better understanding of the reported baseball statistics (believe it or not, I just watched my first baseball game a few months ago).

On the note of collaboration, I attempted a few times to find someone to help, especially when it came to 10-K filings data. I tried to set up the Google Sheets to find people to help during that one class day where attendance wasn't required, but there weren't many students and it didn't pick up well.

6 References

"Corona High School." MaxPreps, www.maxpreps.com/ca/corona/corona-panthers/.

"GPT4." OpenAI, openai.com/chatgpt.

Jiang, Kev. "Optimizing Baseball Strategies through Monte Carlo Simulations." GitHub, github.com/kevjiang/baseball-sim/tree/master.