Where Should the Strongest Batter Bat? A Simulation Study

Michael Kalmus

INTRODUCTION

In Major League Baseball, the batting order or lineup defines the sequence in which each player bats during a game. Usually, the order is set by the team's manager, and the batting lineup is an integral part to any team's strategy. Consequently, the spot where each player is placed within the lineup has a large effect on his opportunity and ability to affect the team. For example, the player who bats first (referred to as the "lead-off" hitter) naturally will have the most opportunity for plate appearances given the batting order picks up where it left of when innings change. However, the players who bat second and third in the lineup have nearly as many plate appearances with more opportunities to drive in runs because more runners may be on base. Intuitively, it would make sense to place the strongest batters earlier in the lineup and the weakest batters late in the lineup; however, fans have seen many variations of batting orders over the years [1] which contradict both statistics and intuition. In this paper, batter ability is reflected in terms of batting average (BA), which is found by dividing the total amount of hits for a player by their plate appearances.

While some team managers prioritize the fastest player as the lead-off hitter, others view on-base percentage (OBP) as the most pivotal statistic. Similarly, recent years have seen many teams change their second batter from relatively average batters to the best players in the league. In fact, in the 2015-2019 seasons, fans have seen players like Mike Trout, Kris Bryant, and Martin Prado in the second spot, each with high batting averages [1]. Traditionally, these players would have batted fourth in the lineup because the probability of having a runner already on-base is highest when the fourth batter appears to the plate [2]. This statistic is the basis for yet another method of placing a team's best or most consistent batter in the fourth spot as his probability of gaining runs for the team is highest when the most possible runners are on base.

Overall, the heuristics employed in creating a baseball lineup to maximize expected runs have greatly changed both in recent years and over the course of the MLB's existence. More interestingly, some of these heuristics violate traditional intuition, which would tell a manager to always put the best hitter at the spot in which the most runners are on base, which statistics show is the fourth lineup spot. Thus, this paper aims to study the location within the lineup that teams should place their best overall hitter and to evaluate if the recently popular method of placing the best overall hitter in the second lineup slot is valid.

This study takes a simulation approach due to the ease of simulating different "what-if?" scenarios: for example, simulation allows to easily see the outcomes of games with the same players under different batting orders. This is notable because some hypothetical situations may scarcely occur in practice, like placing the best batter towards the end of a lineup. However, these situations can be simulated with relative ease. Moreover, simulation provides a framework to analyze future research questions. In the future, I hope to use this framework to study the validity of new emerging strategies such as the Yankees using multiple relief pitchers and to use the open-source simulation to encourage improvement and collaboration from others.

To model baseball games with a given batting lineup, I take a Markov chain modeling approach similar to other researchers, where the state of the game at any given at-bat can be classified in terms of the runners on base and the number of outs. Using historical play-by-play data for the 2015 season, I first compute transition matrices for each type of hit to show how runners advance for a given hit. Then, I simulate a finite amount of games (optimal number of replications is different for each lineup scenario) using the hit probabilities and transition matrices assuming that the probability a batter hits is related to his batting average. Lastly, I calculate the amount of runs scored in each game and repeat the process for different scenarios in which the player with the highest batting average hits in all nine spots of the lineup, with the base case being him in the second spot. To analyze the different systems, I use a paired-t test to construct confidence intervals for the difference in runs between all systems and the standard case and conclude that no batting position for the best batter is better than the standard for a team whose batting skill is similar to the 2018 Yankees, a team that has found success placing its best batter second.

BACKGROUND

Baseball and baseball statistics have been widely explored topics in recent years, largely due to the significant effort by a few individuals and organizations discussed later in this section and paper. In fact, an entire discipline called Sabermetrics has been created to denote the statistical study of baseball [4, pp. ix]. Due to the wealth of publicly available baseball data, literature on baseball statistics in general is extensive: in academic journals [5], blog posts by respected baseball statisticians [2, 6, 7], and books [4] alike. However, most baseball studies take a statistical analysis approach rather than a simulation approach, and the lineup problem is not well-discussed in peer-reviewed academic journals: in fact, searching the University of Michigan Library (which searches through all catalogs, articles, databases, and journals that the University is subscribed to including the Journal of Quantitative Sports and Google Scholar) yields less than 10 articles published between 2009 and 2019 that study the lineup problem [8, 9, 10].

Of the peer-reviewed articles that study the lineup problem, many focus on creating an optimal lineup rather than analyzing the validity of recent lineup heuristics as studied in this paper. For example, Hirotsu and Bickel [8] studied optimal lineups with a focus on little-league baseball games with run limits (where all rules are the same except a run limit exists on how much each team can score per half inning). While their methodology presents a concise description of the Markov Chain Monte Carlo (MCMC) simulation, their results are not applicable to the current study due to the wide range of skillsets present in little league games that is not present in the MLB. Other researchers that have studied the lineup problem take a dynamic programming approach as in Norman and Clarke [9] and Inakawa and Fujita [10]. In their paper, Norman and Clarke study the lineup problem for games of Cricket. Though the game of Cricket is naturally different than baseball, they present a dynamic programming approach that can be and has been applied to baseball games. In fact, Inakawa and Fujita apply a similar dynamic programming approach to find optimal baseball lineups. This approach contains some of the same methods used in this study, largely that baseball is a game with a finite amount of states and that there exists a probability that can be calculated of changing to each state. They consider many complex states, including stealing bases, intentional walks, and sacrifice bunts and interestingly conclude that the difference in win percentage between the best and worst batting orders for a given team was only 2.35%, meaning that creating an optimal batting order has little impact on

whether a team will win [10]. In Hirotsu and Bickel [8], the researchers reach the same conclusion when studying little league baseball with run limits.
In researching older publications, Bukiet [11] was one of the first to take the Markov Chain approach to baseball. In the article, the authors discuss key concepts that are still present in modeling baseball games today including the calculation of a transition probability matrix and the calculation of expected runs given a team with varying skill levels. An interesting conclusion made in this paper was that the optimal lineup maximizes both win probability and expected runs. In-line with the intuition described in the introduction, they also found that both the best and second-best batters should bat either second, third, or fourth. As with other studies, they found that optimal batting orders compared to the current state produces only 4 more wins in a 164-game season. This paper differs from the current study in that the authors do not focus explicitly on the strongest hitter. Furthermore, the paper does not fully focus on the lineup problem: the purpose of the paper was to illustrate the use of Markov Chain modeling in baseball, and as such, the authors study optimal lineups, runs score per inning, as well as the effect of trades in the study. A notable limitation to this paper when using the methodology in current-day baseball is that the authors base the current lineups off the 1997 lineups. As mentioned in the introduction, the heuristics of lineups have changed many times throughout the years, so the comparison to the baseline would be different. Since the publication of this paper, the Markov chain approach has been used to study other baseball problems including win and loss streaks over time (i.e. a season)[12].

In exploring the web, one can quickly learn that a much of the current work in baseball statistics and analysis comes from a few individuals in the form of blog posts. Earlier in this paper, blog posts from Tom Tango [6] and a book co-authored by Jim Albert [3] were referenced. Tango and Albert are subject-matter experts on baseball statistics and are widely respected as analysts: the book co-authored by Albert [3] provides hundreds of pages relating to the statistical analysis of baseball and includes discussions ranging from the available public data to calculating run expectancies and win probabilities. More importantly, this book provides a framework to calculate many of the necessary items for a simulation or Markov Chain analysis that were used in previously discussed papers; however, the simulation presented does not account for runner advancement by hit type nor different lineups [3, pp. 201-215].

Additionally, literature pertaining specifically to the simulation of different baseball lineups is scarce, as most simulations are not open source as they are sold or packaged as video games (i.e. the popular videogame MLB The Show). Of the open source simulations, many are not validated, use impractical assumptions, or do not study the lineup problem [13, 14]. In his blog which pertains specifically to analyzing baseball with R, Jim Albert discusses his methodology in simulating a half-inning of baseball in R [13]. In this study, Albert discusses concepts necessary in any simulation, some of which were discussed earlier such as calculating transition probabilities. More notably, he goes farther and finds baserunning probabilities for a given hit: for example, if a single is hit, his method uses historical data to find the probability that a on first, second, or third base will move to any other base. While Albert discusses essential elements in any baseball simulation, the study is limited in that it only simulates a half-inning of baseball, and more importantly that it does not take into account the abilities of different players. Thus, it provides a great overview of half-inning simulations, but makes an unrealistic assumption that each player bats at the same skill level (i.e. a pitcher has the same probability of hitting as the team's best batter).

Similarly, in 2018, Randall Olson studied the lineup problem using a simulation approach in Python [14]: like studies in the previously discussed peer-reviewed articles, Olson does not specifically study the validity of recent lineup changes but rather aims to find if creating an optimal lineup matters at all. Like Albert, Olson uses league probabilities for a full season to find state transition probabilities. His simulation is interesting because rather than optimizing a lineup using methods like dynamic programming, he instead simulates many different batting scenarios for 9-inning games. This approach differs from the previously discussed simulation in that Olson does not assume players have equal skill, but instead uses both batting average and league probabilities to calculate expected runs; however, his model does not account for runner advancement given each hit type. Furthermore, he only studies teams where eight of the nine players bat at the same level, and the ninth player is far worse or far better than the team, which is not true in practice as MLB players have differing batting averages.

Overall, current literature on baseball statistics is extensive, but literature on the lineup problem – especially in peer-reviewed journals – is scarce. Furthermore, many past and current studies that study the lineup problem use approaches different than simulation, and within the open-source simulations, many make assumptions that are not true in practice. Furthermore, the discussed open-source simulations do not account for both runner advancement and batting lineups. As such, this study aims to analyze the positioning of the best hitter, similar to Olson's simulation, but with a model built from the ground-up in R and that accounts for both player ability and runner advancement given a hit type.

DATA

A key reason why baseball statistics is such a widely studied subject is because of the immense amount of publicly available data. In this section, we outline the data sources and some third-party functions utilized in this simulation. In building a simulation to address the research question, we must have data that allows for the calculation of state transition matrices, the calculation of hit type probabilities, and individual player batting statistics.

The integral data source for this study, as well as almost every other statistical baseball study conducted in recent years, is RetroSheets [15]. RetroSheets is an organization dedicated to the collection and preservation of baseball data, and as such, the organization's website contains play-by-play data for every baseball game since 1921. It also provides summaries of individual games dating back to the advent of the MLB. For this simulation, we focus on Retrosheet play-by-play data for the 2015 season, which is used in calculating transition matrices, hit type probabilities, and true runs scored (for comparison with simulated results). 2015 was chosen to be able to compare preliminary results with statistician Jim Albert, but the simulation can take inputs for any season.

Raw RetroSheet data is extremely difficult to understand and work with; however, Jim Albert created an open-source function to download and parse the data for a given season which can be loaded in R using the Devtools package [16]. His function takes advantage of software created by the Chadwick Bureau [17], which formats the Retrosheet data and adds appropriate column labels to all 97 columns. Albert's function was used for this study due to its compatibility with Mac operating systems. To describe the data generally, each record is one play of a game. By using the column called EVENT_ID, we can identify every single action in any baseball and thus calculate the transition probabilities for a given season.

Using RetroSheet data, one can compute the state of any baseball game over time and thus have all the required data to create a basic simulation which can be built upon in the future. To add individual player batting averages, the presented model uses data from baseball-reference.com [18]. The following sections outline how we use the discussed data to build a simulation model.

METHODS

Baseball as a Markovian Game of Discrete Events

The key assumption made in this simulation is that baseball is a game of discrete events for which the state of a game at any time can be classified using the number of outs and the runners on base. Since there are three bases which can be occupied or not (home plate is only for fielding and batting), each base has two states. Furthermore, at each at-bat, there can be either 0, 1, 2 outs. In total, this means the game would have 24 states, but an absorbing 25th state exists when the batter hits a third out [3, pp. 212-214]. After the 3rd out, the half-inning terminates, the batting team switches with the fielding team, and the state resets. Further notable characteristics also exist, such as the number of runners on base and runner advancement per hit (i.e. if a single is hit, what is the probability a runner on first goes to second? Third?) Because each state has a probability that can be calculated, the expected runs in a given inning by a given team can be found using state transitions from a starting state to an ending state [3, pp. 105-128] and transitioning for nine innings of three outs.

Data Preparation for Transition Matrices

To acquire transition matrices, Retrosheet play-by-play data is parsed using the methods discussed in the Data section of this paper (Chadwick Software and parsing function), and each record of the play-by-play data is characterized in terms of its state. The state characterization used takes the form of "ABC D," where A, B, and C denote whether or not a runner is on first, second or third respectively, and D denotes the number of outs at a given at-bat. For example, a state of "101 2" shows runners on only first and third base with two outs. Following logic in Marchi and Albert's book [3, pp. 111-134], I find for each record: 1) the inning in which the event occurs, 2) the state at the time of the event (runners on each base and number of outs), and 3) the new state after the event. The dataframe is then filtered: events in which the state does not change are ignored and only full half innings are considered (total outs in the inning equal to three). Lastly, the state with three outs is simplified to not include runners on base since it is an absorbing state that causes the next state to reset.

Building the Simulation

In Chapter 9 of their book [3, pp. 201-221], Marchi and Albert describe programming a simulation but the simulation discussed is limited in that it accounts neither for lineups nor player advancement for a given hit type. The presented simulation used the discussed logic as a base and uses the presented code to find states for a given at-bat; however, this simulation goes further in order to analyze different lineup scenarios while accounting runner advancements given a type of hit.

First, transition matrices for runner advancement are calculated following similar logic presented by Albert in [13]. A notable difference is that Albert hard-codes each entry to the matrices, where this simulation calculates the appropriate entries based on Retrosheet data; for this reason, the presented simulation is more flexible as it can get transition probabilities for any season where Albert's logic pertains only to 2015. To validate the logic in this simulation, the matrices for 2015 hit transitions were compared to Albert's to ensure the probabilities were the same.

An assumption in this model is that runner advancement transition matrices only need to be found for singles and doubles because it is assumed that home runs always empty the bases and triples always result in only a runner on third. In terms of the types of event, the model only considers singles, doubles, triples, home runs and outs and assumes player movement is independent of the number of outs. Thus, some events such as errors, stealing, and bunting and their run-generating potentials are not accounted for in this iteration. Additionally, some matrix values are manually created to have a probability of 1 or 0 due to this assumption: for example, in reality it is possible to go from state "100" (runner on first) to state "100" (runner still on first) if the batter hits a single and the runner who was previously on first base is thrown out while running to second base: this case is referred to as a "fielder's choice" in baseball. Since the model does not account for fielder's choice, it assumes this case and similar cases do not occur. To illustrate this, the table below shows a subset of the transition matrix when a single is hit.

		End State							
		000	100	010	001	110	101	011	111
Start State	100	0	0	0	0	0.736	0.264	0	0

Table 1: Excerpt of Transition Matrix for Single

Table 1 shows that if runners are currently on first and third and a single is hit, there is a 73.6% chance that the at-bat ends with runners on first and second (batter hits a single and runner on first advances to second), and a 26.4% chance the at-bat ends with runners on first and third (runner currently on first advances two bases). By filtering the dataset by hit type, we can find both the transition matrices and the overall probabilities of each hit type. Using 2015 data, the hit probabilities were found to be [0.67, 0.20, 0.02, 0.12] (rounded) for singles, doubles, triples, and home runs, respectively. The values were then compared to both Albert's methodology and historical data and found to match.

I then created a function to transition to new states given a type of hit following Albert's logic in [13] and could simulate full games assuming all batters have equal probabilities of each event type. The presented model differs from Albert's in that it can account for individual lineups representing a team of varying skill. To do so, the model takes an input of a batting lineup where each entry is a player's batting average. Then, to judge whether or not a player makes a hit, a uniform random number between 0 and 0.90 is generated (0.9 was used as it was found to most closely approximate true games), and if the number is less than a player's batting average he hits according to overall hit probabilities, and if it is greater than or equal to the batting average, the player gets out. The presented logic is modified from Randall Olson's methodology in [14], though his simulation makes the simplifying assumption that all runners advance one base for each hit and the presented model does not. Using the transition matrices, hit probabilities, and a lineup input, the model iterates through all nine batters for nine-innings of three outs (extra innings assumed to not occur). Then, to analyze different lineups, I created a function that switches the best batter's spot in the lineup with that of every other batter to create nine different systems to compare, including the standard case of the best batter in the second spot. The following section discusses the optimal replication results and results after simulating many games under different lineup scenarios for one team.

ANALYSIS, RESULTS AND CONCLUSIONS

First, the model generates nine possibilities of a lineup using the 2018 Yankees batting averages as a base. The 2018 Yankees were used because the team has found success by placing their best batter second, and I was interested in seeing if the simulation would find that this would statistically be the best heuristic. The batting averages had a minimum of 0.18, a maximum of 0.28, and a mean of around 0.25 (the league average): this is akin to many teams in the MLB. In generating the lineups, the best batter was simply switched with each position for a total of 9 systems including the standard case of the best hitter in spot 2. When running simulations, a relative error of 0.10 and an overall confidence of 80% were desired – since 9 systems are compared, alpha is adjusted to 0.079 as per the Bonferroni correction [19, pp. 545]. Equation 1 below illustrates finding the adjusted alpha value:

$$\propto_{adj} = \frac{1 - \propto_0}{\#\ systems} = \frac{1 - 0.20}{9} = 0.079$$

Equation 1: Bonferroni Correction to Achieve 80% Confidence over 9 systems

Thus, as a result of Equation 1, each interval must be created at a 100(1-0.079) = 91.2% confidence level to achieve an overall confidence of 80%. After determining the required alpha, the logic in Law and Kelton Chapter 9 [19, pp. 500-506] was utilized to find the optimal replications to approximately achieve the desired relative error of 10% (gamma = 0.10). Table 2 below shows the results of this analysis.

Spot of Best Hitter	n^*_r (γ = 0.10)
2 (standard)	152 games
1	170 games
3	162 games
4	149 games
5	163 games
6	161 games
7	167 games
8	185 games
9	175 games

Table 2: Optimal Replications for each System

Then, the model runs each scenario for its required replications. Figure 1 below shows the results of means runs scored with the best batter in all lineup positions and confidence intervals at a confidence of 91.2% each for an overall confidence of 80% and desired relative error of 10%.

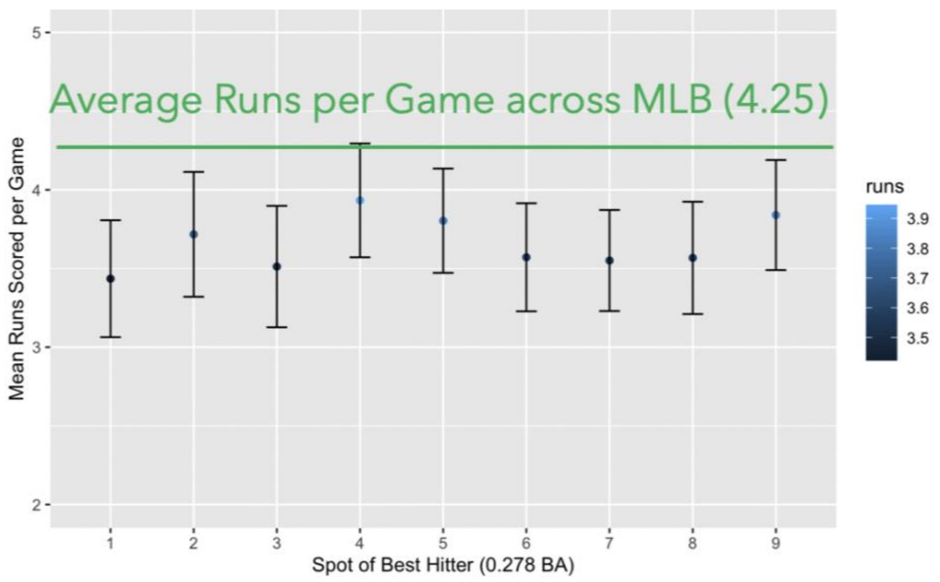

Figure 1: Model Results of Means Runs Scored (y) vs. Spot of Best Hitter (x)

By inspecting the graph, one can see the simulated results show the most runs scored when the strongest batter is in the fourth spot and the least runs scored with the best batter in the first spot. The figure actually shows little difference in mean runs scored resulting from lineup changes, with the interval for the smallest mean being [3.06 runs, 3.81 runs] and the interval for the largest mean being [3.57 runs, 4.29 runs].

Since the model makes some simplifying assumptions as discussed in the Methods section, it almost always underestimates runs scored when comparing to the true mean of 4.25 runs per game across the MLB. For this reason, the model does not accurately predict true mean runs scored in this iteration; however, it provides results that are valid in analyzing the lineup problem when comparing mean runs relatively.

The base case of the best hitter in the second spot was the fourth best lineup out of all nine, and as such, paired t-tests were conducted to analyze if the other scenarios were significantly different than the standard case or not. Eight comparisons are made so alpha must be adjusted again as per the Bonferroni correction: to achieve the desired confidence of 80% across all systems, alpha was corrected to 0.10 representing individual levels of 90%. Table 3 shows the results of this paired t-test, in which replications were discarded as per the method presented in Law and Kelton Chapter 10.2.1 [19, pp. 560-561].

Lineup Spot	Case 1 Interval
1 (vs. 2)	[-0.366, 0.728]
3 (vs. 2)	[-0.296, 0.658]
4 (vs. 2)	[-0.692, 0.262]
5 (vs. 2)	[-0.521, 0.440]
6 (vs. 2)	[-0.375, 0.683]
7 (vs. 2)	[-0.351, 0.660]
8 (vs. 2)	[-0.124, 0.956]
9 (vs. 2)	[-0.829, 0.306]

Table 3: Paired t-test Results for Comparing all Systems to Standard Case

As shown in the table, all of the confidence intervals include zero: thus, the conclusion is drawn that all scenarios with the best batter in each lineup spot are not significantly different than the standard case. In more practical terms, this means the model shows that placing the player with the highest batting average in each lineup spot by switching spots with the current player does not have a significant effect on mean runs scored per game.

Law and Kelton also propose another paired t-test method allowing for the creation of confidence intervals without discarding data [19, pp. 562-563], which involves computing an adjusted degrees of freedom for system comparisons and then creating intervals using the t-value that corresponds to the adjusted degrees of freedom. In the interest of not discarding data and knowing that each system is independent of the standard and each other system, the modified test was performed, and the results are shown below in Table 4.

Lineup Spot	Adjusted Degrees of Freedom	Case 2 Interval
1 (vs. 2)	308.041	[-0.887, 0.324]
3 (vs. 2)	261.187	[-0.926, 0.516]
4 (vs. 2)	220.972	[-1.054, 0.762]
5 (vs. 2)	225.642	[-1.097, 0.764]
6 (vs. 2)	251.059	[-1.067, 0.767]
7 (vs. 2)	234.821	[-0.754, 0.927]
8 (vs. 2)	215.860	[-0.624, 1.055]
9 (vs. 2)	225.385	[-0.893, 1.139]

Table 4: Paired t-test Results using Adjusted Degrees of Freedom

Similar to the previous test, the confidence intervals all contain zero and thus indicate that the lineup spot of the best batter does not matter when his spot is switched with any other batter. The following section offers a brief discussion of the model’s results, limitations, and future work.

CONCLUSIONS, DISCUSSION AND FUTURE WORK

Overall, the results from all simulated games showed that when using the batting averages of a team that has found success placing its hitter in the second spot as inputs, the positioning of the strongest batter within the lineup does not matter as the difference in runs scored between each system and the standard were found to not be significantly different. The presented results were in-line with much of the literature discussed in the background of this paper [8, 10, 11, 14].

In practice, there are many other factors that contribute to a team's success and runs scored besides batting order, many of which are more difficult to model using a Markov Chain approach: this includes team dynamic, streaks, momentum, amount of practice, and many others. Additionally, batting average is not the only metric to judge a player's batting skill, as other statistics like on-base percentage and slugging percentage are also utilized. Lastly, while runs scored for a team is essential, another important metric is whether or not a team wins against another team. Moving forward, I hope to relax some of the assumptions in this model and create a baseball simulator that can capture more of these variables and predict both win probabilities and runs scored for different lineups.

A notable aspect of this research endeavor was that it presents an open-source framework to analyze infinitely many other scenarios in the future. To my knowledge informed by research, it is the only open-source simulation written in R that tries to model baseball games while accounting for runner advancement probabilities and player skill. With code available on GitHub [20], I encourage others to simulate different scenarios and contribute to creating a more complex model that relaxes some of the assumptions presented in this paper.

In the future, I will use this model to study lineups that are overall below average and above average by modifying the difference in batting average between the best hitter and worst hitter. I also plan to improve the accuracy of the model by incorporating pitcher-batter matchups for individual players, calculating transition probability matrices for individual teams, calculating hit probabilities for individual players, relaxing assumptions on the event types that can occur, and analyzing win probabilities for team matchups. By accomplishing these goals, I can study different batting positions for specific teams and more accurately predict mean runs scored and win probabilities. Furthermore, I can input past lineup modification decisions that have been made in practice and validate the results by comparing to historical data.

ACKOWLEDGEMENTS

I would like to offer special thanks to baseball statisticians and Sabermetric subject-matter experts Tom Tango, Jim Albert, Max Marchi. Tango's blog posts were instrumental in understanding current Sabermetrics, Albert and Marchi's previously referenced book was essential in understanding how to work with publicly available baseball data and in translating knowledge of probability and statistics to code, and Albert's open-source half-inning simulator helped me better understand the logic needed for model implementation.

REFERENCES

[1] A. Gleeman, “Banjo Hitter: Reimagining the #2 Hitter,” *Baseball Prospectus*, 05-Apr-2018. [Online]. Available: https://www.baseballprospectus.com/news/article/38931/banjo-hitter-reimagining-2-hitter/.

[2] J. Dewan, “Where should your best hitters bat?,” *Bill James Online*, 21-Nov-2006. [Online]. Available: https://www.billjamesonline.com/stats100/.

[3] M. Marchi, J. Albert, and B. Baumer, *Analyzing baseball data with R*, 2nd ed. Boca Raton: Chapman & Hall/CRC, 2018.

[4] B. Baumer and A. Zimbalist, *The sabermetric revolution: assessing the growth of analytics in baseball*. Philadelphia: University of Pennsylvania Press, 2014.

[5] D. M. Davis, “Markov Analysis of APBA, a Baseball Simulation Game,” *Journal of Quantitative Analysis in Sports*, vol. 7, no. 3, Jul. 2011.

[6] T. Tango, “Cy Young Predictor 2019,” *Tangotiger Blog*, 29-Sep-2019. [Online]. Available: http://tangotiger.com/index.php/site/comments/cy-young-predictor-2019.

[7] S. Rothman, “Sandlot Stats,” *The Sandlot Stats Blog*. [Online]. Available: https://sandlotstats.com/wp/.

[8] N. Hirotsu and J. E. Bickel, “Optimal batting orders in run-limit-rule baseball: a Markov chain approach,” *IMA Journal of Management Mathematics*, vol. 27, no. 2, Dec. 2014.

[9] J. M. Norman and S. R. Clarke, “Optimal batting orders in cricket,” *Journal of the Operational Research Society*, vol. 61, no. 6, pp. 980–986, Dec. 2017.

[10] A. Kira, K. Inakawa, and T. Fujita, “A Dynamic Programming Algorithm For Optimizing Baseball Strategies,” *Journal of the Operations Research Society of Japan*, vol. 62, no. 2, pp. 64–82, Apr. 2019.

[11] B Bukiet et al. “A Markov Chain Approach to Baseball.” *Operations Research*, vol. 45, no. 1, 1997, pp. 14–23. *JSTOR*, www.jstor.org/stable/171922.

[12] J. L. Palacios, "Runs of Markov Chains and Streaks in Baseball," *Methodology and Computing in Applied Probability*, vol. 12, *(4)*, pp. 659-665, 2010. Available: https://proxy.lib.umich.edu/login?url=https://search.proquest.com/docview/756679703?accountid=14667. DOI: http://dx.doi.org/10.1007/s11009-009-9150-6.

[13] J. Albert, “Simulating a Half-Inning of Baseball,” *Exploring Baseball Data with R*, 20-Jun-2016. [Online]. Available: https://baseballwithr.wordpress.com/2016/06/20/simulating-a-half-inning-of-baseball/.

[14] R. Olson, “Simulating Baseball in Python,” *GitHub*, 09-Jul-2018. [Online]. Available: https://github.com/rhiever/Data-Analysis-and-Machine-Learning-Projects/blob/master/python-baseball-simulator/Simulating baseball in Python.ipynb.

[15] Retrosheets, “Retrosheet Event Files,” *Retrosheet.org*. [Online]. Available: https://www.retrosheet.org/game.htm.

[16] J. Albert, “Downloading Retrosheet Data,” *GitHub*, 27-Dec-2018. [Online]. Available: https://bayesball.github.io/VB/Getting_Retrosheet_Files.html.

[17] T. Turkoy, “Software Tools for Game-Level Baseball Data,” *Chadwick Bureau*. [Online]. Available: http://chadwick.sourceforge.net/doc/index.html.

[18] “2018 New York Yankees Statistics,” *Baseball Reference*. [Online]. Available: https://www.baseball-reference.com/teams/NYY/2018.shtml. [Accessed: 16-Nov-2019].

[19] A. M. Law, *Simulation modeling and analysis*, 5th ed. New York, NY: McGraw-Hill Education, 2015.

[20] M. Kalmus, “Baseball Simulation in R,” *GitHub*. [Online]. Available: https://github.com/mkalmus/Baseball_Simulation_in_R. [Accessed: 09-Dec-2019].

CODE HELP REFERENCES

1) Finding states at each event: Chapter 5 of [3]
2) Base Simulation logic (calculating runs and outs and transitioning function): [13] but heavily modified, partly using logic from [14]
3) Sorting lists of lists and converting to dataframe logic comes from Stackoverflow user "Rick" last accessed 12/10/19 at https://stackoverflow.com/questions/28100593/how-to-sort-a-list-of-lists-in-r
4) Turning columns in df from lists to non-lists comes from stackoverflow user Alistaire last accessed 12/10/19 at https://stackoverflow.com/questions/38860380/unnesting-a-list-of-lists-in-a-data-frame-column