Optimal batting order in baseball

Francis Frégeau

November 2022

Introduction

In a game of baseball, the manager must determine the batting order before the game begins. Two copies of said order must be produced: one for the umpire, and one for the opposite manager [1]. This is no easy task as the manager must ensure that his lineup is strong from start to finish. Should the best batter go first? Should the weakest be last? There exists a few guidelines [2] as of today, but most of them are backed by nothing but mere heuristics and coaching insights.

Once one steps out of informal blogs and popular baseball websites, they will find that some work on the subject has already been done within the academic community, with Markov chains being a recurring method of choice [3] [4] [5]. We seek to build on this work (particularly that of Katsunori Ano [3]) by using extra transition matrices to account for base stealing, as well as attempting to optimise the expected number of points scored as a function of the batting order.

Innings modeled as Markov chains

Unlike most sports, baseball can neatly be described as a system of states where transitions occur until 3 batters have been eliminated. This is because whilst an inning is in progress, a total of 3 bases can either be unoccupied or occupied, and between 0 and 2 players can be be eliminated. These possibilities tally up to $2^3 \cdot 3 = 24$ possible states alongside a 25th one representing the inning's end for a given team whenever 3 players have been eliminated.

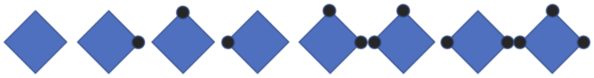


Figure 1: 8 possible base states

Play-by-play data can be obtained from various websites [6], which can then be used in order to compute the transition matrices $\{M_j\}$ of batters included in the manager's list. However, some of the play-by-play transitions (i.e.: base stealing) are associated with events that do not pertain to batters attempting to hit a ball. If one is to take the aforementioned transitions into account, then they must compute additional transition matrices $\{W_j\}$ so that the state of the game S_t after t batters have had their turn is of the form

$$\mathbf{P}\left(S_t = i\right) = \left(\prod_{j=1}^{t} M_j W_j\right)_{1,i}$$

assuming state 1 is the starting state (0 base occupied and 0 players out). The matrices $\{W_j\}$ must be statistics of all batters $\{k \leq j : B_k\}$ currently occupying a base, and we choose a simple average of the form

$$W_j = \mathbf{P}_j\left(\text{steal}\right) \cdot \frac{\sum_{k=1}^{j} \mathbf{P}\left(\mathbf{I}_{k,j} = 1\right) V_k}{\sum_{k=1}^{j} \mathbf{P}\left(\mathbf{I}_{k,j} = 1\right)} + \left(1 - \mathbf{P}_j\left(\text{steal}\right)\right) \cdot I_{25}$$

where $\mathbf{P}\left(\mathbf{I}_{k,j} = 1\right)$ denotes the probability that player B_k is occupying any of the 3 bases at time j, V_k denotes the base-stealing transition matrix of player B_k, and $\mathbf{P}_j$ (steal) denotes the probability that any of the players $\{k \leq j : B_k\}$ attempt to steal a base. Note that the latter probability is computed as

$$\mathbf{P}_j\left(\text{steal}\right) = \sum_{k=1}^{j} \mathbf{P}\left(\text{player } k \text{ steals}\right) \cdot \mathbf{P}\left(\mathbf{I}_{k,j} = 1\right)$$

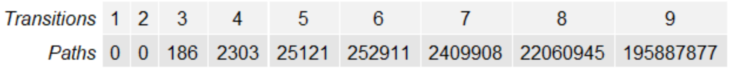
which a simplification as it should technically be capped to a sum of 3 terms given no more than 3 players can be on base at all time. However, $\mathbf{P}\left(\mathbf{I}_{k,j} = 1\right)$ decays rapidly and is almost null whenever the player B_k has entered the game 3 transitions ago (i.e.: $j \geq k + 3$).

Suitable statistic for optimal batting order

Each transition has an associated number of scored points which is a function of the number of players occupying bases, namely:

$$f(i,j) = \max\left\{n_i - n_j - O(i,j),\ 0\right\}$$

where n_k represents the number of players on bases in state k, and $O(i,j)$ the number of players that have been eliminated during the transition from state i to j. Each particular path defined by a unique series of transitions has its associated scored points value. Alas, the number of such possible paths rapidly becomes intractable as the number of transitions increases, rendering the task of optimising the number of points scored computationally infeasible.



Transitions	1	2	3	4	5	6	7	8	9
Paths	0	0	186	2303	25121	252911	2409908	22060945	195887877

Figure 2: Number of possible paths from state 1 to state 25

There is, however, one statistic which could serve as a decent proxy for the number of points scored during an inning based on a strong correlation between the two metrics ($\rho = 0.779$ using data from 2011 to 2021): the number of batters who came into play. Since 99% of innings do not make use of more than 9 batters, which is precisely the number of batters that the managers ought to include in their lists, we will seek to maximize the quantity

$$\mathbf{E}\left[X|\psi\right] = \sum_{t=1}^{9} P_{\psi}\left(S_t \neq 25\right)$$

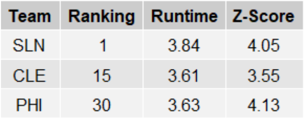
with respect to the player ordering ψ, which would be the expected number of transitions X before the inning stops if $\forall\psi\ \forall t > 9:\ P_{\psi}\left(S_t = 25\right) = 1$ [7]. In other words, we seek to find the ordering of $\{M_j\}$ and its associated set of off-bat transition matrices $\{W_j\}$ which would produce the chain with the longest expected run time $\mathbf{E}\left[X\right]$.

Deriving the optimal batting order

There are no analytical solution to this problem, but brute-force is computationally feasible within a reasonable time frame if one possesses a strong enough computer to go thought the $9! = 362\ 880$ possible arrangements of a set of 9 players. All that is needed to compute $P\left(S_t \neq 25\right)$ is the matrix $\prod_{j=1}^{t} M_j W_j$, so that we must derive $\{M_j\}$ and $\{V_k\}$ beforehand, whereas $\{W_j\}$ must be computed on the fly as base-stealing transition matrices depend on the ordering ψ. The language R was used to code the functions needed to compute $P\left(S_t \neq 25\right)$, where parallel computing was of great use (Intel i7-12700k CPU, 16 cores used).

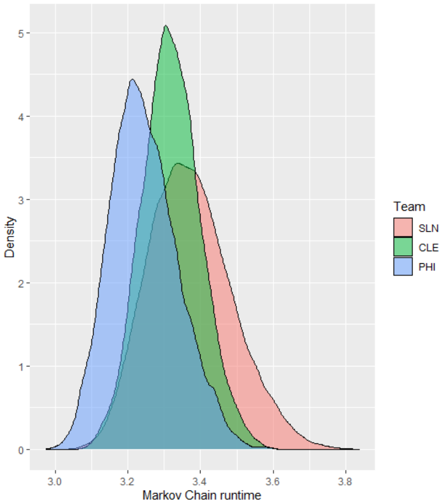
Results

Using play-by-play data from the 2014 and 2015 seasons obtained via a custom web-scrapper [8], we were able to derive the optimal batting order for the top 9 most used batters of various teams. Results for 3 teams ranked 1st, 15th and 30th (last) in the 2015 season are presented in the figure below.



Team	Ranking	Runtime	Z-Score
SLN	1	3.84	4.05
CLE	15	3.61	3.55
PHI	30	3.63	4.13

(a) Rankings and maximum runtimes



(b) Runtimes density plot

Figure 3: Results of the brute-force optimisation for 3 MLB teams

As can be seen in the density plot, the top 9 batters from lower-ranked teams seem to have lower feasible runtimes, with the St-Louis Cardinals (SLN) clearly dominating its two counterparts. It's also worth noting that albeit having very different runtime distributions, the Cleveland Browns (CLE) and the Philadelphia Phillies (PHI) have near-identical maximum runtime values.

Conclusion

We've developed a new approach to calculating the optimal batting order in a game of baseball by building on previous work [3] using Markov chains. The two novel components of our method are the addition of in-between batting events transition matrices $\{W_j\}$ to account for base stealing, as well as maximising the expected number of transitions undertaken by the Markov process as a proxy to maximizing the expected number of points scored. The latter addition enabled us to find an exact solution to the optimisation problem by rendering it computationally feasible, whereas the former added a touch of realism.

Theoretical results suggests that ordering has a noticeable effect on the expected duration of an inning, and hence is likely impact the expected number of points scored. That being said, our model doesn't offer any tangible insight as to what particular metric of interest could determine the optimal batting order of an arbitrary player. As such, it is very much a black-box model despite relying solely on traditional statistics rather than machine learning.

Lastly, while brute-forcing the solution for 9 players is feasible, using such an approach for $n > 9$ players would require considerable computing power as there would now be $\frac{n!}{(n-9)!}$ possible orderings whose respective runtimes need to be computed. As such, the model is only really appropriate for selecting the optimal ordering *after* 9 batters have been chosen.

References

[1] Phillip Mahony. *Baseball Explained.* McFarland Books, 2014.

[2] "The Batting Order". In: *Baseball Canada* (). DOI: https://baseball.ca/the-batting-order.

[3] Katsunori Ano. "Improved Optimal Batting Order With Several Effects For Baseball". In: *Nanzan University* (2000). DOI: https://www.researchgate.net/publication/2435808_Improved_Optimal_Batting_Order_With_Several_Effects_For_Baseball.

[4] Joel S. Sokol. "A Robust Heuristic for Batting Order Optimization Under Uncertainty". In: *Journal of Heuristics (9), 353–370* (2003). DOI: https://link.springer.com/article/10.1023/A:1025657820328.

[5] David W. Smith. "Effect of batting order (not lineup) on scoring". In: *The Baseball Research Journal (35)* (2006). DOI: https://go.gale.com/ps/i.do?id=GALE%7CA176987861&sid=googleScholar&v=2.1&it=r&linkaccess=abs&issn=07346891&p=AONE&sw=w&userGroupName=anon%7Ec8253680.

[6] *FanGraphs website.* URL: https://www.fangraphs.com.

[7] Wojciech Wojdows Pat Muldowney Krzysztof Ostaszewski. "The Darth Vader Rule". In: *Tatra Mountains Mathematical Publications* 52 (2012), pp. 53–63. DOI: 10.2478/v10127-012-0025.

[8] Francis Frégeau. *MLB Webscrapper.* Github repository, 2021. URL: https://github.com/c-Stats/mlb_webscrapper.