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 formulating an alternative way to compute the expected points scored per innings, as well as deriving the optimal batting order thanks to CPUs being overwhelmingly more powerfull today than they were 20 years ago.

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 25^{th} 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

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

where I_{25} is the identity matrix, $\mathbf{P}\left(\mathbf{I}_{k,j} = 1\right)$ 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 $\mathbf{P}_j\left(\text{steal}\right)$ 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.: $j \geq k + 3$).

Deriving the optimal batting order

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

$$\phi(i,j) = \max\left\{n_i - n_j + 1 - O(i,j),\ 0\right\}$$

where n_k represents the number of players on base in state k, and $O(i,j)$ the number of players that have been eliminated during the transition from state i to j. As for the $+1$, it is a factor of correction to account for the fact that a new player has been put into play. (I.e.: if a batter gets on based and the number of people on base goes from 2 to 2, then one player has either scored or been eliminated). Under our model, points can only be scored during batting transitions as stealing the fourth base is very unlikely and thus ignored as an event. The expected number of points scored for a transition is:

$$\sum_{i=1}^{25} \sum_{j=1}^{25} \mathbf{P}\left(S = i\right) \mathbf{P}\left(i \to j\right) \phi(i,j)$$

where $\mathbf{P}\left(i \to j\right)$ denotes the probability of transitioning from state i to state j. This quantity for the t^{th} transition can be expressed as the sum of the entries of the vector

$$\lambda_t^{\mathrm{T}} = \begin{cases} v_0^{\mathrm{T}}\left(M_t \odot \boldsymbol{\phi}\right) & \text{if } t = 1 \\ v_0^{\mathrm{T}}\left(\prod_{j=1}^{t-1} M_{j-1} W_{j-1}\right)\left(M_t \odot \boldsymbol{\phi}\right) & \text{otherwise} \end{cases}$$

where v_0 is positional vector at time 0 (its first entry is 1 whereas all other entries are 0), $\boldsymbol{\phi}$ is a matrix with entries $\boldsymbol{\phi}_{i,j} = \phi(i,j)$, and $\odot$ is the Hadarmard product operator.

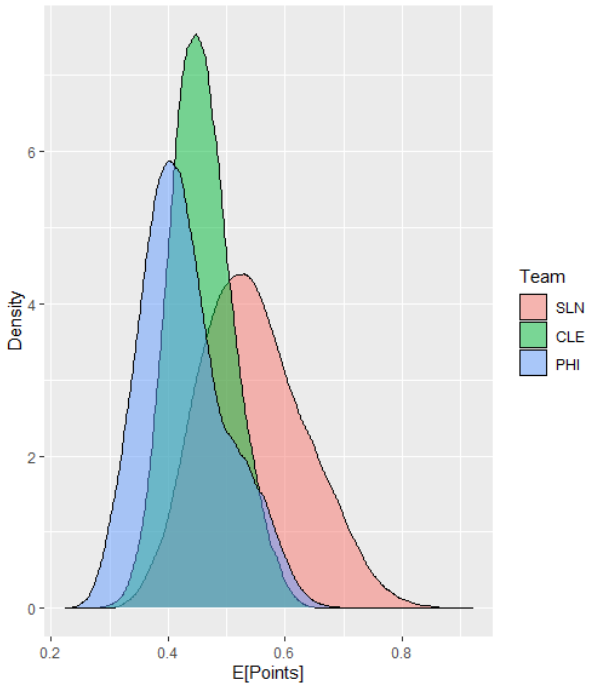
Since 99% of innings do not comprise more than 9 face-offs between a batter and a pitcher (2012-2021 data), we will seek to maximize the conditional expected number of points scored per inning $\mathbf{E}[X|\Psi = \psi]$ with respect to the player ordering ψ under the assumption that the inning is automatically over once all 9 batters have had their turn. Although there is no analytical solution to this problem, 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. Matrices $\{M_j\}$ and $\{V_k\}$ can be derived beforehand, whereas $\{W_j\}$ must be computed on the fly as base-stealing transition matrices depend on the ordering ψ.

Results

Using parallel computing in R [7], we were able to compute optimal batting orders alongside the pdf of $\mathbf{E}[X|\Psi = \psi]$ for various groups of batters (where it was assumed that $\mathbf{P}(\Psi = \psi) = \frac{1}{9!}$ for any ordering ψ). Deriving the two aforementioned quantities took approximately 2 minutes with a decent CPU (12th Gen Intel i7-12700K, 16 cores used), leaving no doubt that the computational hurdles mentioned by Katsunori Ano [3] are no longer relevant. Play-by-play data from the 2014, 2015 and 2016 seasons was obtained using a custom web-scrapper [8], and the top 9 batters from the 1^{st}, 15^{th} and 30^{th} ranked teams of the 2015 and 2016 seasons were compared to one another. Transition matrices were computed using the data from the current season alongside data from the previous one.

Team	Ranking	E[Points]	Max(Points)	Z-Score
SLN	1	0.55	0.92	4.15
CLE	15	0.46	0.66	3.76
PHI	30	0.43	0.73	4.00

(a) General statistics

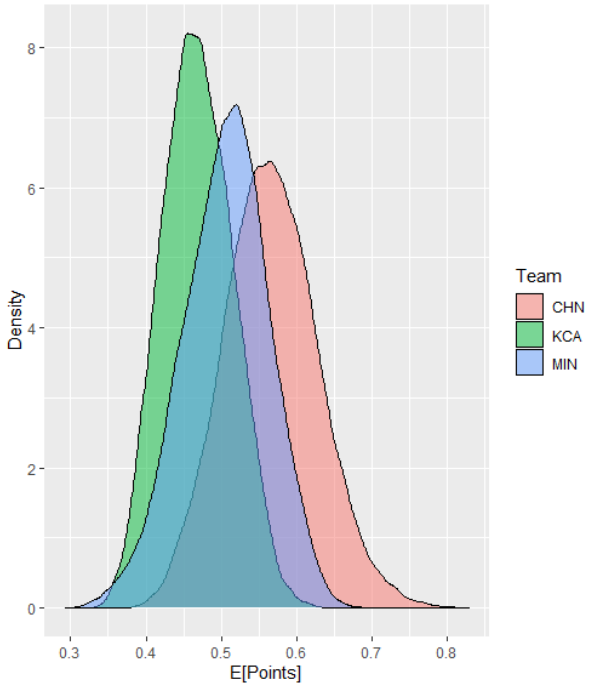


(b) Points density plot

Figure 2: Results of the brute-force optimisation for the 2015 season

Team	Ranking	E[Points]	Max(Points)	Z-Score
CHN	1	0.57	0.83	4.21
KCA	15	0.47	0.66	4.19
MIN	30	0.51	0.70	3.46

(a) General statistics



(b) Points density plot

Figure 3: Results of the brute-force optimisation for the 2016 season

As can be seen in the density plots, orderings have a tremendous impact on $\mathbf{E}[X|\Psi = \psi]$, with pdfs having a range of around 0.4. In other words, the difference between the least and most optimal ordering is approximately 0.4 points scored per inning, which translates into a total of 3.6 points per game.

Furthermore, the top-ranked teams seem to have access to a wide array of possible orderings which dominate any of their rival's, so it is no surprise that their average expected points per inning are significantly greater than their counterparts'. The bottom and middle ranked teams have very similar statistics but very different pdfs; the 2015 data clearly indicates that the middle team has a much better potential than the bottom one, whereas the 2016 data hints at the complete opposite conclusion. Indeed, the 2016 Minnesota team (MIN) should have a clear edge over the 2016 Kansas one (KCA) as the former's pdf dominates the latter one's for MIN orderings which are beyond the distribution's mode. As such, we can hypothesise that sub-optimal batting orders could have played a role in MIN being ranked last in the 2016 season.

However, it should be noted that a lot of unannounced batters make appearances and not all selected batter will get to face the enemy pitcher. Consequently, thoroughly testing the validity of the model (i.e.: by deriving $\mathbf{E}[X]$ for announced batting orders then testing the fit of a statistic of the form

$$\mathbf{P}_{\text{Home}}\left(\text{Win} \mid \mathbf{E}_{\text{Home}}[X], \mathbf{E}_{\text{Away}}[X]\right)$$

) is next to impossible, and as such, only purely theoretical results in favour of the effect of batting

order ψ on $\mathbf{E}[X|\Psi = \psi]$ can be derived.

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 novel components of our work are the way to compute the expected points scored per inning, going through with solving the optimisation problem by brute-force for all of the 9! orderings, and deriving the pdf of $\mathbf{E}[X|\Psi = \psi]$ with respect to ψ. Additionally, we proposed an original way to compute the transition matrices $\{W_j\}$ by computing individual base-stealing transition matrices from play-by-play data, then weighting them via a decaying average based on survival functions.

Theoretical results suggests that ordering has a noticeable effect on 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, and 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 run times need to be computed. Consequently, 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] Microsoft Corporation and Steve Weston. *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*. R package version 1.0.17. 2022. URL: https://CRAN.R-project.org/package=doParallel.

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