Lisa Everest, Katie Shade, Rachel Hausmann
15.053 Project

Optimization of MIT Varsity Women's Softball Batting Order

Introduction:

Softball and baseball are sports that depend on strategically placing players for both offensive positions, in the batting order, and defensive positions, in the field. Utilizing player statistics appropriately can be the difference between a championship team and a losing team. Our project tackles the challenge of optimizing a 9-person softball batting order using players' offensive statistics. In this project, we have optimized the 2015-2016 MIT Varsity Women's Softball batting order. This problem is especially intriguing because the 2015-2016 team placed 5th in the country, at the 2016 Division III Softball National Championships. For this optimization process, we took the 10 players with the most plate appearances, and using their game history, we determined the optimal 9-person batting order for the team.

We used two methods to calculate which batting order was optimal. For both of these methods, we used a collection of functions that describe the transitions between states for each result of an at-bat. Our first method was computer simulation. The second method calculated the expected value of the number of runs scored by a specific 9-person batting order. These will be discussed in detail later.

Since softball and baseball have extremely sports-specific terminology, please refer to the Appendix for all definitions and other relevant information.

Data:

The 2015-2016 MIT Varsity Women's Softball team statistics were found online at http://www.mitathletics.com/sports/w-softbl/2015-16/roster. A sample page can be found at http://www.mitathletics.com/sports/w-softbl/2015-16/bios/shade_katherine_z3nf?view=profile. The raw data used includes: number of plate appearances, number of hits, doubles, triples, and home runs, runs batted in (RBI), number of walks, number of strikeouts, number of sacrifice flies, number of sacrifice hits. This data was then used to calculate our own useful statistics:

- Walks
- Strikeouts
- Singles = Hits - Doubles - Triples - Home Runs
- Doubles
- Triples
- Home Runs
- Outs Advancing Runner = RBI + 2(Sacrifice Flies + Sacrifice Hits)[1]
- Outs Not Advancing Runner = Plate Appearances - (Walks + Strikeouts + Singles + Doubles + Triples + Home runs + Outs Advancing Runner)
- Plate Appearances

These statistics were compiled to calculate the probabilities of all possible events in a game. These probabilities include:

- P{Walk} = Walks / Plate Appearances
- P{Strikeout} = Strikeouts / Plate Appearances

[1] If no runners are on base and this event occurs, then the batter is simply out.

- P{Single} = Singles / Plate Appearances
- P{Double} = Doubles / Plate Appearances
- P{Triple} = Triples / Plate Appearances
- P{Home run} = Home runs / Plate Appearance
- P{Out advancing runners} = Outs Advancing Runner / Plate Appearances
- P{Out not advancing runners} = Outs Not Advancing Runner / Plate Appearances
- P{Plate appearance} = 1

Our method involved many reasonable assumptions, based off of the rules of softball. First, we assumed that outs may be divided into two categories: outs that advance runners on base and outs that do not. Since there is no official statistic for outs that advance runners, we used the equation RBI + 2(Sacrifice Flies + Sacrifice Hits) to estimate this value (see Appendix #6 for a detailed explanation). Then, since outs are only divided into two categories, we can estimate the percentage of outs that do not advance runners on base by subtracting the sum of all statistics from the number of plate appearances.

Next, we made many assumptions for the outcome of each possible event the batter could experience:

- If the batter gets a walk, then the batter goes to first and any runners on base move forward only if "pushed" (if there is a runner on every base behind them, including first)
- If the batter gets a strikeout, then the batter is out and no runners move
- If the batter gets a single, then the batter goes to first base, a runner on first base moves to third base, and runners on second and third base score runs
- If a batter gets a double, then the batter goes to second base, and all runners score
- If a batter gets a triple, then the batter goes to third base, and all runners score
- If a batter gets a home run, then the batter scores, and all runners score.
- If a batter gets an out advancing runners, then all runners advance forward one base (including scoring if there was a runner on third)
- If a batter gets an out not advancing runners, then the batter is out and runners do not move

Methods:

Using the probabilities calculated and the assumptions we made, we then wrote two programs to estimate the number of points that would be scored by the team given a batting order. The first simulates a game and the second actually calculates the expected value of the score.

Many simulated games were run on all possible batting orders. Given that there are 10 possible players to choose from, there are 10! = 3,628,800 potential batting orders to be tested. To simulate a 7 inning softball game, the game has a current state that is updated after each batter. This state contains information such as the current inning, the current batter, the location of runners on base, the number of outs, and the current score. Using the data for each player to calculate the probability of each outcome for that specific player and a random number generator, the result of each at bat was determined, and the state was updated. The simulation continued until the 7 innings (21 outs total) were completed. The final score was then taken as one simulated game for that specific batting order.

Simulating one game was quick for the computer to run, but simulating millions of games took much longer. The simulation program was used to reduce the 3,628,800 possibilities to the top 1,000 possibilities. When running a simulation, it is very possible for an order to become "unlucky" and have a lower average score than expected. To reduce this risk, the simulations were run in multiple sets, each time decreasing the number of top orders remaining and increasing the number of simulated games averaged for each order. With a greater number of games averaged, the simulation program will be more accurate as the number of trials increases. Once this process was completed

multiple times, the top 1,000 orders were passed into the program that calculates their expected value to get a more exact answer than the simulation will give.

To calculate the expected value of the score, we have a set of all states to be explored and a dictionary mapping states to their probabilities. These states are defined the same as those in the simulation program. For the expected value program, there is also a special state for the end of the game. Until we run out of states to explore, we pop a state (removing it from the set) and find all the possible states that could come next and the probability of each. We add each state to the set of states (because of how sets work, if it is already there, it will not be added twice) and update the probability of the new state in the dictionary. When the end state happens, we add the score multiplied by the probability of the state to the expected score. Once we have emptied our set of states, we have the expected score given a batting order.

To summarize, the expected value solution had a more desirable result, since there is no variability involved in calculating expected values, but it also took significantly longer to calculate the runs generated by a specific batting order than the simulation solution. In fact, it would have taken over a year to run the expected value solution on all 10! possible batting orders! To avoid this, we used the simulation program to reduce the number of possible orders being tested as previously described.

To better ensure a more accurate result in case of "unlucky" orders being pruned in the simulations, we iterated through changing the batting order slightly, as shown in the sensitivity analysis section, and then calculated the batting order for each simple switch of players. By doing this check on similar batting orders, we reduced the possibility of mistakenly not considering a top-contending batting order.

Results and Discussion:

Before running the simulation strategy and dynamic programming strategy on the various possible 9-person batting orders, we first ran both strategies on batting orders of 9 copies of a single person. For example, we ran both strategies on a batting order composed entirely of Katherine Shade #16 (i.e. the order {16, 16, 16, 16, 16, 16, 16, 16, 16}). The simulation strategy came up with 4.22368 as the average number of runs generated, while the dynamic programming strategy came up with 4.228066 as the expected number of runs generated. The number of runs generated for these 10 possible batting orders (composed entirely of one person, where there are 10 possible people) made relative sense, based off of each person's batting average and performance in the 2015-2016 season.

Player (#)	**Expected Score**
Jasmin (1)	8.68776186556
Monica (2)	1.98921763441
Ali (3)	2.29952068258
Natalie (5)	2.26538909426
Amanda (6)	6.53783667339
Zoe (10)	3.67723674568
Erika (11)	0.85718737970

Kim (12)	2.42623338596
Katie (16)	4.22806619089
Tori (23)	4.74103994603

With the validity of each computer program confirmed, we moved on to the optimization process. Through the methods described earlier, we discovered that the optimal batting order was: {16, 6, 1, 23, 10, 3, 2, 12, 5}, with an expected score of 3.936641 runs per game. The next best batting order was: {16, 6, 1, 23, 10, 3, 12, 2, 5}, with an expected score of 3.934563. The difference in the expected number of runs generated per game, between these two batting orders, is only 0.002 runs. In fact, among the top 50 batting orders, the expected number of runs generated per game varied by less than 0.06 expected runs, a very small amount.

Sensitivity Analysis:

To see how sensitive the expected score is to small changes in batting order, we tried swapping two adjacent batters in the order and calculating the new order's expected score. The table below details the expected score of each order. The smallest difference was only 0.002 expected runs and the largest difference was 0.027 runs. For comparison, when the order is reversed, the difference is 0.268 runs, much more than any of the small changes. From this, we can conclude that the expected score is not very sensitive to small changes in the order, but does significantly change when the order is significantly changed.

Description (order is 0 indexed)	Order	Expected Score
Best	16, 6, 1, 23, 10, 3, 2, 12, 5	3.93664102306
swapped batters 0 and 1	6, 16, 1, 23, 10, 3, 2, 12, 5	3.92136094648
swapped batters 1 and 2	16, 1, 6, 23, 10, 3, 2, 12, 5	3.92666636829
swapped batters 2 and 3	16, 6, 23, 1, 10, 3, 2, 12, 5	3.93243205439
swapped batters 3 and 4	16, 6, 1, 10, 23, 3, 2, 12, 5	3.92684890576
swapped batters 4 and 5	16, 6, 1, 23, 3, 10, 2, 12, 5	3.90939009645
swapped batters 5 and 6	16, 6, 1, 23, 10, 2, 3, 12, 5	3.92853588487
swapped batters 6 and 7	16, 6, 1, 23, 10, 3, 12, 2, 5	3.93456351421
swapped batters 7 and 8	16, 6, 1, 23, 10, 3, 2, 5, 12	3.92898913630
Reversed order	5, 12, 2, 3, 10, 23, 1, 6, 16	3.66831593602

Conclusion:

Through this project, we discovered a new statistic that could potentially be useful for the MIT Varsity Softball team and other softball or baseball teams. This statistic is the number of runs generated by a single player if that player batted in every spot in the batting order. This would be a very direct way to evaluate the potential of a player, and also to compare batters. Although a batting average allows people to see how often a player gets a hit, and although people use the batting average to compare players, it may not be the most intuitive metric. With the expected number of runs generated by a team composed of only one player, however, we can get a crystal clear measure of the value a single player can add to a team.

Taking a look at the real batting order used in the 2015-2016 NCAA Division III Softball National Championship, we see some interesting results. The 2015-2016 MIT Varsity Softball Team placed 5th in the nation with rotating batting orders of either {1, 6, 23, 16, 10, 3, 12, 2, 5} and {1, 6, 16, 23, 10, 3, 12, 2, 5}. Running the computer program that calculates the expected number of runs generated on these two batting orders, we arrive at {1, 6, 23, 16, 10, 3, 12, 2, 5} generating 3.9122311 runs and {1, 6, 16, 23, 10, 3, 12, 2, 5} generating 3.91712 runs. Compared to our optimal batting order {16, 6, 1, 23, 10, 3, 2, 12, 5}, which generates 3.93664 runs, the two real batting orders only did roughly .02 runs worse than the optimal order. Thus, the real batting orders were very close to optimal. In fact, the 5th place national ranking of the 2015-2016 team may be partially attributed to the strategic placement of players in the batting order.

In conclusion, we were able to use our two computer programs to deduce the optimal batting order, based on ten possible players from the 2015-2016 MIT Varsity Softball Team. We can therefore use this code for future seasons in order to optimize these future batting orders. Not only did this project have an impact on our own understanding of optimization and our understanding of the success of the 2015-2016 team, but it will also have an impact on future teams.

Appendix:

Partially adapted from: http://m.mlb.com/glossary/standard-stats

1. The batting average is calculated as the number of the batter's safe hits divided by her official times at bat
    a. A safe hit is credited to a batter when the batter safely reaches first base after hitting the ball into fair territory (i.e. within the foul lines and therefore not a foul ball), without the benefit of an error (a mistake by the fielder) or a fielder's choice (fielder chooses not to try to get the batter out at first base)
2. The on-base percentage is calculated as: (hits + walks + hit-by-pitch) / (bats + walks + hit-by-pitch + sacrifice fly balls)
    a. A sacrifice fly ball is a batted ball that satisfies four conditions:
        i. There are less than two outs when the ball is hit
        ii. The ball is hit to the outfield (fair or foul) or to infield foul territory
        iii. The batter is out because the fielder catches the ball in the air (or if the batter would have been out, if not for the fielder dropping/missing the ball - i.e. an error)
        iv. A runner who is already on base scores on the play
3. The slugging percentage is calculated as: (singles + doubles*2 + triples*3 + homeruns*4) / at bats
4. A strikeout includes strikeouts where the batter swings or looks (doesn't swing) at the third strike
5. A walk occurs when the batter advances to first base due to the pitcher throwing four balls. We included the number of HBP (Hit by Pitch) occurrences in the number of walks in our data, and HBP occurs when the batter gets hit by the pitcher and therefore advances to first base.
6. An explanation on the equation:

    Outs Advancing Runners = RBI + 2(Sacrifice Flies + Sacrifice Hits)

    By definition of the statistics, RBI's, sacrifice flies, and sacrifice hits are outs that advance runners. Thus, we add these 3 statistics to compute the outs that would advance runners. A greater weight is used for sacrifice flies and sacrifice hits because we do not have statistics for sacrifice bunts, which are also outs that advance runners.

Code: https://github.com/rachelhausmann/15.053_softball_optimization