The Ultimate Lineup

Chad Kelterborn
Carthage College
ckelterborn@carthage.edu

May 3, 2013

Abstract

We explore the optimization of a youth baseball teams batting order, considering each players individual statistics. Specifically, we wanted to see if an alternating batting order of walkers and hitters (e.g. Highest Walk Average, Highest Batting Average, Second Highest Walk Average) would be a better lineup than the conventional batting order by descending batting average. For our calculations, we take into account each individuals Hit Average, On-base Percentage, Walk Average and Strike-out Average. From our simulations, we found that an alternating lineup produces fewer runs at 7.7913 runs per game than a descending lineup which produces 7.9716 runs per game. Although the descending lineup was found to be better than 99% of all possible lineups, our simulation found that the lineups that scored more runs per game on average than the descending lineup shared certian characteristics in the ordering of the lineup.

1 Introduction

In our investigation, we analyze the construction of the ultimate lineup for an actual youth baseball team. In Major League Baseball, managers place their best batters at the top of the batting order. Following this ideology, youth baseball coaches routinely create their lineups with the best batter at the top of the batting order and the worst batter at the bottom of the batting order, enabling the best hitters to get more at bats. This lineup design often causes a "black hole" to exist at the bottom of the lineup in which very few hits are ever made by that group of hitters. This raises the question: Does stacking a youth baseball team's batting order in the aforementioned manner provide the team with the optimum capacity to score runs and win games? Or does there exist a better lineup, one that would produce more runs and, ultimately, win more games, for a given team? How does the grouping of batters based on their individual abilities affect a team's performance?

A youth baseball team has between 10 and 12 players on a team. Additionally, all players on the team hit in a batting order that is fixed at the beginning of each game. There are no substitions and all players must bat. With these conditions we can see that there are, based on the number of players on a team, 10! or more possible lineups for a given team. In our project, we look to create the ultimate lineup provided statistics from three actual youth baseball teams. Each team had 11 players, so there exist 11! or 39,916,800 possible lineups for each team. With so many possible lineups we strongly believe that there is a better way to organize batters than based solely on descending hit average. To achieve this goal we constructed a computer simulation using the programming language C++. We then determined the number of games that our simulation needed to run for each lineup attempting to maximize efficiency and minimizing the variance of the average runs scored per game. The same lineup played 10 sets of "x" number of games from which we were able to determine the variance in runs scored per game. Once we determined that the optimum number of games necessary to maximize efficiency and minimize variance was 1,000 games, we were able to continue with our analysis of our initial question: Does alternating the batters based on their walk and hit average (e.g. Walker, Hitter, Walker, Hitter...) produce on average more runs per game than the traditional descending lineup? Additionally, what is the best conceivable lineup for a youth baseball team provided their statistics?

2 Definitions and Development

We will now define the terminology intrinsic to our analysis as well as specify how these values will be beneficial to us during our investigation. While sharing many similarities with Major League Baseball, youth baseball has several unique rules. A youth baseball game is shorter in length, lasting 6 innings as compared to 9 innings in a Major League Baseball game; and in both cases, each inning lasts for 3 outs. A youth baseball team consists of 10 - 12 players. Prior to the start of each game, these players are set in a fixed batting order (lineup). This batting order cannot change over the course of the game; as a result there are no substitutions of players throughout the game and each player must bat.

Definition 1. A **contact** is an at bat where the batter makes contact with the baseball and hits it into fair territory. If there are any baserunners and fewer than 2 outs, then the baserunners safely advance one base.

The contact can be thought of as a flyout or a groundout with the batter being the one who gets out, all other baserunners are automatically safe.

Definition 2. An **advancement** is an at bat where the batter advances the baserunners. This can be achieved in one of three manners: a hit, a walk, or a contact.

Advancement is an idea we created to statistically keep track of a batter's ability to advance the baserunners.

Definition 3. The **hit average** is a representation of how frequently an individual makes a hit. It is defined as the total number of hits divided by the total number of at bats (inclusive of walks).

$$HA = (\text{total hits})/(\text{total at bats}).$$

The hit average differs from the more commonly used batting average in that the hit average accounts for all at bats made by a player, including the at bats that resulted in a walk. On the other hand, the batting average accounts for all of the at bats that did not result in a walk. This subtle difference leads to the fact that a batter's hit average is less than his batting average.

Definition 4. The **walk average** represents how frequently an individual makes a walk. It is defined as the total number of walks divided by the total number of at bats.

$$WA = (\text{total walks})/(\text{total at bats}).$$

In youth baseball, there are certain batters who make a sizable number of walks, and to keep track of this statistic we created the walk average. It helps us break down the on-base average to better discern which batters are walkers and which are hitters.

Definition 5. The **on-base average** is how frequently an individual safely reaches a base from an at bat. It is defined as the total number of hits and walks divided by the total number of at bats.

$$OBA = (\text{total hits } + \text{ total walks})/(\text{total at bats}).$$

The on-base average is synomous with the commonly used on-base percentage. In our analysis, it is also defined as the sum of the hit average and the walk average.

Definition 6. The **advancement average** represents how frequently an individual advances the baserunners. This quantity is defined as the total number of at bats where the batter did not strikeout.

$$AA = 1 - (\text{total strikeouts})/(\text{total at bats}).$$

Advancement average represents a batter's ability to advance baserunners.

Definition 7. Descending lineup refers to the commonly used batting order in youth baseball. The best batter (the batter with the highest hit average) is placed at the top of the batting order and the worst batter (the batter with the lowest hit average) is placed at the bottom of the batting order. The remaining batters are placed by descending hit average.

Definition 8. Manipulated lineup is any lineup except the descending lineup.

Definition 9. Alternating lineup is a special case of the manipulated lineup where the batters are ordered based on alternating descending hit average and walk average. The lineup takes the form of (Hitter, Walker, Hitter, Walker... etc.).

Definition 10. Single is a hit where the batter safely reaches first base. **Double** is a hit where the batter safely reaches second base. **Triple** is a hit where the batter safely reaches third base. **Home Run** is a hit where the batter safely reaches home plate.

Definition 11. Strike Out is an out where three strikes are called against a batter during an at bat.

The data used throughout our analysis was provided by Dr. Snavely's son's baseball team. Over the course of a youth baseball season, each player's batting statistics were recorded. From these statistics we were able to calculate each player's hit average, on-base average, advancement average, and strike out average. We then compiled these statistics to construct the statistic matrix that was used in the simulation, Figure 1.

For our analysis, the computer program simulated a modified six-inning baseball game. Since we were trying to find the best possible lineup for a youth baseball team, we decided that the best possible lineup would be the one that scored on average the most runs per game. As a result, we only simulated each lineup batting, since pitching was not taken into account in our analysis. The first lineup we created was constructed based on the statistics compiled over the course of an entire youth baseball season from an actual team. The batters in this lineup were ordered based on hit average. The batter with the highest hit average was at the top of the lineup and the batter with the lowest hit average was at the bottom of the lineup, (descending lineup). Youth baseball teams often use the descending lineup, so for our analysis we first wanted to determine two important details: on average how many runs per game does this lineup score, and what number of games need to be simulated in order to minimize variance while maximizing efficiency?

We first needed to determine the number of games that each lineup needed to play in order to maximize the efficiency of the program while minimizing the variance of the average number of runs scored per game. To accomplish this we began by playing 10 sets of 100,000 games, 10,000 games, and 1,000 games. For each set (where one set is either 100,000 games, 10,000 games, or 1,000 games depending on the length), we recorded the average number of runs scored per game, from which we were able to calculate the average number of runs scored per game across each of the 10 sets. We then calculated the variance and standard deviation for each simulation length. We determined that simulating 1,000 games for each lineup provided our program with the greatest means of efficiency while maintaining a low variance of the results.

For each trial, the simulation had each lineup play six innings to determine the average number of runs scored per game. Each game simulated a youth baseball game in that the team batted in a fixed batting order for six innings with three outs per inning. The total number of runs scored in those six innings was calculated. This process of playing six-inning games was repeated 1,000 times for each lineup. As the trial was going through the 1,000 games, a runs per game average was maintained for the lineup. At the end of the 1,000 games the lineup's runs per game average was then stored in an external file.

For an 11 player youth baseball team, the descending lineup can be thought of as the first possible ordering of the players. Each subsequent ordering can be though of as a permutation of the descending lineup. In designing our computer program, we created the descending lineup as our first lineup and then had the program walk through all of the potential permutations of that lineup. The program would conduct a trial for the descending lineup, and then using the function next_permutation in C++ the program simulated a trial for the next possible lineup. This process of conducting trials continued until a trial of the last permutation of the descending lineup was simulated.

For our simulation, we made several assumptions to ensure that all of the trials ran efficiently. Since we were only concerned with the average number of runs scored per game, we decided that we could look at only the innings that a team was batting. This decision reduced the number of computations by 50%. Additionally, we had to make assumptions for how often each type of hit occurred. While the batting statistics provided had

total hits, total walks, total strike outs, and total at bats for each batter, the statistics for the number of singles, doubles, triples, and homeruns was not maintained. So, we estimated that 50% of all at bats were singles, 35% were doubles, 15% were triples and 0% were home runs. We did not consider home runs in our analysis because they do not occur that often in youth baseball. Other assumptions that we made include that only the hitter could make an out; as a result, baserunners could not make an out, so there were no double or triple plays. Also, baserunners were not allowed to steal bases and could not be caught stealing a base. Taking all of these factors into account, a run was only allowed to be scored after a batter made either a hit, a walk, or a contact.

3 Results

After running our simulation to conduct trials for all possible lineups, we were able to find that there exist lineups that are projected to produce more runs per game than the descending lineup. Our simulation calculated that over the course of 1,000 games, the descending lineup scored on average 7.924 runs per game. After 1,000 games, we found that the alternating lineup averaged 7.7913 runs per game. Noticing that the descending lineup scored on average more runs per game than the alternating lineup, our next step was to calculate the statistical significance of both results. To accomplish this, we calculated the variance of each team's runs per game. Our calculations found that the variance of the descending lineup's runs per game average to be 7.924 ± 0.035. Also, the variance of the alternating lineup's run per game average was found to be 7.7913 ± 0.0074. Since $7.924 - 0.035 = 7.889 > 7.7987 = 7.7913 + 0.0074$ we can see that our result is statistically significant. Thus the descending lineup scores on average more runs per game than the alternating lineup. Upon answering our initial question, we next wanted to find out if the descending lineup was the best way to order players in a youth baseball team's lineup.

To tackle this problem, we ran our simulation to walk through all of the possible permutations of lineups and have each lineup play $1,000$ games and record the average runs per game. Of all of the permutations, we recorded all of the lineups that scored on average at least 8.3 runs per game. We chose 8.3 runs per game as our minimum in order to make sure that the difference between a lineup and the descending lineup's run per game average was much greater than the variance of both lineup's run per game average. Our simulation returned 4,668 lineups that all scored greater than 8.3 runs per game. The top ten scoring lineups are shown in Figure 2. We can see that the lineup that scores the greatest number of runs per game, 8.758, scores nearly 1 run more per game than the descending lineup.

4 Discussion

Our simulation found that there do in fact exist lineups that score more runs per game on average than the traditional descending lineup. While there are some better lineups than the descending lineup, they are so few that the descending lineup would typically win. The descending lineup in our simulation outperformed over 99% of all $39,916,800$ lineups. In fact only 0.00012 % of all the possible lineups scored on average greater than 8.3 runs per game. This fact perhaps explains why so many youth baseball teams use the descending lineup every game. When considering the ultimate lineup for a youth baseball team, however, we notice that there are certain intrinsic qualities shared amongst the top ten run scoring lineups.

Considering the top ten lineups shown in Figure 2, we notice that eight out of the ten lineups have the batters numbered 0-4 grouped together. This group of batters is found most often in the middle of the lineup. Additionally, we notice that nine out of the ten lineups have a batter with a high walk average in the first batting position. Of these lineups, seven of them have at least the first two positions, however no more than the first three, in the lineup filled by batters with a high walk average. Moreover, we find that batters 0 and 1, the best two batters on the team, are found batting sequentially in eight out of the ten lineups.

Of the ten highest scoring lineups, we see some common characteristics in the lineups. When creating the ultimate lineup for youth baseball, it is the grouping of batters that is most important. In general we see that it is important to have a group of one to three batters with high walk averages at the beginning of the lineup. This group is then followed by some ordering of the team's five best hitters. We see that in this grouping, the best two hitters typically bat consecutively. The remainder of the lineup consists of the batters that have the lowest on base average. By having at the top of the lineup a group (on base group) of one to three batters that have high walk averages, we see that the group (hitting group) of the best hitters then have an opportunity to

drive in some runs. On the other hand, the descending lineup has the best batter batting first, so when it is that batter's turn to first bat there is not anyone on the bases to drive in. So, instead of having the opportunity to drive in a run, the batter must instead focus on getting on base and rely on his teammate's to drive him in.

Another characteristic of the ultimate lineup is that the grouping of the best batters typically starts with either the second, third, or the fourth position of the lineup. We believe that this design seeks to maximize the number of runs that can be scored in the first inning. Since there are three outs in each inning, there is a high probability that at least one of the batters in the on base group will get on base safely. Therefore, when the batters in the hitting group come to bat, more often than not there is at least one batter on base to drive in.

We also see that batters higher in the lineup have more at bats than batters lower in the lineup. The traditional descending lineup ensures that the best hitter will have the most at bats on the team. The ultimate lineup, on the other hand, seeks to get a batter, outside of the hitting group, to get on base as often as possible so that the hitting group can drive in the most number of runs possible. The ultimate lineup achieves this by placing the hitting group in the middle of the lineup. This provides a balance between getting batters with high walk averages on base and providing the batters in the hitting group a large number of at bats.

While these are the characteristics that have been shown to be intrinsic to the ultimate lineup for this youth baseball team, further research is necessary to determine if the ultimate lineup for a youth baseball team depends on the batting statistics of the team. Although preliminary analysis suggests that the ultimate lineup is team dependent, further research will be able to tell us how the lineup is affected by various team's batting statistics. One shortfall of this program is the duration that it takes to run.

Currently, the program takes approximately 11 hours to simulate 1,000 games for each lineup for a given team. One future goal is to look at further streamlining the simulation to reduce the run time while maintaining a low variance of the runs per game.

Conclusion

Our results show that in youth baseball the descending lineup produces on average more runs per game than 99t% of the 39,916,800 possible lineups for an 11 player team. In response to our initial question, whether an alternating lineup produced on average more runs per game than the descending lineup, we found that the descending lineup had 7.9716 runs per game while the alternating lineup had 7.7913 runs per game. We can conclude that the descending lineup does score more runs per game than the alternating lineup and that this is statistcally significant. After completing this problem, we wanted to then find the ultimate lineup, the lineup that scored the most runs per game for a given team's statistics.

From analyzing our results, we notice several important characteristics of the ultimate lineup: first, the first few batters at the beginning of the lineup (the on base group) are comprised of batters with high walk averages and low hit averages; second, the group of batters in the middle of the lineup (the hitting group) are made up of the batters with the highest hit averages on the team; lastly, the remaining positions are filled with the batters who have the lowest on base averages. Although the descending lineup produces more rusn per game than 99% of all possible lineups, there exist lineups that can produce more runs per game than the descending lineup.

Batter	Single	Double	Triple	Homeun	OBA	AA	Strike Outs
0	186	316	372	372	698	907	1000
1	155	264	310	310	6655	690	1000
2	155	264	310	310	500	738	1000
3	145	246	289	289	553	817	1000
4	92	157	184	184	553	738	1000
5	55	93	109	109	326	522	1000
6	35	60	70	70	372	419	1000
7	24	40	47	47	488	489	1000
8	0	0	0	0	512	588	1000
9	0	0	0	0	500	500	1000
10	0	0	0	0	366	439	1000

Figure 1: The actual statistics of eleven batters taken from a youth baseball team that were used in our analysis.

Lineup	1	2	3	4	5	6	7	8	9	10	11	Runs Per Game
1	7	**1**	**0**	**3**	**4**	**2**	5	6	10	8	9	8.75831
2	7	8	9	**0**	**1**	**4**	**3**	**2**	5	6	10	8.7439
3	9	8	7	**3**	**1**	**0**	**2**	**4**	5	10	6	8.69195
4	9	7	8	**0**	**1**	**3**	**4**	**2**	6	5	10	8.68042
5	9	8	**1**	**4**	**3**	**0**	**2**	6	5	10	7	8.66808
6	7	8	9	**4**	**1**	**0**	**2**	**3**	6	10	5	8.64021
7	9	8	0	1	4	2	5	7	3	10	6	8.63401
8	8	**1**	**4**	**0**	**3**	**2**	6	5	9	10	7	8.62585
9	**0**	**1**	**4**	**3**	**2**	5	10	6	8	9	7	8.62031
10	9	8	1	0	4	3	5	2	10	7	6	8.61824

Figure 2: The ten lineups with the highest runs per game average returned from our simulation.

Appendix

```
#include <algorithm>
#include <cstdlib>
#include <ctime>
#include <fstream>
#include <iomanip>
#include <iostream>
using namespace std;

const int N = 11;

int stats[N][8] = {
//Batter #, Single, Double, Triple, Home Run, Walk, Advancement, Out
       { 0, 186, 316, 372, 372, 698, 907, 1000},
       { 1, 155, 264, 310, 310, 655, 690, 1000},
       { 2, 155, 264, 310, 310, 500, 738, 1000},
       { 3, 145, 246, 289, 289, 553, 817, 1000},
       { 4, 92, 157, 184, 184, 553, 738, 1000},
       { 5, 55, 93, 109, 109, 326, 522, 1000},
       { 6, 35, 60, 70, 70, 372, 419, 1000},
       { 7, 24, 40, 47, 47, 488, 489, 1000},
       { 8, 0, 0, 0, 0, 512, 588, 1000},
       { 9, 0, 0, 0, 0, 500, 500, 1000},
       { 10, 0, 0, 0, 0, 366, 439, 1000}};
int sim_game(int lineup[])
{
       int run = 0, out = 0;
       int first = 0, second = 0, third = 0;
       while(out < 18)
       {

              for( int j = 0; j < N; j++)
              {
                     if( out == 18 )
                            break;
                     int number = rand() % 1001;
                     if( number <= stats[lineup[j]][1] ) //Single
                     {
                            if(first == 0 && second == 0 && third == 0) //Bases Empty
                            {
                                   first = 1;
                            }
                            else if(first == 1 && second == 0 && third == 0) //Runner of First
                            {
                                   first = 1;
                                   second = 1;
                            }
                            else if(first == 1 && second == 1 && third == 0) //Runners on First and
                                Second
                            {
                                   first = 1;
                                   second = 1;
                                   third = 1;
                            }
                            else if(first == 1 && second == 0 && third == 1) //Runners on First and
                                Third
                            {
                                   run++;
                                   first = 1;
                                   second = 1;
                                   third = 0;
```

```
                }
                else if(first == 1 && second == 1 && third == 1) //Bases Loaded
                {
                        run++;
                        first = 1;
                        second = 1;
                        third = 1;
                }
                else if(first == 0 && second == 1 && third == 0) //Runner on Second
                {
                        first = 1;
                        second = 0;
                        third = 1;
                }
                else if(first == 0 && second == 1 && third == 1) //Runners on Second and
                    Third
                {
                        run++;
                        first = 1;
                        second = 0;
                        third = 1;
                }
                else if(first == 0 && second == 0 && third == 1) //Runner on Third
                {
                        run++;
                        first = 1;
                        third = 0;
                }
        }
        else if( number > stats[lineup[j]][1] &&
                number <= stats[lineup[j]][2] ) //Double
        {
                if(first == 0 && second == 0 && third == 0) //Bases Empty
                {
                        second = 1;
                }
                else if(first == 1 && second == 0 && third == 0) //Runner of First
                {
                        first = 0;
                        second = 1;
                        third = 1;
                }
                else if(first == 1 && second == 1 && third == 0) //Runners on First and
                    Second
                {
                        run++;
                        first = 0;
                        second = 1;
                        third = 1;
                }
                else if(first == 1 && second == 0 && third == 1) //Runners on First and
                    Third
                {
                        run++;
                        first = 0;
                        second = 1;
                        third = 1;
                }
                else if(first == 1 && second == 1 && third == 1) //Bases Loaded
                {
                        run = run + 2;
                        first = 0;
                        second = 1;
```

```
                third = 1;
        }
        else if(first == 0 && second == 1 && third == 0) //Runner on Second
        {
                run++;
                second = 1;
        }
        else if(first == 0 && second == 1 && third == 1) //Runners on Second and
            Third
        {
                run = run + 2;
                second = 1;
                third = 0;
        }
        else if(first == 0 && second == 0 && third == 1) //Runner on Third
        {
                run++;
                second = 1;
                third = 0;
        }
}
else if( number > stats[lineup[j]][2] &&
        number <= stats[lineup[j]][3] ) //Triple
{
        if(first == 0 && second == 0 && third == 0) //Bases Empty
        {
                third = 1;
        }
        else if(first == 1 && second == 0 && third == 0) //Runner of First
        {
                run++;
                first = 0;
                third = 1;
        }
        else if(first == 1 && second == 1 && third == 0) //Runners on First and
            Second
        {
                run = run + 2;
                first = 0;
                second = 0;
                third = 1;
        }
        else if(first == 1 && second == 0 && third == 1) //Runners on First and
            Third
        {
                run = run + 2;
                first = 0;
                third = 1;
        }
        else if(first == 1 && second == 1 && third == 1) //Bases Loaded
        {
                run = run + 3;
                first = 0;
                second = 0;
                third = 1;
        }
        else if(first == 0 && second == 1 && third == 0) //Runner on Second
        {
                run++;
                second = 0;
                third = 1;
        }
```

```
        else if(first == 0 && second == 1 && third == 1) //Runners on Second and
            Third
        {
                run = run + 2;
                second = 0;
                third = 1;
        }
        else if(first == 0 && second == 0 && third == 1) //Runner on Third
        {
                run++;
                third = 1;
        }
}
else if( number > stats[lineup[j]][3] &&
        number <= stats[lineup[j]][4] ) //Homerun
{
        if(first == 0 && second == 0 && third == 0) //Bases Empty
        {
                run++;
        }
        else if(first == 1 && second == 0 && third == 0) //Runner of First
        {
                run = run + 2;
                first = 0;
        }
        else if(first == 1 && second == 1 && third == 0) //Runners on First and
            Second
        {
                run = run + 3;
                first = 0;
                second = 0;
        }
        else if(first == 1 && second == 0 && third == 1) //Runners on First and
            Third
        {
                run = run + 3;
                first = 0;
                third = 0;
        }
        else if(first == 1 && second == 1 && third == 1) //Bases Loaded
        {
                run = run + 4;
                first = 0;
                second = 0;
                third = 0;
        }
        else if(first == 0 && second == 1 && third == 0) //Runner on Second
        {
                run = run + 2;
                second = 0;
        }
        else if(first == 0 && second == 1 && third == 1) //Runners on Second and
            Third
        {
                run = run + 3;
                second = 0;
                third = 0;
        }
        else if(first == 0 && second == 0 && third == 1) //Runner on Third
        {
                run = run + 2;
                third = 0;
        }
```

```
}
else if( number > stats[lineup[j]][4] &&
        number <= stats[lineup[j]][5] ) //Walk
{
        if(first == 0 && second == 0 && third == 0) //Bases Empty
        {
                first = 1;
        }
        else if(first == 1 && second == 0 && third == 0) //Runner of First
        {
                first = 1;
                second = 1;
        }
        else if(first == 1 && second == 1 && third == 0) //Runners on First and
            Second
        {
                first = 1;
                second = 1;
                third = 1;
        }
        else if(first == 1 && second == 0 && third == 1) //Runners on First and
            Third
        {
                first = 1;
                second = 1;
                third = 1;
        }
        else if(first == 1 && second == 1 && third == 1) //Bases Loaded
        {
                run++;
                first = 1;
                second = 1;
                third = 1;
        }
        else if(first == 0 && second == 1 && third == 0) //Runner on Second
        {
                first = 1;
                second = 1;
        }
        else if(first == 0 && second == 1 && third == 1) //Runners on Second and
            Third
        {
                first = 1;
                second = 1;
                third = 1;
        }
        else if(first == 0 && second == 0 && third == 1) //Runner on Third
        {
                first = 1;
                third = 1;
        }
}
else if( number > stats[lineup[j]][5] &&
        number <= stats[lineup[j]][6] ) //Contact Out
{
        out++;
        if(out == 3 || out == 6 || out == 9 ||
                out == 12 || out == 15 )
        {
                first = 0;
                second = 0;
                third = 0;
        }
```

```
        else if( out == 18)
                break;
        else
        {
                if(first == 0 && second == 0 && third == 0) //Bases Empty
                {
                        first = 0;
                }
                else if(first == 1 && second == 0 && third == 0) //Runner of First
                {
                        first = 0;
                        second = 1;
                }
                else if(first == 1 && second == 1 && third == 0) //Runners on
                    First and Second
                {
                        first = 0;
                        second = 1;
                        third = 1;
                }
                else if(first == 1 && second == 0 && third == 1) //Runners on
                    First and Third
                {
                        run++;
                        first = 0;
                        second = 1;
                        third = 0;
                }
                else if(first == 1 && second == 1 && third == 1) //Bases Loaded
                {
                        run++;
                        first = 0;
                }
                else if(first == 0 && second == 1 && third == 0) //Runner on Second
                {
                        second = 0;
                        third = 1;
                }
                else if(first == 0 && second == 1 && third == 1) //Runners on
                    Second and Third
                {
                        run++;
                        second = 0;
                }
                else if(first == 0 && second == 0 && third == 1) //Runner on Third
                {
                        run++;
                        third = 0;
                }
        }
}
else if( number > stats[lineup[j]][6])
{
        out++;

        if(out == 3 || out == 6 || out == 9 ||
                out == 12 || out == 15 || out == 18)
        {
                first = 0;
                second = 0;
                third = 0;
        }
```

```
                        if( out == 18)
                                break;
                }
        }
    }
    return run;
}


void print(int a[], int n)
{
    for(int i = 0; i < n; i++)
        cout << a[i] << ' ';
    cout << endl;
}

int main()
{
    ofstream fout;
    fout.open("battingstats.txt");

    srand(time(NULL));
    int lineup[N];
    for(int i = 0; i < N; i++)
        lineup[i] = i;

    double best = 0;

    double average = 0;
    int num_games = 100000;
    do
    {
        for(int i = 1; i <= num_games; i++)
        {
            average = average + sim_game(lineup);
        }
        average = average / num_games;
        if(average > best)
            best = average;
        if(average > 8.3)
        {
            cout << average << ' ';
            fout << average << ' ';
            for(int i = 0; i < N; i++)
                fout << lineup[i] << ' ';
            fout << endl;
            print(lineup, N);
        }
    }while( next_permutation(lineup, lineup + N) );

    fout.close();
    system("pause");
    return 0;
}
```