A Probability Theory Framework for Baseball Strategy and Simulations

Kaz Schmanski

Submitted under the supervision of Dr. Mike Weimerskirch to the University Honors Program at the University of Minnesota - Twin Cities in partial fulfillment of the requirements for the degree of Bachelor of Arts, summa cum laude in Mathematics.

Spring 2016

Abstract

In this paper, we look to answer specific questions about the game of baseball in the MLB. These questions pertain to in-game strategies such as base stealing, questions regarding optimal ways of modeling baseball simulations, and recommendations to teams regarding these results. The paper uses a probability theory framework focused on Markov chains to analyze play-by-play data from the 2014 Major League Baseball season. The data is analyzed in aggregate and also in specific cases of player performance. We hope these questions, results, and recommendations can influence further work done in not only baseball analytics, but analytics for the sports world as a whole. The results of this paper are focused on answering questions pertaining to advice to Major League managers and improving baseball board games. Regarding advice to managers, we provide recommendations pertaining to optimal base stealing strategies and lineup optimization techniques. We also provide recommendations for improvement in three aspects of baseball board games. The first recommendation suggests that pitchers pitch more effectively out of the windup position compared to the stretch position, and in order to create a more realistic simulation, board games should account for this difference. The second recommendation analyzes the cases of conditional walks in baseball. We determine that certain players are able to walk at different rates conditional on the plate appearance situation. Finally, we analyze the process of base stealing within a board game and provide a better metric for determining how certain baserunners are able to steal against certain pitchers. We hope these insights can offer improvements to in-game management and in designing better baseball simulations.

Contents

1 Introduction

1.1 Background

In this paper, we hope to gain a better general understanding into the game of baseball as it pertains to strategy, analysis, and its simulations. In order for managers to consider the optimal strategies for a team, it is wise to consider all of the data before making an informed decision regarding the best way to go about managing a game. For this purpose, analytics are becoming an increasingly important aspect in not only professional baseball, but other sports around the world as well. For example, in 2013 the NBA began using a new technology called player tracking as a method of collecting more precise and detailed statistics. In every NBA arena, multiple cameras are positioned to capture the entire court, and the cameras track individual players as well as the ball movement to record statistics like distance covered by each player, their defensive impact, touches per game, and much more. This new technology allows teams to better analyze player performance with these advanced statistics. Clearly, we can see that analytics are important not only in the Major League Baseball, but in other sports as well. In this paper, we will use a Markov chain model as well as a Python baseball inning simulation to analyze base stealing, lineup optimization, pitchers' effectiveness out of the windup compared to the stretch, and players' abilities to draw walks and get successful jumps when base stealing. We hope to gain some insight into these aspects of the game in order to make specific strategic recommendations, suggestions to improve simulations, and to provide some groundwork for further research to be done in baseball.

1.2 Outline

This paper will be organized in the following way. First, we will talk about the data we collected and how we will use the data to draw insights. Next, we will walk through both the Markov chain model and Python simulation to illustrate the ways that we came up with our tools used to analyze the data. Following the demonstration of these tools, we will separate the analysis into two sections: advice to major league managers and baseball simulations. In the advice to major league managers section, we will analyze base stealing strategies and methods of optimizing a lineup. Regarding baseball simulations, we consider current board game situations and use data to suggest improvements to the game. The analysis will be followed by a discussion and then a conclusion. The Appendix will finish off the paper, which will include the Python program and any other supplementary material left out of the main paper.

2 The Data

The data collected for the purposes of this paper came primarily from Retrosheet.org with some supplemental data coming from MLB.com. Retrosheet is a volunteer organization dedicated to collecting data as far back as the beginning of Major League Baseball. The data specifically used for this project was the 2014 play-by-play data for 29 MLB teams. The St. Louis Cardinals home data was the only data not included in the project. This data was not included because of issues dealing with the extraction of the data from Retrosheet and conversion of this data into Excel format. In regards to the 81 games missing from the data, this should not have a significant impact on the results for the purposes of this project. The data missing represents roughly three percent of the data from the season, so the data collected should be representative of the

whole set.

The data from Retrosheet consists of every play that occurred in the 2014 season. A "play" constitutes an event that has significance in terms of the change in the state of the game. Examples of plays are hits, outs, walks, hit by pitches, stolen bases, caught stealing, errors, errors on dropped foul balls, defensive indifferences, balks, etc. The play by play data does not include pitch by pitch data. The data was extracted from Retrosheet was converted to a CSV format and then saved into an Excel Workbook for analysis. The data is set up in a way that illustrates how every play from the 2014 season occurred. Each row in the Excel sheet represents one play. For each play, a large variety of metrics are recorded, including the teams, the location of the game, and the date, as well as aspects of the specific game situation such as current ball/strike count, number of outs, what bases are occupied by what specific players, who is batting, who is pitching, and other useful data. Each play is specified as either a "batter event," in which case the play occurring ended up involving the batter completing a plate appearance, or a "non-batter event," in which case after this play, the current player batting remained batting for the next play. An example of a non-batter event involves a stolen base, caught stealing, defensive indifference, dropped foul ball, a balk, and others. Due to the reliable and simple way the play-by-play data is organized, we can easily maneuver the data and analyze the data from a number of angles.

3 Tools

3.1 Markov Chain Model

We start off the data analysis by first setting up a model of an inning as a Markov chain with a 25 by 25 transition matrix. Recall that a Markov chain

is a random process that models transitions from one state to another over a state space. The transition matrix corresponding to a Markov chain models the probability of the model transitioning from one state to another. Each row in the matrix represents the starting state and each column represents the ending state. We can think of a "state" as a position in the model. For example, we could consider the weather with two states, rainy and sunny, and each day is either rainy or sunny, and it cannot be both. Each entry in the transition matrix would give the probability of the next day being either rainy or sunny, depending on what the weather is like on the current day. For illustration, let's consider the following example, which is an example of an Ehrenfest chain, credited to the Dutch physicist Paul Ehrenfest. We have two boxes, labeled Box 1 and Box 2. Suppose we also have five balls labeled 1, 2, 3, 4, and 5. Each ball is initially placed into a box at random, so the initial configuration of the balls in boxes is random. We select a random number from the set {1, 2, 3, 4, 5} and move the ball corresponding to that number to the opposite box. This procedure is repeated indefinitely, and each ball selected is independent of the previous draw. Suppose B_n is the number of balls in Box 1 after the n^{th} draw. We can model B_n as a sequence of random variables taking values from the set {0, 1, 2, 3, 4, 5}. The transition matrix corresponding to this Markov chain is displayed below.

$$
\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{array}{c}
\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 \\
\frac{1}{5} & 0 & \frac{4}{5} & 0 & 0 & 0 \\
0 & \frac{2}{5} & 0 & \frac{3}{5} & 0 & 0 \\
0 & 0 & \frac{3}{5} & 0 & \frac{2}{5} & 0 \\
0 & 0 & 0 & \frac{4}{5} & 0 & \frac{1}{5} \\
0 & 0 & 0 & 0 & 1 & 0
\end{pmatrix}
\end{array}
$$

This transition matrix models the the number of balls located in Box 1, and the probabilities with which this number changes. Since each ball has an equal probability of being selected, if we have only one ball in Box 1 at a time t, then the probability of having that ball transferred to the other box is only one in five, while the probability of gaining another ball in Box 1 (selecting and moving one of the four balls from Box 2) is four in five. Once the box is either full or empty, then the probability of respectively losing or gaining a ball is 1. An important property of a Markov chain is that the probability of transferring from one state to another is independent of its history. For our example, if we have one ball in Box 1 in time t, the probability of obtaining or losing a ball in the next transition is independent of how we got there. It doesn't matter if we are beginning the model in this state, or if it took 100 turns to get to this state, the transition probabilities will be the same. Therefore, the probability of transitioning to one state depends only on the current state. We will use this assumption to model a baseball inning. That is, the probability of obtaining any configuration of baserunners and outs depends only on the current configuration of baserunners and outs and not on the history of the process.

In the case of this paper, the state space is the set of any combination of outs and baserunners in an inning. Each state is represented uniquely by the number of outs, either zero, one, two, or three, and what bases are occupied, any combination of runners on first, second, and third. There is only one state modeling three outs, in which case there are no runners on base. The states in this model are numbered 0 through 24. State 0 represents the state of the game with no outs and no runners on base. State 24 represents the state of the game with three outs, and therefore represents the end of the inning. In order to calculate the transition probabilities from each one of the 25 states of the Markov chain, we used the play-by-play data collected from Retrosheet and sorted it accordingly. Starting from some state x, to calculate all of the probabilities of transitioning out of state x into another state y, we filtered the data into every play that occurred in 2014 in which at the time of the play, the inning was in state x. We then further divided the data into all of the possible outcomes of the play from state x. The probability of the state transitioning from state x to state y was then the number of times we arrived in state y from state x, divided by the total number of times the chain is in state x. The entire transition matrix was calculated in this fashion and is the matrix is displayed on the next page. Below the transition matrix is the State Key, which specifies the number of outs and the bases occupied for each state in the game.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	.03	.24	.05	.01	0	0	0	0	.69	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	.03	0	.01	.01	.21	.04	.04	0	0∗	.44	.11	0∗	0	0	0	0	.12	0	0	0	0	0	0	0	0
2	.02	.06	.04	.01	.11	.10	.01	0	0∗	.01	.38	.27	0	0	0	0	.01	0	0	0	0	0	0	0	0
3	.02	.17	.04	0	0	.12	0	0	.25	0∗	.01	.38	0	0	0	0	.01	0	0	0	0	0	0	0	0
4	.02	0	.01	0∗	.04	.03	.03	.16	0∗	0∗	0∗	0	.35	.10	.13	0	0	.01	.01	.09	0	0	0	0	0 ∗
5	.03	0	.02	.01	.16	.04	.05	.07	0	.18	.06	0∗	.02	.22	.04	0	.09	0∗	0∗	.01	0	0	0	0	0 ∗
6	.01	.05	.04	.01	.01	.10	.01	.14	0∗	.01	.10	.15	.01	.01	.35	0	0	0	.01	0∗	0	0	0	0	0
7	.02	0∗	.02	.01	.05	.03	.04	.16	0	0∗	0	0	.06	.09	.04	.36	0	.01	0∗	.07	0∗	.01	.04	0	0 ∗
8	0	0	0	0	0	0	0	0	.02	.24	.05	.01	0	0	0	0	.69	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	.03	0∗	.02	.01	.20	.04	.03	0	0∗	.47	.09	0∗	0	0	0	0	.12
10	0	0	0	0	0	0	0	0	.03	.06	.05	0∗	.14	.07	0∗	0	.01	.01	.44	.19	0	0	0	0	.01
11	0	0	0	0	0	0	0	0	.02	.18	.05	.01	0	.15	0	0	.20	.02	0∗	.36	0	0	0	0	.01
12	0	0	0	0	0	0	0	0	.03	0	.02	.01	.05	.03	.04	.15	0∗	0∗	.01	0∗	.38	.09	.07	0	.13
13	0	0	0	0	0	0	0	0	.02	0	.01	.01	.14	.04	.03	.08	0	.17	.05	0∗	.02	.25	.03	0	.13
14	0	0	0	0	0	0	0	0	.02	.07	.05	0∗	.01	.08	0∗	.20	0∗	.01	.09	.15	.01	.02	.27	0	.02
15	0	0	0	0	0	0	0	0	.02	0	.01	.01	.05	.04	.04	.16	0	.01	0∗	0∗	.07	.10	.05	.31	.13
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.02	.24	.04	.01	0	0	0	0	.69
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.02	0∗	.02	.01	.19	.05	.02	0	.69
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.02	.09	.05	.01	.16	.03	0	0	.66
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.02	.14	.05	0∗	0	.15	0	0	.65
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.02	0∗	.03	.01	.05	.04	.03	.12	.71
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.02	0∗	.02	.01	.10	.04	.02	.09	.71
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.02	.07	.04	.01	0	.03	0	.19	.65
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.02	0∗	.02	.01	.04	.04	.02	.12	.73
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

*different from zero to three decimal places

Table 1: State Key

State	Outs	Bases Occupied
0	0	0
1	0	1
2	0	2
3	0	3
4	0	1,2
5	0	1,3
6	0	2,3
7	0	1,2,3
8	1	0
9	1	1
10	1	2
11	1	3
12	1	1,2
13	1	1,3
14	1	2,3
15	1	1,2,3
16	2	0
17	2	1
18	2	2
19	2	3
20	2	1,2
21	2	1,3
22	2	2,3
23	2	1,2,3
24	3	x

As an example, we can see that the probability of transitioning from state 0 back to itself is about 3 percent. In total, there were 43,804 plays that occurred starting from state 0, with no outs and no runners on base. This includes the leadoff batter of an inning, plus the number of plate appearances following a home run with zero outs. The total number of plays that resulted in clearing the bases with no outs (such as a home run, a triple and an error, etc.) was 1,114. Dividing 1,114 by 43,804 gives us .025, which is rounded to .03 in our table, so there is a 2.5 percent probability that the state transitions back to state 0 from state 0. Due to rounding, it appears that some values in the table are lower or higher than their true values. For example, the table indicates

that the probability of the Markov chain transitioning from state 7 to state 24 (going from the bases loaded with no outs to three outs, in which case a triple play has occurred) has a probability of zero, or that it is impossible. The true probability of this happening based on the data is 0.0020. The Python simulation takes each value to three decimal places, so despite what is displayed in the table, the true value of a triple play occurring when the bases are full is positive, and our simulation accounts for this. All of the values with this property, the values marked zero in our matrix in which the true probabilities are greater than zero, are distinguished with an asterisk. Also note that in this model, state 24, the state representing three outs in an inning, is an absorbing state. Once the chain enters state 24, it never leaves. In the case of our paper, once the simulation reaches state 24, we consider the inning to end, tally up all of the runs scored in the inning, and then restart another inning. The details regarding this simulation will be clear in the next subsection. Finally, the gridlines in this transition matrix are used to separate the states with zero, one, two, or three outs in order to make the matrix more readable, and note that all of the transition probabilities in the gray sections are 0. This indicates that once the chain enters a state with a certain number of outs, it's impossible to transition back to a state with a fewer number of outs.

As we mentioned earlier, a key assumption regarding Markov chains is the memoryless property. Therefore, we operate under the assumption that each inning in the game is played according to the same transition matrix. We believe this is a reasonable assumption for the average MLB game, but it would be interesting to separate the data to see if certain situations produce a different transition matrix. For example, suppose if a team is losing by one run in the 9th inning. They likely play with a significantly different strategy as opposed to their "normal" strategy. This team is going to play to maximize the probability of

obtaining one run, rather than operating under the assumption of maximizing their expected number of runs, like a typical team would in a normal game situation. Therefore, we may expect the transition matrix to look different for teams in this situation. Our paper chooses to operate under the assumption that baseball is played as a Markov chain and is memoryless, and we expect the typical average game to follow this pattern.

For the purposes of this project, we will use the transition matrix in order to calculate an Expected Runs table. With this Expected Runs table, we hope to determine the expected number of runs a team will score when they're at any given point in the inning, until the end of the inning. This will become important later in the paper for strategy and board game analysis. With the concept of an Expected Runs table in mind, we turn now to the Python simulation to demonstrate how we calculate this table.

3.2 Python Simulation

The main goal with our Python simulation is to calculate the Expected Runs table, which will help us with many sections of this paper. This table will show how many runs an average team is expected to score from any state of the game until the end of the inning. This way, we can calculate how successful a base stealer has to be in order to balance the risk of getting thrown out, how many runs pitchers would save pitching out of the windup instead of the stretch, and more. To calculate the Expected Runs table, we wrote a Python computer program that acts as a baseball simulation and counts the number of runs scored in an inning. This Python program is included in the Appendix of this paper.

In Python, we begin the inning in state 0 with no runners on base and no outs with the run counter equal to 0. The program then generates a random number between 1 and 1000. This number will determine the outcome of the

play based on the transition matrix we calculated previously. For example, if the transition matrix says that there is a 2 percent probability of transitioning from one state to another, then if the random number falls within a specific range of numbers (the interval being 20 integers long), then the simulation and inning will transition to this state. Each outcome is placed into an interval from 1 to 1000, and the length of the interval depends on the probability of the event occurring based on the transition matrix we calculated previously. Depending on the specific outcome of the play, runs may be added to the runs counter. The program tracks runs in the following way. At the beginning of an inning, the run counter is set to 0. Suppose from state 0, the simulation goes again to state 0. This means that the batter went all the way around the bases without getting out, so we know to add one to the run counter. If for example, the game transitions from state 0 to state 2, with a runner on second base and no outs, and then the game transitions to state 1, where we have a runner on first with no outs, then we know that the runner on second base must have scored, since an out was not recorded, and he is not on base anymore. In this case, a run would be added to the run counter. This type of reasoning is established and accounted for in the simulation. Once three outs are reached in the simulation, the inning ends, and the number of runs scored in this inning is recorded by another total runs counter. The program written for this project simulates 300,000 innings and records the average number of runs scored when starting from each state in the game. These values give us the Expected Runs table, which is displayed below, where each row represents each number of outs and each column represents what bases are occupied.

This table displays the number of runs an average team would expect to score from any given state of the inning to the end of the inning. The first entry in the table shows us that with no outs and the bases empty, a team

Table 2: Expected Runs

	Empty	1	2	3	1 and 2	1 and 3	2 and 3	Loaded
0	0.45	0.84	1.09	1.32	1.42	1.77	1.99	2.25
1	0.24	0.49	0.63	0.91	0.89	1.10	1.47	1.53
2	0.09	0.20	0.29	0.33	0.40	0.41	0.90	0.66

should expect to score 0.45 runs until the end of the inning. Therefore, in a typical game with nine innings, we would expect a team on average to score $(0.45) * 9 = 4.05$ runs per game. In fact, according to the data from MLB.com, in 2014, the 30 MLB teams scored a total of 19,761 runs in 4,860 total games. Dividing 19,761 by 4,860 gives us 4.07 runs per game per team. This corresponds to a less than 0.5 percent discrepancy between our estimate and the data, so on average, it appears that our expected runs table is quite accurate, based on an average of nine innings per game. In fact, according to Retrosheet, the number of innings pitched in 2014 was 43,614.2, where .2 refers to two thirds of an inning (the game ends on a walk off hit for example, where there are two outs in the inning). Dividing by 4,860 games gives us an average of 8.97 innings per game. Therefore, using nine innings per game is realistic in terms of the average MLB game. Based on our table, we can see that with the bases loaded and no outs, on average a team should expect to score slightly more than two runs by the end of the inning. As the number of outs increase, we can see that runs become more difficult to obtain. We will use this Expected Runs table to calculate how successful a baserunner needs to be to justify stealing. Once we know the expected number of runs a team will score in certain states of the game, we can easily see how a stolen base or a caught stealing will impact the expected number of runs a team will score. We can also analyze player statistics and see which players contribute to increasing their team's expected number of runs scored the most, in order to generate an optimized lineup. In addition to lineup optimization, we can analyze pitcher effectiveness from the

windup and compare it to the stretch in terms of how the different outcomes from these positions affect the expected number of runs scored for the batting team. Therefore, we can see that this table will prove useful for us in many aspects for this project.

4 Analysis

4.1 Advice to Major League Managers

In this section, we analyze two aspects of in-game strategy and how managers could better manage a game to increase their team's chance to win. We analyze aspects of base stealing and formulate a method to optimize the way a manager sets up a lineup. We provide recommendations based on our data that suggest there are more effective strategies towards management of a baseball game in these areas.

4.1.1 Base Stealing

The first question we'd like to analyze is, how successful does a base runner need to be in order to compensate for the risk of being thrown out? After all, a stolen base would mean a significant increase in expected runs for the inning. For example, stealing second base with no outs increases the number of runs a team expects to score by 0.25, taken from our Expected Runs table. Getting thrown out, however, could cost a team a chance at scoring a run and possibly cost them the game as well.

In order to calculate the probability of a successful steal that takes into account the change in expected runs, we take a look at the Expected Runs table. If we want to determine how successful the base runner needs to be to justify the risk of stealing, we set equal the expected number of runs the team

scores by not stealing, to the expected number of runs with an attempted steal. This yields the following equation.

$$\mathbf{E}[runs_{stay}] = \mathbf{Pr}[success] * \mathbf{E}[runs_{success}] + \mathbf{Pr}[failure] * \mathbf{E}[runs_{failure}]$$

First, let's analyze the case of a runner on first base with no outs. To calculate how successful he needs to be on average to risk getting thrown out stealing, we take the expected number of runs scored if the runner stays put, which we see is 0.84 from our Expected Runs table, and set that equal to the expected number of runs scored if the runner attempts to steal. If the probability of successfully stealing is p, then if the runner attempts to steal, his expected number of runs scored is $p * (1.09) + (1 - p) * (0.23)$, where $(1 - p)$ is the probability of the runner getting thrown out, 1.09 is the expected number of runs the team will score if the stolen base is successful (from the Expected Runs table), and 0.23 is the expected runs the team will score if the runner is thrown out, in which case there will be one out with no one on base.

The equation $0.84 = p * (1.09) + (1 - p) * (0.23)$ yields the solution $p = 0.71$. This tells us that given the situation of no outs with a runner on first base, if the baserunner is successful in stealing on average 71 percent of the time or better, attempting to steal is worth the risk of getting thrown out. This is because a baserunner with a probability of success greater than 71 percent will increase his team's expected number of runs by attempting to steal compared to staying on first base and not attempting to steal. This number 0.71 corresponds to the entry of necessary base stealing probabilities for the situation of stealing second base with no outs. We wish to calculate the rest of these probabilities. That is, we wish to know how successful a baserunner needs to be in order to attempt to steal second base with one or two outs, and how successful they need to be to attempt to steal third base with zero, one, or two outs. The rest of the

probabilities of the table are calculated in a similar manner to before, and are displayed below.

Table 3: Necessary Base Stealing Probabilities

	Stealing Second	Stealing Third
0 outs	0.71	0.79
1 out	0.74	0.66
2 outs	0.71	0.86

The table indicates that on average, if a baserunner is on first base with any number of outs, if the baserunner has a 74 percent or greater probability of stealing second successfully, they should attempt to steal every opportunity they get because they will increase their team's expected number of runs with a steal attempt. Dee Gordon stole 64 bases in 83 attempts, giving him a success rate of just over 77 percent. Based on our table, Dee Gordon should be stealing much more often given this success rate, as he will significantly increase his team's expected number of runs.

Based on our analysis, how are teams faring with stealing bases properly? Are certain teams stealing when they shouldn't, and are certain teams not stealing enough? We examined MLB team baserunning stats from 2014, courtesy of ESPN.com, to answer this question. Of the 30 teams in the league, only eight were stealing at a better rate than 75 percent. Based on our analysis, these teams are successful enough in stealing to justify the risk of stealing more often. Therefore, the Athletics, Astros, Bluejays, Indians, Nationals, Phillies, Royals, and Yankees should consider taking advantage of their strong ability to steal bases by stealing more often. One thing we can notice about these teams is that six of these eight teams are American league teams. In baseball, base stealing is seen as more of a National League-type strategy. The rationale behind this is that the majority of the best power-hitting players play in the American League, so to make up for the lack of power in the National League, teams

need to steal more often; they need to play "small ball." Despite the American League teams' abilities to hit home runs, based on our analysis we recommend the Athletics, Astros, Bluejays, Indians, Royals and Yankees consider expanding their game into the realm of "small ball," and using their success in stealing bases to increase their expected number of runs scored per game and win more games.

The Cincinnati Reds stole the third-most bases in the league in 2014, with 122 steals in 174 attempts (a 70.1 success rate). Many of these steals probably played a significant impact on the outcome of the game, as did the caught stealing occurrences. Teams like the Cincinnati Reds and the Texas Rangers, who stole 105 bases in 164 attempts (64 percent success rate) should consider stealing less often, or choose their situations more carefully when they do choose to steal. These teams exhibit a successful steal percentage much lower than is required to balance the tradeoff between stealing the base and getting thrown out. Therefore, it appears that certain teams in the league are stealing far too often, while some teams should consider stealing more often.

There are a few limitations with our findings to point out. Often times, a stolen base is attempted for the sole purpose of getting one more run, rather than simply getting more runs in general. In late game situations, when one run can make the difference between winning and losing, a base runner is much more valuable. In our analysis, we are generalizing to the case of a team looking to maximize the number of runs they score in an inning (and therefore a game). However, in certain cases teams may be more interested in obtaining a single more run, and increasing the probability of scoring one run, rather than increasing the expected number of runs scored.

A stolen base has a certain element of surprise to it. If a base runner attempts to steal every time they're on first base, pitchers and catchers will likely do more

to prevent the runner from taking second base, in which case the probability of successfully stealing would decline. It would be unwise to conclude that based on our calculations, with no outs and a runner on first, a success rate of 71 percent or better justifies stealing every time. However, it's worth considering that the elite base stealers tend to steal at a greater rate than this threshold. Managers should always take into consideration the inning, score, the pitcher, the catcher, and other conditions, but overall, our analysis indicates that some base runners and teams should be stealing more often.

4.1.2 Lineup Optimization

Another important aspect of managing a baseball game is constructing the batting order on a daily basis to maximize the number of runs scored in order to give your team the best chance to win. Managers throughout the league follow different strategies. Some managers prefer to put the fast players in the top of the order so they can steal bases and be in scoring position when the power hitting third and fourth hitters come to the plate. Some managers opt to place players who simply get on base often in the first two spots in the order, so the highest average hitting players have a chance to bring them in. Other National League managers, namely Tony LaRussa, like to bat the pitcher in the eighth position rather than the traditional ninth position, with the mentality that if you have a player who can get on base in the 9th spot, it makes the players in the top of the lineup have a better chance of bringing in an additional run. Finally, managers like Joe Torre tended to mix up the lineup depending on what players were hitting well at the moment. Even if a player is only batting .240 on the season but has been hitting .350 in the last five games, Torre could put him in the early part of the lineup. We can see that every manager has a unique strategy when it comes to placing players in the batting order, but what strategy is optimal? Which strategy will score the greatest number of runs over

the course of the game? We will attempt to respond to this question and provide some insights from a probability-theory perspective.

In order to answer questions about lineup optimization, we will focus specifically on the 2014 San Francisco Giants roster. This team won the World Series in 2014 and has a unique lineup in the sense that they don't have any "true" power hitters. Buster Posey and Hunter Pence were the only players who hit more than 20 home runs, and Buster Posey led the team with only 22. The team batted .255 total which was good for fourth in the National League, so they certainly were a quality hitting team. We will analyze whether or not based on our model they were successful in optimizing their lineup. In order to optimize the lineup, we will focus on quantifying the number of runs each players were worth over the course of the season, and structure a lineup based on this statistic. A commonly used statistic in baseball analytics is WAR, or Wins Above Replacement. This statistic estimates the number of wins a certain player contributed to their team's effort compared to how the average major league bench player would have. A WAR of 5.0 would indicate that this player contributed five additional wins to the team's season than would be expected if that player were replaced by an average MLB non-starter. If a team consisted of players with with a cumulative 0 WAR, then the average player on that team would be the equivalent of an average major league bench player. Even the worst teams in the history of the modern era posted 45 win seasons, so we could assume a team with a cumulative WAR of 0.0 would win about 45 games per season, or about 1/4 of their games. Therefore, a team with a cumulative war of 30.0 could expect to win roughly 75 games, or nearly half of their total games. Our statistic is similar in purpose to WAR but calculated differently, and yields different results.

To calculate our statistic Runs Worth (RW), we first need to quantify how

many runs a typical single, double, triple, homerun, and walk are worth. Suppose we have a situation where the bases are empty with no outs, and a player hits a single. While this play doesn't technically score a run for the team, it does put the team in a better position to score at least one run in the inning. Therefore, while the single isn't worth one whole run, it should be worth at least part of a run. We will use the Expected Runs table and the relative frequencies of each state occurring in a game to quantify the RW for walks and all four types of hits.

Let's first focus on the case of a single. Throughout this calculation, we will be conservative and suppose that we play "station to station" baseball. In other words, if a player is on second base and the batter hits a single, we will assume the player moves up one base, even though in reality a player on second base is likely to score on a single. This will provide us with a conservative and safe estimate. On the other hand, if we have a runner on second base with third and first base empty, we will assume with probability 1 that the baserunner will advance to third base on a single. This is a reasonably accurate assumption, although it's not perfect, because we are disregarding the cases of infield singles and other situations where a base runner may not advance. We will assume for our analysis that it is always the case a baserunner will move up at least as many bases as the hit obtained, so a runner on third will always score on a single, a runner on second will always score on a double, and so on. If we start in state 0, with no runners on base and no outs, how many runs would a single be worth? After the player hits a single, we are now expected to score 0.84 runs because we now have a runner on first base with no outs. Before the single, we were expected to score 0.45 runs in the inning. Therefore, the single in this situation is worth how many expected runs we gained by a result of the play. In state 0 of the game, a single is worth $0.84 - 0.45 = 0.39$ runs. If we are in

state 1 and the batter hits a single, we assume the runner on first moves up to second and doesn't advance further. With runners on first and second with no outs, we are now expected to score 1.42 runs in the inning, compared to 0.84 runs before. Therefore, a single in this case is worth $1.42 - 0.84 = 0.56$ runs. We calculate this metric for the rest of the states of the game in the same fashion to determine how much from each state a single increases our expected runs scored in the inning. We then multiply each of these values by the proportion of time they occur to give the weighted average of how much a single is worth in a given at bat. In the case of a single, this number is 0.37. On average, a single is expected to increase the number of runs we score by 0.37. We would of course expect doubles to be worth more than singles, triples to be worth more than doubles, and home runs to be worth the most. We would also expect outs to be worth negative runs, because they decrease the expected number of runs scored in an inning. However, we do not include outs into the Runs Worth statistic. This is to keep the results positive, and to avoid "double counting" outs. Because a hit or a walk is worth positive runs, then a batter with more hits and walks is going to be worth more runs, and the more frequent they get a hit, the less frequent they get an out. Therefore, counting outs may undervalue a player's contribution to his team. Including outs into our Runs Worth statistic would also result in negative values for every player on the Giants roster, which is not a realistic measure. The results we obtained for RW are displayed in the table below.

Table 4: Runs Worth (RW)

Outcome	Runs
Single	0.372
Double	0.683
Triple	1.015
Home Run	1.402
Walk	0.315

With these numbers in hand, we can quantify the RW for players in the 2014 Giants roster to see which players produced the highest RW. To calculate the RW for a player, we use the following formula with the values coming from the RW table:

$$RW_{player} = (Singles) * (0.372) + (Doubles) * (0.683) + (Triples) * (1.015) + (HomeRuns) * (1.402) + (Walks) * (0.315)$$

The formula produces the following results for the 2014 Giants roster:

Table 5: Runs Worth (RW) for SF Giants

Rank	Player	Position	Runs Worth
1	Hunter Pence	RF	119.40
2	Buster Posey	C	110.71
3	Pablo Sandoval	3B	99.81
4	Brandon Crawford	SS	86.54
5	Mike Morse	LF	83.52
6	Gregor Blanco	OF	65.47
7	Angel Pagan	CF	61.58
8	Joe Panik	2B	40.98
9	Brandon Belt	1B	39.86
10	Brandon Hicks	2B	33.10
11	Joaquin Arias	IF	23.56
12	Tyler Colvin	LF	21.15
...	...	...	...

Due to the small sample size of plate appearances from pitchers, we don't see any pitchers in this table. In fact, Madison Bumgarner is the highest ranking pitcher on the team in terms of RW, and he was worth approximately 11 runs, so we can see that even the best hitting pitchers generally contribute very little to a team's runs. Also note that because of the Giants being a National League team, they're unable to use a designated hitter unless playing at an American League park during interleague play. This table indicates that Pence, Posey,

and Sandoval were by far the best hitters on the club in terms of how many runs their production was worth over the course of the season. Based on this knowledge, how do we order the lineup to maximize our expected number of runs we would score in the game? There are a number of theories for the best way to optimize a lineup, and we will examine multiple. The first strategy we will focus on is arranging the lineup in order from best to worst hitters; a simple, yet likely effective strategy. The second strategy we will cover involves putting your best hitters in the number 1, 4, and 2 spots in that order, according to Tom Tango's "The Book: Playing the Percentages in Baseball." Tango's reasoning for this order of batters will be apparent shortly.

A simple way of looking at a lineup is the fact that the players higher up in the lineup will get more chances to bat. Therefore, we should place the best hitters in the early part of the lineup in order to maximize their opportunities to bat. If we wanted to order the players in descending order of RW, the lineup for the Giants would appear as follows.

Table 6: SF Giants Batting Order

Lineup Position	Player	Position	Runs Worth
1	Hunter Pence	RF	119.40
2	Buster Posey	C	110.71
3	Pablo Sandoval	3B	99.81
4	Brandon Crawford	SS	86.54
5	Mike Morse	LF	83.52
6	Gregor Blanco	CF	65.47
7	Joe Panik	2B	40.98
8	Brandon Belt	1B	39.86
9	Pitcher		

This configuration is particularly interesting because for the majority of the season, Buster Posey typically batted in either the third or fourth position. Because Posey gets many extra base hits, it's seen as a better thing if there are runners on base when he bats so he can drive them in. Meanwhile, in our chart

Angel Pagan and Joe Panik are towards the bottom of the lineup (in Pagan's case he isn't even one of the three best outfielders), while these players spent significant time in the first or second spots of the Giants lineup. The data may suggest that Joe Panik, who was only worth 41 runs in 2014, may not be the best candidate for a leadoff or second hitter, and a player like Brandon Crawford who normally batted towards the bottom of the lineup may be a more desirable leadoff hitter.

Tom Tango's "The Book" covers many topics including base stealing, bunting, and lineup optimization. According to "The Book," your best hitters should go in places in the order where getting out is most costly. In other words, each place in the batting order is ranked in terms of how important it is to avoid an out in this position based on the state of the game (outs and baserunners) and weighted by how often the lineup position occurs. Tango ranks the following lineup positions as the most important regarding avoiding outs: 1, 4, 2, 5, 3, 6, 7, 8, 9. Tango's findings provide an alternative idea to the traditional baseball theory that your best hitter should be in the third batting spot. If we use our metric of Runs Worth and use this as a way to measure how "good" a player is, and use Tango's method of lineup optimization, we obtain the following configuration:

Table 7: SF Giants Batting Order, Tango Method

Lineup Position	Player	Position	Runs Worth
1	Hunter Pence	RF	119.40
2	Pablo Sandoval	3B	99.81
3	Mike Morse	LF	83.52
4	Buster Posey	C	110.71
5	Brandon Crawford	SS	86.54
6	Gregor Blanco	CF	65.47
7	Joe Panik	2B	40.98
8	Brandon Belt	1B	39.86
9	Pitcher		

Again, we see Joe Panik towards the bottom of the lineup with Angel Pagan not making the starting lineup altogether, while Buster Posey bats in a position we would typically see, the fourth position. It would be interesting to run a simulation to test how many runs on average each of these lineups would score and compare which method would be more effective. Alternatively, there are other ways to rank players in terms of hitting ability. Some managers prefer to stack the players with the highest on base percentages in the top of the lineup, while others stick to the philosophy of putting your top two power hitters in the 3rd and 4th spots. An alternative would be ranking players according to WAR and placing them in the lineup due to Tango's, or another seemingly successful structure, such as a method using Linear Weights. Linear Weights is a system suggested by Thorn and Palmer in their 1985 book "The Hidden Game of Baseball." The set up of Linear Weights is similar to our system using Runs Worth. Each at bat outcome is given a weight based on how much the outcome on average will increase or decrease the expected number of runs scored. The linear weights for singles, doubles, triples, home runs, and walks are similar to our values obtained using Runs Worth, and linear weights also includes stolen bases, caught stealing, and outs. The authors also established linear weights for pitching performances. We chose to leave out base stealing statistics in order to analyze the effect of batter performance rather than effects of base running. We also chose to leave events like fly outs and groundouts out of the analysis for the sake of risking "overcounting" negative events. Essentially, the more positive events the batter obtains, the more runs he will be worth compared to the other players. Therefore, players who obtain fewer positive outcomes may be penalized more if we chose to include fly outs and groundouts into the Runs Worth statistic. Clearly, there may be no one perfect way to quantify player performance to one simple statistic, and to therefore adjust a lineup

accordingly. Either way, it appears that there are numerous ideas as to how to rank players and structure the lineup, and we hope the insight creating a Runs Worth statistic can provide insight towards further research.

4.2 Baseball Simulations

In this section, we take a look at optimizing aspects of a baseball simulation. The baseball simulations we are attempting to improve are board games in which a series of random numbers chooses the outcome of each plate appearance. There are certain aspects of the game that allow for randomness to appear; the weather can have an effect on the distance a fly ball travels, the umpires may have an effect on a safe versus out call, and we may have some bizarre events like fan interference. As many aspects as these board games cover, there are a few that we suggest could improve the game and make it more realistic and in line with data. We suggest that pitchers perform better out of the windup compared the stretch, and players are able to walk more or less effectively based on the situation with which they're faced. We also conclude that poor baserunners are less likely to attempt to steal against a pitcher bad at holding runners on base than the typical linear model suggests. These are aspects that are not incorporated into these simulations, so they provide room for these games to improve.

4.2.1 Pitcher Effectiveness - Windup or Stretch?

In order to understand the game more in depth and develop a more realistic board game simulation, we take a look at pitching performance and how it compares when a pitcher throws from the windup versus throwing from the stretch. Traditionally, nearly all pitchers throw from the stretch if there is a runner on base and from the windup with no runners on base. There are cases

of some pitchers not using the windup at all, as well as situations where a pitcher will deliver from the windup with a runner on third base, runners on second and third, or when the bases are loaded. However, for the sake of our analysis we will ignore these rare cases and focus on comparing pitchers with the bases empty, assuming they will use the windup, and contrasting with their performance when there is at least one base occupied, in which case we assume they will be using the stretch. Traditional baseball logic says that pitchers are more comfortable, and therefore more effective, when there are no runners on base. Therefore, under the traditional logic, it would make sense if we included these pitching differences into a baseball board game in order to more accurately model pitching performance. We will keep this hypothesis in mind as we go into the analysis.

In order to answer the question, "are pitchers more effective out of the windup or the stretch?" we divide the play-by-play data from Retrosheet into two groups. One group, the "windup" group, consists of every play that occurred in which at the time of this play, there were no runners on base. The other group, the "stretch" group, consisted of data from every play in which there was at least one runner on base. Over the course of the 2014 season, 100,826 plays occurred with no one on base, and 76,956 plays occurred in which there was at least one base occupied. In order to determine a pitcher's effectiveness, each of these situations were broken down into the following outcomes: generic out (which is considered either a groundout, lineout, or flyout), strikeout, walk, hit by pitch, error, fielder's choice, single, double, triple, and home run. Intentional walks were excluded for the sake of consistency, as there were only 10 intentional walks with no runners on base. Intentional walks may also skew pitching data and effectiveness because most of the time, an intentional walk is issued because of a specific game situation or hitter at the plate, and generally a pitcher giving

up an intentional walk is not illustrative of his performance, but rather an indicator of the specific in-game situation. Therefore, for the purposes of this paper we omit intentional walks. The rationale behind separating each play into these categories is to determine whether the ability for a pitcher to get an out via generic out, strikeout, or fielder's choice is significantly higher out of the windup than out of the stretch. Another way to think of a pitcher's effectiveness is essentially the ability for a pitcher to get an out. A pitcher wants to keep runs from scoring, and therefore, getting a runner out is eliminating his opportunity to score, so we can consider "better" pitcher performance to be the case where a pitcher obtains an out a larger percentage of the time. Therefore, a higher percentage of outs when a pitcher is pitching from the windup would indicate that the pitcher is indeed more effective out of the windup than out of the stretch.

Of the 100,826 plays that occurred with no runners on base, the number of strikeouts was 21,317, or 21.1 percent. Of the 76,956 plays occurring with at least one runner on base, a total of 12,989 strikeouts occurred, or 19.5 percent of total plays. We would like to find out whether or not this difference is statistically significant. If it is, we can conclude that the higher percentage of strikeouts occurring with no runners on base (a pitcher pitching from the windup, which is theoretically more comfortable and effective) is not due to random chance. Essentially, statistical significance would tell us that a pitcher's ability to get a strikeout is statistically higher when he is pitching out of the windup, and that it is not due to randomness in the data. In order to calculate if this difference is significant, we use an online p-value calculator from Social Science Statistics. This calculator gives the probability that, based on the number of observations and proportions of events that occurred in both groups, we would have obtained these results due to random chance alone. A high p-value indicates that there

is a relatively large chance that the data could have been obtained due to randomness, and that there is likely no causation. Therefore, we want to keep the risk of claiming causality when there is none, so we will use a significance level of .05, meaning that any p-value less than .05 we will reject the hypothesis that there a pitcher is not more effective out of the windup, and any p-value higher than .05 we will be unable to say that a pitcher is more or less effective out of the windup compared to the stretch.

This p-value calculator for the strikeout case gives us a p-value of 0.0001. This indicates that there is a 0.01 percent likelihood that the ratio of strikeouts occurring from the windup is not statistically higher than the ratio of strikeouts from the stretch. Therefore, we can conclude from this p-value that there is a statistically significant increase in strikeouts when a pitcher is pitching from the windup. In essence, we've just shown that without a doubt, pitchers are more effective in terms of striking batters out when they're on the mound with no runners on base. The remaining results are displayed in the table below, as well as a key metric, looking at pitcher's effectiveness in generally being able to get a batter out, using either a strikeout, generic out, or fielder's choice.

Table 8: Pitcher Performance, Percentage of Total Outcomes

Outcome	Windup	Stretch	Windup Minus Stretch
Generic Out (GO)	47.77	47.34	0.43
Strikeout (K)	21.14	19.72	1.42
Base on Balls (Walk)	6.91	7.39	-0.48
Hit By Pitch (HBP)	0.80	1.00	-0.20
Ability to Get an Out (GO + K + FC)	68.91	67.73	1.18

We're not only interested in a pitcher's effectiveness in striking batters out in these two different scenarios. To measure a pitcher's overall effectiveness, we'd like to measure the proportion of outs they get out of the windup versus out of the stretch and compare the two to see if we can find some sort of statistical significance. In order to compare the proportion of outs in each of the two

cases, we added up the number of generic outs, strikeouts, and fielder's choices for plays that occurred from the windup and the stretch. An out occurred 68.91 percent of the time out of the windup, and 67.73 percent of the time out of the stretch. Using the p-value calculator we obtain a p-value of 0.00001. This leads us to conclude that pitchers are able to record an out a statistically significantly higher proportion of the time when pitching out of the windup than when pitching out of the stretch. Based on our results here, it appears that pitchers are certainly more effective out of the windup compared to the stretch, in terms of being able to record outs more frequently from the windup, and also giving up fewer walks and hit by pitches from the windup. Therefore, this difference could be modeled in a baseball board game in which about 118 out of every 1000 rolls corresponds to a pitcher from the stretch giving up a walk or a hit instead of recording an out. However, are we able to quantify just how much better pitchers perform on average from the windup? We will attempt to gain some insight on this question now. If a pitcher were allowed to pitch out of the windup when there were runners on base, how many fewer runs would they give up? We hypothesize this number will be greater than zero, because a pitcher is more effective out of the windup.

A pitcher who pitches from the windup, the more comfortable position, records an out approximately 1.18 percent more often than when they pitch from the stretch. We can effectively split this into two causes, namely walks and singles. A pitcher gives up a walk 0.48 percent more often from the stretch, and hits batters with a pitch 0.20 percent more often from the stretch. Therefore, we can combine these two into one "walk" figure, because in terms of expected runs, these are the same outcome. We can then suggest that pitchers either walk or hit (we will say walk) a batter 0.68 percent more when pitching from the stretch. Subtracting this value from 1.18, we have 0.50 percent of outs

unaccounted for when pitching from the windup. But pitchers give up a single approximately 0.48 percent, which is approximately 0.50 percent, more often from the stretch compared to the windup. We can then combine this figure with our walks percentage to obtain $0.50 + 0.68 = 1.18$. Pitchers give up a single 0.50 percent more often pitching from the stretch, give up a walk 0.68 percent more often, showing that they obtain an out 1.18 percent less pitching from the stretch compared from the windup. We can use these numbers obtained here to quantify the number of runs pitchers give up pitching from the stretch and see just how many runs these pitchers are giving up. In essence, we will see how many fewer runs teams would score if pitchers could pitch from the windup, their more comfortable position.

From each state of the game, we will calculate how many expected runs pitching from the windup would save. From state 0, there are no runners on base with no outs, so the pitcher will be pitching from the windup. Therefore, we don't need to consider this state. We will now calculate the number of runs we would save pitching from the windup from state 1, with no outs and a runner on first base. Pitching from the windup, we know our outcomes on average would be the same 98.82 percent of the time, as we mentioned previously. The remaining 1.18 percent of the time, we would expect to get an out compared to pitching from the stretch. Expecting to get an out corresponds to a decrease from 0.84 runs to 0.49 runs (a 0.35 decrease in expected runs scored), according to our Expected Runs table. Therefore, the expected number of runs we would give up being able to pitch from the windup would be a difference of $(0.0118) * (-0.35) = -0.004095$ runs. We want to compare this to the expected figure pitching from the stretch to see how many "extra" runs we would expect to give up from this situation.

If we pitch from the stretch, we would expect about 0.68 percent of our outs

becoming walks, and 0.50 percent of our outs becoming singles, as we mentioned above. Walking a batter with runner on first base and no outs brings us to the state of runners on first and second base with no outs, and increase from 0.84 expected runs to 1.42 (0.60 expected runs increase). We will assume for the sake of being conservative, that a single in any situation simply moves the base runner up one base, so we don't consider cases of runners going from first to third on a single or going from second to home on a single. Therefore, we assume teams are playing "station to station" baseball which will make the analysis more conservative and easier to illustrate. Therefore, the remaining 0.50 percent of outs becoming singles brings us from 0.84 expected runs to 1.42 expected runs, as is the case with a walk. The number of runs we would expect to give up pitching from the stretch in these 1.18 percent of plays that differ from the windup is $(0.0068) * (0.60) + (0.0050) * (0.60) = .0070$. Taking this number and subtracting the expected difference in number of runs pitching from the windup and without rounding off the figures, we have that the total number of expected runs scored from the stretch is .0109. What does this figure tell us? The number .0109 tells us that every time a game reaches state 1, where we have no outs with a runner on first base, if a pitcher were allow to pitch from the windup instead of the stretch (without having to worry about the baserunner stealing), then on average, the team on offense would expect to score 0.0109 runs fewer than if the same pitcher pitched from the less comfortable position of the stretch.

Just how significant is this number 0.0109 for this specific state of the game? According to the play-by-play data obtained from Retrosheet, this state occurred 9600 times over the course of the 2014 season. Combining these two numbers, we have that the total number of runs the windup would save compared to the stretch is $9600 * (0.0109) = 104.52$ runs per season. Therefore, if

every pitcher were allowed to pitch from the windup compared to the stretch when there is a runner on first base with no outs, we would expect to see about 105 fewer scored over the course of the season. This number is helpful, but we're more interested in how many fewer runs would be scored over the entire course of the season if pitchers simply pitched from the windup all of the time. In order to figure this out, we take every other state from the game and simply apply the same techniques we did to get the number from state 1. That is, we figure out how many expected runs we would save for each state of the game pitching from the windup, and multiply this by the number of occurrences of that state to get the final number. We then sum over all of the states, 0 through 24, excluding the states with no runners on base (states 0, 8, 16, and 24) to obtain the end result, which is the number of runs the stretch "costs" over the course of the season, or the number of runs the windup could potentially save. This final number is 743.60 runs. There are 2430 games played over the course of an MLB season, so this corresponds to roughly one run every three games.

If we expect a team to score 0.45 runs per inning (from our Expected Runs table) then each team would expect to score 4.05 runs per game, totaling 9.1 runs per game for both team. Multiplying this by 2430, we get that over the course of the season, we would expect to see a total of 22,113 runs scored. If we instead forced (or allowed) pitchers to pitch from the windup no matter the state of the game, according to our analysis this would decrease the expected number of runs scored from 22,113 to $22,113 - 744 = 21,369$ runs. This is a 3.36 percent decrease in total runs scored over a season, which certainly adds up over the course of 2,430 games played. Therefore, we can see that pitchers perform quantifiably better out of the windup compared to the stretch, and baseball board games would create a more accurate representation of a game with this aspect involved.

4.2.2 Are Walks Situational?

Walks are a useful statistic that have recently been used more extensively in player analysis. Players who tend to get on base often are valuable, and players with a high number of walks have a unique ability to get on base which helps their stock. In 2014, Carlos Santana led the major leagues in walks with 113. His exceptional ability to walk, however useful, is overshadowed by his inability to hit for average, as he hit just .231. Jose Bautista was second in the majors with 104 walks. Combined with a .286 average, this corresponds to an on base percentage of .403, compared to Santana's OBP of .365. Therefore, a player like Bautista, who walked in over 15 percent of all plate appearances, is a valuable asset to a team looking to have players who can get on base. After all, the ability to get on base can also be thought of as the ability to not get out, and outs in baseball are precious. When a batter gets out, they are eliminating the opportunity for them to score a run. Therefore, batters and baserunners who are able to get on base more often, or in other words "not get out," are helping their team by providing an opportunity to score a run. In order to understand baseball and to better analyze players, we will look to answer the question, "are walks situational?" In other words, can certain players walk more or less frequently depending on the situation? If they can, than this aspect of certain batters should be incorporated into baseball board games in order to make them more realistic.

Because of the usefulness of walks, teams should likely pay attention to this statistic and scout players who routinely show solid plate discipline by getting on base via the base on balls. However, is a player like Bautista likely to walk 15 percent of the time regardless of the situation? In other words, can we always rely on the probability of .15 that Bautista will get on base by a walk, or will this statistic change based on the situation of the inning? Will he walk more

frequently when the bases are loaded, or walk less frequently when there are runners in scoring position? Many baseball simulations simply use the fraction of walks per plate appearance as a constant figure. That is, regardless of who's pitching, what bases are occupied, a player will walk a constant percentage of the time. There is good reason to believe this is not the case, and this research will present an argument that the probability of a player obtaining a walk can certainly depend on the state of the game. The situations we chose to analyze were the following: 2 outs with no runners on base, at least one runner in scoring position (second or third base) with 1st base open, bases loaded, bases empty with no outs, and 9th inning or later. With the second scenario, with first base open and at least one runner in scoring position, this would provide an opportunity for a pitcher to "pitch around" a powerful hitter. In this case we might expect walk percentage to increase. In regards to the final scenario, dealing with walks in the 9th inning or later, we chose this situation because late in the game, there is an additional benefit to getting on base if the game is close. Therefore, if walks truly are conditional, we might expect players to walk more often in this situation. A typical table of a player we would analyze for situational walks is displayed below for illustration.

Table 9: Situational Walk Percentage

Player Name	Opportunities	Walks	Walks Per Opportunity	P-value
Season				
2 outs, no one on				
RISP, 1st base open, any outs				
Bases loaded, any outs				
No one on, no outs				
9th or extra inning				

In order to answer the question, "are walks conditional?" this paper looks into the case of every player who hit 25 or more home runs over the course of the 2014 season. This resulted in 27 players who differed greatly in the number

of walks they achieved over the season. Carlos Santana led the group with 113 walks. Adam Jones sat at the bottom of the list, with only 19 walks all season. The entirety of the list is contained in the Appendix. For each of these players, the goal was to determine if their rate of walks over the course of the season differed depending on the situation of the inning and game. If we found statistical significance, that would indicate that certain players may have the quality of walking more or less frequently, depending on the situation. This would imply that walks are indeed conditional. The methodology to determine if walk percentage differed significantly is as follows. In order to clarify the analysis, we will focus on Todd Frazier, who walked 52 times in 660 plate appearances, giving him a rate of walking in 7.88 percent of plate appearances. Frazier batted 96 times when there were two outs with no runners on base. In these opportunities, he walked 4 times, giving him a walk rate of 9.68 percent. In order to test if this difference is significant, we used a p-value calculator to determine the probability that based on these ratios and number of opportunities, how likely it is that the outcome could have happened due to random chance. Essentially, the p-value returns the probability that there is no significant difference. In the case of 2 outs with no runners on base, for Frazier the p-value was .0968, meaning that there is a 9.68 percent chance that this could have been due to random chance, and that Frazier does not likely walk less often in this situation.

When Frazier came to bat with the bases loaded, he walked 3 times in 14 opportunities, giving him a walk rate of 21.4 percent. Certainly, this is a much higher figure than his average rate of 7.9 percent. This value is indeed statistically significant as well, yielding a p-value of .034 (using a significance level of .05). This indicates that based on this data, there is only a 3.4 percent chance that these numbers could have been due to random chance alone. Essentially, there is only a 3.4 percent chance that Frazier does not walk a significantly

higher percentage of the time due to his ability to draw more walks in this situation. This is a particularly exciting result, because it shows that a player like Frazier has the ability to walk more often in specific situations, like with the bases loaded, where a walk scores a run. These results, and the ones similar, could have indeed been due to a small sample size. They could have also been a result of poor pitcher performance under pressure. It's difficult to say whether or not the batter or pitcher has more impact on whether or not a player walks, but the fact that certain players have significantly higher and lower walk rates over the course of an entire season indicate that certain players may indeed walk at different rates depending on the situation. The statistically significant results are displayed in the table below.

Table 10: Walks, Season Average

Player	Season Average (percentage)
David Ortiz	12.5
Victor Martinez	10.9
Josh Donaldson	10.9
Todd Frazier	7.9
Adam LaRoche	14.0
Andrew McCutchen	13.0
Brandon Moss	11.6

Table 11: Situational Walk Percentage

Player	Situation	Situation Average (percentage)
David Ortiz	RISP, 1st base open	2.4
Victor Martinez	RISP, 1st base open	4.8
	bases loaded	37.5
Josh Donaldson	RISP, 1st base open	18.9
Todd Frazier	bases loaded	21.4
Adam LaRoche	RISP, 1st base open	22.9
Andrew McCutchen	RISP, 1st base open	19.4
Brandon Moss	bases empty, no outs	5.9

These results indicate that in many cases, certain players have the ability

Table 12: Situational Walk Difference

Player	Situation	Situation Average minus Season Average	p-value
David Ortiz	RISP, 1st base open	-10.0	0.004
Victor Martinez	RISP, 1st base open	-6.1	0.027
	bases loaded	26.6	0.009
Josh Donaldson	RISP, 1st base open	8.0	0.009
Todd Frazier	bases loaded	13.5	0.034
Adam LaRoche	RISP, 1st base open	8.9	0.010
Andrew McCutchen	RISP, 1st base open	6.4	0.047
Brandon Moss	bases empty, no outs	-5.6	0.027

to walk more or less often, depending on the situation and how it may benefit their team. A negative value in table 12 indicates that in this specific situation, the batter in question walks in a smaller proportion of plate appearances than their season average. Therefore, a value above zero would indicate that in this situation, they walk in a higher proportion of plate appearances than they did over the course of the season. We can see that in the case of analyzing the top 26 home run hitters in the Majors, seven of them (or 27 percent) had situations where they walked statistically different from their season average. Therefore, the typical model of walks used by many baseball simulations may be considered inaccurate, and it could be worth the additional research to determine which players are best at walking in certain situations. This analysis can also be used in in-game managing, in which a manager may wish to pitch differently to a batter like Victor Martinez if he comes up with the bases loaded, knowing that Martinez walks a larger percentage of the time than normal in this situation. It would be wise to examine the data for each player faced, and determine an optimal defensive strategy in order to address the unique aspects of each batter, especially in a close game where baserunners are important.

4.2.3 Baserunner Jumps and Bad Hold Pitchers

The final aspect of baseball board games we wish to discuss is the concept of a baserunner's ability to get a good jump off of a pitcher in order to create a situation to steal a base. In Dynasty League Baseball, a common baseball board game, pitchers are assigned a hold rating, which dictates how good they are at preventing a baserunner from getting a good jump when stealing a base. A baserunner's ability to get a good jump positively influences their attempts to steal; if a baserunner is able to get a good jump, they will attempt to steal the base. If a baserunner wishes to steal a base, they must first get a good jump before attempting to steal the base, so if the baserunner is unable to get a good jump, they won't be able to attempt to steal. Pitchers with higher hold ratings are able to prevent runners from stealing more effectively than pitchers with lower hold ratings, because their high hold ratings prevent a baserunner from getting a good jump. While a pitcher is given a hold rating, a baserunner is also given a stealer rating on a scale from one to ten, with one being the worst and ten being the best. A high stealer rating indicates that the baserunner is able to get a good jump a high proportion of the time.

With the current design of the board game, when a baserunner wishes to attempt to steal, their ability to get a good jump depends on not only their own stealer rating, but also the pitcher's hold rating. The model set up in Dynasty League Baseball when determining the rate of a successful jump is given as follows:

$$j = 0.55 - 0.10p + 0.04s$$

where p is the pitcher's hold rating and s is the baserunner's stealer rating. Therefore, the model assumes a linear weight to both the pitcher's hold rating and baserunner's stealer rating when determining the rate of a successful jump.

However, is this linear model realistic? In other words, if we have a pitcher with a poor hold rating and a baserunner with a poor stealer rating, will this result in a similar jump rate compared to an average pitcher and an average base stealer? Our results indicate that this is not the case. That is, we hypothesize that a poor base stealer is much less likely to get a good jump off of a poor pitcher than this model predicts, and we can use base stealing data from the 2014 season to explore this model further.

First, let's consider the case of an average pitcher on the mound with a baserunner with a stealer rating of at least 2. An average pitcher has a hold rating of $p = 2.8$ and of the baserunners with stealer ratings at least 2, the average baserunner has a stealer rating of $s = 3$. The successful jump rate of an average 2+ rated base stealer attempting to steal off of an average pitcher is then $j = 0.55 - (0.10) * (2.8) + (0.04) * (3) = 0.39$. Therefore, we can expect an average 2+ rated baserunner in this situation to get a good jump 39 percent of the time off of an average holding pitcher. We can also compute the successful jump rates comparing average and poor holding pitchers with average and poor baserunners, where a poor holding pitcher (we denote these pitchers "F" type pitchers) has a hold rating of $p = 0$. Poor baserunners, a 1 on a scale of 1-10, have a stealer rating of $s = 1$. We can then compute the following successful jump rates, where AD represents the average pitcher with hold ratings A through D, F represents pitchers with the worst hold rating F, 1 corresponds to the baserunners with the worst stealer ratings, and 2+ represents the remaining baserunners.

Table 13: Successful Jump Rates

Pitcher/Baserunner	1	2+
AD	0.31	0.39
F	0.59	0.67

Professor Dennis Stanton suggests the successful jump rates for 1-rated baserunners facing F-rated pitchers should be much lower than 0.59 as the model predicts. He proposes a figure closer to 0.41 to more accurately model successful jump rates for these baserunners facing these pitchers. The intuition behind this is the following. Suppose we have an extremely poor baserunner on first base with an extremely poor holding pitcher on the mound. No matter how poor the pitcher is at holding a baserunner on base, a bad baserunner will theoretically never attempt to steal. Therefore, the rate at which a bad baserunner gets a good jump should apparently be much lower, and we will use base stealing data to examine just how much lower this jump rating should be.

To analyze the 2014 MLB data, we separate the baserunners into two groups. One group will consist of all baserunners with a stealer rating of 1, and the other group consists of all others, in which case they have stealer ratings of 2 or higher (denoted 2+). In addition to separating baserunners, we separate pitchers into two groups. One group of pitchers, the AD group, consists of all the pitchers with hold ratings of A through D. The other group of pitchers, the F group, is the group of all pitchers with the worst hold ratings in the league. We can then analyze all of the possible stolen base opportunities for each group of baserunners and pitchers, and see how often each group of baserunners attempted to steal on each group of pitchers. The results are displayed in the table below.

Table 14: Stolen Base Attempts and Opportunities

Pitcher-Baserunner Matchup	Steal Attempts	Opportunities	Rate
F vs 1	34	530	0.064
AD vs 1	1335	30517	0.044
F vs 2+	84	132	0.636
AD vs 2+	2346	7629	0.308

This table indicates that baserunners with a rating of at least 2 steal much more often as a percentage of total opportunities than baserunners with a stealer

rating of 1. We also note that baserunners of any group steal more often against F-hold pitchers than AD-hold pitchers, but 2+ baserunners steal in more than double the opportunities against F-hold pitchers. While 2+ baserunners take special advantage of F-hold pitchers, baserunners with a stealer rating of 1 only steal 2 percent more often against F-hold pitchers. This indicates that despite the model predicting a baserunner being able to get a good jump, baserunners with a poor stealer rating don't steal, even when the pitcher is bad at holding runners on. This is in line with our hypothesis that even against poor hold pitchers, bad baserunners fail to get a good jump a large percentage of the time.

For a given baserunner, how much more often will they be inclined to steal with an F-hold pitcher on the mound compared to an AD-hold pitcher? Answering this question will help us construct a better figure for the successful jump rate of 1-rated base stealers. We can define the Inclination to Steal for a baserunner against a pitcher as the rate of stolen base attempts in opportunities divided by the jump rate. This essentially tells us how eager a baserunner is to steal, measured by the percentage of the time they attempt to steal in opportunities, given their jump rate. Mathematically, this equation is:

$$Inc_{s,p} = \frac{\frac{\text{steal attempts}}{\text{steal opportunities}}}{j_{s,p}}$$

where $j_{s,p}$ is the successful jump rate predicted by the model for the cases of 2+ runners against AD and F-hold pitchers and 1-rated baserunners against AD pitchers. We will be calculating a better jump rate for 1-rated baserunners against F-hold pitchers. Below is the following chart of Inclination to Steal for baserunners against pitchers.

This table shows us that given a good baserunner, a runner with a rating of 2 or greater, they will be inclined to steal 95 percent of the time when facing

Table 15: Inclination to Steal

Pitcher/Baserunner	1	2+
AD	0.141	0.789
F	?	0.950

an F-hold pitcher. Just how much of an impact does the performance of the pitcher make? Well, if a 2+ runner is stealing against an F-hold pitcher, they are inclined to steal 95 percent of the time. If the same runners are stealing against a better hold pitcher, they're inclined to steal 78.9 percent of the time. The percentage increase in inclination to steal with dealing with an F-hold pitcher compared to a better holding pitcher is $\frac{0.950}{0.789} = 1.204$, so a 2+ baserunner is inclined to steal 20.4 percent more often when facing an F-hold pitcher compared to facing a better holding pitcher.

Baserunners with a stealer rating of 1 are inclined to steal on AD-hold pitchers in 14.1 percent of opportunities. If we also consider that they will be 20.4 percent more inclined to steal when facing an F-rated pitcher, then we know that their inclination to steal against F-rated pitchers will be $(1.204)*(0.141) = 0.170$, or they will be inclined to steal in 17 percent of opportunities against this group of pitchers. We can then use the data regarding these 1-rated baserunners in order to solve for the optimal j^*_{1F}:

$$0.170 = Inc_{1,F} = \frac{\frac{\text{steal attempts}}{\text{steal opportunities}}}{j^*_{1,F}} = \frac{\frac{34}{530}}{j^*_{1,F}}$$

This equation yields the result $j^*_{1,F} = 0.38$. This calculation indicates that based on the 2014 base stealing data, the successful jump rate for 1-rated baserunners facing F-hold pitchers should be closer to 0.38, rather than the model's successful jump rate of 0.59. Using the same p-value calculator from earlier, we find that the probability these numbers are not statistically different is p = 0.0000. Therefore, we can reject the null hypothesis that $j_{1,F} = 0.59$ in favor of the

alternative hypothesis that $j^*_{1,F} = 0.38$.

What does this new jump rate of 0.38 indicate? A lower jump rate for poor base stealers facing poor hold pitchers means that there is a lower probability that this type of base stealer will attempt to steal. This makes sense, because even if a pitcher is horrible at holding runners on base, if we have a very slow baserunner who is bad at stealing bases, a poor holding pitcher won't magically make them more effective at stealing, so they should have a lower than predicted probability of getting a successful jump, which leads to a steal attempt. Therefore, Professor Stanton's hypothesis that this jump rate should be in the neighborhood of 0.41 presents a much more accurate figure. Dynasty Baseball has the opportunity to improve the game by matching successful jump rates based on real base stealing data, and improving the game in this way does not affect the playability of the game. Therefore, we've discussed three ways in which the Dynasty Baseball League can improve their board game simulation of a game. They can incorporate a more accurate jump rate for poor baserunners against poor hold pitchers, update the walk rates for certain players in different situations, and provide different results for pitchers pitching out of the windup and out of the stretch.

5 Discussion

5.1 Project Limitations

Based on the nature of the project, we are only able to conclude things on an average, or league-wide level. An interesting, and likely more precise method to test our conclusions would be to look at them at a more granular level. That is, analyze each player on each team in order to determine if certain players' performances match our conclusions here. This type of analysis is outside the

scope of this project, but it's worth considering that this may indeed be a more accurate option. However, that is not to say that team-wide and league-wide analysis cannot be accurate, but it may be less accurate than analysis at a more specific level.

The lack of the 81 St. Louis Cardinals home game data has the chance of altering the results of the project. It is always wise to collect as much data as possible prior to analysis. Based on technical difficulties with pulling the data off the Retrosheet website, this data wasn't able to be collected and included in the analysis. However, the Cardinals' data represents just three percent of all of the MLB data. In this case, we believe the rest of the 97 percent of the data provides a representative sample of the population of MLB teams, and therefore the absence of three percent of the data should not skew the results of this project significantly. However, a more complete analysis would of course include this missing data.

Our use of the same transition matrix for all states of the game has the potential to be an inaccurate assumption. It would be an interesting idea to find a way to filter the game data into situations where a team is down by one run late in the game, or find another meaningful in-game situation where a team may play differently, to test the hypothesis that the transition matrix should be the same on average over all states. Should this kind of produce results that suggest different parts of the game produce different transition matrix, our results may be changed significantly, and the robustness of our results could be a limitation as well.

5.2 Further Research

As baseball continually becomes a game more focused on the numbers, there is always more research to be done. Based on our research, further research

could include analyzing which players are best at picking their spots to steal bases, whether certain pitchers are better at preventing a stolen base, and which catchers are the best defensively at preventing a steal. Research in lineup optimization can always be improved, as the ability to score runs will always be important, especially in recent years where we have seen dominant pitching performance. Even if a better algorithm could be constructed to earn a team a few additional runs over the course of the year, it may mean the difference between the playoffs and missing the postseason all together. We hope that with our creation of the RW statistic, further ideas could be formulated and improve the method of lineup optimization outlined in this paper and Tom Tango's "The Book." We could also suggest in looking at a larger sample of players in determining which players are able to walk more or less frequently in different situations, and how having these players in the lineup affects the rest of the players in the lineup. Are certain pitchers going to pitch differently knowing that the player behind them in the order walks more frequently with runners on base? These questions, and more, would make for interesting further research in this topic.

6 Conclusion

In this paper we have concluded that there are multiple ways in which teams could use data in order to gain a better understanding of the game, and therefore come up with better strategies to maximize the probability of scoring more runs, and thus winning games. We have demonstrated that there are considerations to be made regarding base stealing strategies, ordering the lineup, different ways to analyze conditional walks, analyzing pitcher effectiveness in different scenarios, and modeling base stealing from poor baserunners. Hopefully this paper will give way to new research in these topics and others, as managers, players, and

other people involved in an MLB organization attempt to understand fully the game of baseball, and maximize their team's chances of winning games.

7 Appendix

The following is the Python code, condensed and not in the exact syntactical form of the actual program for sake of saving space.

```
import random
inning = 1 sumofruns = 0 while inning<300000: runs = 0 state = 0
terminate = False while not terminate: x = random.randint(1,1000)
if state == 0: if x <= 25: runs +=1 state = 0
elif x > 25 and x <= 260: state = 1 elif x > 260 and x<= 309: state = 2 elif x > 309 and x<=314: state=3 elif x>314: state=8
elif state == 1: if x<= 26: runs +=2 state = 0
elif x > 26 and x<=39: runs+=1 state = 2
elif x > 39 and x<=45: runs+=1 state = 3
elif x>45 and x<=253: state=4 elif x>253 and x<=295: state=5 elif x>295 and x<=330: state=6 elif x>330 and x<=331: state=8 runs+=1 elif x>331 and x<=769: state=9 elif x>769 and x<=878: state=10 elif x>878 and x<=882: state=11 elif x>882: state=16
elif state == 2: if x<=22: runs+=2 state=0 elif x>22 and x<=77: runs+=1 state=1 elif x>77 and x<=121: runs+=1 state=2 elif x>121 and x<=127: runs+=1 state=3 elif x>127 and x<=234: state=4 elif x>234 and x<=330: state=5 elif x>330 and x<=339: state=6 elif x>339 and x<=341: runs+=1 state=8 elif x>341 and x<=354: state=9 elif x>354 and x<=729: state=10 elif x>729 and x<=994: state=11 elif x>994: state=16
elif state==3: if x<=19: runs+=2 state=0 elif x>19 and x<=190: runs+=1 state=1 elif x>190 and x<=230: runs+=1 state=2 elif x>230 and x<=349: state=5 elif x>349 and x<=596: runs+=1 state=8 elif x>596 and x<=600: state=9 elif x>600 and x<=608: state=10 elif x>608 and x<=992: state=11 elif x>992: state=16
elif state==4: if x<=20: runs+=3 state=0 elif x>20 and x<=30: runs+=2 state=2 elif x>30 and x<=34: runs+=2 state=3 elif x>34 and x<=75: runs+=1 state=4 elif x>75 and x<=103: runs+=1 state=5 elif x>103 and x<=137: runs+=1 state=6 elif x>137 and x<=298: state=7 elif x>298 and x<=300: runs+=2 state=8 elif x>300 and x<=304: runs+=1 state=9 elif x>304 and x<=306: runs+=1 state=10 elif x>306 and x<=660: state=12 elif x>660 and x<=756: state=13 elif x>756 and x<=887: state=14 elif x>887 and x<=897: state=17 elif x>897 and x<=909: state=18 elif x>909 and x<=998: state=19 elif x>998: state=24
elif state==5: if x<=31: runs+=3 state=0 elif x>31 and x<=50: runs+=2 state=2 elif x>50 and x<=55: runs+=2 state=3 elif x>55 and x<=218: runs+=1 state=4 elif x>218 and x<=253: runs+=1 state=5 elif x>253 and x<=301: runs+=1 state=6 elif x>301 and x<=372:
```

state=7 elif x>372 and x<=551: runs+=1 state=9 elif x>551 and x<=612: runs+=1 state=10 elif x>612 and x<=615: runs+=1 state=11 elif x>615 and x<=632: state=12 elif x>632 and x<=856: state=13 elif x>856 and x<=897: state=14 elif x>897 and x<=989: runs+=1 state=16 elif x==990: state=17 elif x==991 or x==992: state=18 elif x>992 and x<=999: state=19 elif x==1000: state=24

elif state == 6: if x<=9: runs+=3 state=0 elif x>9 and x<=60: runs+=2 state=1 elif x>60 and x<=101: runs+=2 state=2 elif x>101 and x<=108: runs+=2 state=3 elif x>108 and x<=119: runs+=2 state=4 elif x>119 and x<=220: runs+=1 state=5 elif x>220 and x<=228: runs+=1 state=6 elif x>228 and x<=365: state=7 elif x>365 and x<=367: runs+=2 state=8 elif x>367 and x<=374: runs+=1 state=9 elif x>374 and x<=475: runs+=1 state=10 elif x>475 and x<=627: runs+=1 state=11 elif x>627 and x<=632: runs+=1 state=12 elif x>632 and x<=642: state=13 elif x>642 and x<=994: state=14 elif x>994 and x<=999: state=18 elif x==1000: state=19

elif state==7: if x<=21: runs+=4 state=0 elif x>21 and x<=23: runs+=3 state=1 elif x>23 and x<=39: runs+=3 state=2 elif x>39 and x<=45: runs+=3 state=3 elif x>45 and x<=98: runs+=2 state=4 elif x>98 and x<=131: runs+=2 state=5 elif x>131 and x<=167: runs+=2 state=6 elif x>167 and x<=322: runs+=1 state=7 elif x>322 and x<=325: runs+=2 state=9 elif x>325 and x<=386: runs+=1 state=12 elif x>386 and x<=475: runs+=1 state=13 elif x>475 and x<=519: runs+=1 state=14 elif x>519 and x<=875: state=15 elif x>875 and x<=881: runs+=1 state=17 elif x>881 and x<=884: runs+=1 state=18 elif x>884 and x<=952: runs+=1 state=19 elif x>952 and x<=955: state=20 elif x>955 and x<=960: state=21 elif x>960 and x<=998: state=22 elif x>998: state=24

elif state==8: if x<21: runs+=1 state=8 elif x>21 and x<=256: state=9 elif x>256 and x<=304: state=10 elif x>304 and x<=309: state=11 elif x>309: state=16

elif state==9: if x<26: runs+=2 state=8 elif x>26 and x<=28: runs+=1 state=9 elif x>28 and x<=43: runs+=1 state=10 elif x>43 and x<=50: runs+=1 state=11 elif x>50 and x<=247: state=12 elif x>247 and x<=290: state=13 elif x>290 and x<=321: state=14 elif x>321 and x<=322: runs+=1 state=16 elif x>322 and x<=789: state=17 elif x>789 and x<=874: state=18 elif x>874 and x<=877: state=19 elif x>877: state=24

elif state==10: if x<=26: runs+=2 state=8 elif x>26 and x<=85: runs+=1 state=9 elif x>85 and x<=136: runs+=1 state=10 elif x>136 and x<=140: runs+=1 state=11 elif x>140 and x<=278: state=12 elif x>278 and x<=345: state=13 elif x>345 and x<=347: state=14 elif x>347 and x<=352: runs+=1 state=16 elif x>352 and x<=364: state=17 elif x>364 and x<=800: state=18 elif x>800 and x<=991: state=19 elif x>991: state=24

elif state==11: if x<=19: runs+=2 state=8 elif x>19 and x<=196: runs+=1 state=9 elif x>196 and x<=245: runs+=1 state=10 elif x>245 and x<=252: runs+=1 state=11 elif x>252 and x<=406: state=13 elif x>406 and x<=607: runs+=1 state=16 elif x>607 and x<=631: state=17 elif x>631 and x<=634: state=18 elif x>634 and x<=991: state=19 elif x>991: state=24

elif state==12: if x<=26: runs+=3 state=8 elif x>26 and x<=42: runs+=2 state=10 elif x>42 and x<=47: runs+=2 state=11 elif x>47 and x<=98: runs+=1 state=12 elif x>98 and x<=130: runs+=1 state=13 elif x>130 and x<=165: runs+=1 state=14 elif x>165 and x<=314: state=15 elif x>314 and x<=315: runs+=2 state=16 elif x>315 and x<=318: runs+=1 state=17 elif x>318

and x<=324: runs+=1 state=18 elif x==325: runs+=1 state=19 elif x>325 and x<=703: state=20 elif x>703 and x<=797: state=21 elif x>797 and x<=871: state=22 elif x>871: state=24

elif state==13: if x<=20: runs+=3 state=8 elif x>20 and x<=34: runs+=2 state=10 elif x>34 and x<=43: runs+=2 state=11 elif x>43 and x<=181: runs+=1 state=12 elif x>181 and x<=221: runs+=1 state=13 elif x>221 and x<=255: runs+=1 state=14 elif x>255 and x<337: state=15 elif x>337 and x<=509: runs+=1 state=17 elif x>509 and x<=561: runs+=1 state=18 elif x>561 and x<=564: runs+=1 state=19 elif x>564 and x<=588: state=20 elif x>588 and x<=842: state=21 elif x>842 and x<=871: state=22 elif x>871: state=24

elif state==14: if x<=16: runs+=3 state=8 elif x>16 and x<=90: runs+=2 state=9 elif x>90 and x<=142: runs+=2 state=10 elif x>142 and x<=146: runs+=2 state=11 elif x>146 and x<=155: runs+=1 state=12 elif x>155 and x<=235: runs+=1 state=13 elif x>235 and x<=238: runs+=1 state=14 elif x>238 and x<=439: state=15 elif x==440: runs+=2 state=16 elif x>440 and x<=445: runs+=1 state=17 elif x>445 and x<=534: runs+=1 state=18 elif x>534 and x<=684: runs+=1 state=19 elif x>685 and x<=692: state=20 elif x>692 and x<=713: state=21 elif x>713 and x<=983: state=22 elif x>983: state=24

elif state==15: if x<=22: runs+=4 state=8 elif x>22 and x<=36: runs+=3 state=10 elif x>36 and x<=45: runs+=3 state=11 elif x>45 and x<=95: runs+=2 state=12 elif x>95 and x<=132: runs+=2 state=13 elif x>132 and x<=168: runs+=2 state=14 elif x>168 and x<=329: runs+=1 state=15 elif x>329 and x<=336: runs+=2 state=17 elif x>336 and x<=339: runs+=2 state=18 elif x==340: runs+=2 state=19 elif x>340 and x<=411: runs+=1 state=20 elif x>411 and x<=511: runs+=1 state=21 elif x>511 and x<=560: runs+=1 state=22 elif x>560 and x<=865: state=23 elif x>865: state=24

elif state==16: if x<=22: runs+=1 state=16 elif x>22 and x<=259: state=17 elif x>259 and x<=303: state=18 elif x>303 and x<=308: state=19 elif x>308: state=24

elif state==17: if x<=24: runs+=2 state=16 elif x==25: runs+=1 state=17 elif x>25 and x<=46: runs+=1 state=18 elif x>46 and x<=53: runs+=1 state=19 elif x>53 and x<=240: state=20 elif x>240 and x<=290: state=21 elif x>290 and x<=311: state=22 elif x>311: state=24

elif state==18: if x<=17: runs+=2 state=16 elif x>17 and x<=103: runs+=1 state=17 elif x>103 and x<=154: runs+=1 state=18 elif x>154 and x<=159: runs+=1 state=19 elif x>159 and x<=314: state=20 elif x>314 and x<=345: state=21 elif x>345: state=24

elif state==19:

if x<=19: runs+=2 state=16 elif x>19 and x<=154: runs+=1 state=17 elif x>154 and x<=202: runs+=1 state=18 elif x>202 and x<=206: runs+=1 state=19 elif x>206 and x<=354: state=21 elif x>354: state=24

elif state==20: if x<=22: runs+=3 state=16 elif x==23: runs+=2 state=17 elif x>23 and x<=48: runs+=2 state=18 elif x>48 and x<=53: runs+=2 state=19 elif x>53 and x<=106: runs+=1 state=20 elif x>106 and x<=146: runs+=1 state=21 elif x>146 and x<=172: runs+=1 state=22 elif x>172 and x<=287: state=23 elif x>287: state=24

elif state==21:

if x<=17: runs+=3 state=16 elif x==18 or x==19: runs+=2 state=17 elif x>19 and x<=39: runs+=2 state=18 elif x>39 and x<=45: runs+=2 state=19 elif x>45 and x<=145: runs+=1

```
state=20 elif x>145 and x<=187: runs+=1 state=21 elif x>187 and x<=205: runs+=1 state=22
elif x>205 and x<=295: state=23 elif x>295: state=24
    elif state==22: if x<=19: runs+=3 state=16 elif x>19 and x<=86: runs+=2 state=17 elif
x>86 and x<=129: runs+=2 state=18 elif x>129 and x<=134: runs+=2 state=19 elif x>134 and
x<=162: runs+=1 state=21 elif x>162 and x<=347: state=23 elif x>347: state=23
    elif state==23: if x<=21: runs+=4 state=16 elif x==22: runs+=3 state=17 elif x>22 and
x<=43: runs+=3 state=18 elif x>43 and x<=54: runs+=3 state=19 elif x>54 and x<=96:
runs+=2 state=20 elif x>96 and x<=134: runs+=2 state=21 elif x>134 and x<=157: runs+=2
state=22 elif x>157 and x<=273: runs+=1 state=23 elif x>273: state=24
    elif state == 24: terminate = True inning+=1 sumofruns += runs
    print("inning count:", inning)
    print("Average runs per inning:") print(sumofruns/inning)
```

Below is the table of players analyzed for the purposes of determining if walks can be modeled in a conditional way. The players we analyzed were all players who hit 25 home runs or more, and they're listed in descending order of home runs hit.

Table 16: Players and Walks

Player Name	Plate Appearances	Walks	Walks Per Opportunity
Nelson Cruz	678	55	0.08
Chris Carter	572	56	0.10
Giancarlo Stanton	638	94	0.15
Jose Abreu	622	51	0.08
Mike Trout	705	83	0.12
Jose Bautista	673	104	0.15
David Ortiz	602	75	0.12
Edwin Encarnacion	542	62	0.11
Victor Martinez	641	70	0.11
Anthony Rizzo	616	73	0.12
Lucas Duda	596	69	0.12
Josh Donaldson	695	76	0.11
Todd Frazier	660	52	0.08
Adam Jones	682	19	0.03
Justin Upton	641	60	0.09
Albert Pujols	695	48	0.07
Adrian Gonzalez	660	56	0.08
Carlos Santana	660	113	0.17
Chris Davis	525	60	0.11
Adam LaRoche	586	82	0.14
Marlon Byrd	637	35	0.05
Miguel Cabrera	685	60	0.09
Matt Kemp	599	62	0.10
Andrew McCutchen	648	84	0.13
Devin Mesoraco	440	41	0.09
Brandon Moss	580	67	0.12
Kyle Seager	654	52	0.08