Markov League Baseball
Baseball Analysis Using Markov Chains

Introduction:

A Markov chain is a stochastic model that is well utilized when analyzing baseball. Markov chains date back to 1907 and appeared in statistical literature for baseball as early as 1960 (Pankin, 2016). Applying Markov chains to baseball typically employs situational analysis and studies the probabilities and effects on expected scoring of moving from one base runners and outs combination to another, which will be discussed in the literature review of “Markov League Baseball: Baseball Analysis Using Markov Chains” by VJ Asaro and the Cal Poly Statistics Department.

Literature Review:

Application to Baseball:

The purpose of this statistical paper is to use Markov chains to predict winning percentages of Major League Baseball teams in a single season by first observing expected number of runs and applying it to player batting and pitching analysis. The datasets used in this study are collected from http://www.baseball-reference.com/ and data manipulation and analytics are performed using R. The statistics being obtained consist of: League Standard Pitching, League Standard Batting, and MLB League Standings for the year of 2015. A Markov chain contains a number of states and creates a probability transition matrix denoted as (P) during the study. Furthermore, when applying this concept in terms of baseball, these multiple states represent the different runners on base and each combination of the number of outs. For clarification, if there is a runner on first and only one out, this is a different state than if you consider bases loaded with 1 outs. In the paper, these situations would be denoted as follows: (1,1) and (123,1) respectfully. For this Markov chain, there are 24 total combinations of baseball states, excluding the absorbing state; three outs. Note that an absorbing state is when the probability of going from that state to itself is 1.0 (4, Asaro). For baseball, the absorbing state will be the third out of an inning because you are not able to transition to any other state as it is the end of the inning.

Computing the Probabilities from Baseball Reference Data:

We use the following probabilities calculated using R in order to construct the transition probabilities of going from one state to the next.

P(1B)	= 1B / PA	P(1B)	= 0.016
P(2B)	= 2B / PA	P(2B)	= SF / PA
P(3B)	= 3B / PA	P(3B)	= DP / PA
P(HR)	= HR / PA	P(HR)	= 1- [P(H)+P(BB,HBP)+P(E)]
P(BB, HBP)	= (BB + HBP) / PA	P(BB, HBP)	= P(Out) - P(DP)

1B=Single	HR=Home Run	DP=Double Play
2B=Double	E=Error	BB,HBP=Walk, Hit by Pitch
3B=Triple	SF=Sacrifice Fly	PA=Plate Appearance

For example, the probability of the transition from zero base runners and zero outs to one base runner and zero outs is represented by:

$$(0,0) \rightarrow (1,0) = P(1B)+P(BB,HBP)+P(E).$$

When constructing the transition probability matrix, there are three assumptions that need to be made.

1. Field errors only account for advancing one base. Multi-base errors are not included for this model.
2. Stolen bases, triple plays, pass balls, or wild pitches are also not included in this model.
3. Base running behavior and situational hitting are not included in this model.

Constructing the Transition Probability Matrix:

Including the absorbing state, this model contains 25 states meaning the transition matrix is 25x25, P, which can be split into multiple submatrices: A (yellow) matrix, B (blue) matrix, C2 matrix, and D matrix.

1. A → probabilities of transitioning to a state with the number of outs remaining constant.
2. B → probabilities of transitioning to a state where the number of outs increases by one.
3. C2 → probabilities of a double play with zero outs.
4. D → probabilities of transitioning to the absorbing state (three outs).

Note: white submatrices contain all zero because the number of outs are irreducible from state i to state j.

Expected Runs:

The first step to finding the expected runs is to do so for a half-inning. E, is a matrix found by using the matrix equation: **E = (I-Q)-1** where Q is a 24x24 submatrix of P (the three-outs state is excluded) and I is a 24x24 identity matrix (8,Asaro). Therefore, E represents the expected number of visits to each state, beginning from each state until three outs are achieved. The Rscore matrix has the same dimensions and contains the weighted probabilities of scoring a number of runs beginning in each state. Row sums of Rscore are calculated to compute the eRuns matrix, which contains values of the expected number of runs to be scored beginning from each state after one plate appearance. We use both E and eRuns to calculate the expected number of runs.

E(Runs) [24x1] = E*eRuns (****Note**: Multiply E(Runs) by 9 for expected runs in full 9 innings**)

To put this into context, consider an E(Runs) matrix that is based off of Mike Trout's career averages. If Mike Trout is up to bat and there are currently two baserunners and zero outs (12,0), its is expected that there are going to be 1.81 runs for the rest of the inning. Additionally, if Trout were up-to-bat starting in a state of zero baserunners and zero outs for all 9 innings, it is expected that (9 * 0.7822) = 7.04 runs in a complete game.

Player Analysis Using Expected Runs:

The expected Markov runs are implemented to player batting statistics to project a players' Markov runs score. Markov runs are also a strong tool for evaluating a player's offensive productivity (11, Asaro). The On-base Plus Slugging (OPS) is a popular statistic also, which is denoted as:

OPS = OBP + SLG% (11, Asaro).

SLG% is slugging percentage which equals the total number of bases divided by the number of at-bats for a hitter. This statistic measures the power of a hitter. OBP is the percentage of the time that a hitter safely reaches base in a number of plate appearances.

SLG = ((1B) + (2 x 2B) + (3 x 3B) + (4 x 4B)) / (AB)

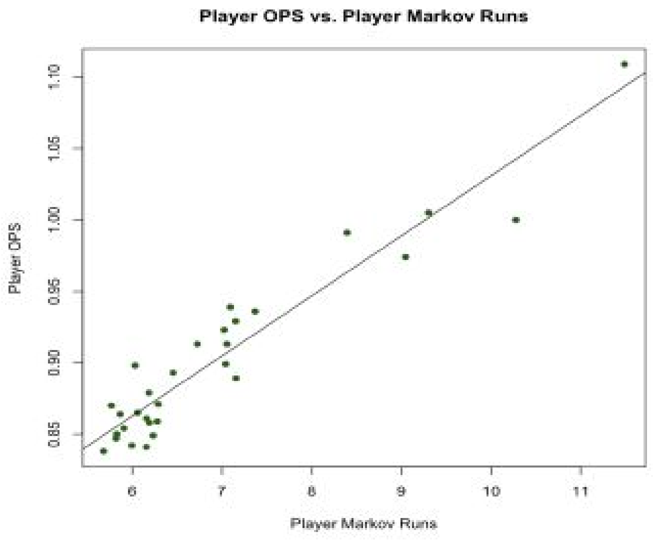
OBP = (H + BB + HBP) / (AB + BB + HBP + SF)

Therefore, OPS measures how often a player at bat gets on base, as well as, the impact of how he gets on base. The Markov runs appear to be highly correlated in relation to the OPS statistic (12, Asaro). Below is a table display from the paper ("Table 3: Top 5 OPS players and Top 5 Markov Runs players", 12, Asaro):

Player Name	**OPS Rank**	**OPS**	**Markov Runs**	**Markov Runs Rank**
Bryce Harper	1	1.109	11.49	1
Paul Goldschmidt	2	1.005	9.30	3
Joey Votto	3	1.00	10.28	2

		0		
Mike Trout	4	0.991	8.39	5
Miguel Cabrera	5	0.974	9.04	4

Figure 4: Player OPS vs. Player Markov Runs



The table indicates the top 5 players in On-base Plus Slugging are also in the top 5 in Markov runs with some slight variability in the rankings. In fact, Markov runs and OPS display a linear relationship when plotted for the top 30 batters in 2015 season as seen in the Figure 4 from the paper (“Player OPS vs. Player Markov Runs”,13, Asaro).

Team Analysis and Win Expectancy:

Markov runs can also be used to calculate a team’s productivity. When looking at a team’s productivity analysis, the team must be evaluated by both their hitting and pitching statistics. We find the expected number of runs allowed for a pitcher as well the expected number of runs scored for a batter in order to determine a team’s winning percentage.

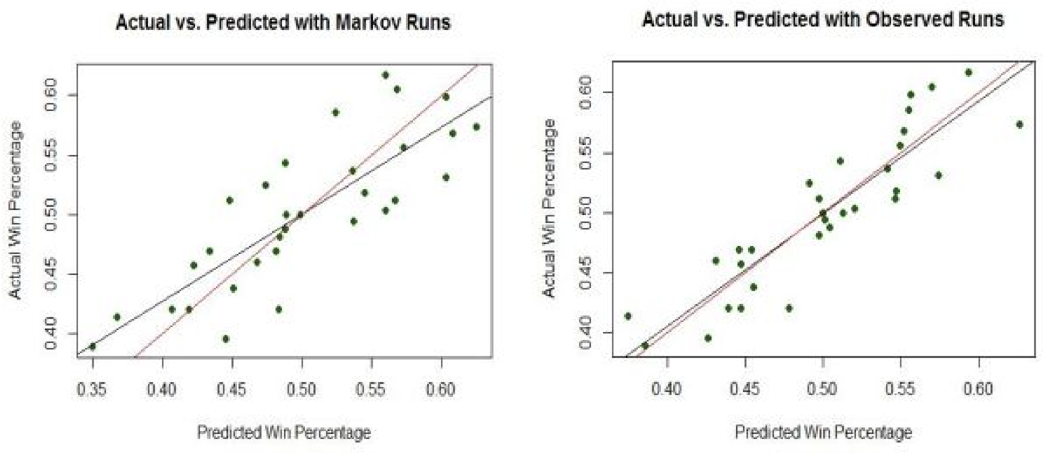
Bill James developed an expected win percentage formula using runs scored and runs allowed for a team that is utilized in this paper (Baseball Reference Bullpen). The parameters of the formula consist of the calculated Markov run scored and Markov runs allowed for a team.

$$E(WIN\%) = R^{1.81} / (R^{1.81} + RA^{1.81})$$

Figure 5a on the left shows the relationship between the expected winning percentage using Markov runs versus the actual win percentage for the 2015 season. Figure 5b is similar; however, this graph represents the relationship between the expected winning percentage using the actual runs and runs allowed for the teams versus the actual winning percentage.

Figure 5: Predicted Win Pct vs. Actual Win Pct with Markov Runs (5a),

Predicted Win Pct vs. Actual Win Pct with Observed Runs (5b)



Regression Output for Figure 5a:

Markov Runs	**Estimate**	**Std. Error**	**t-value**	**p-value**
Intercept	0.135	0.047	2.839	0.008
E(WIN%)	0.73	0.094	7.778	<0.001
R^2	0.676			

Residual Std Error	0.037			

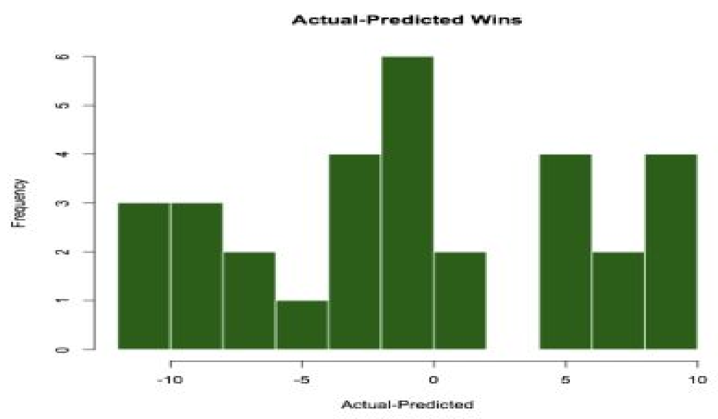
Regression Output for Figure 5b:

Actual Runs	**Estimate**	**Std. Error**	**t-value**	**p-value**
Intercept	0.027	0.044	0.628	0.535
E(WIN%)	0.94	0.086	10.937	<0.001
R^2	0.805			
Residual Std Error	0.028			

Using the Markov runs to predict winning percentage accounts for approximately 68% of the variation in the actual winning percentage. Conversely, using actual runs to predict winning percentage, we see that it accounts for 81% of the variation in the actual winning percentage. Both Figure 5a and 5b contain the line of best fit (black) and the red lines denote the perfect prediction of winning percentage. We include these two lines to compare how accurate the respective models are to a perfect prediction. It comes as no surprise that the actual runs and runs allowed better predicted the actual winning percentage.

To show just how successful this Markov chain process predicted win percentages, we plot the distribution of the predicted number of wins subtracted from the actual number of wins of a team for a full 162 game season.

Figure 6: Distribution of Actual Wins minus Predicted Wins using Markov Runs



The distribution is centered around 0, and the mean absolute difference was 5.4 wins. Therefore, after using Markov runs and the revised pythagorean theorem of the E(WIN%) baseball formula to predict the number of wins in a 162 game season average was only 5.4 games off from the actual 17 number of wins.

Conclusion:

There are numerous situational and strategic hitting in baseball that are not accounted for by this Markov process; yet, the model built in this paper efficiently simulates basic characteristics of the game. To reiterate, the Markov chain could be adjusted to implement stolen bases, situational hitting, pinch hitters, type of pitch, and/or pitcher vs. batter matchup (is the pitcher left handed or right and is the batter hitting left handed or right). The inclusion of these factors may not have severe impacts to your final numbers; however, it would help your simulations be closer to the realities of baseball.

Markov chains can have a strong influence in the MLB. Additional applications of the Markov chain in baseball include trade analysis, simulation of batting orders, predicting the type of pitch that will be thrown, and salary vs. productivity evaluation. For example, consider you are looking to trade player X for player Y. In order to ensure this trade would be beneficial to your team's overall success, you can use the concept of Markov runs to evaluate your trade decision. Remove player X from the team's average and replace it with scraped data for Player Y. Evaluate whether your team's expected runs (Markov runs) increases or decreases to determine if another you need to request another player for the trade. Similarly, using this Markov model, you may compute 9 different P matrices to simulate a batting order for desired players. You can then use these matrices in order to calculate an optimal batting order that produces the highest number of Markov runs.

Additional Research and Application:

This paper stimulated my interest to continue using stochastic processes and Markov chains for baseball analytics. I wanted to continue using R for the data analysis, so I looked at many Github repositories and became interested to use Markov chains to determine the probabilities of throwing a specific pitch being thrown based on the prior pitch. After much troubleshooting, I was able to successfully scrape all the data from the 2015 season from MLB Gameday and follow along with the provided code, as well as do some of my own findings.

The multi-state Markov models package ('msm') is used for this analysis, which will allow to restate after each state. A unique identifier is created for each batter, gameID, and inning and by doing so, we are able to look an individual pitchers' overall pitch proportions by using applications of the dplyr package. Once we have the overall proportions, we use the msm package to construct a pitch type transition probability matrix for the respective player.

For example, if we look at Chris Sale, pitcher of the Boston Red Sox, we filter and select to extract his pitching statistics to first find this overall pitch proportions:

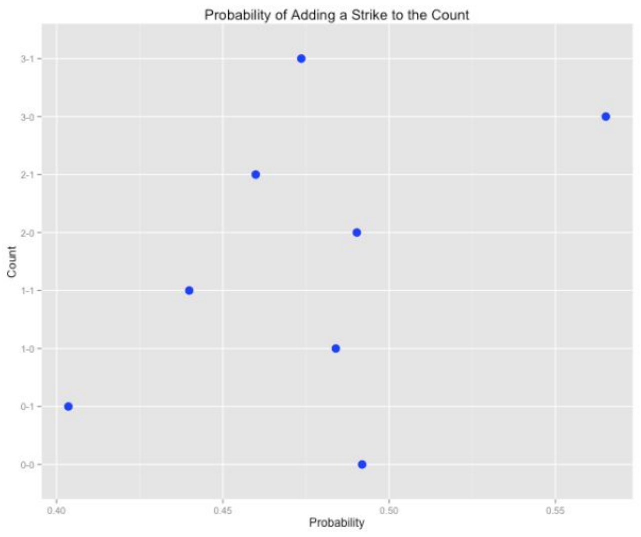
2-Stream FB	Changeup	Slider
0.528	0.277	0.195

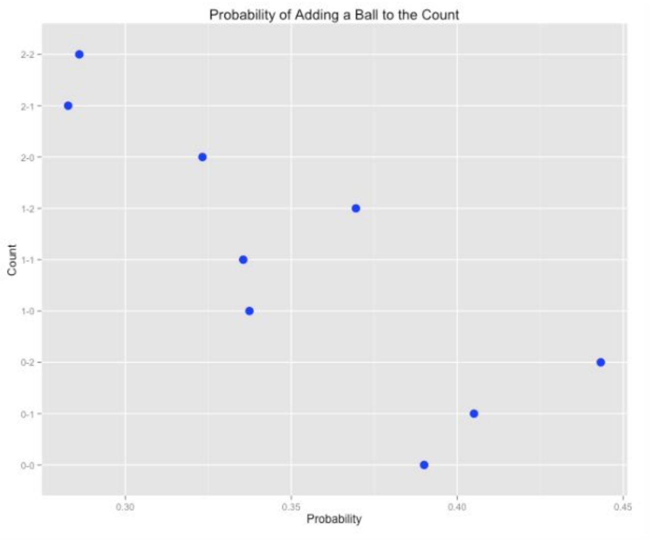
Right off the bat, pun intended, you can see that Chris Sale throws a 2-stream fastball more than 50 percent of the time. Now, using msm you create the transition probability matrix to find the probability of Chris Sale throwing either a fastball, changeup, or slider given his previous pitch.

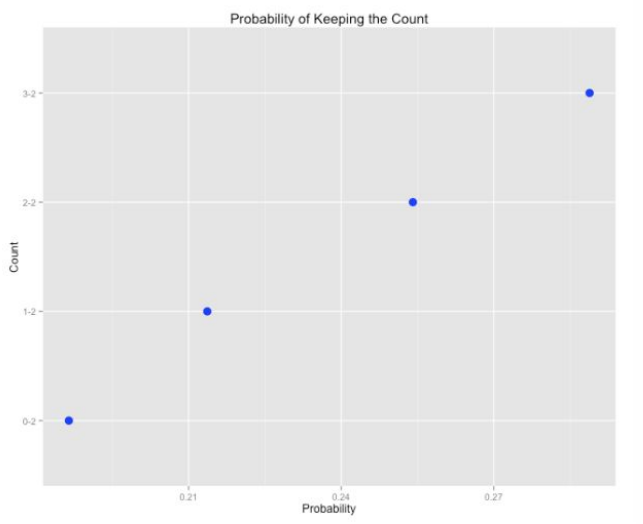
From: To:	2-Seam FB	Changeup	Slider
2-Seam FB	0.556	0.254	0.190
Changeup	0.467	0.370	0.164
Slider	0.507	0.278	0.215

The transition matrix shows you that even though Chris Sale throws a fastball approximately half of the time, he is almost 11 percent more likely to come back with another changeup given his previous pitch was a changeup in comparison than if his prior pitch had been a fastball or slider. It should also be noted that if he threw a 2-seam fastball on the previous pitch, the probability of him throwing the same pitch again increases when compared to his overall pitch proportions. This is extremely interesting because this proves that the Markov chain process analysis gives a more accurate result than just looking at overall proportions and it greater simulates the realities of game of baseball. A recommended extension of the analysis was to try and enhance the accuracy of the Markov model by using both the pitch type and the pitch location. This intrigued me and I tried to see if I could implement this in my additional research; however, after much time I wasn’t able to correctly code for including pitch locations. An example of this would be that if Chris Sale threw a fastball in the bottom third of the strike zone on the previous pitch, then based on an updated transition probability matrix, we could find the probability that Sale is going to follow up with a fastball that is higher up in the strike zone on the next pitch.

Another additional research topic that I chose to pursue was looking at pitch sequences that pass through different pitch counts and see the effect on expected run values. For this analysis, I downloaded pitching data baseball-reference for pitching, the variable of interest being pitch sequence. All the pitch counts are extracted from a single character string of either “balls” or “strikes” using a created function that returns the beginning and end count for each pitch. You can use this data then to represent the sequence of pitches as a Markov Chain. The transition matrix, P, for transitions in pitch counts gives the probability of moving to the next count (i.e. P[“0-1”, “1-1”] will give you the probability of moving from a 0-1 count to a 1-1 count). Using ggplot, here are some visual representations from the probability transition matrix. The first plot (top left) shows the probability of adding a strike to the count for each count situation. There is a high probability of the next throw being a strike when the initial count is 3-0, but there is a low probability of throwing a strike on the next pitch when the initial count is 0-1. The second plot (top right) is exactly the same; however this time it is displaying the probability of adding a ball to the count. It is observed that it is more likely to add a ball on a 0-2 count, and there is a smaller probability of adding a ball to a 2-1 or 2-2 count. Lastly (bottom), this is a representation of the probability plot of keeping the count the same showing that the count will most likely remain the same with a 3-2 count, which is as expected.







Conclusion:

This research has really put me on a path of how I may want to continue my statistic career post-graduation. Manipulating and analyzing sports data has been a very fun experience, especially when using applications of the Markov chain. It drives a desire of wanting to take these studies a little bit further and try to use the same application for other sports as well. When picking a topic for this final project, I came across numerous stochastic process for analysis ranging from predicting basketball possession outcome using a multiresolution stochastic model to predicting the outcome of a soccer match.

Bibliography:

Asaro, V. (2015-2016). Markov League Baseball (Cal Poly Statistics Department, Ed.).
Retrieved May 5, 2017, from
http://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1063&context=statsp

cpsievert. (2014, March 24). Starting and updating PITCHf/x database with pitchRx and dplyr.
Retrieved May 5, 2017, from Exploring Baseball Data with R website:
https://baseballwithr.wordpress.com/2014/03/24/422/

Malter, D. (2016, March 31). Using markov chains to predict pitches. Retrieved May 5, 2017, from
Exploring Baseball Data with R website:
https://baseballwithr.wordpress.com/2016/03/31/
using-markov-chains-to-predict-pitches/

Malter, D. (2016). Using markov chains to predict pitches. Retrieved May 5, 2017, from Github.io
website: http://danmalter.github.io/r/2016/03/28/Markov-chains.html#

Pankin, M. D. (2016). Baseball as a markov chain. Retrieved May 5, 2017, from Pankin website:
http://www.pankin.com/markov/intro.htm